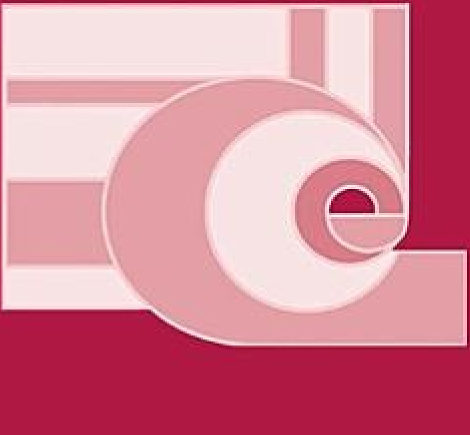
LNCS 6608

Alexander Gelbukh (Ed.)

Computational Linguistics and Intelligent Text Processing

12th International Conference, CICLing 2011
Tokyo, Japan, February 2011
Proceedings, Part I

1 Part I



Springer

Alexander Gelbukh (Ed.)

Computational Linguistics and Intelligent Text Processing

12th International Conference, CICLing 2011
Tokyo, Japan, February 20-26, 2011
Proceedings, Part I

Springer

Volume Editor

Alexander Gelbukh
Instituto Politécnico Nacional (IPN)
Centro de Investigación en Computación (CIC)
Col. Nueva Industrial Vallejo, CP 07738, Mexico D.F., Mexico
E-mail: gelbukh@gelbukh.com

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-19399-6 e-ISBN 978-3-642-19400-9
DOI 10.1007/978-3-642-19400-9
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011921814

CR Subject Classification (1998): H.3, H.4, F.1, I.2, H.5, H.2.8, I.5

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

CICLing 2011 was the 12th Annual Conference on Intelligent Text Processing and Computational Linguistics. The CICLing conferences provide a wide-scope forum for the discussion of the art and craft of natural language processing research as well as the best practices in its applications.

This set of two books contains four invited papers and a selection of regular papers accepted for presentation at the conference. Since 2001, the proceedings of the CICLing conferences have been published in Springer's *Lecture Notes in Computer Science* series as volume numbers 2004, 2276, 2588, 2945, 3406, 3878, 4394, 4919, 5449, and 6008.

The set has been structured into 13 sections:

– Lexical resources
– Syntax and parsing
– Part-of-speech tagging and morphology
– Word sense disambiguation
– Semantics and discourse
– Opinion mining and sentiment analysis
– Text generation
– Machine translation and multilingualism
– Information extraction and information retrieval
– Text categorization and classification
– Summarization and recognizing textual entailment
– Authoring aid, error correction, and style analysis
– Speech recognition and generation

The 2011 event received a record high number of submissions. A total of 298 papers by 658 authors from 48 countries were submitted for evaluation by the International Program Committee, see Tables 1 and 2. This two-volume set contains revised versions of 74 papers, by 227 authors, selected for presentation; thus the acceptance rate for this set was 25%.

The books feature invited papers by

– Christopher Manning, Stanford University, USA
– Diana McCarthy, Lexical Computing Ltd., UK
– Jun'ichi Tsujii, U. of Tokyo, Japan, and U. of Manchester and NacTeM, UK
– Hans Uszkoreit, Saarland University and DFKI, Germany

who presented excellent keynote lectures at the conference. Publication of extended full-text invited papers in the proceedings is a distinctive feature of the CICLing conferences. Furthermore, in addition to the presentation of their invited papers, the keynote speakers organized separate vivid informal events; this is also a distinctive feature of this conference series.

Table 1. Statistics of submissions and accepted papers by country or region

Country or region	Authors	Papers[1]	
	Subm.	Subm.	Accp.
Australia	17	7	3
Austria	2	1.33	0.33
Belgium	4	2	1
Brazil	5	2	1
Canada	11	6.33	2
China	47	17.67	5.67
Colombia	3	2	–
Croatia	3	2	–
Cuba	2	0.67	–
Czech Rep.	14	8.50	3
Egypt	9	2.67	1.67
Finland	3	2	–
France	36	16.68	4.83
Georgia	1	1	–
Germany	29	12.58	3.50
Greece	6	2	1
Hong Kong	5	2	1
India	85	41.75	6.42
Iran	23	18	3
Ireland	14	7	1
Israel	3	1.75	–
Italy	17	6.25	2.25
Japan	71	29.67	14
Jordan	1	0.50	–
Korea (South)	10	4.29	1
Macao	4	1	–
Malaysia	5	2	–
Mexico	13	6.92	2
Myanmar	7	2	–
Nigeria	3	1	–
Norway	7	2.17	–
Pakistan	6	3.57	–
Peru	2	0.50	–
Poland	2	2	–
Portugal	25	9.67	2
Romania	7	3.33	–
Russia	8	2.33	–
Saudi Arabia	1	1	–
Singapore	7	2.50	1
Spain	39	14.30	4.30
Sweden	5	1.39	1
Taiwan	13	5	–
Thailand	6	3	1
Tunisia	9	3.15	–
Turkey	8	4.17	1
UK	13	6.67	0.50
USA	39	17.87	4.53
Viet Nam	8	4.33	1
Total:	658	298	74

[1] By the number of authors: e.g., a paper by two authors from the USA and one from the UK is counted as 0.67 for the USA and 0.33 for the UK.

With this event we introduced a new policy of giving preference to papers with verifiable and reproducible results: we encouraged the authors to provide, in electronic form, a proof of their claims or a working description of the suggested algorithm, in addition to the verbal description given in the paper. If the paper claimed experimental results, we encouraged the authors to make available to the community all the input data necessary to verify and reproduce these results; if it claimed to advance human knowledge by introducing an algorithm, we encouraged the authors to make the algorithm itself, in some programming language, available to the public. This additional electronic material will be permanently stored on CICLing's server, www.CICLing.org, and will be available to the readers of the corresponding paper for download under a license that permits its free use for research purposes.

In the long run we expect that computational linguistics will have verifiability and clarity standards similar to those of mathematics: in mathematics, each claim is accompanied by a complete and verifiable proof (usually much

Table 2. Statistics of submissions and accepted papers by topic[2]

Accepted	Submitted	% accepted	Topic
13	40	33	Lexical resources
13	47	28	Practical applications
11	39	28	Clustering and categorization
11	44	25	Other
10	28	36	Acquisition of lexical resources
10	29	34	Statistical methods (mathematics)
10	52	19	Machine translation & multilingualism
9	25	36	Syntax and chunking (linguistics)
9	31	29	Semantics and discourse
9	58	16	Information extraction
7	46	15	Text mining
6	12	50	Symbolic and linguistic methods
6	50	12	Information retrieval
5	13	38	Parsing algorithms (mathematics)
5	16	31	Noisy text processing and cleaning
5	18	28	Summarization
4	11	36	Text generation
4	16	25	Opinion mining
4	17	24	POS tagging
3	7	43	Speech processing
3	8	38	Cross-language information retrieval
3	15	20	Word sense disambiguation
3	20	15	Formalisms and knowledge representation
2	6	33	Emotions and humor
2	13	15	Named entity recognition
1	5	20	Spelling and grammar checking
1	7	14	Anaphora resolution
1	7	14	Textual entailment
1	8	12	Question answering
1	11	9	Natural language interfaces
1	12	8	Morphology
–	6	0	Computational terminology

[2] As indicated by the authors. A paper may belong to several topics.

greater in size than the claim itself); each theorem —and not just its description or general idea—is completely and precisely presented to the reader. Electronic media allow computational linguists to provide material analogous to the proofs and formulas in mathematics in full length—which can amount to megabytes or gigabytes of data—separately from a 12-page description published in a book. A more detailed argumentation for this new policy can be found on www.CICLing.org/why_verify.htm.

To encourage the authors to provide algorithms and data along with the published papers, we introduced a new Verifiability, Reproducibility, and Working Description Award. The main factors in choosing the awarded submission were technical correctness and completeness, readability of the code and documentation, simplicity of installation and use, and exact correspondence to the claims of the paper. Unnecessary sophistication of the user interface was discouraged; novelty and usefulness of the results were not evaluated—those parameters were evaluated for the paper itself and not for the data.

The following papers received the Best Paper Awards, the Best Student Paper Award, as well as the Verifiability, Reproducibility, and Working Description Award, correspondingly (the best student paper was selected from the papers of which the first author was a full-time student, excluding the papers that received a Best Paper Award):

1st Place:	*Co-related Verb Argument Selectional Preferences*, by Hiram Calvo, Kentaro Inui, and Yuji Matsumoto;
2nd Place:	*Self-Adjusting Bootstrapping*, by Shoji Fujiwara and Satoshi Sekine;
3rd Place:	*Effective Use of Dependency Structure for Bilingual Lexicon Creation*, by Daniel Andrade, Takuya Matsuzaki, and Jun'ichi Tsujii;
Student:	*Incorporating Coreference Resolution into Word Sense Disambiguation*, by Shangfeng Hu and Chengfei Liu;
Verifiability:	*Improving Text Segmentation with Non-systematic Semantic Relation*, by Viet Cuong Nguyen, Le Minh Nguyen, and Akira Shimazu.

The authors of the awarded papers (except for the Verifiability Award) were given extra time for their presentations. In addition, the Best Presentation Award and the Best Poster Award winners were selected by a ballot among the attendees of the conference.

Besides its high scientific level, one of the success factors of CICLing conferences is their excellent cultural program. The attendees of the conference had a chance to visit Kamakura—known for the Kamakura period of ancient history of Japan—where they experienced historical Japanese cultural heritage explained by highly-educated local volunteers and saw Shinto (traditional religion of Japan) shrines and old Buddhist temples characteristic of Japan. They recalled recent history at the Daigo Fukuryu Maru Exhibition Hall, which tells the story of a Japanese boat exposed to and contaminated by nuclear fallout from a thermonuclear device test in 1954. Finally, the participants familiarized themselves with modern Japanese technology during guided tours to Toshiba Science Museum and Sony Square; the latter can only be accessed by special invitation from Sony. And of course they enjoyed Tokyo, the largest city in the world, futuristic and traditional at the same time, during an excursion to the Japanese-style East Gardens of the Imperial Palace and a guided tour of the city, by bus and boat (see photos on www.CICLing.org).

I would like to thank all those involved in the organization of this conference. In the first place these are the authors of the papers that constitute this book: it is the excellence of their research work that gives value to the book and sense to the work of all other people. I thank all those who served on the Program Committee, Software Reviewing Committee, Award Selection Committee, as well as the additional reviewers, for their hard and very professional work. Special thanks go to Ted Pedersen, Grigori Sidorov, Yasunari Harada, Manuel Vilares Ferro, and Adam Kilgarriff, for their invaluable support in the reviewing process.

I thank the School of Law and Media Network Center of Waseda University, Japan, for hosting the conference; the Institute for Digital Enhancement of Cognitive Development (DECODE) of Waseda University for valuable collaboration in its organization; and Waseda University for providing us with the best conference facilities. With deep gratitude I acknowledge the support of Professor Waichiro Iwashi, the dean of the School of Law of Waseda University, and Professor Toshiyasu Matsushima, Dean and Director of Media Network Center of Waseda University. I express my most cordial thanks to the members of the local Organizing Committee for their enthusiastic and hard work. The conference would not have been a success without the kind help of Professor Mieko Ebara, Ms. Mayumi Kawamura, Dr. Kyoko Kanzaki, and all the other people involved in the organization of the conference and cultural program activities.

My most special thanks go to Professor Yasunari Harada, Director of DECODE, for his great enthusiasm and infinite kindness and patience; countless nights without sleep, after a whole day of teaching and meetings, spent on the organization of the conference, from the strategic planning to the finest details. I feel very lucky to have had the opportunity to collaborate with this prominent scientist, talented organizer, and caring friend. From him I learnt a lot about human relationships as well as about planning and organization.

With great gratitude I acknowledge the financial support of the Kayamori Foundation of Information Science Advancement, which greatly helped us to keep the fees low. I would like to express my gratitude to the Kamakura Welcome Guide Association for making our visit to this historical city of Japan a memorable and enjoyable one. Thanks are also due to Sony and Totsusangyo Corporation, Toshiba Science Museum, and Daigo Fukuryu Maru Exhibition Hall, for arranging special visits and guided tours for CICLing 2011 participants. I would like to specifically recognize the help of Mr. Masahiko Fukakushi, Executive Officer and Corporate Senior Vice President of Toshiba Corporation, in arranging our visit to Toshiba Science Museum and the help of Dr. Atsushi Ito, Distinguished Research Engineer at KDDI R&D Laboratories, in providing wireless Internet access to the attendees of the conference.

The entire submission and reviewing process was supported for free by the EasyChair system (www.EasyChair.org). Last but not least, I deeply appreciate the Springer staff's patience and help in editing this volume and getting it printed in record short time—it is always a great pleasure to work with Springer.

February 2011 Alexander Gelbukh
General Chair

Organization

CICLing 2011 was co-hosted by the School of Law and Media Network Center of Waseda University, and was organized by the CICLing 2011 Organizing Committee, in conjunction with the Natural Language and Text Processing Laboratory of the CIC (Center for Computing Research) of the IPN (National Polytechnic Institute), Mexico, and the Institute for Digital Enhancement of Cognitive Development of Waseda University, with partial financial support from the Kayamori Foundation of Information Science Advancement.

Organizing Chair

Yasunari Harada

Organizing Committee

Glen Stockwell
Ryo Otoguro
Joji Maeno
Noriaki Kusumoto
Akira Morita
Kyoko Kanzaki
Kanako Maebo
Mieko Ebara
Mayumi Kawamura
Alexander Gelbukh

Program Chair

Alexander Gelbukh

Program Committee

Sophia Ananiadou
Bogdan Babych
Sivaji Bandyopadhyay
Roberto Basili
Pushpak Bhattacharyya
Nicoletta Calzolari
Sandra Carberry
Kenneth Church
Dan Cristea
Silviu Cucerzan
Mona T. Diab
Alex Chengyu Fang
Anna Feldman
Daniel Flickinger
Alexander Gelbukh
Roxana Girju
Gregory Grefenstette
Iryna Gurevych
Nizar Habash
Sanda Harabagiu
Yasunari Harada
Ales Horak
Eduard Hovy
Nancy Ide

Software Reviewing Committee

Award Committee

Additional Referees

Website and Contact

The website of the CICLing conference series is www.CICLing.org. It contains information about the past CICLing conferences and their satellite events, including published papers or their abstracts, photos, video recordings of keynote talks, as well as the information about the forthcoming CICLing conferences and the contact options.

Table of Contents – Part I

Table of Contents – Part II

Influence of Treebank Design on Representation of Multiword Expressions

Eduard Bejček, Pavel Straňák, and Daniel Zeman

Charles University in Prague, Institute of Formal and Applied Linguistics
{bejcek,stranak,zeman}@ufal.mff.cuni.cz

Abstract. Multiword Expressions (MWEs) are important linguistic units that require special treatment in many NLP applications. It is thus desirable to be able to recognize them automatically. Semantically annotated corpora should mark MWEs in a clear way that facilitates development of automatic recognition tools. In the present paper we discuss various corpus design decisions from this perspective. We propose guidelines that should lead to MWE-friendly annotation and evaluate them on numerous sentence examples. Our experience of identifying MWEs in the Prague Dependency Treebank provides the base for the discussion and examples from other languages are added whenever appropriate.

1 Motivation

Grammatical theories have been thriving recently in computational linguistics. They describe phenomena of natural language in increasing detail with the purpose of creating a description that analyses and/or generates language as natural as possible.

Several treebanks have been developed during the past decade, new ones are still being created and the old ones are being enriched with additional annotations. A corpus is often designed and developed with the vision of further, deeper annotation, with the aim to add semantic information in future. Multiword expressions (MWEs; such as idioms, phrasemes, multiword named entities) are an important part of most natural languages. Usually they form a significant portion of vocabulary, particularly in special domains where terminology is in play, but not only there.

Although some grammatical theories have accounted for MWEs decades ago (see e.g. [1]), in treebanks, multiword expressions are one of the least developed phenomena. Recently, however, their processing started to attract attention, as they are proving to be important for information extraction, machine translation and other crucial tasks of NLP [2]. Therefore they should be an integral part of any serious semantic annotation.

In this paper, we discuss some decisions of a treebank design that have direct influence on representation of MWEs. A good treebank design can contribute to both more natural and more useful representation of MWEs, or even enable to capture certain rare forms of MWEs. We will also discuss the decisions that make the representation of MWEs harder or inefficient (see Section 3).

We base the discussion on our experience with MWEs in the Prague Dependency Treebank 2.0 (PDT 2.0[1])[3]. Examples from other treebanks are presented for comparison. Examples that are not specifically marked are taken from PDT 2.0.

The rest of the paper is organized as follows: In Section 2, we provide some background on multiword expressions, and why they are important in NLP. In Section 3, we discuss the way MWEs are currently represented in selected treebanks, and what are the problems of these representations. Section 4 constitutes the core of the paper. We present a variety of linguistic phenomena and decisions of their representation that affect processing of MWEs to varying degree. We summarise our findings in Section 5.

2 Introduction to MWEs

Multiword expressions are a boundary phenomenon on the interface of grammar and lexicon. We understand them, in accordance with [4,5,6] and other authors, as phrases that contain some idiosyncratic elements that differentiates them from normal expressions. The idiosyncratic element can be morphological, syntactic, or semantic.[2] As a practical guideline we add that the idiomaticness must be significant enough to justify adding the given MWE into a lexicon.[3]

The idiosyncracy that defines the class of multiword expressions causes problems for various NLP applications.

- *Morphological* analysers have to analyse "words" that only exist in modern language as a part of an idiom (e.g. "criss" in criss-cross) in one fixed form. Even if it is a form of say singular, instrumental case, it does not fill such a morphological function.
- *Syntactic* analysers (and treebank designers before them) have to cope with analysis of idioms and other MWEs, in which the relations between the parts (words) do not have the meaning expressed by dependency relations or phrase structure types of the given grammar. The problem is usually solved by creating artificial annotation (grammar) rules with little to no linguistic motivation. Rules for analysis of named entities (NEs) like addresses or personal names can serve as good examples (see the relevant sections in [9]).
- *Semantic* idiosyncrasy limits the forms or even completely changes the translation equivalents of a MWE. One cannot translate "spill the beans" into a foreign language literally and keep its meaning. It is a big challenge for machine translation, especially in terminology (Supreme Court, Secretary of the Treasury, etc.).

The problems with handling MWEs in NLP applications are precisely why it is important to represent them correctly in treebanks. We will demonstrate that proper representation of MWEs can alleviate later problems with their treatment.

[1] `http://ufal.mff.cuni.cz/pdt2.0/`

[2] Some authors prefer still wider definition of MWEs and include also expressions that are fully regular and compositional on all layers of description, but are *statistically* significant. For instance the phrase "salt and pepper" is significantly more frequent than "pepper and salt" [7,6].

[3] For a description of a lexicon of MWEs see for instance [8].

For the purpose of this paper the problem whether a particular expression is a MWE or not is not crucial. What is important, is an agreement that MWEs exist, and that in representing them the linguistic phenomena discussed below have to be tackled.

3 Representation of MWEs

A handful of corpora provide MWE annotation on the layer of tokenisation. That means a MWE is actually converted to one-word expression. Not only is it an indivisible meaning from the perspective of deep syntax; it is also one token from the point of view of morphology and surface syntax. Even if the treebank does not have a dedicated deep-syntactic layer of annotation, the idiomaticness of the MWE can be captured by the annotation; the price is that it is no more possible to describe the inner structure of the MWE as well, should one desire that. Tokenisation-based annotation is typically limited to contiguous MWEs (otherwise, one would have to reorder tokens, apart from joining them). The CoNLL Shared Task Treebanks ([10], [11]) of Portuguese, Spanish and Catalan belong to this class. For instance, consider the following Spanish sentence:

(1)

sentence:	Durante	la	presentación	del	libro	"	**La_prosperidad_por_-**
lit.:	During	the	presentation	of-the	book	"	**The_prosperity_through_-**

_medio_de_la_investigación_._La_investigación_básica_en_EEUU	"	,
_means_of_the_research_._The_research_basic_in_U.S.	"	,

editado	por	la	**Comunidad_de_Madrid**	,	el	secretario	general	de	la
edited	for	the	**Community_of_Madrid**	,	the	secretary	general	of	the

Confederación_Empresarial_de_Madrid-CEOE	–	CEIM	–	,
Confederation_of-Company_of_Madrid-CEOE	–	CEIM	–	,

Alejandro_Couceiro	,	abogó	por	la	formación	de	los	investigadores	en
Alejandro_Couceiro	,	advocated	for	the	formation	of	the	investigators	in

temas	de	innovación	tecnológica	.
themes	of	innovation	of-technology	.

trans.: During the presentation of the book “Prosperity through Research. The Basic Research in the U.S.”, edited for the Community of Madrid, the Secretary General of the Confederación Empresarial de Madrid (CEOE), Alejandro Couceiro, advocated for the training of researchers in the field of technological innovation.

We believe that MWEs should be viewed as single units, but not on the morphological layer, as in the above mentioned Iberian treebanks. Even in terms of surface syntax, it is usually possible[4] to view MWEs as relations between words. It is the layer of the meaning of a sentence, i.e. deep syntax, where it is natural to tackle MWEs as single units, because units of this layer are supposed to be “meanings” [1,12,13]. In the PDT 2.0, the deep syntactic layer is called the *tectogrammatical layer* [9] and we demonstrate (mainly in Section 4) that it is the layer of description most suitable to represent MWEs.

[4] Even though sometimes quite awkward.

The same is true for other treebanks that already include some deep structures: in the beginning of treebanking, all treebanks (including PDT 1.0) were based only on the surface syntax. Most of them, however, have been accepting some deep syntactic features. These include PropBank [14] and NomBank [15] for the Penn Treebank, Chinese PropBank [16] and annotation of some named entities integrated in the in recent Chinese Treebank (see Figure 1), Salsa project [17] for the German Tiger treebank, and several others. The main problem of most of these annotation projects is, however, that the deep structures are annotated without any relation to the (surface) syntax, thus often ending up in conflict with it.

An illustration of this problem is given in Figure 1. The NEs, as well as coreference, were annotated on plain text and are stored separately from the syntactic annotation of the Chinese Treebank. This results in many cases in a unit of coreference annotation or a NE that does not form a phrase and thus cannot be represented in the tree. This points towards an error of either syntactic or deeper annotation, because any unit that is a member of a coreference relation or that forms a named entity should also form a phrase in a phrase structure tree.

3.1 List Structures

Some MWEs really have no internal syntactic structure in the given language. For instance embedded passages in a foreign language cannot be analysed using the grammar of the "main" language of the treebank.

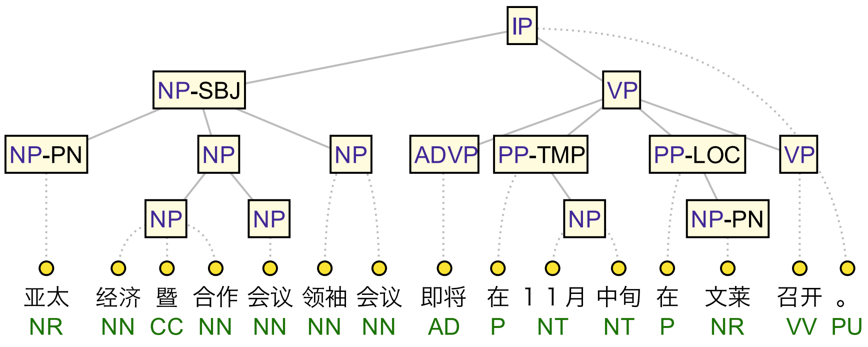


Fig. 1. A sentence from the Chinese Treebank 7, romanised *yàtài jīngjì jì hézuò huìyì lǐngxiù huìyì jíjiāng zài 11yuè zhōngxún zài wénlái zhàokāi*, meaning *"Asia-Pacific Economic Cooperation [APEC] Summit will be held in mid-November in Brunei."* lit. *"Asia-Pacific economy and cooperation conference leader meeting upcoming in November mid in Brunei hold."* The first five terminal nodes together constitute a named entity (MWE) that is the Chinese translation of *APEC*. However, the syntactic annotation does not contain any nonterminal spanning just this expression. The `NP-SBJ` span includes two additional terminals and describes an event (meeting of APEC leaders) rather than the institution. On the other hand, its second child `NP` fails to cover the node of *Asia-Pacific*. Thus the MWE cannot be properly marked without changing the parse tree first.

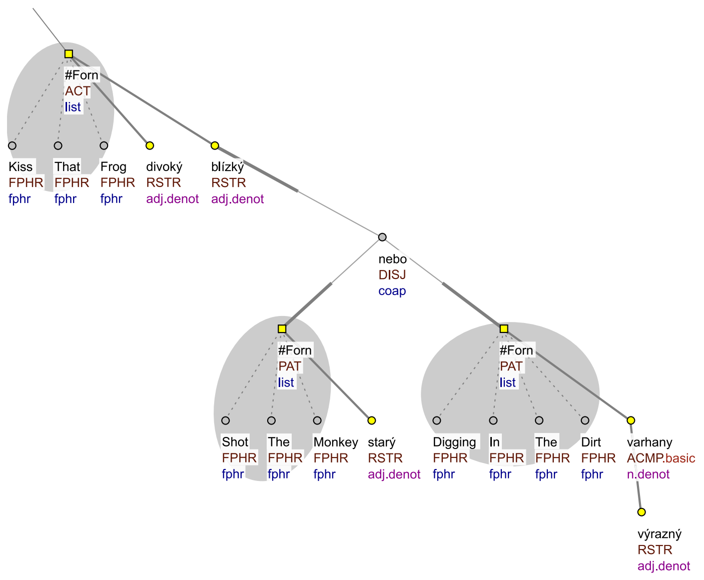


Fig. 2. *. . . divoké Kiss That Frog blízké staršímu Shot The Monkey nebo Digging In The Dirt s výraznými varhanami.*
trans.: . . . wild Kiss That Frog similar to older Shot The Monkey or Digging In The Dirt with striking organ.
Foreign expressions (English in the Czech sentence) represented as lists. The first MWE is modified by an attribute "similar (to)" and a coordination of the other two MWEs that are also further modified. (Example from the PDT 2.0.)

In PDT 2.0 these constructions are represented as lists of words with a generated root node that has a t-lemma[5] substitute specifying the type of the list (an Idiomatic Phrase, or a Foreign Phrase).

The list members (words in a list structure) cannot have children, since the whole point of creating these list structures is to specify either that there are no syntactic relations inside these objects, or that we cannot describe them. The whole structure can of course have children (e.g. attributes). Such children are represented as brothers of the members of list structures, and are distinguished by their tectogrammatical function, as seen in Figure 2.

We believe that creating list structures with artificial rigid and flat structures serves no point. Lemmas of the parts of such structures are foreign morphological forms (e.g. "shot"), and the dependency edges do not really represent any dependency relations. Thus we believe that non-analysable idioms and foreign phrases should be represented just as a single node.[6]

[5] A lemma of a node of a tectogrammatical tree, i.e. a tree on the tectogrammatical layer.

[6] One may also want to annotate the original structure according to the foreign grammar in parallel to the one-node representation assigned to the phrase once it entered the host language and became a MWE.

4 Linguistic Phenomena Reflected by Treebank Features

We present an overview of common linguistic phenomena that complicate capturing the MWEs. Every phenomenon is described and exemplified, the problem is discussed and a potential solution in the dependency treebank is proposed.

Two principles are to be borne in mind while making decisions on the structure of a treebank:

1. Structure of a tree must not obstruct marking any MWE.
2. Representation of a MWE has to enable identification of the same MWE automatically in the text.

How does one represent a MWE in a treebank? As the tree structure is non-linear, the best representation is a set of nodes that make up a particular MWE. This set has to be unambiguous, i.e. two different MWEs should not be represented by the same set of nodes.[7] On the other hand, slight variations in form of the same MWE should lead to the same representation so that the various forms of the MWE can be matched against each other. The set of nodes itself for a particular MWE highly depends on the treebank grammar and it is generally not guaranteed that every peculiar MWE can be mapped to a tree structure. For example, the MWE may contain a word that has been elided and does not have a corresponding node in the tree structure. In other cases, deep syntactic structure may contain a complex node spanning several surface words, some of which belong to a MWE and some of which do not; one would have to be able to mark only a part of a complex node in order to delimit the MWE properly.

The second principle leads to this aim: each and every instance of the particular MWE should be described by absolutely identical structure in data. In that case, it would be easy to find other instances of the same MWE automatically (using the same treebank or formalism). Following subsections illustrate that this is not as natural as it might seem.

4.1 Morphology

MWEs are hard to recognize automatically in an unprocessed text. Lemmatisation (or at least stemming) is the minimum requirement—even in English, not speaking of highly inflected languages such as Czech.

Consider the two instances of the German idiom *auf die lange Bank schieben* ("put off") in Examples (2) and (3) and Figure 3. The first one is in infinitive, the second one is passive. However, their lemmatised strings are identical, which makes it possible to recognize them as instances of the same MWE.

(2)	*sentence:*	Die	EU	dürfe	die	Entscheidung	nicht	**auf**	**die**	**lange**	**Bank**	**schieben** …
	lemmatised:	Der	EU	dürfen	der	Entscheidung	nicht	**auf**	**der**	**lang**	**Bank**	**schieben** …
	lit.:	The	EU	could	the	decision	not	**on**	**the**	**long**	**bench**	**shift** …

trans.: The EU could not put the decision off …

[7] i.e. The structure of a subtree plus the words (lemmas) themselves.

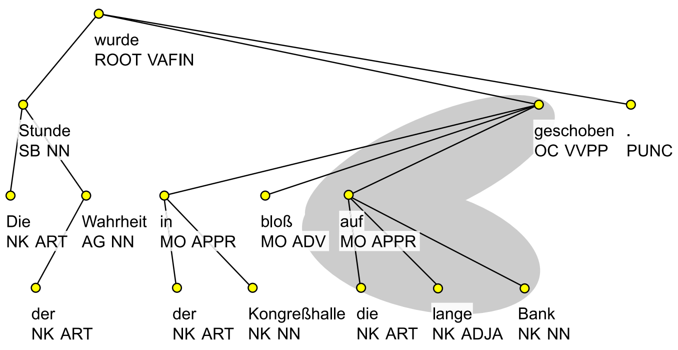


Fig. 3. An example from the CoNLL 2009 German treebank: *Die Stunde der Wahrheit wurde in der Kongreßhalle bloß auf die lange Bank geschoben. The moment of truth in the congress hall was just put off (lit. "shifted on the long bench").* The idiom has been passivised in this sentence but it still can be identified using lemmas.

(3)

sentence:	Die	Stunde	der	Wahrheit	wurde	in	der	Kongreßhalle	bloß	**auf**
lemmat.:	Der	Stunde	der	Wahrheit	werden	in	der	Kongreßhalle	bloß	**auf**
lit.:	The	moment	of	truth	was	in	the	congress-hall	just	**on**

die	**lange**	**Bank**	**geschoben**.
der	**lang**	**Bank**	**schieben**.
the	**long**	**bench**	**shifted**.

trans.: The moment of truth in the congress hall was just put off.

One might think that lemmatisation is a solved problem and that an annotated corpus is unlikely to lack it. As a matter of fact, out of the 23 treebanks from the CoNLL 2006 and 2007 shared tasks, lemmatisation was missing from significant number of them (Bulgarian, Chinese, Danish, German, Japanese and Swedish 2006 and English and Chinese 2007).

4.2 Word Form Alternations

There are many changes of word forms other than inflection mentioned in previous Section 4.1. Although these changes are more significant than morphological alternations, they still do not necessarily change the meaning of the MWE.

Lemmas in their usual sense cannot provide for unification of the alternations mentioned below, since the alternations differ morphologically and a considerable number of lemmatisers would assign a different lemma to each of them. In order to capture the relation between morphologically different expressions for a semantically identical concept, we need a sort of *generalized lemma*, common for all word form alternations.[8]

[8] Functional Generative Description (FGD, [12]), the theory behind PDT, introduces such a generalized concept called the "tectogrammatical lemma". The deep (tectogrammatical) layer of PDT 2.0 assigns a `t-lemma` attribute to nodes but it fails to merge some of the alternations mentioned here.

An alternative approach would be to annotate each MWE with its exact lemma, and create links between "variants" in the lexicon. The drawback here would be the large amount of lemma variants (some of them created productively on a regular basis) all written in the lexicon. The additional complexity could however bring also some additional information, i.e. in case of lemmas whose relation can be described by lexical functions [18]. Some variants of lemmas cannot, however, be distinguished by a lexical function, e.g. variants of diminutives in Czech. Some of the (spelling) variants are even unified on the level of morphology, while some other are not, and we unify them only in the MWE lemmas. Thus we have decided to employ the simple and uniform approach of using the same MWEs for all lemma variants. We can list and further analyse and classify all the variant realisations of all MWEs at a later point. We view the application of a lexical function in this respect as a form of a modification of a MWE, very much like any other modification, with similar restrictions: Some words in some MWEs can be modified, while other words or even whole MWEs cannot. Thus the approaches can be complimentary in our view.

Gender Inflection. The first alternation type we want to mention is present in many languages, including English, French or German. Since gender inflection of nouns is not productive in any of these languages, the alternate forms are assigned separate lemmas. Examples include pairs like "waiter"/"waitress", "actor"/"actress", "écrivain" or "homme de lettres"/"femme de lettres" etc. Examples of such pairs used in Czech MWEs are quoted below. We believe that the core meaning of the MWE remains the same across genders and it should be differentiated by a flag, not by a separate MWE in the lexicon.

We indicate the approximate occurrence ratio in PDT 2.0 in parentheses.

(4) mistr / mistryně světa (ratio 76:1)
master / she-master of-the-world
world champion

(5) státní zástupce / zástupkyně (ratio 2:1)
public prosecutor / prosecutrix
prosecuting attorney

(6) poštovní doručovatel / doručovatelka (ratio 1:2!)
postman / -woman
postman / postwoman

We propose that in each of the pairs, both variants should map to the same generalized lemma. One may wonder whether the actual string representing the generalized concept in (6) should match the masculine form (as is the usual default), or the feminine form (because in this particular case it seems to be more common in Czech data), or somehow embrace both (e.g. *poštovní_doručovatel(ka)*). However, these are only technical subtleties that are not significant from the perspective of the general concept-oriented approach.

Abbreviations. Writing systems of most languages have a means of abbreviating words and long multiword named entities. Examples of abbreviated and unabbreviated forms referring to the same concept are given in (7) and (8). Again, we propose that the corpus annotation assign the same generalized lemma to both members of each such pair.

(7) Václav Havel / V. Havel

(8) země bývalého Sovětského svazu / země bývalého SSSR
states of-former Soviet Union / states of-former USSR

Aspect. In quite a few languages (the Slavic family being an example) aspect alternation is lexicalised (or at least not fully productive), which means that perfective and imperfective verbs get different surface lemmas. The following Czech examples (9) and (10) illustrate aspectual variations of MWEs.[9]

(9) zaujímat stanovisko / zaujmout stanovisko
take a stand *imperfective* / *perfective*

(10) pohlavně zneužívat / pohlavně zneužít
sexually abuse *imperfective* / *perfective*

Diminutives. Unless diminutive formation is fully productive in a language, the diminutive typically gets a (surface) lemma different from the base word. Yet the core meaning of a MWE is usually preserved in a "diminutivized" variant such as in the following Czech example (11):

(11) rodinný dům / domek
family house / small-house

Others. For the sake of completeness we bring up some other related pairs, although it is arguable whether it is necessary to unify them all. They have very close meanings and one has to consider them carefully. The variants in (12) are lexical meronyms but their encompassing MWEs are almost synonymous (furthermore, the second one is rarely used). The second expression (13) has the same meaning, only the first form is fixed phrase and the second is less formal. The pair in (14) has exactly the same properties in English. And the last one (15) illustrates an ellipsis[10] of a part of a word; the two expressions are totally synonymous in the context of telecommunications.

(12) občanský zákoník / občanský zákon — *meronym*
civil code / civil law

(13) náčelník generálního štábu / šéf generálního štábu — *synonym*
chief of-general staff / head of-general staff

(14) cenová regulace / regulace cen — *synonymous, though different*
price_{adj} control / regulation of-prices *syntactic structures*

(15) telekomunikační systém / komunikační systém — *ellipsis*
telecommunication system / communication system

[9] Aspect is an exception that *is* unified in the `t-lemma` attribute of PDT 2.0, except for a few omissions.

[10] If we substitute the Greek prefix "tele-" in *tele*communication in (15) with its translation *remote*, the fact that it is an ellipsis becomes obvious.

4.3 Word Order

Lemmatisation is not sufficient (not even the generalized one) when the word order comes into play. In languages with a free word order, the same MWE can surface in various permutations. For instance, consider the phraseme "sehrát roli" (to play a role) in (16) and (17). The two instances differ in word order. The first sentence is neutral with respect to topic-focus articulation (i.e. it keeps the default Subject-Verb-Object order), whereas the second sentence accents the Subject ("communistic interpretation of history") by placing it into the focus position (resulting in the Object-Verb-Subject order).

(16) *sentence:* Klubíčko vztahů, které **sehrály roli** v této kauze,
lit.: Entanglement of-relations, that played role in this case,

se pokoušíme rozmotat...
we-are-trying to-disentangle...

trans.: We are trying to disentangle the entanglement of relations that played a role in this (legal) case.

(17) *sentence:* Svou **roli sehrál** i komunistický výklad historie.
lit.: Its role played even communistic interpretation of-history.

trans.: Even the communistic interpretation of history played its role.

The word order differences are a good reason why MWE detection should be done on dependency trees (as opposed to simple bracketing). Instead of looking at sequences of adjacent tokens, one can query parent-child pairs that remain the same regardless of word order. See Figure 4 for the dependency trees of (16) and (17).

4.4 Discontinuity on Surface

Discontinuous MWEs pose a problem similar to the word-order issue. Even a very lexicalised phrase (such as a verbal phraseme) can be disconnected with other words breaking in. In a phrase-based bracketing one would have to capture a MWE with gaps. The results would differ across sentences (different positions and sizes of the gaps) and there seems to be no reasonable algorithm to recognize them automatically.

Examples (18) and (19) illustrate that continuity is not related to MWE boundaries. There seems to be the phrase "hrát na nervy" ($\sim$ to fray one's nerves) twice – but only the first one (the one with gaps) is a real phraseme; the words in (19) came together just by coincidence.

(18) *sentence:* **Na nervy** to muselo **hrát** i našemu olympijskému vítězi.
lit.: **On nerves** it must-have **play** also to-our Olympic winner.

trans.: It must have been making nervous even our Olympic winner.

(19) *sentence:* Je to balzám **na nervy hrát** s Jenseny.
lit.: Is it balm **for nerves to-play** with Jensen's.

trans.: To play with Jensen's is like a balm for your nerves.

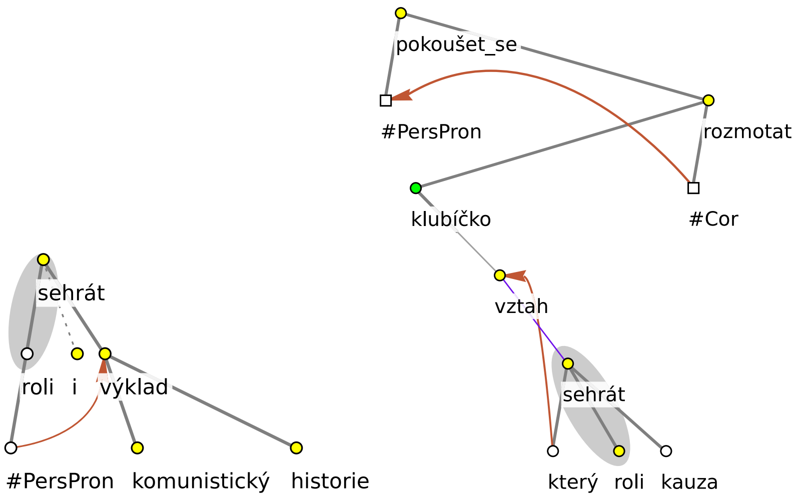


Fig. 4. Although the MWEs look diverse in the text (examples 16 and 17), they are identical and so are their subtrees. (It is not important whether a node is the left or the right son of its parent – the order of nodes represents the topic-focus articulation and does not affect the MWE.)

Similarly as in Section 4.3, we argue for the dependency structure as the basis for MWE detection. A dependent node of a MWE ("nervy" in this case) is connected to the governing node ("hrát"), no matter how far it is or in which direction. On the other hand, the parts of the would-be MWE in (19) are unrelated in the dependency tree, which blocks them from being considered as a MWE.

Dependency subtrees (with word order information stripped) provide sufficient means of representation for a vast majority of MWEs. They adhere to the second principle and assign the same representation to all instances of a MWE, regardless of word order and gaps. Unfortunately, there are still phenomena that cause problems.

4.5 Ellipsis

In (20) both "Ministry of the Interior" and "Ministry of Defense" should be recognized as MWEs. The problem is that there is only one word "ministry". The annotation mechanism would have to enable reusing one node in two different MWEs. Even if it did, a surface-oriented dependency tree (where there is a 1-1 mapping between nodes and tokens) would not provide enough information to detect the MWEs automatically (there would be no dependency link between "ministry" and "interior" or "defense", respectively).

(20)	dvě	klíčová	ministerstva	– vnitra	a	obrany
	two	key	ministries	– of-the-Interior	and	of-Defence

This example illustrates why we need a deep syntactic tree in which elided nodes are reconstructed. Figure 6 illustrates how the example is structured in the tectogrammatical

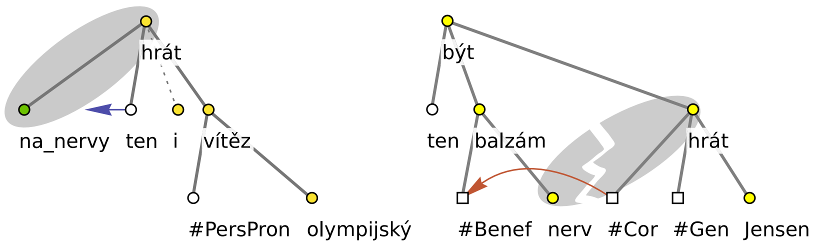


Fig. 5. Seemingly two occurrences of the phraseme "hrát na nervy" (∼ to fray one's nerves) in Examples (18 and 19). The dependency tree of (19) indicates that the idiomatic interpretation would be false. Such MWE across subtrees cannot be correct.

layer of the PDT 2.0. Thanks to the generated copies of "ministerstvo", the links to the required attributes are readily available and the MWEs can be detected easily.

Finally, there is an even worse problem with coordination: a coordination of two modifiers ascribed to an already modified noun, see (21):

(21)	*coord.:*	základní	a	náhradní	vojenská	služba
	lit.:	basic	and	substitute	military	service

trans.: military service and unarmed service

To be able to recognize both MWEs, we would like to see two complete (and disjunct) subtrees, one for "základní vojenská služba" and another for "náhradní vojenská služba". Node reconstruction in a deep syntactic tree could achieve that by generating copies of both the nodes "vojenská" and "služba". Unfortunately, this is not the case in PDT 2.0 where only the noun is copied, see Figure 7.

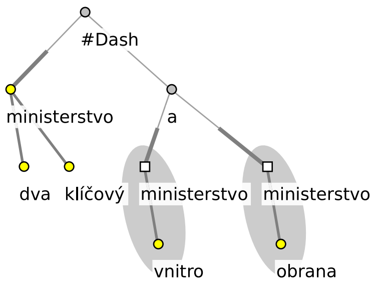


Fig. 6. A coordination with generated nodes (displayed as squares) enables annotation of words elided in the text (20)

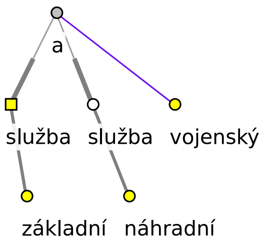


Fig. 7. The word "vojenský" (military) modifies the whole coordination in PDT 2.0, instead of modifying each coordinated node "služba" (service) with the same meaning

To summarize this section, we propose that elided nodes in coordination should be regenerated by copying *and* that their modifiers should be copied along, i.e. modification of the whole coordination should not be allowed.

5 Conclusion

We have discussed a number of linguistic phenomena that affect representation and automatic detection of multiword expressions in corpora. Each phenomenon led to a type of additional information that is needed in the corpus in order to detect MWEs properly. Such information can be added either manually in annotated corpora, or by previous steps of automatic processing of the text.

The following features of a treebank have been identified as useful for appropriate and efficient representation of MWEs:

- Surface lemmatisation to overcome the impact of inflection.
- Generalized lemmatisation to unify surface lemmas referring to the same semantic concept.
- Dependency structure to abstract from word order variation and discontinuity.
- Restoring nodes for elided words. In case of coordinated modifiers, restoring can be achieved relatively easily by copying the modified node to each coordination member.

We have tested our proposals while annotating MWEs in PDT 2.0, using the deep syntax of its tectogrammatical layer. They proved to be helpful from the perspective of both the principles set in Section 4.

Our annotated data, a lexicon of MWEs in PDT 2.0, and the tools we have used are freely available at `http://ufal.mff.cuni.cz/lexemann/mwe/`.

Acknowledgement

The research has been supported by the grant P406/11/1499 of the Czech Science Foundation, the grant GAUK 4307/2009 of the Grant Agency of Charles University in Prague, and projects no. LC536, MSM 0021620838, and “Clarin”, no. 7E08026 of the Ministry of Education of Czech Republic.

References

1. Mel’čuk, I.A., Polguère, A.: A formal lexicon in The Meaning-Text Theory (or how to do lexica with words). Computational Linguistics 13(3-4), 261–275 (1987)
2. Ren, Z., Lü, Y., Cao, J., Liu, Q., Huang, Y.: Improving statistical machine translation using domain bilingual multiword expressions. In: Proceedings of the Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications, pp. 47–54. Association for Computational Linguistics, Singapore (2009)
3. Bejček, E., Straňák, P.: Annotation of multiword expressions in the Prague dependency treebank. Language Resources and Evaluation (44), 7–21 (2010)
4. Sag, I.A., Baldwin, T., Bond, F., Copestake, A., Flickinger, D.: Multiword expressions: A pain in the neck for NLP. In: Gelbukh, A. (ed.) CICLing 2002. LNCS, vol. 2276, p. 1. Springer, Heidelberg (2002)
5. Baldwin, T., Bannard, C., Tanaka, T., Widdows, D.: An empirical model of multiword expression decomposability. In: Proceedings of the ACL 2003 Workshop on Multiword Expressions, pp. 89–96. Association for Computational Linguistics, Morristown (2003)

6. Pecina, P.: Lexical Association Measures: Collocation Extraction. Studies in Computational and Theoretical Linguistics, vol. 4. Institute of Formal and Applied Linguistics, Prague (2009)
7. Baldwin, T.: Multiword expressions, CSSE, University of Melbourne (2004)
8. Straňák, P.: Annotation of Multiword Expressions in The Prague Dependency Treebank. PhD thesis, Charles University in Prague (2010)
9. Mikulová, M., Bémová, A., Hajič, J., Hajičová, E., Havelka, J., Kolářová, V., Kučová, L., Lopatková, M., Pajas, P., Panevová, J., Razímová, M., Sgall, P., Štěpánek, J., Urešová, Z., Veselá, K., Žabokrtský, Z.: Annotation on the tectogrammatical level in the Prague Dependency Treebank. annotation manual. Technical Report 30, ÚFAL MFF UK, Prague, Czech Rep. (2006)
10. Buchholz, S., Marsi, E.: CoNLL-X shared task on multilingual dependency parsing. In: Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X), pp. 149–164. Association for Computational Linguistics, New York City (2006)
11. Nivre, J., Hall, J., Kübler, S., McDonald, R., Nilsson, J., Riedel, S., Yuret, D.: The CoNLL 2007 shared task on dependency parsing. In: Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007, pp. 915–932. Association for Computational Linguistics, Praha (2007)
12. Sgall, P., Hajičová, E., Panevová, J.: The Meaning of the Sentence in Its Semantic and Pragmatic Aspects. Academia/Reidel Publ. Comp., Praha (1986)
13. Kahane, S.: The Meaning-Text Theory. In: Dependency and Valency, Handbooks of Linguistics and Communication Sciences, vol. 25(1-2), p. 32. De Gruyter, Berlin (2003)
14. Palmer, M., Gildea, D., Kingsbury, P.: The Proposition Bank: A corpus annotated with semantic roles. Computational Linguistics Journal 31(1) (2005)
15. Meyers, A.: Using treebank, dictionaries and GLARF to improve NomBank annotation. In: Proceedings of The Linguistic Annotation Workshop, LREC 2008, Morocco (2008)
16. Xue, N.: Labeling Chinese predicates with semantic roles. Computational Linguistics 34(2), 225–256 (2008)
17. Burchardt, A., Erk, K., Frank, A., Kowalski, A., Padó, S., Pinkal, M.: The SALSA corpus: a German corpus resource for lexical semantics. In: Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006), Citeseer (2006)
18. Mel'čuk, I.: Lexical functions: A tool for the description of lexical relations in a lexicon. In: Wanner, L. (ed.) Lexical Functions in Lexicography and Natural Language Processing. SLCS, vol. 31, pp. 37–102. John Benjamins, Amsterdam (1996)

Combining Contextual and Structural Information for Supersense Tagging of Chinese Unknown Words

Likun Qiu[1], Yunfang Wu[1], and Yanqiu Shao[2]

[1] Key Laboratory of Computational Linguistics (Peking University), Ministry of Education 100871 Beijing, China
[2] Institute of Artificial Intelligence, Beijing City University, 100083 Beijing, China
{qiulikun,wuyf}@pku.edu.cn, yqshao@bcu.edu.cn

Abstract. Supersense tagging classifies unknown words into semantic categories defined by lexicographers and inserts them into a thesaurus. Previous studies on supersense tagging show that context-based methods perform well for English unknown words while structure-based methods perform well for Chinese unknown words. The challenge before us is how to successfully combine contextual and structural information together for supersense tagging of Chinese unknown words. We propose a simple yet effective approach to address the challenge. In this approach, contextual information is used for measuring contextual similarity between words while structural information is used to filter candidate synonyms and adjusting contextual similarity score. Experiment results show that the proposed approach outperforms the state-of-art context-based method and structure-based method.

Keywords: Supersense Tagging, Contextual Information, Structural Information, Chinese Unknown Words.

1 Introduction

Lexical-semantic resources such as *WordNet* [1] have influenced NLP research significantly. These resources have been successfully applied in a wide range of research [2, 3, 4, 5]. However, to keep up with the pace of language evolution, lexicographers should update the resources by hand, which is time-consuming and labor-intensive. To reduce human effort, a technology called supersense tagging [6] is presented to help lexicographers classify unknown words and insert them into an existing resource. Here, *supersense* refers to a semantic class to which the unknown word belongs, e.g., “tool”, “organization”, “person”, etc. A similar name of that task is semantic classification [7].

To address the problem of supersense tagging, two kinds of information might be utilized, i.e., contextual information and structural information.

Pilot studies on English unknown words mainly used contextual information [6, 8]. The methods of these studies are based on the *distributional hypothesis*, which means that words having similar meaning usually appear in similar context. The experiments of these studies show the effectiveness of contextual information for English.

However, pilot studies on Chinese unknown words [7, 9, 10] mainly used structural information. The experiments of these studies show the effectiveness of structural information for Chinese. To understand the idea of structure-based methods, we must notice that Chinese words are composed of meaningful characters, e.g., means ‘person’. Therefore, it is reasonable to believe that an unknown word *w* probably has the same supersense with a known word w_1 if *w* and w_1 share some characters, e.g., w_1 = 市 shi-min ‘citizen’. Structure-based methods can be used in Chinese but not English, because most Chinese characters are meaningful while English characters are not.

Then, one question is, how about using context-based method in Chinese? Since contextual information has been proved effective in English, it is reasonable to believe that it should also be effective in Chinese, as *distributional hypothesis* seems do not have difference among languages. But previous studies gave negative answer. [11] examined a context-based method but only achieved 34% accuracy.

Another question is can contextual information be integrated with structural information to improve the whole performance of supersense tagging on Chinese? [10] is the only one that has tried this idea. According to [10], structure-based methods are the main part and a context-based method is only used to rank the candidate supersense provided by the structure-based methods. That is a loose-coupled combination. [10]’s experiments show that the use of contextual information doesn’t further improve performance on the basis of structural information. Therefore, contextual and structural information have not been successfully integrated yet.

In this study, tightly-coupled strategy is proposed to combine contextual and structural information together for supersense tagging of Chinese unknown words. Context-based approach forms the main part of this strategy. It is used to measure the contextual similarity between two words. Structure-based approach is tightly-coupled with context-based approach. Firstly, the structure-based approach generates a candidate synonym set. The context-based method would be used to measure the similarity between the test word and each of the word among the candidate synonym set. Secondly, the structure-based approach adjusts the similarity scores computed by context-based approach. The basic idea of the using of structural information in this strategy is the assumption that: the more common characters two Chinese words share, the closer their meanings are. Both the process of candidate synonym set building and contextual similarity score adjusting are based on this assumption. We experiment on a Chinese thesaurus *Cilin* and extract context for test words from a large corpus to validate the effectiveness of the proposed approach. We also compare several weighting schemes in measuring contextual similarity.

The remainder of this paper is organized as follows. Section 2 introduces related work on supersense tagging. The proposed approach is described in Section 3. Experiments results and error analysis are presented in Section 4 and Section 5 respectively. The final section gives out conclusion and presents future work.

2 Related Work

2.1 Supersense Tagging in Chinese

Most previous studies on Chinese used structural information. [7] proposed a similar-example-based method. The similarity of the modifiers of two words that share the same head is computed to represent the similarity of the two words. [9] attempted to detect the morpho-syntactic relationship between the morphemes of a word and filter retrieved candidate synonyms by morpho-syntactic relationship. Both [10] and [12] computed the association between character and semantic categories. Except for the character-category association model, [10] developed dozens of manual rules, which constructed the rule-based model and achieved high precision with low recall. The result of Lu's rule-based model shows that only about 30% Chinese words obey strict rules. For most Chinese words, the mapping from part meaning to whole meaning would encounter too many WSD problems. This is just the limitation of structure-based methods.

[11] did the only study that used contextual information alone to do Chinese supersense tagging. From translation, WordNet is used to providing candidate categories for a Chinese unknown word. Then context comparison is used to choose a final category from those candidate categories. The accuracy of this approach is only about 34%.

[10] combined two knowledge-based models together with a context-based model. It might be considered as two separate methods: one contains the two knowledge-based models and so is structure-based (called structure-based approach); the other contains all the three models and is both context-based and structure-based (called hybrid approach). The hybrid approach included two steps. Firstly, the two knowledge-based models provide some candidate supersenses for the unknown words. Secondly, the context information is computed and compared to determine which supersense should win. The structure-based approach (about 61% F1-score) outperforms previous methods on the same task, but the hybrid approach (only about 37% F1-score) does not further improve performance.

2.2 Supersense Tagging in English

Most previous studies on English used context information. [13] used a vector-space technique to insert words into the *WordNet* hierarchy. [6] implemented a multi-class perceptron classifier, which takes those features commonly used in WSD and named entity recognition, such as standard collocation, spelling and syntactic relations. Their originality was to use the *WordNet* glosses as annotated training data and massively increase the number of training instances with the help of the noun hierarchy. [14] proposed a classification method using both context feature of an unknown word and the strength of the semantic relatedness of its target class to other likely candidate classes. They tried to classify nouns into 137 classes and achieved 35.1% precision.

[8] described an unsupervised approach based on vector-space similarity. Although that approach does not require annotated examples, it significantly outperforms the method of [6] (68% against 53% precision) on English unknown nouns with 26 supersenses.

2.3 Weighting Scheme

The difference between term frequency and term presence is a serious problem in standard IR and sentiment classification. Term frequencies have traditionally been important in standard IR [15], where TFDIF usually performs better than IDF. [16] compared several term weighting schemes, including TFIDF, IDF and their normalized form, on sentiment classification task. The results show: (1) normalized TFIDF and normalized IDF performs best and very similar; (2) BOOL performs not well but is still better than TF; (3) IDF is even worse than TF. This conclusion is similar to that on standard IR [15].

To measure contextual similarity, PMI and TTEST are more popular [8, 10]. Both PMI and TTEST are based on term frequency and co-occurrence frequency. Schemes such as TFIDF and IDF are rarely used to measure contextual similarity.

The task of measuring contextual similarity is quite different from standard IR and sentiment classification, but it is also a kind of classification problem. It is necessary for us to comparing TFIDF, IDF and their normalized form with PMI and TTEST.

3 The Proposed Method

For an unknown word, the proposed method analyzes its structure to collect some known words from a thesaurus to work as its candidate synonyms. Then the contextual similarity is computed between the unknown word and the candidate synonyms. Finally, the method selects some most similar candidate synonyms, and assigns the dominant supersense among those synonyms to the unknown word. (It is assumed that the supersense of the unknown word has been included in the current thesaurus. In other words, no new supersense is created.)

Particularly, given an unknown word w and a thesaurus T, the following three steps are executed:

(1) Candidate Synonym Set Building. A character filtering process and a POS (Part Of Speech) filtering process are completed to acquire some words sharing character and POS with w from T. Those words compose a set, which is called candidate synonym set and denoted by $CS(w)$.
(2) Contextual Similarity Computation. Collect context of w and words in $CS(w)$ from a corpus. Denote context of a word w_1 as $CT(w_1)$, where $w_1=w$ or $w_1 \in CS(w)$. Define contextual similarity of w and $w_i \in CS(w)$ as $sim(w, w_i)=\lambda(w, w_i) * CTS(w, w_i)$, where $CTS(w, w_i)$ denotes the pure contextual similarity, while weighted by $\lambda(w, w_i)$ (structural similarity between a test word and corresponding training word).
(3) KNN Classification. A KNN classifier is employed to assign w a supersense s, which is selected from $\{SS(w_i)\}$. Here, $SS(w_i)$ denotes the supersense of $w_i \in CS(w)$.

In the following, the above three steps are described in three subsections.

3.1 Candidate Synonym Set Building

Most Chinese characters are meaningful, e.g., means person, means organization and means tools. Therefore, a word usually has some synonyms that share character with it. E.g., ji-min ‘stock fund investor’ has a synonym 市 shi-min ‘citizen’. The statistics on some test sets shows that about 98% words obey that assumption (see Table 2), which means that about 98% of words have at least one synonym that shares a character.

According to the above observation, to guess the supersense of an unknown word *w*, we can focus on those known words sharing character with *w* and select a supersense for *w* from the set of supersenses of those words. In this way, the candidate supersense set of *w* is largely shrunk, compared with forming that set by selecting all the supersenses of *T*. Moreover, some noisy data are filtered through this filtering process.

This character-filtering process, one of the key components of structure-based methods, is borrowed to work as the basis for context analysis.

In detail, given an unknown word *w* and a thesaurus *T*, the candidate synonym set *CS(w)* is formed by selecting any word $w_1 \in T$, where w_1 and *w* share at least one character. For example, for *w* = ji-min ‘stock fund investor’, *CS(w)*={ ji-chu ‘basis’, 人 ren-min ‘human’, min-zhu ‘democracy’, dian-ji-zhe ‘founder’…}.

CS(w) can be further filtered by POS. That is, if the POS of *w* is known, then w_1 will be removed from *CS(w)* if w_1 has different POS with *w*. In fact, identifying POS of unknown words is an independent research topic, e.g., [17]. In the experiments of this paper, POS of *w* is assumed known and POS-filtering is employed.

3.2 Contextual Similarity Computation

Three important factors involve in the similarity computation procedure: context extraction and representation, term weighting and similarity measurement. They are described in the following.

3.2.1 Context Extraction and Representation

Two kinds of context extraction strategies are developed: window-based and dependency-based strategies. Window-based strategy takes the words appearing within a certain window size of the target word as its context. [10] examined window size from 6 to 100 (i.e., 3 to 50 on the left and right of the target word respectively) and found that 6 performs the best. [11] also used 6. Therefore, 6 is adopted as the window size in this paper.

Dependency-based strategy takes syntactic dependency information of a target word as its context. A dependency relationship is an asymmetric binary relationship

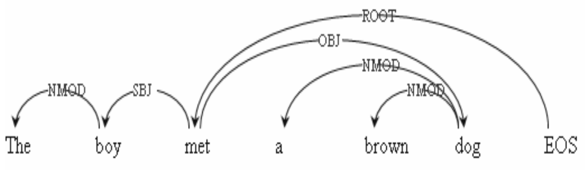


Fig. 1. Example of Dependency Tree

between a word and its modifier [18]. The local context of a word *w* is a triple (*w, r, w'*), where *w'* is a word having *r* relation with *w*. Notice that *w* can be either the target word or modifier in a triple.

A target word may have multiple context triples in one sentence. For example, in sentence "the boy met a brown dog" (Figure 1), the word "dog" has three context triples: (dog, NMOD, a), (dog, NMOD, brown), (dog, OBJ, met). Then, for "dog", three features are acquired: a/NMOD, brown /NMOD, met/OBJ. By parsing a corpus with a syntactic dependency parser, all features of a target word can be collected.

The context of a word is represented by a vector $<v_1, v_2, ..., v_n>$, where *n* is the total number of context words, and v_i is a weighted value.

For *w*= gong-an-ju-zhang 'police chief', four candidate synonyms of *w* are w_1= gong-shi 'minister', w_2= gong-guo 'duchy', w_3= hui-zhang 'chairman' and w_4= te-chang 'one's specialty'. Frequency of some context words is listed in Table 1. For instance, the figures in column ' /VOB' means that, cheng-wei-gong-an-ju-zhang 'become police chief', cheng-wei-gong-shi become minister' and cheng-wei-hui-zhang 'become chairman' occur '1, 2 and 4 times respectively in the corpus, while cheng-wei-gong-guo 'become duchy' and cheng-wei-te-chang 'become one's specialty' occur none in the corpus.

Table 1. Frequency of some context words of *w*, w_1, w_2, w_3 and w_4

	cheng-wei 'become'	wei 'for'	汉 TT wu-han-shi 'Wuhan city'	Q yi-wei 'a'	zai 'at'	Q yi-ge 'a'
w= 安局	1	1	1	1	1	4
w_1=	2	1	0	2	1	0
w_2=	0	0	0	0	2	1
w_3=	4	1	0	10	1	2
w_4=特	0	5	0	0	5	3

3.2.2 Term Weighting

We test various term weighting schemes including BOOL, TFIDF, BMTFIDF, IDF, BMIDF, TTEST and PMI (the formulas of the last six schemes are listed in Figure 2). In those weighting schemes, w is the headword. c is the context word. tf(w,c) is the frequency of c occurring in the context of w. df(c) is the times of c occurring in the context of different headwords, |d| is the number of unique context words in the context of w. avgdl is the average |d| of the context of all headwords, N is the number of headwords in the collection. P(w) is the probability of w occurring in the corpus. P(c) is the probability of c occurring in the corpus. P(w,c) is the probability of w and c co-occurring in the corpus. k1 and b are constants 2.0 and 0.75.

$$W_{TFIDF}(w,c) = tf(w,c)\log(\frac{N}{df(c)}) \tag{1}$$

$$W_{BMTFIDF}(w,c) = \log\frac{N-df(c)+0.5}{df(c)+0.5} \times \frac{(k_1+1)\times tf(w,c)}{k_1\left((1-b)+b\frac{|d|}{avgdl}\right)+tf(w,c)} \tag{2}$$

$$W_{IDF}(c) = \log(\frac{N}{df(c)}) \tag{3}$$

$$W_{BMIDF}(c) = \log\frac{N-df(c)+0.5}{df(c)+0.5} \tag{4}$$

$$W_{TTEST}(w,c) = \frac{P(w,c) - P(w)P(c)}{\sqrt{p(w)P(c)}} \tag{5}$$

$$W_{PMI}(w,c) = \log\frac{P(w,c)}{P(w)P(c)} \tag{6}$$

Fig. 2. Weighting schemes

3.2.3 Similarity Measure

Contextual similarity of w and $w_i \in CS(w)$ is defined as $sim(w, w_i)=\lambda(w, w_i) * CTS(w, w_i)$, where $CTS(w, w_i)$ denotes the pure contextual similarity, while weighted by $\lambda(w, w_i)$ (called λ weighting).

Cosine distance is used to compute $CTS(w, w_i)$. See formula (7) for concrete definition, where n denotes the dimension of the two vectors, while v_j and v_{ij} denote the weighted value of the jth context word of w and w_i respectively.

$$CTS(w,\ w_i) = \frac{\sum_{j=1}^{n} v_j v_{ij}}{\sqrt{\sum_{j=1}^{n} v_j^2}\sqrt{\sum_{j=1}^{n} v_{ij}^2}} \tag{7}$$

The λ weighting is based on structural information. Its basic idea is that: the more common characters two Chinese words share, the closer their meanings are. Therefore, $\lambda(w, w_i)$ is defined as follows.

(1) If w and w_i only share the final or the first character, $\lambda(w, w_i)=1$. E.g., w= ji-min 'stock fund investor', w_i=市 shi-min 'citizen'.

(2) Else, if w and w_i share both the final and the second final character, $\lambda(w, w_i)=\lambda_1$. E.g., w=铝 lv-he-jin 'aluminum alloy', w_i= tie-he-jin 'iron alloy'.

(3) Else, if w and w_i share both the first and the final character, $\lambda(w, w_i)=\lambda_2$. E.g., w= 厂 dian-ji-chang 'electronic appliance factory', w_i= 厂 dian-qi-chang 'electronic motor factory'.

(4) Else, $\lambda(w, w_i)=\lambda_3$.

The three parameters should be set as $\lambda_1 \geq \lambda_2 \geq 1 \geq \lambda_3$.

To optimize all the λs, the following procedure is completed: (1) set the range of λ_1 and λ_2 as [1, 20], and adjust every 1; (3) set the range of λ_3 as [0, 1], and adjust every 0.1.

3.3 KNN Classification

Each word $w_i \in CS(w)$ has a supersense $SS(w_i)$, which is defined in the thesaurus T. Different candidate synonym may have same supersense. A KNN classifier takes the following steps:

(1) Rank $sim(w, w_i)$ from big to small.
(2) Keep $sim(w, w_i)$ unchanged for the first K words and set $sim(w, w_i)=0$ for the remaining.
(3) Collect supersenses of all $w_i \in CS(w)$ to form a supersense set $\{SS(w_i)\}$. For a supersense $s_j \in \{SS(w_i)\}$, denote its similarity sum as $sim(s_j)$, and compute it as $sim(s_j)= \; sim(w, w_i)$, where w_i satisfies $SS(w_i)= s_j$.
(4) Assign w the supersense having the biggest similarity sum, i.e., $Argmax(sim(s_j))$.

4 Experiments

4.1 Evaluation Setup

4.1.1 Data Set

Extended Cilin: (abbreviated as *Cilin* in the following), which is released by Harbin Institute of Technology, is taken as the thesaurus in the experiments [19]. *Cilin* classifies over 70,000 words into 12 major categories, 94 medium categories (for nouns, there are 49 medium categories) and 1428 small categories. Each small category contains several synsets that are close in meaning. The three levels of semantic categories are referred to as major, medium and small supersense in the following.

Two sets were constructed, called TS1 and TS2, which contain 3,000 test words respectively. TS1 is used for development and TS2 are used for test.

TS1 and TS2 were constructed following the procedure in [10], which selects words from *Cilin*, and then filters and groups them with the help of *Contemporary Chinese Corpus* [20]. The *Contemporary Chinese Corpus*, which is segmented and POS-tagged, contains all the news articles (over 1.12 million tokens) of the *People's Daily* newspaper published in China from January to June 1998. To form TS1 (or TS2), filter from *Cilin* those words not occurring in *Contemporary Chinese Corpus* of January 1998, or not POS-tagged as nouns, verbs, or adjectives. Then group those words, which remain in *Cilin* after filtering, by their frequency in *Contemporary*

Chinese Corpus: 1-3 times, 3-6 times, and 7 or more times. Randomly select 1,000 words from the three groups to form TS1 (or TS2).

Context of words is extracted from a self-made corpus, called Raw-Corpus. Raw-Corpus contains about 500,000 news articles (about 20,000,000 sentences), which are collected from four news websites Xinhua (www.xinhuanet. com), Sina (news.sina.com.cn), Sohu (news. sohu.com) and CCTV (news.cctv.com) from January to December 2006.

To extract context of a word, at most 1000 sentences containing that word are retrieved from Raw-Corpus. To reduce noisy data, a retrieved sentence must contain at least 50 Chinese characters. All those retrieved sentences are then segmented and parsed by LTP 2.0 Platform[1], which is developed by HIT Center for Information Retrieval. The LTP 2.0 Platform supports adding new words in functions of word segmentation and dependency parsing.

4.1.2 Baseline, Topline and State-of-Art Methods

For comparison, a baseline method, a topline method and two state-of-art methods are experimented on both the two sets (TS1, TS2).

Baseline Method (Baseline): In this method, when the candidate synonym set is formed, all candidate synonyms vote with equal weight. That is, the KNN classifier works with setting $sim(w, w_i)=1$ and $K=|CS(w)|$. In fact, this method assigns w with the supersense that the most candidate synonyms have.

Topline Method (Topline): In the method, denote the correct supersense of w as s, and the assignment is considered correct if $s \in \{SS(w_i)\}$. That is, as long as one candidate synonym share supersense with w, the topline model is considered correct on w. The topline method only used to help estimate the upper limit of the performance of the proposed method.

State-of-art Context-based Method (SCM): The approach of [8] is the state-of-art one among all those context-based methods. We implemented this approach and called it SCM. Grefenstette's weighted JACCARD measure and TTEST are used to measure contextual similarity between words.

State-of-art Structure-based Method (SSM): The structure-based approach of [10] is the state-of-art one among all those structure-based methods. We also implemented this approach and called it SSM. SSM includes the rule-based model and character-category association model of [10]. Since the third model, i.e., the context-based one, doesn't further improve performance on the basis of the previous two models, it hasn't been adopted in SSM.

4.2 Evaluation Results

4.2.1 Method Comparison

In the experiments, we first compare the proposed approach (with parameters: k=20, λ_1=10, λ_2=10, λ_3=0.4, window-based and PMI weighting, which are adjusted on TS1.)

[1] `http://ir.hit.edu.cn/demo/ltp/`

with the baseline method, topline method and two state-of-art methods. For convenience of comparison with previous studies for English, we also make experiments on classifying all nouns into 49 medium categories. However, except indicating directly, all the following evaluations are made on the results of classifying nouns, verbs and adjectives into 1428 small categories. Note that a polysemous candidate synonym would vote one time for each of its supersense. Moreover, for a polysemous unknown word, the result is considered correct if the assigned supersense matches any of its correct supersense. Table 2 shows the comparison results.

Seen from the table, the proposed approach outperforms the baseline method and the two state-of-art methods. The proposed method achieves improvements about 10% over the SSM, about 20% over the baseline method and 25% over the SCM. The SCM runs even worse than the baseline method. Note that the process of candidate synonym set building is executed in the baseline method but not in the SCM. The SSM outperform the baseline method greatly, with improvement about 10% in terms of F_1-score.

The baseline model achieves above 45% F_1-score on small suspense tagging. The performance is much better than that in English, e.g., [14] achieved 35% when classifying English nouns into 137 classes. The topline method achieves about 98% F_1-score. This suggest that the character-filtering and POS-filtering processes, i.e., the process of candidate synonym set building, are reliable and have little risk of missing correct answers.

4.2.2 Weighting Schemes Comparison

To choose the most suitable weighting scheme, seven variants of the proposed method are implemented. In the seven implementations, TTEST, TFIDF, BOOL, BMTFIDF, IDF, BMIDF and PMI are used as weighting schemes respectively. All other parameters are kept the same as in Section 4.2.1. Table 3 shows the results on TS1.

The results show that PMI outperform best ([10] also show that PMI perform better than TTEST, although the accuracy of Lu's hybrid method using PMI weighting is only about 37%). BMTFIDF, one of the best weighting schemes in standard IR, also performs well in measuring contextual similarity. However, IDF, which usually perform worse than TFIDF in standard IR, outperforms IDF in measuring contextual similarity.

The TTEST implementation of the proposed method performs much better than the SCM, which also uses TTEST as the weighting scheme.

4.2.3 Influences of the Value of *K* in KNN Classifier

Figure 3 shows the F_1-score curve of three variant implementations of the proposed method on TS1 with different values of *K* in the KNN Classifier. *K* varies from 1 to 80. In the three variant implementations, PMI, BMTFIDF and TTEST are used as the weighting scheme respectively and all the other parameters are the same as in Section 4.2.1. From this figure, we can see that the proposed method with PMI weighting achieves the best performance at *K*=20, while BMTFIDF and TTEST weighting at *K*=25.

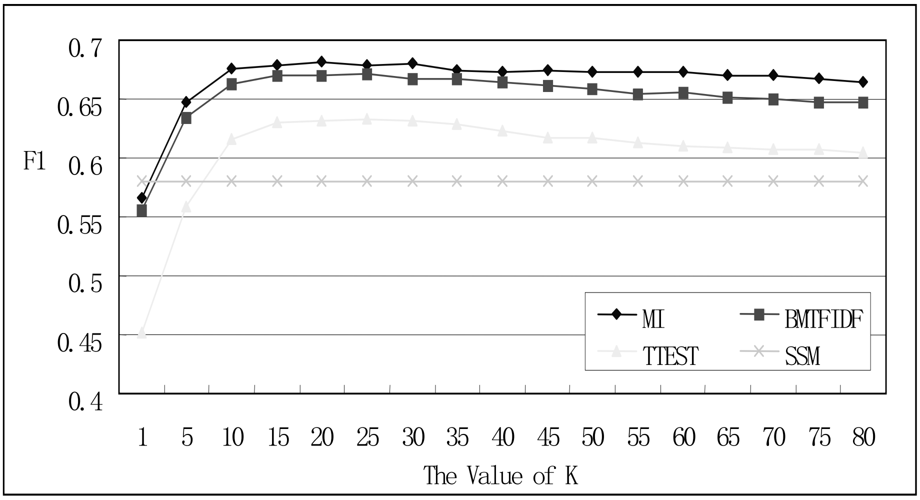


Fig. 3. F_1-score vs. the value of K in the KNN classifier (TS1)

4.2.4 Influences of the Process of Candidate Synonyms Building, the Process of λ Weighting and the Context Extraction Strategy

Besides the value of *K* and weighting schemes, there are several other crucial options in the proposed method, i.e., candidate synonyms set (CSS) building, λ weighting and context extraction strategy. To verify the effectiveness of those options, several implementations of the proposed method are experimented. The first one uses all the best parameters as in Section 4.2.1. All the following implementations are based on the first one. The second one only uses the dependency-based context extraction strategy instead of the window-based one. The third one only doesn't include the process of candidate synonyms set building. The fourth one only sets all the three λs as 1. The last one not only doesn't include the process of candidate synonyms set building, but also sets all the three λs as 1. The results of the five implementations are listed in Table 4.

Table 2. Comparison results

Method	Data Set	F(Small Supersense)	F(Medium Supersense)
Baseline	TS1	0.482	0.559
	TS2	0.458	0.562
Topline	TS1	0.981	0.995
	TS2	0.980	0.997
SCM	TS1	0.431	0.573
	TS2	0.425	0.557
SSM (Lu)	TS1	0.580	0.713
	TS2	0.561	0.692
Proposed Method	TS1	**0.681**	**0.802**
	TS2	**0.671**	**0.767**

Table 3. Comparison results of weighting schemes on TS1[2]

Weighting Scheme	P	R	F
BOOL	0.626	0.625	0.625
TFIDF	0.630	0.629	0.629
TTEST	0.637	0.627	0.632
BMIDF	0.664	0.663	0.663
IDF	0.666	0.665	0.665
BMTFIDF	0.671	0.670	0.670
PMI	**0.682**	**0.680**	**0.681**

Table 4. Comparison results of the using of candidate synonyms set building, context extraction strategy and λ weighting on TS1

Parameter	P	R	F
Best parameter	**0.682**	**0.680**	**0.681**
Without character-filtering and POS-filtering	0.633	0.633	0.633
$\lambda_1=1, \lambda_2=1, \lambda_3=1$	0.664	0.663	0.663
Without character-filtering and POS-filtering, $\lambda_1=1, \lambda_2=1, \lambda_3=1$	**0.461**	**0.461**	**0.461**
Dependency-based Context Extraction Strategy	0.670	0.668	0.669

From this table, we may find that both the process of candidate synonym set building and λ weighting are very crucial. If both of the two processes are not contained in the proposed method, the performance decreases greatly (46.1% in terms of F_1-score) and is very close to that of the Curran's SCM method. The results not only show the effectiveness of the combination strategy in the proposed method, and also show that context-based method alone doesn't perform as well as structure-based method (e.g. Lu's SSM method). However, since both the previous two processes are based on the same idea, only missing one of them doesn't lead to great decrease. As for context extraction strategy, the dependency-based one performs a little worse than the window-based one.

5 Error Analysis

There are four types of errors. First, for an unknown word *w*, there is no word in thesaurus *T* that shares character with *w*, i.e., $CS(w)=\emptyset$. In such a case, no supersense is assigned to *w*. TS1 and TS2 only have 4 and 2 such types of words respectively.

Second, although some words do share characters with *w*, all of them do not share supersenses with *w*. There are 83 and 80 such types of words in the two sets respectively.

[2] In SCM, *K*=10, which is the best parameter for SCM.

Third, *w* does not appear in the Raw-Corpus, i.e., $CT(w_i)=<0, ..., 0>$. Thus, $sim(w, w_i)=0$ holds for all $w_i \in CS(w)$. In this case, the proposed method degrades to the baseline model. There are 50 and 32 such types of words in the two sets respectively. Since contextual information is absent, many of them would produce wrong results.

Fourth, although one or more candidate synonyms share supersense with *w*, the proportion of those words is too small to help that supersense win among all the candidate supersenses. Most of the errors are of this type.

6 Conclusions and Future Work

This paper proposes a tightly-coupled approach to combine contextual and structural information for supersense tagging of Chinese unknown words. The experiment results show the effectiveness of the proposed approach. Although context-based methods used alone perform not as well as structure-based methods, the combination of contextual and structural information performs much better than using them alone.

There are several avenues that might be taken for future work. First, since PMI is usually used for feature selection and TFIDF used for term weighting, adding a feature selection process into the proposed method might make some more improvements. Second, our experiment results show that IDF weighting performs better than TFIDF weighting, which is very different from the results in standard IR and sentiment classification. We would try to explain this difference in future work. Finally, although this paper only experimented on Chinese, the idea might also be applicable to other Asian languages such as Japanese.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments and suggestions. This work was supported by the Open Project Program of the National Laboratory of Pattern Recognition (NLPR), National Natural Science Foundation of China (No. 60703063), National Social Science Foundation of China (No. 08CYY016) and the Open Projects Program (KLCL-1003) of Key Laboratory of Computational Linguistics(Peking University), Ministry of Education.

References

1. Fellbaum, C. (ed.): WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
2. Clark, S., Weir, D.: Class-based probability estimation using a semantic hierarchy. Computational Linguistics 28(2), 187–206 (2002)
3. Ponzetto, S.P., Strube, M.: Exploiting Semantic Role Labeling, WordNet and Wikipedia for Coreference Resolution. In: Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pp. 192–199 (2006)
4. Herrera, J., Peñas, A., Verdejo, F.: Textual Entailment Recognition Based on Dependency Analysis and WordNet Machine Learning Challenges, pp. 231–239. Springer, Heidelberg (2006)

5. Esuli, A., Sebastiani, F.: PageRanking WordNet Synsets: An Application to Opinion Mining. In: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pp. 424–431 (2007)
6. Ciaramita, M., Johnson, M.: Supersense Tagging of Unknown Nouns in WordNet. In: Proceedings of the 2003 Conference on Empirical Methods on Natural Language Processing, pp. 168–175 (2003)
7. Chen, K., Chen, C.: Automatic semantic classification for Chinese unknown compound nouns. In: Proceedings of the 18th International Conference on Computational Linguistics, pp. 173–179 (2000)
8. Curran, J.R.: Supersense Tagging of Unknown Nouns using Semantic Similarity. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics, pp. 26–33 (2005)
9. Tseng, H.: Semantic classification of Chinese unknown words. In: Proceedings of ACL-2003 Student Research Workshop, pp. 72–79 (2003)
10. Lu, X.: Hybrid Models for Semantic Classification of Chinese Unknown Words. In: Proceedings of North American Chapter of the Association for Computational Linguistics - Human Language Technologies 2007 Conference, pp. 188–195 (2007)
11. Chen, H., Lin, C.: Sense-tagging Chinese Corpus. In: Proceedings of the 2nd Chinese Language Processing Workshop, pp. 7–14 (2000)
12. Chen, C.: Character-sense association and compounding template similarity: Automatic semantic classification of Chinese compounds. In: Proceedings of the 3rd SIGHAN Workshop on Chinese Language Processing, pp. 33–40 (2004)
13. Widdows, D.: Unsupervised methods for developing taxonomies by combining syntactic and statistical information. In: Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, Edmonton, Alberta Canada, pp. 276–283 (2003)
14. Pekar, V., Staab, S.: Word classification based on combined measures of distributional and semantic similarity. In: Proceedings of 10th Conference of the European Chapter of the Association for Computational Linguistics, pp. 147–150 (2003)
15. Salton, G., Buckley, C.: Term weighting approaches in automatic text retrieval. Information Processing and Management 24(5), 513–523 (1988)
16. Kim, J., Li, J., Lee, J.: Discovering the Discriminative Views: Measuring Term Weights for Sentiment Analysis. In: Proceedings of the 47th Annual Meeting of the Association of Computational Linguistics, pp. 253–261 (2009)
17. Qiu, L., Hu, C., Zhao, K.: A method for automatic POS guessing of Chinese unknown words. In: Proceedings of the 22nd International Conference on Computational Linguistics, pp. 705–712 (2008)
18. Hudson, R.: Word Grammar. Basil Blackwell Publishers Limited., Oxford (1984)
19. Mei, J., Zhu, Y., Gao, Y., Yin, H. (eds.): Tongyici Cilin. Commercial Press, Hong Kong (1984)
20. Yu, S., Duan, H., Zhu, X., Swen, B.: The basic processing of Contemporary Chinese Corpus at Peking University. Journal of Chinese Information Processing 16(5), 49–64 (2002)

Identification of Conjunct Verbs in Hindi and Its Effect on Parsing Accuracy

Rafiya Begum, Karan Jindal, Ashish Jain,
Samar Husain, and Dipti Misra Sharma

Language Technologies Research Centre, IIIT-Hyderabad, India
rafiyabegum@gmail.com,
{karan_jindal,ashishjain}@students.iiit.ac.in,
{samar,dipti}@mail.iiit.ac.in

Abstract. This paper introduces a work on identification of conjunct verbs in Hindi. The paper will first focus on investigating which noun-verb combination makes a conjunct verb in Hindi using a set of linguistic diagnostics. We will then see which of these diagnostics can be used as features in a MaxEnt based automatic identification tool. Finally we will use this tool to incorporate certain features in a graph based dependency parser and show an improvement over previous best Hindi parsing accuracy.

Keywords: Conjunct verbs, Diagnostics, Automatic Identification, Parsing, Light verb.

1 Introduction

There are certain verbs that need other words in the sentence to represent an activity or a state of being. Such verbs along with the other words, required for completion of meaning, are together called *Complex Predicates* (CP). CP exist in great numbers in South Asian languages [1], [2], [3]. A CP is generally made via the combination of nouns, adjectives and verbs with other verbs. The verb in the CP is referred as light verb and the element that the light verb combines to form a CP is referred as host [4].

[5] says that in Hindi/Urdu, the light verb is taken as a contributing 'semantic structure' which determines syntactic information such as case marking whereas *host* contributes the 'semantic substance', i.e. most of the meaning the complex predicate has. [6] has talked about four types of complex predicates: (a) In Syntactic Complex Predicates the formation takes place in the syntax. (b) In Morphological Complex Predicates, a piece of morphology is used to modify the primary event predication. Well known example is morphological causatives. (c) Light Verbs cross linguistically do not always form a uniform syntactic category. They are not always associated with a uniform semantics, but they always muck around with the primary event predication. (d) In Semantics, complex predicates represent the decomposition of event structure.

In CPs, 'Noun/Adjective+Verb' combinations are called conjunct verbs and 'Verb+Verb' combinations are called compound verbs. In this paper, we are focusing on conjunct verbs in Hindi and their identification using set of diagnostics and then we will see which of these diagnostics can be used to automate the identification process using statistical techniques and showed their usefulness in data driven dependency parsing [41]. This work can also greatly help in automatically augmenting a lexical network such as the Hindi WordNet[1]. Previous automatic identification approaches made use of parallel English corpora [19], [20] which makes use of the property that single verb in English will break into two components i.e. noun/adjective and verb in Hindi. [21] also makes use of English corpus for extracting collocation based features. To the best of our knowledge ours is the first work towards automatic identification of conjunct verbs in Hindi using only Hindi corpus. We have achieved a maximum accuracy of 85.28%. Incorporating this as a feature in graph based dependency parsing shows an improvement of 0.39% in label and 0.28% in label attachment accuracy.

The paper is arranged as follows: Section 2 gives overview of conjunct verbs in Hindi. In Section 3, we describe behavioral Diagnostics to Identify Complex Predicates. In Section 4, we discuss the subjective evaluation of diagnostics. In Section 5 and 6, we define the system for automatic identification of conjunct verb and discuss experimental results respectively. We evaluate the effect of conjunct verb on parsing accuracy and compare it with the current state-of-the-art parser in Section 7. We conclude the paper is Section 8.

2 Conjunct Verbs in Hindi

Conjunct verb in Hindi is formed by combining a noun or an adjective with a verb. These verbs have the following structure [7]:

Noun/Adjective + Verb (Verbalizer)

The most frequent *verbalizers* in Hindi are *karnaa* 'to do', *honaa* 'to be', *denaa* 'to give', *lenaa* 'to take', *aanaa* 'to come'. Take (1) as a case in point.

(1) *raam ne* *siitaa ko* ***kshmaa*** ***kiyaa***
ram Erg. sita Acc. forgiveness do-Past
'Ram forgave Sita.'

(2) *raam ne* *shyaam kii* ***madad kii***
ram Erg. shyam Gen. help do-Past
'Ram helped Shyam.'

In example (1), *kshmaa* 'forgiveness' is a noun which is combined with the verb *karnaa* 'to do' to express the sense of the verb 'to forgive'. In example (2), conjunct verb is *madad karnaa* 'to help' and the noun *madad* 'help' is linked with the object *shyaam* 'Shyam' by the postposition (Hindi case marker) *kii* 'of'.

There are two approaches [8] which define conjunct verbs: 'Lexical approach' and 'Semanticist approach'. The aim of Lexical approach is to offer either a formal or

[1] Developed by the wordnet team at IIT Bombay, `http://www.cfilt.iitb.ac.in/webhwn`

structural justification for the recognition of the category of conjunct verbs and to specify and delimit the noun or adjective plus verb sequence as conjunct verbs. [12], [13], [14], [15] have followed Lexical approach. In 'Semanticist Approach', they tried to explore the semantic structure of the language completely abandoning the lexicalist interpretationists's goal of specifying and delimiting the noun or adjective plus verb sequence as conjunct verbs. [16], [17] have followed semanticist approach.

In this paper we also discuss the syntactic analysis of the conjunct verbs i.e., how they are treated at the syntactic level annotation of the data.

3 Diagnostics to Identify Complex Predicates

The following are some of the diagnostics mentioned in the literature [4], [18] for deciding which Noun+Verb (N+V) combinations are conjunct verbs:

I. Coordination Test (D1): This test shows that nouns of conjunct verb don't allow coordination. However it is possible to conjoin the entire N+V combination.

(3)*l*og pratiyogita meN* ***rucii*** *aur* ***bhaag le*** *rahe the*
People competition in interest and participation take Prog be-Past
'People were taking interest and participation in the competition.'

(4) *log pratiyogita meN* ***rucii le*** *rahe the aur* ***bhaag le*** *rahe the*
People competition in interest take Prog was and participation take Prog was
'People were taking interest and participation in the competition.'

Example (3) is ungrammatical because *rucii* 'interest' and *bhaag* 'participation' are conjoined by *aur* 'and', whereas these nouns are part of CP. Sentence (4) is grammatical because here the N+V combination i.e., *rucii le* 'take interest' and *bhaag le* 'participate' has been conjoined with *aur* 'and'.

II. Constituent Response Test (Wh-Questions) (D2): CP internal nouns can't be questioned. Only N+V combination can be questioned.

(5) *raam ne* ***jamhaaii lii***
ram Erg yawn take-Past
'Ram yawned.'

(6)**raam ne kya lii?*
ram Erg what take-Past
'What did Ram take?'

(7) *raam ne kya kiya?*
raam Erg what do-Past
'What did Ram do?'

Example (6) is ungrammatical because only noun of CP i.e., *jamhaaii* 'yawn' given in example (5) has been questioned. Whereas in (7), the N+V combination, *jamhaaii le* 'take yawn' has been questioned.

III. Relativization (D3): CP internal nominals cannot be relativized.

(8)**vah* **snaan** *[jo* **bahut pavitra hai]** *raam ne gangaa taT par* ***kiyaa***
that baath which lot pure is ram Erg ganga bank on do-Past
'The bath which Ram did on the bank of river Ganga is very pure.'

Sentence (8) is ungrammatical because *snaan* 'bath' which is noun internal to CP has been relativized by the relative clause.

IV. Adding the accusative case marker (D4): CP internal nominal will not allow the accusative marking.

(9)*r*aam ne us* ***jamhaaii ko liyaa ...***
ram Erg that yawn Acc take-Past ...
'Ram took that yawn…..'

Sentence (9) is ungrammatical because *jamhaaii* 'yawn' which is noun internal to CP has taken an accusative case marker.

V. Adding the Demonstrative Pronoun (D5): CP internal nominal will not take Demonstrative Pronoun.

(10) *raam ne yah nirdesh diyaa*
ram Erg. this order give-Past
'Ram gave this order.'

In sentence (10), the demonstrative pronoun *yah* 'this' is modifying the N+V combination i.e., *nirdesh diyaa* 'gave order' and not just the Noun, *nirdesh* 'order'. To justify the above diagnostics we did a survey of these tests among native speakers of Hindi Language.

4 Diagnostics Evaluation

We conducted a survey among 20 native language speakers of Hindi to ascertain the usefulness of the diagnostics described in the previous section in identification of conjunct verb (CV). We took conjunct verbs and applied the above diagnostics to see how they fare in a subjective evaluation. Table 1 below shows the results of the test. '+ve'/'-ve' reflect the usefulness of diagnostics D1-D5 for each verb. A diagnostic is deemed '+ve' if it got the desired response from >50% of the subject. A noun/adjective-verb pair is accepted as a conjunct verb (indicated by 'yes') if >=3 diagnostics are '+ve', it is not accepted as a conjunct verb (indicated by 'no') if all the diagnostics are '-ve'. The decision is 'unsure' (indicated by 'maybe') if >=3 diagnostics are '-ve'. If a diagnostic is not applicable for a verb we use a hyphen ('-') to indicate this. The cells that show +ve/-ve indicate no majority in total number of responses. For ease of exposition, Table 1 shows the result only for only 7 verbs. The study considers a total of 20 verbs.

Table 1. Results of the subjective evaluation

Noun+Verb	**D1**	**D2**	**D3**	**D4**	**D5**	**CV**
rucii le ‘take interest’	+ve	+ve	+ve/-ve	+ve	+ve	Yes
maar khaa ‘get beaten’	+ve	+ve	-ve	+ve	+ve	Yes
bhaag le ‘participate’	+ve	+ve	+ve	+ve	-ve	Yes
snaan kar ‘bathe’	-ve	+ve	+ve	+ve	+ve	Yes
chalaang maar ‘jump’	-	+ve	+ve	+ve	+ve	Yes
bhojan kar ‘eat’	-ve	-	-ve	+ve	-ve	may be
havaa khaa ‘feel air’	+ve	+ve	+ve	+ve	+ve	Yes

After exploring the behavioral diagnostics to identify conjunct verb, we will now move on to automate this task of identification. The tool will try to use the diagnostics that can be incorporated.

5 Automatic Identification of Conjunct Verb

In the previous sections, various tests were explored for manual identification of conjunct verbs. Now, we will explain the methodology used for building a statistical tool for automatic identification of conjunct verb. We didn’t focus on compound verbs (verb + verb) because already a high accuracy of 98% has been reported [22]. We have learned a binary classification using maximum entropy model, which will classify a noun/adjective-verb pair into either conjunct verb or literal class (non-conjunct verb).

5.1 Corpus

We have used two different dataset that are part of Hyderabad Dependency Treebank annotated according to CPG framework [9].

1. Dataset-1: Has 4500 manually annotated sentences (200k words approx.). It was released as part of the ICON’10 tools contest on Indian Language Parsing [39]. This dataset was used as a training data.

2. Dataset-2: Has 1800 sentences. It was released as part of the ICON'09 tools contest on Indian Language Parsing [40]. This dataset was used as a testing data.

Training data has around 3749 unique consecutive noun/adjective-verb pairs out of which 1987 are unique noun/adjective and 350 unique verbs. Semantic category of each object is mined from the Hindi WordNet. The language model consisting of trigrams of words is created for training data, which is later used for extraction of various features. Testing data has 3613 noun/adjective-verb pairs out which 998 are conjunct verbs and remaining are literal expressions.

5.2 Features

Each of noun/adjective-verb pair is represented as a vector of following feature set. The features are categorized into three categories (1) Lexical (word based features like f1, f2, f3), (2) Binary features (f4, f5), (3) Collocation based (f6, f7). These features will helps in classifying a noun/adjective-verb pair into literal or conjunct verb class.

a. Verb (f1): Some verbs govern whether an object-verb pair is conjunct verb or not as compared to other verbs. They are more likely to occur as light verbs. Example of such a verb is '*kar*' (to do) which accounts for large part of conjunct verb expressions. On the other hand verbs like '*chalnA*' (to walk) occur as literal expression in most cases. Hence, verb will be a good feature for classification task.

b. Object (Noun, Adjective) Lexical (f2): Some objects are more biased towards occurring with a light verb as compared to other objects. These objects have high chances of forming conjunct verb expression with a light verb as compared to other objects.

c. Semantic Category of Object (f3): In some of the theoretical work [5], [6] importance of semantic category of a noun/adjective in identifying conjunct verb has been shown. We incorporated this feature for nouns/adjectives by extracting it from the Hindi WordNet. We referred to the first sense of topmost ontological node of a noun/adjective. Some of the possible semantic categories are 'Artifact', 'Abstraction', 'State', 'Physical Object' etc. Total semantic categories are 83; noun/adjective will fall into any of these categories, so this will help in case of unknown nouns/adjectives.

For Example: in the expression '*viSvAsaGAwa-karana*' (meaning 'to betray'), the Semantic type of '*viSvAsaGAwa*' is "Anti Social".

d. Post-Position Indicator (f4): is a Boolean feature which will indicate whether a noun/adjective is followed by a post position and then verb i.e. a post-position marker is present between noun/adjective and verb or not. Basic intuition behind this feature is that if a noun/adjective is followed by a post position than it's a possible candidate of being a part of verb argument structure. Hence, possibly the particular noun/adjective-verb pair doesn't belong to conjunct verb class, as mentioned in diagnostic number 4 (D4) in section 3.

e. Demonstrative Indicator (f5): is a Boolean feature indicating presence of DEM (demonstrative tag) before noun/adjective-verb pair. This diagnostic is explained in section 3 as D5.

f. Frequency of Verbs corresponding to particular Object (f6): If a noun/adjective is occurring with few verbs than it is highly probable that the given noun/adjective-verb pair is a multi-word expression.So the frequency of the number of different verbs occurring with a particular object will be a good indicator for conjunct verbs. For example: a noun '*svIkAra*' (to accept) occurs only with two different verbs –'*kar*' (to do) and '*hE*' and noun '*kAnUna*' (law) occurs with five different types of verbs – '*bawA*' (to tell), '*kar*', '*baxala*' (to change), '*lA*' (to bring) and '*paDa*' (to study). Therefore, '*svIkAra*' is more likely to form a conjunct verb expression.

g. Verb Argument Indicator (f7): This feature computes the average number of post-position occurring before a unique noun/adjective-verb pair. The reason for exploring this feature is that if an expression has large number of post position occurring before it then its verb's argument structure is likely to be satisfied because each post-position is preceded by a noun/adjective which may potentially be the argument of the verb. Hence this noun/adjective-verb pair is more probable to form a conjunct verb.

5.3 Maximum Entropy

The features extracted above are used for binary classification of a noun/adjective-verb expression into conjunct verb and non-conjunct verb using the maximum entropy model [23]. Maximum entropy has already been widely used for a variety of natural language tasks, including language modelling [24], [25], text segmentation [26], part-of-speech tagging [28], and prepositional phrase attachment [27].

The maximum entropy model estimates probabilities based on the principle that the model is consistent with the constraint imposed maintaining uniformity otherwise. The constraints are derived from training process which expresses a relationship between the binary features and the outcome [29] [30]. Some of the features on which training is performed are distinct valued features (f1, f2) while others are real valued feature (f6, f7). These features are mapped to binary features. We used maximum entropy toolkit[2] to conduct our experiments.

6 Experiments and Results

The trained system on the corpus of 4500 sentences is tested on 1800 sentences for measuring its accuracy. The binary classification of noun/adjective-verb test expressions into conjunct verbs and non-conjunct verbs are done. We took different set of features for our experiments by trial and error method to come up with the best model. The best model gives us the highest accuracy of around 85.28%. For the baseline for our task we included Verb (f1) and Object (f2) as feature. Table 2 gives the overview of useful features which helped in improving the accuracy.

[2] `http://homepages.inf.ed.ac.uk/s0450736/maxent toolkit.html`

Table 2 shows that when the semantic feature (f3) was introduced, it lead to an improvement of around '0.75%', which proves the relevance of this feature. Inclusion of both Boolean features f4 and f5 showed a large jump in accuracy of about '3.15%'. Recall that f3 and f4 corresponds to D4 and D5 in section 3. Addition of feature f6 improved our system by '0.54%' showing dominance of particular objects (as discuss during f6 definition) in conjunct verbs. We have not considered features which will show the steep decrease in accuracy, e.g. feature f7 on addition shows a decrease of '7.78%' with respect to the best accuracy reached so far, and moreover it is even less than the baseline also. We define features (f1+f2+f3+f3+f5) and (f1+f2+f3+f3+f5+f6) as System-1 and System-2 respectively.

Table 2. Showing system accuracy with different feature set

Feature set	Accuracy
f1 + f2	(81.59)
f1+f2+f3	(82.34)
f1+f2+f3+f4+f5	(84.74)
f1+f2+f3+f4+f5+f6	(85.28)
f1+f2+f3+f4+f5+f7	(77.44)

7 Effect of Conjunct Verb on Parsing Accuracy

It had been observed that Dependency framework is the better way to analyze morphological rich free word-order languages (MoRFWO) (such as Czech,Turkish, Hindi, etc). Various data driven [32], [33], [34] and hybrid approaches [31] has been tried but still the current state-of-the-art parsing accuracy hasn't reached to a level which is comparable to English. Complex linguistics phenomenon is considered as a most vital factor for low accuracy of Hindi parsing apart from long distance dependencies, non-projective sentences and less corpus size. In past various morphological [34], semantic [35] and clause boundary [32] features have been tried to give language specific features in data driven parsing. All these features help in increasing the overall Hindi dependency parsing accuracy, but the gap between labeled and unlabeled accuracy is still large. Previous works [34], [43] have pointed that error due to complex predicates are significant in Hindi dependency parsing. Recall that in a conjunct verb it is the noun/adjective-verb complex that forms the predicate thereby controlling the argument structure. This means that unlike a sentence with a normal verb the predicate information in a sentence with conjunct verb is distributed.

In this section, we investigate the effect of using conjunct verb specific features on parser accuracy. MST [36], [37] Parser was used to parse sentence, the MaxEnt based tool described in section 5.3 provides the feature. An improvement of 0.39% in label and 0.28% in label attachment accuracy is achieved.

7.1 Experiments and Results

We considered the MST+MaxEnt setting mentioned in [38] as Baseline for our experiments. All the parsing related experiments are performed on Dataset-2 as described in section 5.1. Using the output of System-1 and System-2 as described in Section-6, we added conjunct verb feature in each consecutive noun/adjective-verb pair in the dataset. Feature is added in the feature column of CONLL [42] format by giving an extra indicator like 'pof' (for conjunct verb) and 'npof' (for non-conjunct verb), which led to an increase in parsing accuracy using MST. Total number of noun/adjective-verb pairs is 3613 out of which 962 and 942 are marked as 'pof' and remaining as 'npof' by System-1 and System-2 respectively. The parsing result is shown in Table 3.

Table 3. Average LA (Labeled Attachment), UA (Unlabeled Attachment) and L (Label) accuracies on 12-fold cross validation

	LA (%)	UA (%)	L (%)
Baseline	68.77	85.68	71.90
System 1	**69.05**	**85.68**	**72.29**
System 2	68.52	85.04	71.93

Table 4. 2nd and 3rd column represents the number of correctly identified 'pof' and 'npof' labels. Baseline-1 and Baseline-2 gives the number of labels that are correctly identified by the Baseline System group into 'pof' and 'npof' labels in comparison to System-1 and System-2 respectively. These stats are the summation of 12 testing set which are tested during 12-fold cross validation.

	'pof' labels	'npof' labels
Baseline-1	715	1628
System-1	715+**36**	1628+**21**
Baseline-2	713	1630
System-2	713+**42**	1630+**15**

7.2 Observations

System-1 shows an increase of 0.39% in label and 0.28% in label attachment accuracy, this increase accounts to the 0.3%, 1.87%, 2.94% and 0.43% increase in labels accuracy of 'k1', 'k2', 'pof', 'k7p'[3] respectively. These labels occur in the same environment as 'pof', hence the confusion. Both the System-1 and System-2 helps in reducing the 'npof' label (like 'k1', 'k2', 'k7p' etc.) confusion for those chunks which are given conjunct verb feature, by correctly identifying 21 and 15 more labels compare to baseline respectively as shown in Table 4. Similarly, number of correctly identified conjunct verb labels increase by 36 and 42 in System-1 and System-2 respectively. This increase shows the positive effect of giving label specific feature to noun/adjective-verb pairs. Even if there is an increase in both systems

[3] k1, k2 can be roughly translated as agent and theme respectively. 'pof' is the relation between noun/adjective-verb in a conjunct verb, 'k7p' shows place relation. The dependency labels in the Treebank are syntactico-semantic in nature. For more details refer [10].

output, the overall accuracy of System-2 is less compare to both System-1 and Baseline results. This decrease is because of indirect wrong learning leading to ambiguity between different labels.

8 Conclusions and Future Work

We have analyzed some of the diagnostics for manual identification of conjunct verb and there relevance in automatic identification. We successfully showed the importance of these diagnostics in statistical techniques by observing the significant increase in overall accuracy of identifying conjunct verbs and there positive effect on parsing accuracy. In future we will try to automate behavioral diagnostics (like D1 and D3) on the availability of large corpus. Although some diagnostics like Constituent Response Test (Wh-Questions) cannot be automated, they can give some theoretical grounding to conjunct verb identification and can complement the statistical tool.

We tried to include some context through feature like f7, but they didn't help. Since, additional context proves helpful in many tasks; we will have to explore this feature. The parsing accuracy showed improvement by incorporating the features given by our tool. Other NLP application tasks such as Machine Translation can also be tried.

References

1. Hook, P.E.: The Hindi compound verb: What it is and what it does? In: Singh, K.S. (ed.) Readings in Hindi-Urdu Linguistics. National Publishing House, Delhi (1974)
2. Mohanan, T.: Wordhood and Lexicality. NLLT 13, 75–134 (1995)
3. Alsina, A.: Complex Predicates. CSLI Publications, Stanford (1995)
4. Mohanan, T.: Arguments in Hindi. Center for the Study of Language and Information, Leland Stanford Junior University, United States (1994)
5. Butt, M.: Conscious Choice And Some Light Verbs In Urdu. In: Verma, M.K. (ed.) Manohar (1993)
6. Butt, M.: Complex Predicates Compendium, Tromso (May 2005)
7. Agnihotri, R.K.: Hindi, An Essential Grammar, pp. 121–126. Routledge, London (2007)
8. Bahl, K.C.: Studies in the Semantic Structure of Hindi. Motilal Banarasidass, Bihar (1974)
9. Bharati, A., Chaitanya, V., Sangal, R.: Natural Language Processing: A Paninian Perspective, pp. 65–106. Prentice Hall of India, New Delhi (1995)
10. Begum, R., Husain, S., Dhwaj, A., Sharma, D.M., Bai, L., Sangal, R.: Dependency Annotation Scheme for Indian Languages. In: The Third International Joint Conference on Natural Language Processing (IJCNLP), Hyderabad, India (2008)
11. Bhatt, R., Narasimhan, B., Palmer, M., Rambow, O., Sharma, D.M., Xia, F.: A Multi Representational and MultiLayered Treebank for Hindi/Urdu. In: The Third Linguistic Annotation Workshop (The LAW III) in Conjunction with ACL/IJCNLP, Singapore (2009)
12. Kellogg, S.H.R.: A Grammar of the Hindi Language. Routledge and Kegan Paul Ltd., London (1875)
13. Bailey, T.G.: Teach Yourself Urdu. In: Firth, J.R., Harley, A.H. (eds.), London (1956)
14. Guru, K.P.: Hindi Vyakarana. Kasi Nagari Pracarini Sabha, Kasi (1922)

15. Sharma, A.: A Basic Grammar of Modern Hindi. Government of India, Ministry of Education and Scientific Research, Delhi (1958)
16. Bahri, H.: Hindi Semantics. The Bharati Press, Allahabad (1959)
17. Bahl, K.C.: A Reference Grammar of Hindi. The University of Chicago, Chicago (1967)
18. Bhattacharyya, P., Chakrabarti, D., Sarma, V.M.: Complex predicates in Indian languages and wordnets. Language Resources and Evaluation 40(3-4), 331–355 (2006)
19. Mukerjee, A., Soni, A., Raina, A.M.: Detecting Complex Predicates in using POS Projection across parallel Corpora Aligned Hindi sentence. In: Workshop on Multiword Expressions, Sydney, pp. 11–18 (2006)
20. Sinha, R.M.K.: Mining Complex Predicates In Hindi Using A Parallel Hindi-English Corpus. In: ACL International Joint Conference in Natural Language Processing, p. 40 (2009)
21. Venkatapathy, S., Joshi, A.: Relative Compositionality of Noun+Verb Multi-word Expressions in Hindi. In: ICON (2005)
22. Chakrabarti, D., Mandalia, H., Priya, R., Sarma, V., Bhattacharyya, P.: Hindi Compound Verbs and their Automatic Extraction. In: International Conference on Computational Linguistics, pp. 27–30 (2008)
23. Ratnaparkhi, A.: Maximum Entropy Models for Natural Language Ambiguity Resolution. Ph.D. thesis (1998)
24. Rosenfeld, R.: Adaptive Statistical Language Modeling: A Maximum Entropy Approach. Ph.D. thesis, Carnegie Mellon University (1994)
25. Chen, S.F., Rosenfeld, R.: A Gaussian prior for smoothing maximum entropy models. Technical Report CMUCS-99-108, Carnegie Mellon University (1999)
26. Beeferman, D., Berger, A., Lafferty, J.: Statistical models for text segmentation. Machine Learning 34(1-3), 177–210 (1999)
27. Ratnaparkhi, A., Reynar, J., Roukos, S.: A maximum entropy model for prepositional phrase attachment. In: ARPA Human Language Technology Workshop, pp. 250–255 (1994)
28. Ratnaparkhi, A.: A maximum entropy model for part-of-speech tagging. In: Empirical Methods in Natural Language Conference (1996)
29. Nigam, K., Lafferty, J., McCallum, A.: Using Maximum Entropy for Text Classification. In: IJCAI 1999 Workshop on Machine Learning for Information Filtering, pp. 61–67 (1999)
30. Berger, A., Pietra, D.: A Maximum Entropy Approach to Natural Language Processing. Computational Linguistics, Vol 22, 39–71 (1996)
31. Bharati, A., Husain, S., Vijay, M., Deepak, K., Sharma, D.M., Sangal, R.: Constraint Based Hybrid Approach to Parsing Indian Languages. In: The 23rd Pacific Asia Conference on Language, Information and Computation (PACLIC 23), Hong Kong (2009)
32. Gadde, P., Jindal, K., Husain, S., Sharma, D.M., Sangal, R.: Improving Data Driven Dependency Parsing using Clausal Information. In: NAACL-HLT 2010, Los Angeles, CA (2010)
33. Nivre, J., Hall, J., Kubler, S., McDonald, R., Nilsson, J., Riedel, S., Yuret, D.: The CoNLL 2007 Shared Task on Dependency Parsing. In: CoNLL Shared Task Session of EMNLP-CoNLL (2007)
34. Ambati, B.R., Husain, S., Nivre, J., Sangal, R.: On the Role of Morphosyntactic Features in Hindi Dependency Parsing. In: NAACL-HLT 2010 Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2010), Los Angeles, CA (2010)
35. Ambati, B.R., Gade, P., GSK, C.: Effect of Minimal Semantics on Dependency Parsing. In: RANLP 2009 Student Research Workshop (2009)

36. McDonald, R., Pereira, F., Ribarov, K., Hajic, J.: Non-projective dependency parsing using spanning tree algorithms. In: HLT/EMNLP, pp. 523–530 (2005)
37. McDonald, R., Crammer, K., Pereira, F.: On-line large-margin training of dependency parsers. In: ACL 2005, pp. 91–98 (2005)
38. Ambati, B.R., Gadde, P., Jindal, K.: Experiments in Indian Language Dependency Parsing. In: ICON 2009 NLP Tools Contest: Indian Language Dependency Parsing, pp. 32–37 (2009)
39. Husain, S., Mannem, P., Ambati, B., Gadde, P.: The ICON-2010 tools contest on Indian language dependency parsing. In: ICON 2010 Tools Contest on Indian Language Dependency Parsing, Kharagpur, India (2010)
40. Husain, S.: Dependency Parsers for Indian Languages. In: ICON 2009 NLP Tools Contest: Indian Language Dependency Parsing, Hyderabad, India (2009)
41. Bharati, A., Husain, S., Ambati, B., Jain, S., Sharma, D., Sangal, R.: Two Semantic features make all the difference in Parsing accuracy. In: International Conference on Natural Language Processing (2008)
42. Buchholz, S., Marsi, E.: CoNLL-X shared task on multilingual dependency parsing. In: Tenth Conference on Computational Natural Language Learning (CoNLL), pp. 149–164 (2006)
43. Ambati, B.R., Husain, S., Jain, S., Sharma, D.M., Sangal, R.: Two methods to incorporate local morphosyntactic features in Hindi dependency parsing. In: NAACL-HLT 2010 Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2010), Los Angeles, CA (2010)

Identification of Reduplicated Multiword Expressions Using CRF

Kishorjit Nongmeikapam[1], Dhiraj Laishram[1], Naorem Bikramjit Singh[1], Ngariyanbam Mayekleima Chanu[2], and Sivaji Bandyopadhyay[3]

[1] Dept. of Computer Sc. & Engg., Manipur Institute of Technology, Manipur University, Imphal, India
[2] Dept. of Education Technology, Kanan Devi Memorial College of Education, Imphal, India
[3] Dept. of Computer Sc. & Engg., Jadavpur University, Jadavpur, Kolkata, India
{kishorjit.nongmeikapa,dhirajlaishram, naorembikramjit10,mayekleima.ng}@gmail.com, sivaji_cse_ju@yahoo.com

Abstract. This paper deals with the identification of Reduplicated Multiword Expressions (RMWEs) which is important for any natural language applications like Machine Translation, Information Retrieval etc. In the present task, reduplicated MWEs have been identified in Manipuri language texts using CRF tool. Manipuri is highly agglutinative in nature and reduplication is quite high in this language. The important features selected for running the CRF tool include stem words, number of suffixes, number of prefixes, prefixes in the word, suffixes in the word, Part Of Speech (POS) of the surrounding words, surrounding stem words, length of the word, word frequency and digit feature. Experimental results show the effectiveness of the proposed approach with the overall average Recall, Precision and F-Score values of 92.91%, 91.90% and 92.40% respectively.

Keywords: Multiword Expressions (MWE), Reduplicated MWE, Conditional Random Field (CRF), Manipuri.

1 Introduction

Manipuri (or Meiteilon) belongs to the Tibeto-Burman family of languages mainly spoken in Manipur, a state in the North East India. Manipuri is a schedule language of Indian constitution. This language is also spoken in some parts of the other countries like Myanmar and Bangladesh. Manipuri is highly agglutinative in nature, monosyllabic, influenced and enriched by the Indo-Aryan languages of Sanskrit origin and English. The affixes play the most important role in the structure of the language. The majority of the roots found in the language are bound and the affixes are the determining factor of the class of the words in the language. Manipuri uses two scripts; the first one is purely of its own origin (Meitei Mayek) while another one is a borrowed Bengali script. In the present task, the processing has been done on the Bengali script.

MWE is composed of an ordered group of words which can stand independently and carries a different meaning from its constituent words. For example in English: *'to and fro', 'bye bye', 'kick the bucket' etc.* MWEs include compounds (both word-compounds and phrasal compounds), fixed expressions and technical terms. A fixed expression MWE is one whose constituent words cannot be moved randomly or substituted without distorting the overall meaning or allowing a literal interpretation. Fixed expressions range from word-compounds, collocations to idioms. Some of the proverbs and quotations can also be considered as fixed expressions. Reduplicating words are usually collocated words MWE.

The paper is organized in the following manner. Section 2 gives a brief discussion about related works, Section 3 details about Manipuri Reduplicated MWEs, Section 4 gives the list of prefixes, suffixes and an example of highly agglutinative word, Section 5 gives the idea about how words are stemmed, Section 6 gives the concept of CRF, the Reduplicated MWEs identification using CRF is discussed in Section 7 which is followed by the experiments and evaluation while the conclusion is drawn in Section 8.

2 Related Works

The concept of Reduplicated MWE originated from the concept of MWEs. So far few works of Reduplicated MWEs and MWEs are found on Indian Languages as well as on other Language. First work of Manipuri Reduplicated MWEs can be seen in [1], other combined works on identification of named entities and Reduplicated MWEs can be seen in [2]. For Bengali works on Reduplicated MWEs can be found in [3]. Works on MWE identification for Bengali and Hindi languages can be found in [4], [5] and [6]. MWE works on languages other than Indian languages can be found in [7], [8], [9] and [10]. It is observed that very little work has been done for Reduplicated MWEs.

3 Manipuri Reduplicated MWEs

The difficulties faced during the POS tagging task of Manipuri motivated us to work on the identification of MWEs and reduplicated MWEs in Manipuri. Some example Reduplicated MWEs which are difficult in POS tagging are words like *ইমুন ইমুন মুনবা* (*'i-mun i-mun mun-ba'*) which means, *'completely ripe'*, *ঙোশোক ঙোশোক ঙৌবা* (*'ŋəɯ-srok ŋəɯ-srok ŋəɯ-ba'*) which means *'shining white'* etc. Works for identification of Reduplicated MWEs in Manipuri is found in [1]. The process of reduplication is defined in [11] as: 'reduplication is that repetition, the result of which constitutes a unit word'. These single unit words are the MWEs. The reduplicated MWEs in Manipuri are classified mainly into four different types. These are: 1) Complete Reduplicated MWEs, 2) Mimic Reduplicated MWEs, 3) Echo Reduplicated MWEs and 4) Partial Reduplicated MWEs. Apart from these four types there are also cases of a) Double reduplicated MWEs and b) Semantic Reduplicated MWEs.

3.1 Complete Reduplicated MWEs

In the complete reduplication MWEs the single word or clause is repeated once forming a single unit regardless of phonological or morphological variations. Interestingly in Manipuri these complete reduplication MWEs can occur as Noun, Adjective, Adverb, *Wh-* question type, Verbs, Command and Request. For example, মরিক মরিক (*'mərik mərik'*) which means *'drop by drop'*.

3.2 Partial Reduplicated MWEs

In case of partial reduplication the second word carries some part of the first word as an affix to the second word, either as a suffix or a prefix. For example, চৎথোক চৎসিন (*'cət-thok cət-sin'*) means *'to go to and fro'*, শামী লানমী (*'sa-mi lan-mi'*) means *'army'*.

3.3 Echo Reduplicated MWEs

The second word does not have a dictionary meaning and is basically an echo word of the first word. For example, *thk-si kha-si* means *'good manner'*. Here the first word has a dictionary meaning *'good manner´* but the second word does not have a dictionary meaning and is an echo of the first word.

3.4 Mimic Reduplicated MWEs

In the mimic reduplication the words are complete reduplication but the morphemes are onomatopoetic, usually emotional or natural sounds. For example, করক করক (*'krək krək'*) means *'cracking sound of earth in drought'*.

3.5 Double Reduplicated MWEs

In double Reduplicated MWE there consist of three words, where the prefix or suffix of the first two words is reduplicated but in the third word the prefix or suffix is absent. An example of double prefix reduplication is ইমুন ইমুন মুনবা (*'i-mun i-mun mun-ba'*) which means, *'completely ripe'*. It may be noted that the prefix is duplicated in the first two words while in the following example suffix reduplication take place, ঙাশোক ঙাশোক ঙাবা (*'ŋəω-srok ŋəω-srok ŋəω-ba'*) whichmeans *'shining white'*.

3.6 Semantic Reduplicated MWEs

Both the reduplication words have the same meaning as well as the MWE. Such type of MWEs is very special to the Manipuri language. For example, পামবা কৈ (*'pamba kəy'*) means *'tiger'* and each of the component words means *'tiger'*. Semantic reduplication exists in Manipuri in abundance as such words have been generated from similar words used by seven clans in Manipur during the evolution of the language.

4 Prefixes, Suffixes and an Example of Highly Agglutinative Manipuri Word

Altogether 72 (seventy two) affixes are listed in Manipuri out of which 11 (eleven) are prefixes and 61 (sixty one) are suffixes. Table 1 shows the 10 prefixes. The prefix ম (mə) is used both as formative and pronomial prefix but it is included only once in the list. Similarly, Table 2 lists 55 (fifty five) suffixes as some of the suffixes are used with different forms of usage such as গুম (gum) which is used as particle as well as proposal negative, দা (də) as particle as well as locative and না (nə) as nominative, adverbial, instrumental or reciprocal.

To prove with the point that Manipuri is highly agglutinative let us site an example word: "*পুশিনহনজারমগাদাবানিদাকো*" ("pusinhənjərəmgədəbənidəko"), which means "(I wish I) myself would have caused to carry in (the article)". Here there are 10 (ten) suffixes being used in a verbal root, they are "*pu*" is the verbal root which means "to carry", "*sin*"(in or inside), "*hən*" (causative), "*jə*" (reflexive), "*rəm*" (perfective), "*gə*" (associative), "*də*" (particle), "*bə*" (infinitive), "*ni*" (copula), "*də*" (particle) and "*ko*" (endearment or wish).

Table 1. Prefixes in Manipuri

Prefixes used in Manipuri
অ, ই, ই, খু, চা, ত, থ, ন, ম and শে

Table 2. Suffixes in Manipuri

Suffixes used in Manipuri
কন, কুম, কো, খরে, খৎ, খাই, খি, খোয়, গা, গনি, গী, গুম, ঙৈ, চা, চো, থ, থৎ, থেক, থোক, দা, দি, দুনা, দে, না, নত্তে, নি, নিং, নু, নে, পী, ফাৎ, বা, বু, মক, মল, মিন, মুক, লে, লা, লক, ল্ম, লি, লী, লূ, লু, লে, লো, লোয়, শনু, শি, শিং, শিন, শু, হৎ and হন

5 Manipuri Word Stemming

Manipuri words are stemmed by stripping the suffixes in an iterative manner. As mentioned in Section 4 a word is rich with suffixes and prefixes. In order to stem a word an iterative method of stripping is done by using the acceptable list of prefixes (11 numbers) and suffixes (61 numbers) as mentioned in the Table 1 and Table 2 above.

6 Concept of CRF

The concept of Conditional Random Field [12] is developed in order to calculate the conditional probabilities of values on other designated input nodes of undirected graphical models. CRF encodes a conditional probability distribution with a given set of features. It is an unsupervised approach where the system learns by giving some training and can be used for testing other texts.

The conditional probability of a state sequence X=(x_1, x_2,..x_T) given an observation sequence Y=(y_1, y_2,..y_T) is calculated as :

$$\mathrm{P(Y|X)} = \frac{1}{Z_X}\exp(\sum_{t=1}^{T}\sum_{k}\lambda_k \mathrm{f}_k(\mathrm{y}_{t-1},\mathrm{y}_t,\mathrm{X},\mathrm{t})) \tag{1}$$

where, f_k(y_{t-1},y_t, X, t) is a feature function whose weight λ_k is a learnt weight associated with f_k and to be learned via training. The values of the feature functions may range between -∞ … +∞, but typically they are binary. Z_X is the normalization factor:

$$Z_X = \sum_{y}\exp\sum_{t=1}^{T}\sum_{k}\lambda_k \mathrm{f}_k(\mathrm{y}_{t-1},\mathrm{y}_t,\mathrm{X},\mathrm{t})) \tag{2}$$

which is calculated in order to make the probability of all state sequences sum to 1. This is calculated as in Hidden Markov Model (HMM) and can be obtained efficiently by dynamic programming. Since CRF defines the conditional probability P(Y|X), the appropriate objective for parameter learning is to maximize the conditional likelihood of the state sequence or training data.

$$\sum_{i=1}^{N}\log \mathrm{P}(\mathrm{y}^i \mid \mathrm{x}^i) \tag{3}$$

where, {(x^i, y^i)} is the labeled training data.

Gaussian prior on the λ's is used to regularize the training (i.e., smoothing). If λ ~ N(0,ρ^2), the objective function becomes,

$$\sum_{i=1}^{N}\log \mathrm{P}(\mathrm{y}^i \mid \mathrm{x}^i) - \sum_{k}\frac{\lambda_i^2}{2\rho^2} \tag{4}$$

The objective function is concave, so the λ*'s* have a unique set of optimal values.

7 Reduplicated MWE Identification Using CRF

Figure 1 shows the system architecture for identification of Reduplicated MWEs using CRF. The important processes required in identification of Reduplicated MWE extraction using CRF are feature selection, preprocessing which includes arrangement of tokens or words into sentences with other notations, creation of model file after training and finally the testing with the test corpus. For the current work, C++ based CRF++ 0.53 package[1] which is readily available as open source for segmenting or labeling sequential data is used.

[1] `http://crfpp.sourceforge.net/`

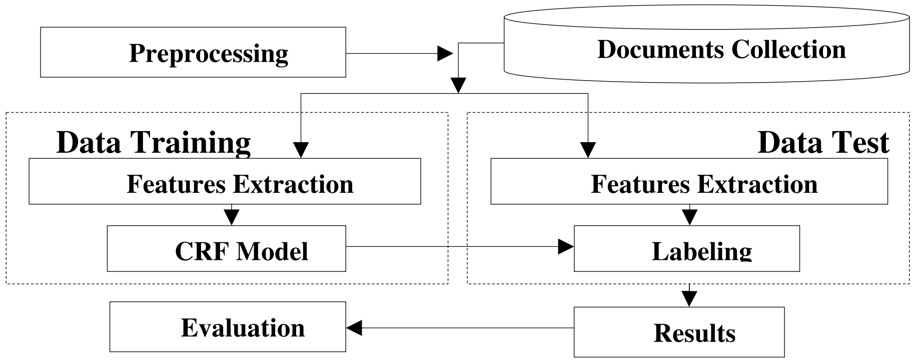


Fig. 1. System Architecture for identification of Reduplicated MWEs

Following sub sections explain the overall process in detail:

7.1 Feature Selection

The feature selection is important in CRF. The various features used in the system are,

F= {SW_{i-m}, …, SW_{i-1}, SW_{i}, SW_{i+1},… , SW_{i-n} , number of acceptable suffixes, number of acceptable prefixes, acceptable suffixes present in the word, acceptable prefixes present in the word, Surrounding POS tag, word length, word frequency, digit feature }

The details of the set of features that have been applied for MWEs in Manipuri are as follows:

Surrounding Stem words as feature: Stemming is done as mentioned in Section 5 and the preceding and following stem words of a particular word are used as features since the preceding and following words influence the present word in case of reduplicated MWEs.

Number of acceptable suffixes as feature: The suffix plays an important role in Manipuri since it is a highly agglutinative language. For every word the numbers of suffixes are identified during stemming, if any and the number of suffixes is used as a feature.

Number of acceptable prefixes as feature: Like suffixes the prefixes plays an important role too for Manipuri since it is a highly agglutinative language. For every word the numbers of prefixes are identified during stemming, if any and the number of prefixes is used as a feature.

Acceptable suffixes: 61 suffixes have been manually identified in Manipuri and the list of suffixes is used as one feature. As mention with an example in Section 4, suffixes are appended one after another and the maximum number of appended suffixes can be ten. So taking into account of such case, for every word ten columns separated by a space are created for every suffix present in the word. A “NIL” notation is being used in those columns when the word consists of less or no acceptable suffixes.

Acceptable prefixes as feature: 11 prefixes have been manually identified in Manipuri and the list of prefixes is used as one feature. For every word the prefix is identified and a column is created mentioning the prefix if the prefix is presents, otherwise the "NIL" notation is used.

Surrounding POS tag: Reduplicated MWEs can be a combination of noun-noun, verb-noun, adjective-noun POS patterns etc, so the POS of the surrounding words are considered as an important feature.

Length of the word: Length of the word is set to 1 if it is greater than 3 otherwise, it is set to 0. Very short words are rarely Reduplicated MWEs.

Word frequency: A range of frequency for words in the training corpus is set: those words with frequency <100 occurrences are set the value 0, those words which occurs >=100 but less than 400 are set to 1 and so on. The word frequency is considered as one feature since Reduplicated MWEs are rare in occurrence compared to those of determiners, conjunctions and pronouns.

Digit features: Reduplicated MWEs are generally strings of characters and not digits. Thus the digit feature is an important feature. A binary notation of '1' is used if the word consist of a digit else '0'.

7.2 Pre-processing and Feature Extraction

A Manipuri text document is used as an input file. The training and test files consist of multiple tokens. In addition, each token consists of multiple (but fixed number) columns where the columns are used by a template file. The template file gives the complete idea about the feature selection. Each token must be represented in one line, with the columns separated by white spaces (spaces or tabular characters). A sequence of tokens becomes a **sentence**. Before undergoing training and testing in the CRF the input document is converted into a multiple token file with fixed columns and the template file allows the feature combination and selection which is specified in section 7.1.

Two standard files of multiple tokens with fixed columns are created: one for training and another one for testing. In the training file the last column is manually tagged with all those identified Reduplicated MWEs by marking B-RMWE for the beginning of Reduplicated MWE and I-RMWE for the rest of the Reduplicated MWE else 'O' for those which are not reduplicated MWE whereas in the test file we can either use the same tagging for comparisons or only 'O' for all the tokens regardless of whether it is Reduplicated MWE or not.

7.3 Model File after Training

In order to obtain a model file we train with CRF using the training file. This model file is a ready-made file by the CRF tool for use in the testing process. In other words the model file is the learnt file after the training of CRF. We do not need to use the template file and training file again since the model file consists of the detail information of the template file and training file.

7.4 Testing

The test file is the test data to which we want to assign sequential tags of the Reduplicated MWEs by the system, i.e., tagging with B-RMWE for the beginning of Reduplicated MWE and I-RMWE for the rest of the Reduplicated MWE else ‘O’ for those words which are not Reduplicated MWEs. As mentioned earlier in section 7.2 this file has to be created in the same format as that of training file, i.e., of fixed number of columns with the same field as that of training file.

The output of the testing process is a new file with an extra column which is tagged with B-RMWE for the beginning of Reduplicated MWE and I-MWE for the rest of the Reduplicated MWE else ‘O’ for those words which are not part of any Reduplicated MWEs.

8 Experiment and Evaluation

Manipuri corpus are collected and filtered to rectify the spelling and syntax of a sentence by a linguist expert from Linguistic Department, Manipur University. In the corpus some words are written in English, such words are rewritten into Manipuri in order to avoid confusion or error in output. The corpus we have collected includes 55,000 tokens which are of Gold standard. A total of 50,000 words are considered for training and testing is done on the rest 5000 words.

A number of problems have been faced while doing the experiment due to typical nature of the Manipuri language. In Manipuri, word category is not so distinct. The verbs are also under bound category. Another problem is to classify basic root forms according to word class although the distinction between the noun class and verb classes is relatively clear; the distinction between nouns and adjectives is often vague. Distinction between a noun and an adverb becomes unclear because structurally a word may be a noun but contextually it is adverb. Further a part of root may also be a prefix, which leads to wrong tagging. The verb morphology is more complex than that of noun. Sometimes two words get fused to form a complete word.

8.1 Best Feature Selection

Experiments are performed in order to identify the best feature so that maximum Reduplicated MWEs are identified in a given text. Training is done with features created by a Gold standard and manually POS tagged data by a linguist and tested with 5000 tokens. The notations used by the linguist for the Reduplicated MWE are B-RMWE for the beginning of Reduplicated MWE and I-RMWE for the rest of the Reduplicated MWE else ‘O’ for those which are not part of any reduplicated MWE. Among the features mentioned in Section 7.1 different combinations have been experimented in order to identify the best feature set that produces good result. The Table 4 shows the result in terms of recall (**R**), precision (**P**) and F-measure (**F**). Table 3 suggests the meaning of the notations used.

Table 3. Meaning of the notations

Notation	Meaning
SW[-I,+J]	Stem Words spanning from the i-th left position to the j-th right position
POS[-I, +J]	POS tags of the words spanning from the ith left to the jth right positions
NoPre	Number of prefix present in the word
NoSuf	Number of suffix present in the word
Pre	Prefixes present in the word
Suf_j	Suffixes present in the word, where j= 1 to 10 (Refer section 7.1)
DF	Digit feature
Len	Length of the word
Frq	Frequency of the word

Table 4. Results on the development set

Feature	R (in %)	P (in %)	FS (in %)
SW[-2,+2], POS[-2,+2], NoPre, NoSuf, Pre, Suf_j, Len, Frq, DF	**92.91**	**91.90**	**92.40**
SW[-3,+3], POS[-3,+3], NoPre, NoSuf, Pre, Suf_j, Len, Frq, DF	90.12	88.67	89.39
SW[-3,+2], POS[-3,+2], NoPre, NoSuf, Pre, Suf_j, Len, Frq, DF	87.76	84.78	86.24
SW[-4,+3], POS[-4,+3], NoPre, NoSuf, Pre, Suf_j, Len, Frq, DF	78.87	76.87	77.86
SW[-4,+3], POS[-4,+3], NoPre, NoSuf, Pre, Suf_j, Len, Frq, DF	68.64	65.89	67.24
SW[-2,+2], POS[-2,+2], NoPre, NoSuf, Pre, Suf_j, Len, Frq, DF	64.78	63.99	64.38
SW[-3,+3], POS[-3,+3], NoPre, NoSuf, Pre, Suf_j, Len, Frq, DF	52.88	50.14	51.47
SW[-4,+3], POS[-4,+2], NoPre, NoSuf, Pre, Suf_j, Len, Frq, DF	25.76	42.78	32.16
SW[-4,+4], POS[-4,+4], NoPre, NoSuf, Pre, Suf_j, Len, Frq, DF	21.54	36.24	27.06

A lot of variations are noticed in Table 4 as we experiment with various feature combinations. We have started the combination from four stem words prior and four stem words succeeding a given stem word, POS tag of previous four words and the following four words, number of prefixes and suffixes, prefixes and suffixes appended in each word extended up to ten suffixes and one prefix, word length, frequency and a digital feature.

8.2 Best Result

With the experiments performed above we are able to find the best feature selection for the CRF. The best combination is thus as follows:

F={ SW_{i-2},SW_{i-1}, SW_i, SW_{i+1} ,SW_{i-2}, POS tag(s) of the current and 2 preceding and 2 following word(s), Number of Prefix, Number of Suffixes, Prefix of the word, Upto ten suffixes (if present) in the word, Length of the word, word frequency, Digital feature}

Table 5. Best result on the test set

Reduplicated MWE Types	*Recall*	*Precision*	*F-Measure*
Complete and Mimic	**98.33**	**92.16**	**95.15**
Partial	**95.68**	**96.56**	**96.12**
Echo	**90.37**	**91.21**	**87.92**
Double	**96.48**	**92.78**	**94.59**
Semantic	**83.72**	**86.78**	**85.22**

The best result for different Reduplicated MWEs shows a result as in Table 5. The overall best result for all types of MWEs is shown in Table 6.

Table 6. Overall best result on the test set

Model	Recall	Precision	F-Score
CRF	**92.91**	**91.90**	**92.40**

8 Conclusion

In this paper experiments have been carried out using the CRF tool for identification of Reduplicated MWEs in Manipuri and results achieved are promising. More features need to be identified in future to improve the score. Besides using the CRF tool, other machine learning methods need experimentation for identification of Reduplicated MWEs. Moreover, reduplicated MWE identification is also necessary for authorship stylometry task.

References

1. Kishorjit, N., Bandyopadhyay, S.: Identification of Reduplicated MWEs in Manipuri: A Rule based Approached. In: Proceedings of 23rd International Conference on the Computer Processing of Oriental Languages (ICCPOL 2010), Redwood City, San Francisco, USA, pp. 49–54 (2010)
2. Singh, T.D., Bandyopadhyay, S.: Web Based Manipuri Corpus for Multiword NER and Reduplicated MWEs Identification using SVM. In: 23rd International Conference on the Computational Linguistics (COLING), Beijing, pp. 35–42 (2010)
3. Chakraborty, T., Bandyopadhyay, S.: Identification of Reduplication in Bengali Corpus and their Semantic Analysis: A Rule-Based Approach. In: 23rd International Conference on the Computational Linguistics (COLING), Beijing, pp. 73–76 (2010)
4. Agarwal, A., Ray, B., Choudhury, M., Sarkar, S., Basu, A.: Automatic Extraction of Multiword Expressions in Bengali: An Approach for Miserly Resource Scenarios. In: Proceedings of ICON 2004, pp. 165–174. Macmillan, Basingstoke (2004)
5. Dandapat, S., Mitra, P., Sarkar, S.: Statistical investigation of Bengali noun-verb (N-V) collocations as multi-word-expressions. In: Proceedings of MSPIL, Mumbai, pp. 230–233 (2006)
6. Kunchukuttan, A., Damani, O.P.: A System for Compound Nouns Multiword Expression Extraction for Hindi. In: Proceedings of ICON 2008, Macmillan, Basingstoke (2008)
7. Enivre, J., Nilson, J.: Multiword Units in Syntactic Parsing. In: Proceedings of MEMURA 2004 Workshop, Lisbon, pp. 39–46 (2004)
8. Koster, C.H.A.: Transducing Text to Multiword Unit. In: Proceedings of MEMURA 2004 Workshop, Lisbon, pp. 31–38 (2004)
9. Odijik, J.: Reusable Lexical Representation for Idioms. In: Proceedings of LREC 2004, Lisbon, pp. 903–906 (2004)
10. Diab, M.T., Bhutada, P.: Verb Noun Construction MWE Token Supervised Classification. In: Workshop on Multiword Expression, ACL-IJCNLP, Singapore, pp. 17–22 (2009)
11. Singh, C.Y.: Manipuri Grammar, pp. 190–204. Rajesh Publications, Delhi (2000)
12. Lafferty, J., McCallum, A., Pereira, F.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: Procceedings of the 18th International Conference on Machine Learning (ICML 2001), Williamstown, MA, USA, pp. 282–289 (2001)

Computational Linguistics and Natural Language Processing

Jun'ichi Tsujii

Department of Computer Science, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
School of Computer Science, University of Manchester
National Centre for Text Mining (NaCTeM)
Manchester Interdisciplinary Biocentre, 131 Princess Street,
Manchester M1 7DN, UK
tsujii@is.s.u-tokyo.ac.jp

Abstract. Researches in Computational Linguistics (CL) and Natural Language Processing (NLP) have been increasingly dissociated from each other. Empirical techniques in NLP show good performances in some tasks when large amount of data (with annotation) are available. However, in order for these techniques to be adapted easily to new text types or domains, or for similar techniques to be applied to more complex tasks such as text entailment than POS taggers, parsers, etc., rational understanding of language is required. Engineering techniques have to be underpinned by scientific understanding. In this paper, taking grammar in CL and parsing in NLP as an example, we will discuss how to re-integrate these two research disciplines. Research results of our group on parsing are presented to show how grammar in CL is used as the backbone of a parser.

1 Introduction

The two terms, Computational Linguistics (CL) and Natural Language Processing (NLP), have often been used interchangeably. However, these two terms represent two different streams of research which emphasize different aspects of our field. For example, while research on grammar and its formalisms in CL and research on parsing in NLP are closely related, their objectives are quite different. On one hand, researchers in CL have focused on revealing how surface strings of words systematically correspond to their meanings (in a compositional way) and have been interested in developing formalisms by which the correspondences are described. On the other hand, those in NLP are interested in more practical engineering issues involved in processing natural languages by computer, such as efficient algorithms for a program (parser) which computes the structure and/or the meaning of a given sentence.

While some of parsers used in NLP are based on a grammar in CL, in order for them to be practical, they should not only be efficient and robust but also be able to choose the most plausible interpretation of a sentence among many possible interpretations. Probabilistic modeling, which has been extensively studied of lately, has been successful not only in making such choices but also in improving both the efficiency and robustness of parsers. However, since probabilistic models are beyond the scope of research on grammars in CL formalisms, the two research streams have become increasingly dissociated from each other.

We discuss in this paper our research strategy for re-connecting these two streams of research to evolve a new broader filed in which research on grammar representation and processing are properly integrated. In particular, we argue that representation of grammar and processing based on it should be clearly distinguished. Straightforward application of a grammar in CL to parsers in NLP would not be fertile as we expected. At the same time, statistical modeling without proper linguistic theories would be as futile as CL without proper consideration of processing issues. We discuss several researches [1–9], which we have been engaged in. They will shed light on the interesting relationship between representation of grammar and processing.

2 Grammar Formalisms in CL

Confusion between grammar formalisms and processing started at the very beginning of research on grammar in theoretical linguistics. Chomsky proposed a set of transformation rules in order to associate a pair of sequences of words which are very different from each other as sequence but share the same core meaning [10]. The Standard Theory was proposed in [11], which postulated two levels of structures, the surface and deep structures. The former is mapped to phonological realization (sequence of words) and the latter with meaning through semantic interpretation.

Though the main aim of introduction of transformation rules was to associate surface sequences of words with somewhat standardized "meaning" representations, the whole process of linking the two levels was misinterpreted as a mental process which has some psychological reality. It was wrongly claimed that transformational grammar was not suited for parsing because it modeled generation of sentences instead of recognition.

The tendency of directly associating a grammar representation with process has become less common, due to the advent of grammar formalisms in CL, such as GPSG [12] (Generalized Phrase Structure Grammar), LFG [13] (Lexical Functional Grammar), HPSG [14] (Head-Driven Phrase Structure Grammar), CCG [15] (Combinatorial Category Grammar), LTAG [16] (Tree Adjoining Grammar) etc.. These formalisms have made it clear that the role of grammar is to define an infinite set or/and to describe relationships between surface sequences of words and their meanings. A grammar models neither psychological processes in human mind nor computational processes for computer software. These formalisms, inspired by computer science, are accompanied by clear semantics, and emphasize the declarative nature of grammar representation.

The separation of declarative representation of grammar from processing has significant implication on NLP (Natural Language Processing). Research on parsing, for example, is concerned with the efficiency or psychological reality of algorithms which, given a sequence of words, compute the structure as a sentence and/or its meaning. Due to the inherent ambiguities of natural language, the search space is very large. Therefore, issues such as how to search the space efficiently, how to find the most plausible interpretations among many, etc. have been the major research issues in parsing.

However, the declarative nature of the grammar formalisms and the freedom they provides have not been fully explored in NLP research.

3 Grammar Conversion

The clear separation of grammar representation and processing is a norm in formal language theory (FLT). For example, the same grammar can be used by diverse algorithms (Left-Corner, CYK, Earley, Shift-Reduce, etc), all of which are guaranteed to be complete and sound. These base algorithms of CFG can also be augmented with probabilistic models, and one can study the formal property, the efficiency and effectiveness of these algorithms [17, 18]. Furthermore, in FLT, grammars and the language defined by them are also clearly separated. A grammar can be transformed into seemingly different grammars without affecting the language defined by them. A grammar in CFG can be transformed into other grammars of standard forms such as Chomsky Normal Form, Graibach Normal Form, etc. The base algorithms of parsing are defined for one of these standard forms.

Compared with CFG, grammar conversion among grammars in the different formalisms in CL has hardly been studied. We neither know whether two grammars in different formalisms define the same set, nor whether they have the same number of derivation histories (i.e. the same number of interpretations) for a given sequence of words.

We aimed in [19] to remedy this situation by showing that a grammar of LTAG [20] can be converted to a HPSG-style of grammar. In this work, we treated HPSG from two different perspectives, HPSG as a formal descriptive framework and HPSG as a linguistic theory. As a formal framework, HPSG has the same generative power as Type0, while TAG, the formal framework for LTAG, belongs to the class called "mildly context sensitive" [21]. The greater generative power of HPSG as a formal framework allows us to obtain a trivial encoding of LTAG [22]. However, such conversion leads to a grammar which does not follow the basic assumptions made by HPSG as a linguistic theory [14]. We require our HPSG-style grammar to satisfy the minimum principles of HPSG as linguistic theory.

For example, a lexical entry for a word must express the characteristics of the word in the HPSG style, such as its sub-categorization frame and grammatical category. A rule schema must represent constraints on the configuration of immediate constituents and not be a construction-specific rule. These restrictions enable us not only to define a formal link between the two frameworks (TAG and HPSG as a framework), but also to clarify the relationship between specific grammars in LTAG and HPSG as a linguistic theory.

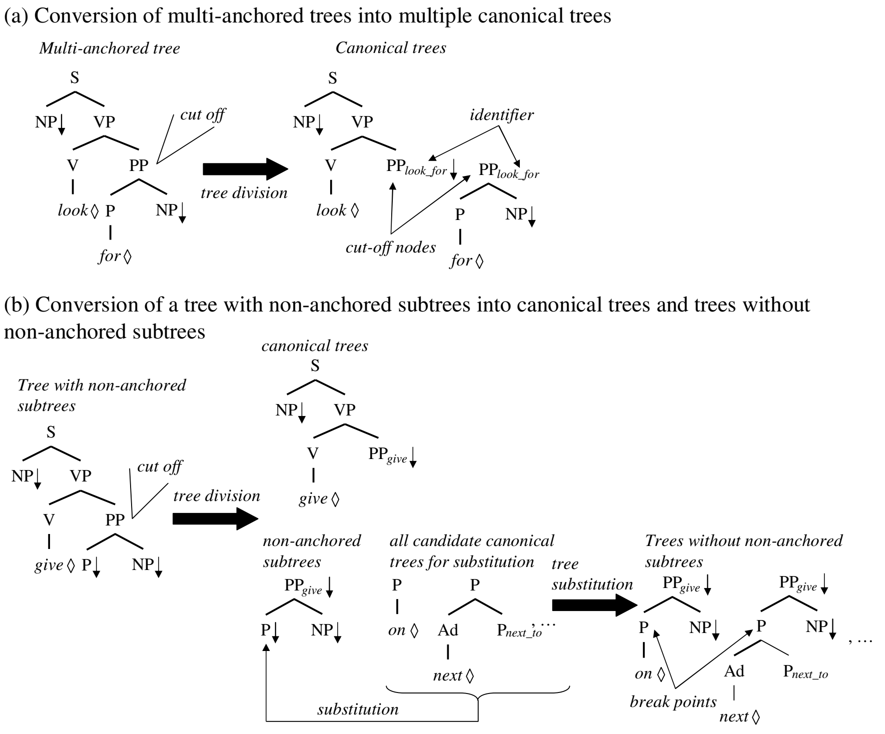

(a) Conversion of multi-anchored trees into multiple canonical trees
Multi-anchored tree
Canonical trees
cut off
tree division
identifier
cut-off nodes
look ◊
for ◊
(b) Conversion of a tree with non-anchored subtrees into canonical trees and trees without non-anchored subtrees
canonical trees
Tree with non-anchored subtrees
cut off
tree division
give ◊
non-anchored subtrees
all candidate canonical trees for substitution
tree substitution
Trees without non-anchored subtrees
on ◊
next ◊
substitution
break points

Fig. 1. Conversion of LTAG trees into canonical elementary trees

LTAG (lexicalized TAG) is a linguistic theory which use TAG as its descriptive framework. As a linguistics theory, LTAG commits to specific principles which individual grammars should follow. For example, it requires all the elementary trees in a grammar be anchored by lexical items. The first approximation of a grammar of LTAG by HPSG can be obtained simply by folding the leaf-trees of an elementary tree into the Subcat feature of the lexical head in HPSG. However, grammar conversion reveals interesting insights on differences between the two linguistic theories. LTAG, for example, allows elementary trees to be multi-anchored. It also allow non-anchored sub-trees to be included as parts of elementary trees. The former is used to treat continuous or discontinuous idioms, which causes difficulties in HPSG. The latter allow generalization which HPSG cannot make in a simple manner.

These constructions can also be converted into HPSG-style canonical elementary trees (Fig. 1 (a) and (b)). Such conversion suggests interesting extensions of HPSG (See [23] for the details). With such extensions to HPSG, we showed that we could construct a HPSG-like grammar which is strongly equivalent to an original LTAG grammar. The actual LTAG grammar we used is the XTAG English grammar at 2001 [24], which was one of the most comprehensive grammars of English. Experiments confirmed that the number of derivation trees by our grammar is exactly the same as by the original XTAG grammar.

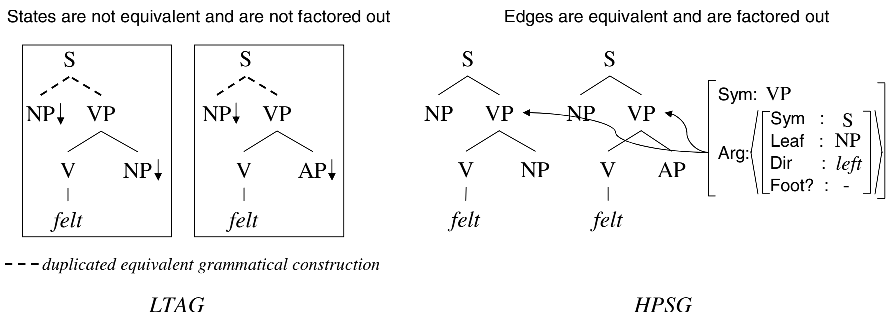

States are not equivalent and are not factored out
Edges are equivalent and are factored out
S
NP↓ VP
V NP↓
felt
V AP↓
S
NP VP
V NP
V AP
Sym: VP
Arg:
Sym : S
Leaf : NP
Dir : left
Foot? : -
duplicated equivalent grammatical construction
LTAG
HPSG

Fig. 2. Difference between factoring schemes in LTAG and HPSG

More importantly, parsers based on the two grammars in the LTAG and HPSG formalisms showed very different performances in terms of parsing efficiency. The LTAG parser we used for comparison is the one reported in [25]. It is based on the Head-Corner algorithm proposed in [26]. The parser for HPSG is a naïve implementation of CKY. As Table 2 shows, the naïve CKY parser is much faster, by a factor of 13, than the original LTAG parser. Both parsers use dynamic programming, in which the merging operation of the same states plays the essential role in preventing combinatorial explosion. The main cause of inefficiency in LTAG parsing is that the head-corner parsing can merge states only when partial parses share the same elementary trees. On the other hand, the HPSG-grammar based parser slices elementary trees into a set of binary branching sub-trees so that it can merge when partial parses share the same top-level binary trees.

This experiment shows that the two equivalent grammars in terms of declarative semantics may have very different characteristics from the view of processing.

Table 1. Classification of elementary trees in the XTAG English grammar (LTAG) and lexical entries converted from them (HPSG)

Grammar	*A*	*B*	*C*	*D*	**Total**
LTAG	326	763	54	50	1,193
HPSG	326	1,989	1,083	2,474	5,872

A: Canonical elementary trees
B: Multi-anchored trees without non-anchored sub-trees
C: Single-anchored trees with non-anchored sub-trees
D: Multi-anchored trees with non-anchored sub-trees

Table 1 and Table 2 illustrate the interesting contrasts of the two grammars. Table 1 shows that the LTAG grammar captures generalization better than the HPSG one. 1,193 elementary trees in XTAG have to be expanded into 5,872 lexical entries in the converted HPSG. On the other hand, the parser using the HPSG grammar outperforms the LTAG parser which keeps elementary trees

Table 2. Parsing performance with the XTAG English grammar for the ATIS corpus

Parser	Parse time (sec.)
Naïve HPSG Parser	1.54
LTAG Parser	20.76

as they are during parsing. It is important to note that the HPSG parser can reconstruct the same derivation trees and descriptive trees as the LTAG parser produces, without keeping elementary trees.

4 Deriving Partial Constraints from Grammar - CFG Filtering

Though formalisms in CL do not use unbridled transformation, they also provide their own devices which are beyond CFG. Since they normally use feature-value representation to characterize phrases and constraints among them, parsing algorithms for them use unification instead of symbol equality among non-terminal symbols in CFG, which is computationally expensive [27].

Another alternative is to avoid unification from processing altogether. A bundle of feature-value pairs can be represented by a non-terminal symbol in CFG, and if we ignored certain features (e.g. compositionally built semantic representation), the number of possible bundles of feature-value pairs would be finite. The discussion in Sect. 3 also indicates that a grammar in one formalism (e.g. LTAG) can be converted into another one in another formalism (e.g. HPSG). Surface differences in representation are not essential in terms of the language actually defined.

Following this line of thought, we derived a CFG grammar from a given HPSG grammar [28]. Since we remove some of the features in the given HPSG to limit the number of non-terminals, the derived CFG is less constrained than the original grammar. On the other hand, a sequence accepted by the original grammar is guaranteed to be accepted by the derived CFG.

The derived CFG approximates the original grammar in the same way as a FA approximates a push-down automaton PDA by restricting the depth of the stack [29] (For RG approximation of CFG, see [30]).

We derive a CFG from a HPSG grammar by recursively instantiating daughters of an ID rule of the HPSG grammar with lexical entries and generated feature structures, as shown in Fig. 3. This procedure terminates when new feature structures are not generated. In order to guarantee termination of the whole process, we impose restrictions [31] on the features (i.e. ignore some of the features which lead to infinitely many feature structures) (See [28] for the details). The CFG thus derived have a large set of non-terminal symbols, and the generalizations captured by the grammar representation are lost. However, parsing based on a CFG with atomic symbols as non-terminals is far faster than parsing using feature unification. In this work, we viewed unification as a device

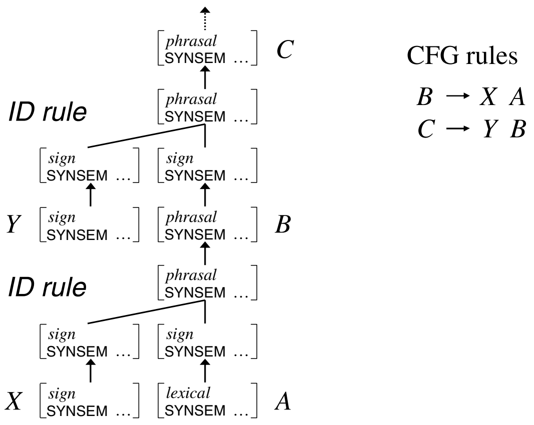


Fig. 3. Extraction of CFG from HPSG

for grammar representation which has clear declarative semnatics [32]. As long as the semantics of grammar representation is not changed, we can tranform the representation into other forms for processing.

Once parsing finishes, we can easily reconstruct the derivation history by the original grammar and perform further checking by unification. Since we use the original grammar at this stage, the meaning representation can be constructed at this stage.

The number of unification required is far less than that parsers using the original grammar alone. This is because all partial parses which fail to contribute to successful parses have been removed by the first phase of CFG parsing. Such global filtering of unnecessary rule applications is found to be far more effective than local filtering by quick checks.

The overall architecture using CFG filtering is very similar to that for LFG. In LFG, the CFG component is explicit in grammar representation as c-structure. A LFG parser generally adopts two-stage architecture [33], which is essentially the same as CFG filtering in our framework. The essential difference is in grammar representation. LFG, a multi-strata syntactic theory, considers c-structure as an essential level of representation, while HPSG, a mono-strata theory, directly map a sequence of words to the meaning. The derived CFG in our parser is relevant to processing but has no theoretical status in grammar representation. We can design a derived CFG for the sole purpose of maximizing the efficiency, by choosing features.

Though in different context, [34] observed an interesting fact which may shed light on the relationship between mono-strata theories and multi-strata ones. In this work, he made logical axiomatization of the GB theory [35], a multi-strata theory (surface structure and D-structure), as a logic program, and then applied a series of program transformation. In the end, he discovered that D-structure disappeared in the final program which maps PF (Phonetic form) directly to LF (Logical Form).

5 Parsing as a Search Problem

While a declarative grammar in CL defines a set of all possible interpretations, a parser in NLP has to choose the most or more plausible ones than the rest.

Pragmatics or constraints based on extra-linguistic knowledge may give further constraints on interpretations to narrow down possible interpretations. However, as many indicated, pragmatic clues are mostly preferential, not constraints [36–39]. In order for the whole scenario of natural language understanding based on pragmatic knowledge to work efficiently, a sentential parser should be very efficient in producing single or a limited number of plausible parses, which are to be checked their pragmatic plausibility. It should be able to do so with limited accesses to semantic or pragmatic resources [40].

While evidences show that the first phase of linguistic processing (i.e. before the conscious re-analysis phase) may not produce complete interpretations [41], the existence of garden path sentences indicates that deterministic processing of some sort, and thus selection of more plausible parsing paths than the rest, takes place during parsing in human processing before processing based on pragmatics [42, 43].

Preferential ranking can be captured by a probabilistic model. Since the probabilistic version of CFG (Probabilistic CFG) and Viterbi-type parsing algorithms for PCFG have been studied extensively, several attempts have been made to (1) reduce the cost of exhaustive viterbi-type algorithms of PCFG by clever pruning [44–46], and (2) to combine PCFG-parsing with parsing by feature-based grammar [47–49]. The research results in (1) show that clever pruning can drastically reduces the cost of parsing by reducing the total number of rule application with minimum search errors. This is important since each rule application in parsing for feature-based grammar involves expensive operation of unification.

The idea in (2) is rather simple. They used the CFG backbone, or a shallow grammar built independently of the feature-based grammar, to choose the (most) plausible derivation trees among many. Then they proceed to the stage of full unification. However, since the probabilistic model of the first phase is dissociated from feature-values in the original grammar, it failed to exploit rich information encoded in the original grammar.

Although a grammar in the declarative formalisms is not concerned with selection of plausible interpretation, this does not mean that the feature-value representation does not contribute to the selection. On the contrary, to pack them in single non-terminal symbols loses useful information for preference rating as we lose useful generalization in grammar representation. Preference cues are not monolithic but multi-layered in the same way as linguistic constraints are.

One possible solution is to build probabilistic models directly for feature-based grammar, and apply the same line of thought on PCFG to feature-based grammar. That is, one has to (1) build a probabilistic model for feature-based grammar formalisms, (2) estimate model parameters without combinatorial explosion of possible interpretations, and (3) develop efficient algorithms with clever pruning based on the probabilistic model.

Table 3. Viterbi parsing versus beam thresholding versus iterative parsing

	Precision	Recall	F-score	Avg. Time (ms)	No. of failed sentences
viterbi (none)	87.80%	87.52%	87.66%	94374	2
beam search (num+width)	88.63%	82.45%	85.43%	90	25
iterative (iterative)	87.40%	87.03%	87.21%	101	2

As for (1), log-linear model or maximum entropy models [50] are being successfully deployed for constructing discriminative models for grammars with many features [51–54]. It was also shown by [55], [56], and [3] that estimation of model parameters and selection of most plausible interpretations can be done by using packed structures (i.e. feature forests) in a way similar to a dynamic programming method used for PCFG.

Both [55] and [56] used simple exhaustive Viterbi-type of algorithms. They enumerate all possible interpretations in packed feature forests and then choose the most plausible interpretations. Though the packed representation reduces the computational cost, the efficiency of their algorithms is inherently limited by the inefficiency of exhaustive parsing. Since then, we have introduced to parsing based on a feature-based grammar (a grammar in HPSG: see [57] for the detail) many clever search strategies which were proven effective for PCFG-parsing. These include local and global beam search [4], iterative widening search [8], etc.

Table 3 shows the performances of our parsers with three different strategies [5]. While the exhaustive Viterbi showed the best performance in terms of quality, it was extremely slow. On the other hand, the beam search combined with iterative widening showed good performance in terms of both quality and efficiency. The search errors (i.e. errors caused by pruning of this method) are very small (0.45 %). Please also note tha the sentence length was limited to 14 words in these experiments because of inefficiency of Viterbi parsing.

6 Super-Tagging and Staged-Architecture

Whether search strategies such as beam search, iterative widening, etc. work effectively and efficiently is highly dependent on how good the FOM (Figure of Merit) is. When the parses with the highest value of the FOM are not the correct ones, they are counted as model errors (errors caused by the model). On the other hand, even if the correct ones would have the highest FOMs, the parsing paths which would reach to them would be pruned in early stages of search. This happens when we use beam search. These are counted as search errors. Furthermore, when we employ iterative widening, an inappropriate FOM tends to increase the number of iterative cycles and the efficiency deteriorates. The efficiency deterioration occurs especially when early stages of parsing assign wrong FOM values to partial parses of small constituents without much evidence.

Since we use a probabilistic model as the FOM, we examined how different types of probabilistic models affect the performance [6].

The one used in [6] is, in essence, the same as PCFG. That is, the inside probability of a mother node is computed based on the inside probabilities assigned to the daughters (See the formula 1).

(Model 1)

$$p_{model1}(T|\mathbf{w}) = p_0(T|\mathbf{w})\frac{1}{Z_{\mathbf{w}}}\exp\left(\sum_u \lambda_u f_u(T)\right) \tag{1}$$

$$Z_{\mathbf{w}} = \sum_{T'} p_0(T'|\mathbf{w})\exp\left(\sum_u \lambda_u f_u(T')\right)$$

$$p_0(T|\mathbf{w}) = \prod_{i=1}^{n} p(l_i|w_i)$$

The probabilities of smaller constituents are percolated up to those of larger ones in a bottom-up fashion. At the bottom of a parse are words. The probabilities of assignment of lexical entries to words are estimated without considering the context. Words are the smallest constituents. The effect of the context on lexical assignment to a word will be indirectly considered as the probabilities of larger constituents which include the word. Wrong assignments would be redressed at later stages when larger constituents are constructed.

This model would make errors in the very early stage of processing, i.e. lexical entry assignments, since lexical assignment probabilties do not take into consideration the context in which words occur. Such errors may lead to parsing failure and thus trigger another cycle of iteration (efficiency deterioration). In some cases, they may survive till the end inside the highest FOM (model or search errors).

Another FOM focuses on lexical assignment. This model estimates the probability of lexical assignment to a word by taking the local context of the word in a sentence. The model uses the words in local context of (+1/-1), the POS tags of the word in local context (+3/-2), their bi-grams, etc. (See [6] for the details). The model assigns the probability of a parse according to (2).

(Model 2)

$$p_{model2}(T|\mathbf{w}) = \prod_{i=1}^{n} p(l_i|\mathbf{w}, i) \tag{2}$$

The third FOM is the combination of these two models.

(Model 3)

$$p_{model3}(T|\mathbf{w}) = \frac{1}{Z_{model3}} p_{model2}(T|\mathbf{w})\exp\left(\sum_u \lambda_u f_u(T)\right) \tag{3}$$

$$Z_{model3} = \sum_{T'} p_{model2}(T'|\mathbf{w})\exp\left(\sum_u \lambda_u f_u(T')\right)$$

Table 4. Experimental results for Penn Treebank Section 23

	Section 23 ($\leq$ 40 + Gold POSs)				**Section 23 ($\leq$ 100 + Gold POSs)**			
	LP	**LR**	**F_1**	**Avg. Time**	**LP**	**LR**	**F_1**	**Avg. Time**
model 1	87.65	86.97	87.31	468 ms	87.26	86.50	86.88	604 ms
model 2	87.71	87.02	87.36	111 ms	87.23	86.47	86.85	129 ms
model 3	89.79	88.97	89.38	132 ms	89.48	88.58	89.03	152 ms
	Section 23 ($\leq$ 40 + POS tagger)				**Section 23 ($\leq$ 100 + POS tagger)**			
	LP	**LR**	**F_1**	**Avg. Time**	**LP**	**LR**	**F_1**	**Avg. Time**
model 1	85.33	84.83	85.08	509 ms	84.96	84.25	84.60	674 ms
model 2	85.26	84.31	84.78	133 ms	85.00	84.01	84.50	154 ms
model 3	87.66	86.53	87.09	155 ms	87.35	86.29	86.81	183 ms

The performances of these three models are shown in Table 4. As expected, the third model showed the best performance. Compared with the model 1, the model 3 improves the accuracy but also the efficiency significantly. This means that improvement in lexical assignment is one of the keys for efficient parsing by avoiding iteration cycles or backtracking. This confirms the observation by others research groups in different grammar formalisms that super-tagging significantly improved the efficiency of their parsers [58–62].

It is also interesting to see that the model 2 performed rather well. The model did not construct any probabilistic models for larger constituents, thus avoid complex computation of probabilities on feature-bundles. It chooses the parse with the highest FOM of (1.2) among all possible parses. Parsing in this model only ensures that a sequence of lexical entries assigned to words actually lead to legitimate parses. Furthermore, since the super-tagger in this model is one of the simplest, we can improve the performance by introducing rich features in context and using more sophisticated models of sequential lablling [63].

Compared with probabilistic models on trees or embedded feature structures, even sophisticated models for sequential tagging are much faster. Therefore, if we could improve the performance of super-tagging in terms of accuracy, a model centered a super-tagger without complex probabilistic models for larger constituents could be both efficient and accurate.

Figure 4 is a staged architecture for efficient HPSG parsing [7]. The first two stages are sequential tagging parts, POS and super-tagger. The super-tagger, a simple one as in the Model 2, produces a raked list of sequences of super-tags. The third stage is to filter out sequences of super-tags which do not lead to complete parses. This phase uses the CFG skeleton derived from the original HPSG. While the super-tagger only sees a local context of five words, the CFG parsing check whether a given sequence of supertags is globally consistent with the given grammar. The final phase is a deterministic shift-reduce parser which checks all constraints and re-constructs the derivation history (or construct the meaning representation). This staged architecture produces an extremely efficient parser by avoiding probabilistic models on feature-value pairs, and by restricting clever

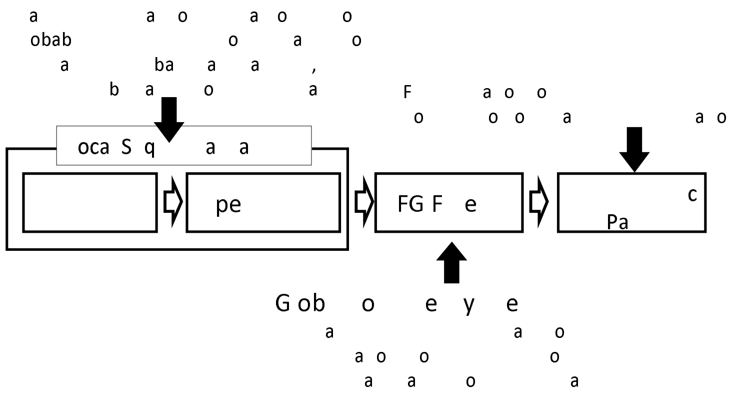


Fig. 4. Staged architecture with supertagging

Table 5. Results of parsing on Section 23

	Sentences ≤ 40 words				Sentences ≤ 100 words			
	LP	LR	F_1	Avg. Time	LP	LR	F_1	Avg. Time
model 1	85.33	84.83	85.08	509 ms	84.96	84.25	84.60	674 ms
model 3	87.66	86.53	87.09	155 ms	87.35	86.29	86.81	183 ms
staged architecture	87.15	86.65	86.90	25.9 ms	86.93	86.47	86.70	29.6 ms

search only to the early stage of POS tagger and super-tagger (See Table 5). Currently, we focus on improving the performance of super-tagger and thus that of the whole parser in terms of accuracy [9, 64].

7 Concluding Remarks

The declarative formalisms in CL give more freedom than procedural formalisms such as TG to the research on processing. Based on them, we can explore possible designs of architectures or algorithms for processing.

However, because of the multi-layered nature of linguistic constraints and the complex relationship between the surface (a sequence of words or phonemes) and the meaning, these formalisms for grammar representation also use formalism-specific descriptive devices with procedural flavor, such as feature unification, substitution and adjoining, function application and type-raising, etc.

Though they have clear semantics, these devices make the formalisms opaque compared with simple CFG. They may have caused confusion among researchers in NLP about representation and processing. The coufusion is not so conspicuous as initial confusion among psycholinguists caused by TG [65]. Parsing research, for example, had invested great effort on making unification efficient, while we show in this paper that most of unifications could be avoided or be replaced by the simple symbol equality checking. The computational cost of unification turned out to be not as crucial for efficient parsing as we had thought. We confused unification as a device for grammar representation with actual operation to be performed in parsing.

The surface differences in the formalisms also make comparison of grammars difficult. It is not trivial to see whether two grammar representations in different formalisms actually define the same language or not. Conversion from a grammar in LTAG to a HPSG-style grammar shows that grammars in different formalisms are not as different as they look. It also shows that such basic constructs in representation as elementary trees in LTAG can be dispensed with in an actual parser. Without explicit representation of elementary trees, the parser in Sect. 3 can preserve derivation trees and thus compute the meaning according to the original grammar. The parser is far more efficient than the one with explicit representation of elementary trees.

The relationship between grammar representation and processing is similar to the one between the computational level and the algorithmic level by [66]. The division would be useful from the practical point of view as well as from the conceptual one. It enables us to design processing algorithms in systematic ways, based on declarative grammar representation. Maintainability in representation and efficiency in processing will be achieved simultaneously. We can focus on efficiency in designing computational architectures and algorithms, while the completeness and soundness of grammar can be addressed at the level of grammar representation. We showed in this paper that the breakdown of a thoery at the computational level into the one of the algorithmic level is far from trivial. As the design of supertagger shows, it may involve information accesses which a theory of the computational level does not anticipate.

References

1. Oepen, S., Flickinger, D., Tsujii, J., Uszkoreit, H. (eds.): Collaborative Language Engineering. A Case Study in Efficient Grammar-based Processing. CSLI Lecture Notes. CSLI Publications, Stanford (2002)
2. Miyao, Y., Makino, T., Torisawa, K., Tsujii, J.: The LiLFes abstract machine and its evaluation with the LinGO grammar. Journal of Natural Language Engineering 6, 47–61 (2000)
3. Miyao, Y., Tsujii, J.: Feature Forest Models for Probabilistic HPSG Parsing. Computational Linguistics 34, 35–80 (2008)
4. Ninomiya, T., Tsuruoka, Y., Miyao, Y., Tsujii, J.: Efficacy of beam thresholding, unification filtering and hybrid parsing in probabilistic HPSG parsing. In: Proc. of IWPT 2005, pp. 103–114 (2005)
5. Ninomiya, T., Tsuruoka, Y., Miyao, Y., Tsujii, J.: Fast and Scalable HPSG Parsing. TAL 46 (2005)
6. Ninomiya, T., Matsuzaki, T., Tsuruoka, Y., Miyao, Y., Tsujii, J.: Extermely lexicalized models for accurate and fast HPSG parsing. In: Proc. of EMNLP 2006, pp. 155–163 (2006)
7. Matsuzaki, T., Miyao, Y., Tsujii, J.: Efficient HPSG Parsing with Supertagging and CFG-filtering. In: The Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (2007)

8. Tsuruoka, Y., Tsujii, J.: Iterative CKY parsing for probabilistic context-free grammars. In: Su, K.-Y., Tsujii, J., Lee, J.-H., Kwong, O.Y. (eds.) IJCNLP 2004. LNCS (LNAI), vol. 3248, pp. 52–60. Springer, Heidelberg (2005)
9. Zhang, Y., Matsuzaki, T., Tsujii, J.: A Simple Approach for HPSG Supertagging Using Dependency Information. In: The Proceedings of 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics, NAACL-HLT 2010 (2010)
10. Chomsky, N.: Syntactic Structures. Mouton, The Hague (1957)
11. Chomsky, N.: Aspects of the theory of syntax. MIT Press, Cambridge (1965)
12. Gazdar, G., Klein, E.H., Pullum, G.K., Sag, I.A.: Generalized Phrase Structure Grammar. Harvard University Press, Blackwell, Oxford, Cambridge (1985)
13. Bresnan, J.: Lexical Functional Syntax. Blackwell, Malden (2001)
14. Pollard, C., Sag, I.A.: Head-Driven Phrase Structure Grammar. University of Chicago Press, Chicago (1994)
15. Steedman, M.: The Syntactic Process. MIT Press, Cambridge (2000)
16. Abeilllé, A., Rambow, O.: Tree Adjoining Grammar: An Overview. In: Abeilllé, A., Rambow, O. (eds.) Tree Adjoining Grammars: Formalisms, Linguistic Analyses and Processing, pp. 1–68. CSLI Publications, Stanford (2000)
17. Sornlertlamvanich, V., Inui, K., Tanaka, H., Tokunaga, T., Takezawa, T.: Empirical support for new probabilistic generalized LR parsing. Journal of Natural Language Processing 6, 3–22 (1999)
18. Nederhof, M.J., Satta, G.: Probabilistic Parsing Strategies. Journal of ACM 53, 406–436 (2006)
19. Yoshinaga, N., Miyao, Y., Torisawa, K., Tsujii, J.: Parsing comparison across grammar formalisms using strongly equivalent grammars. TAL 44, 15–39 (2003)
20. Schabes, Y., Abeille, A., Joshi, A.: Parsing strategies with 'lexicalized' grammars: Application to tree adjoining grammars. In: The Proceedings of COLING 1988, pp. 578–583 (1988)
21. Joshi, A., Vijay Shanker, K., Weir, D.: The Convergence of Mildly Context-Sensitive Grammar Formalisms. Technical Report MS-CIS-90-01, University of Pennsylvania (1990)
22. Keller, B.: Feature Logics, Infinitary Descriptions and Grammars. CSLI Publications, Stanford (1994)
23. Yoshinaga, N., Miyao, Y.: Grammar conversion from LTAG to HPSG. WEB-SLS: the European Student Journal on Language and Speech (2002)
24. XTAG Research Group: A lexicalized Tree Adjoining Grammar for English. Technical Report IRCS-01-03, University of Pennsylvania (2001)
25. Sarkar, A.: Practical experiments in parsing using tree adjoining grammars. In: Proc. of the fifth TAG+, pp. 193–198 (2000)
26. Van Noord, G.: Head Corner Parsing for TAG. Computational Intelligence 10, 525–534 (1994)
27. Schieber, S.M.: An Introduction to Unification-Based Approaches to Grammar. Center for the Study of Language and Information, Stanford University, Stanford, CA (1986)
28. Torisawa, K., Nishida, K., Miyao, Y., Tsujii, J.: An HPSG parser with CFG filtering. Natural Language Engineering 6, 63–80 (2000)
29. Perreira, F.C.N., Wright, R.N.: Finite-State Approximation of Phrase-Structure Grammars. In: Roche, E., Schabes, Y. (eds.) Finite-State Language Processing. The MIT Press, Cambridge (1997)
30. Nederhof, M.J.: Practical Experiments with Regular Approximation of Context-Free Languages. Computational Lingusitics 26, 17–44 (2000)

31. Schieber, S.M.: Using restrictions to extend parsing algorithms for complex-feature-based formalisms. In: Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics (1985)
32. Carpenter, B.: The logic of typed feature structures. Cambridge University Press, Cambridge (1992)
33. Boullier, P., Benout, S.: Efficient and robust LFG parsing: SXLFG. In: Proc. of IWPT 2005, pp. 1–10 (2005)
34. Johnson, M.: Deductive parsing with multiple levels of representation. In: Proceedings of Association for Computational Linguistics, pp. 241–248 (1988)
35. Haegeman, L.: Introduction to Government and Binding Theory. Blackwell Textbooks in Linguistics. Wiley-Blackwell (1991)
36. Fass, D., Wilks, Y.: Preference semantics, ill-formedness, and metaphor. Computational Lingusitics 9 (1983)
37. Hobbs, J.R., Bear, J.: Two Principles of Parse Preference. In: Proceedings of COLING 1990, pp. 162–167 (1990)
38. Hobbs, J.R., Stickel, M.E., Appelt, D.E., Martin, P.A.: Interpretation as Abduction. Artificial Intelligence 63, 69–142 (1993)
39. Bednarek, M.: Semantic preference and semantic prosody re-examined. Corpus Linguistics and Linguistic Theory 4, 119–139 (2008)
40. Fodor, J.: The modularity of mind. MIT Press, Cambridge (1983)
41. Ferreira, F., Christianson, K., Hollingworth, A.: Misinterpretations of Garden-Path Sentences: Implications for Models of Sentence Processing and Reanalysis. Journal of Psycholinguistic Research 30 (2001)
42. Abney, S.: A computational model of human parsing. Journal of Psycholinguistic Research 18, 129–144 (1989)
43. Weinberg, A.: Parameters in the theory of sentence processing: Minimal Commitment theory goes east. Journal of Psycholinguistic Research 22, 339–364 (1993)
44. Chitrao, M.V., Grishman, R.: Statistical parsing of messages. In: Proceedings of the DARPA Speech and Natural Language Workshop, pp. 263–266 (1990)
45. Charniak, E., Goldwater, S., Johnson, M.: Edge-based best-first chart parsing. In: Proceedings of the Sixth Workshop on Very Large Corpora, pp. 127–133 (1998)
46. Klein, D., Manning, C.D.: A* Parsing: Fast Exact Viterbi Parse Selection. In: HLT-NAACL (2003)
47. Kasper, W., Krieger, H.U., Spilker, J., Weber, H.: From Word Hypotheses to Logical Form: An Efficient Interleaved Approach. In: Proceedings of Natural Language Processing and Speech Technology. Results of the 3rd KONVENS Conference, pp. 77–88 (1996)
48. Briscoe, T., Carrol, J.: Generalized probabilistic LR-parsing of natural language (corpora) with unification grammars. Computational Linguistics 9 (1993)
49. Kiefer, B., Krieger, H.U., Prescher, D.: A Novel Disambiguation Method For Unification-Based Grammars Using Probabilistic Context-Free Approximations. In: Proc. of COLING 2002 (2002)
50. Berger, A., Pietra, S.D., Pietra, V.D.: A Maximum Entropy Approach to Natural Language Processing. Computational Linguistics 22, 39–71 (1996)
51. Riezler, S., Prescher, D., Kuhn, J., Johnson, M.: Lexicalized Stochastic Modeling of Constraint-Based Grammars using Log-Linear Measures and EM Training. In: Proc. of ACL 2000, pp. 480–487 (2000)
52. Malouf, R., van Noord, G.: Wide Coverage Parsing with Stochastic Attribute Value Grammars. In: Proc. of IJCNLP 2004 Workshop Beyond Shallow Analyses (2004)

53. Kaplan, R.M., Riezler, S., King, T.H., III, J.T.M., Vasserman, A.: Speed and accuracy in shallow and deep stochastic parsing. In: Proc. of HLT/NAACL 2004 (2004)
54. Miyao, Y., Tsujii, J.: Probabilistic dismabiguation models for wide-coverage HPSG grammar. In: Proc. of ACL 2005, pp. 83–90 (2005)
55. Geman, S., Johnson, M.: Dynamic programming for parsing and estimation of stochastic unification-based grammars. In: Proc. of ACL 2002, pp. 279–286 (2002)
56. Miyao, Y., Tsujii, J.: Maximum Entropy Estimation for Feature Forests. In: Proc. of HLT 2002, pp. 292–297 (2002)
57. Miyao, Y., Ninomiya, T., Tsujii, J.: Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the penn treebank. In: Su, K.-Y., Tsujii, J., Lee, J.-H., Kwong, O.Y. (eds.) IJCNLP 2004. LNCS (LNAI), vol. 3248, pp. 684–693. Springer, Heidelberg (2005)
58. Bangalore, S., Joshi, A.: Supertagging: An approach to almost parsing. Computational Linguistics 25, 237–265 (1999)
59. Clark, S., Curran, J.R.: The importance of supertagging for wide-coverage CCG parsing. In: Proc. of COLING 2004 (2004)
60. Clark, S., Curran, J.R.: Parsing the WSJ using CCG and log-linear models. In: Proc. of ACL 2004, pp. 104–111 (2004)
61. Shen, L., Joshi, A.K.: A SNoW based supertagger with application to NP chunking. In: Proc. of ACL 2003, pp. 505–512 (2003)
62. Zhang, Y., Ahn, B., Clark, S., Wyk, C.V., Curran, J.R., Rimell, L.: Chart Pruning for Fast Lexicalised-Grammar Parsing. In: The Proceedings of 23rd International Conference on Computational Linguistics (COLING 2010), pp. 1471–1479 (2010)
63. Kaji, N., Fujiwara, Y., Yoshinaga, N., Kitsuregawa, M.: Efficient Staggered Decoding for Sequence Labeling. In: The Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010), pp. 485–494 (2010)
64. Zhang, Y., Matsuzaki, T., Tsujii, J.: Forest-guided Supertagger Training. In: The Proceedings of 23rd International Conference on Computational Linguistics, COLING 2010 (2010)
65. Sag, I.A., Wasow, T.: Performance-Compatible Competence Grammar. In: Borsley, R., Borjars, K. (eds.) Non-Transformational Syntax. Blackwell, Cambridge (in press)
66. Marr, D.: Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. Freeman, New York (1982)

An Unsupervised Approach for Linking Automatically Extracted and Manually Crafted LTAGs

Heshaam Faili and Ali Basirat

Department of ECE, University of Tehran, Tehran, Iran
hfaili@ut.ac.ir
Department of Computer Engineering, Islamic Azad University, Science and Research, Branch of Tehran, Iran
a.basirat@srbiau.ac.ir

Abstract. Though the lack of semantic representation of automatically extracted LTAGs is an obstacle in using these formalism, due to the advent of some powerful statistical parsers that were trained on them, these grammars have been taken into consideration more than before. Against of this grammatical class, there are some widely usage manually crafted LTAGs that are enriched with semantic representation but suffer from the lack of efficient parsers. The available representation of latter grammars beside the statistical capabilities of former encouraged us in constructing a link between them.

Here, by focusing on the automatically extracted LTAG used by MICA [4] and the manually crafted English LTAG namely XTAG grammar [32], a statistical approach based on HMM is proposed that maps each sequence of former elementary trees onto a sequence of later elementary trees. To avoid of converging the HMM training algorithm in a local optimum state, an EM-based learning process for initializing the HMM parameters were proposed too. Experimental results show that the mapping method can provide a satisfactory way to cover the deficiencies arises in one grammar by the available capabilities of the other.

Keywords: Supertagging, HMM Initialization, XTAG Derivation Tree, Semantic Representation, Grammar Mapping, Automatically Extracted Tree Adjoining Grammar, MICA.

1 Introduction

Tree Adjoining Grammar (TAG) introduced by Joshi [17] as a Mildly Context Sensitive Grammar is supposed to be powerful enough to model the natural languages [18]. In Lexicalized case (Lexicalized Tree Adjoining Grammar, LTAG,), any elementary tree of a TAG could be considered as a complex description of the lexical item that provides a domain of locality, which specifies syntactic and semantic dependencies [2]. Lexicalization, Extended Domain of Locality (EDL) and Factoring of Recursion from the Domain of dependencies (FRD) are three important keys of using this formalism in NLP applications and theorems.

Regarding the LTAG creation methodologies, LTAGs could be classified into two major classes, manually crafted LTAGs and automatically extracted LATGs [14]. The first class, manually crafted, contains the grammars that are developed by some experts and are enriched with some representations like feature structures, e.g., English XTAG grammar [32]. The difficulties and time consuming of developing these grammars inclined some researchers to develop some methods to extract LTAG from a corpus automatically [9, 30]. In [23, 9, 30, and 24] some statistical models are proposed that extract LTAG from tree banks automatically. These models take syntactic structure of a corpus and produce an LTAG as output by hiring some statistical and machine learning algorithms (e.g., Chen's grammar for English [9], Xia's grammar for English [30], Habash's grammar for Arabic [15], and Park's grammar for Korean [24]). The coverage of these grammars is strictly bounded by the training corpus.

In spite of the all capabilities of automatically extracted and manually crafted LTAGs, there are some serious obstacles in using them in real world applications. But, fortunately, the observed weaknesses in each of these classes are against the capabilities of the other. For instance, though there is no applicable parsing algorithm for manually crafted LTAGs, due to their statistical nature, automatically extracted grammars have a lot of potential to be used by some powerful statistical parsers such as Statistical LTAG parser [28] and MICA [4]. On the other hand, despite there is no semantic representation for automatically extracted LTAGs, the manually extracted LTAGs are enriched with semantic representation due to their manually creation.

Here, we are going to rectify some of these deficiencies by utilizing their relevant capabilities in the other grammar. Our focus is on the English XTAG grammar [32] as the manually crafted grammar in one side and the automatically extracted grammar used by the MICA parser [4] in the other side. This grammar, MICA's, has been automatically extracted from Penn Treebank using Chen's approach [9]. The detail information about XTAG and MICA grammar are presented in section 2.

The main idea is based on mapping each elementary tree sequence of one grammar onto an elementary tree sequence of another grammar. The usefulness of the mapping can be considered by the three different ways. First, mapping elementary trees of automatically extracted grammars which represent the syntactical structure of a word, onto an elementary tree of a hand crafted grammar, can enrich the automatically extracted grammar with the semantic representation of hand written elementary trees. Second, this mapping enables the XTAG based applications to use the statistical parsers related to the automatically extracted grammar instead of XTAG parser. Third, it provides a way to evaluate the existence gap between in the manually crafted grammar or vice versa.

In this paper, we have defined the mentioned mapping as a sequence-tagging problem to give the best mapping solution regarding the local and non-local information of each elementary tree. Here an unsupervised sequence tagger based on Hidden Markov Model (HMM) has been presented that produces an XTAG elementary tree sequence given an automatically extracted elementary tree sequence.

The layout of this paper is as follow. Next section introduces the grammars in more detail regarding the mentioned deficiencies and the relevant capabilities. In section three, we present a brief history of previous works have been done on this subject. In section 4 the system architecture of the solution is presented that clarifies how to use the model in real world applications. Problem modeling and other related works for training the HMM is presented in section 5. An innovative way for finding a suitable initial state of HMM parameters is given too. Finally in section 6 after giving an outline about the numerical experiment, the models accuracy is calculated.

2 XTAG vs. MICA

XTAG grammar [32] is a well known and large scale manually crafted LTAG for English that has been hand-crafted at university of Pennsylvania since 1990. It contains 1226 elementary trees that provide good syntactic as well as semantic constraints over the words due to their manually creation. The linguistic knowledge, such as features representation, embedded in XTAG grammar caused it widely get used in many TAG based applications such as parsing, machine translation, information retrieval, generation, semantic interpretation and summarization applications [32, 12, 13].

Despite the widely usage of XTAG grammar, there is no related efficient parser yet. The current available XTAG parser[1] uses the chart-based head-corner parsing algorithm mentioned by Van Noord [29] that runs in $O(n^6)$ time complexity. Several ambiguities in the resulted parse trees beside the low speed of this method caused applications to face big problems and subsequently encouraged some researchers to solve them [2, 3, 13].

Though the history of progress in supertagging accuracy is clearly evidenced on the success of statistical methods in this area, the lack of statistical information for XTAG grammar was an obstacle in using these methods for XTAG based supertagging [2, 3, 13]. Under the circumstance some researchers, e.g. [9], tried to extract a LTAG that is almost compatible with XTAG grammar and contains statistical information. It was expected that these XTAG-like automatically extracted grammars be more compatible with statistical supertagging solutions due to their statistical nature [9].

In [9] a XTAG-like LTAG contains some statistical information about the grammar was extracted from Penn Treebank. This model was a limited version of TAG namely Tree Insertion Grammar (TIG) [27] that didn't absolutely obey TAG formalisms. The compatibility of the grammar with XTAG grammar and then the emergence of MICA [4] as a statistical and powerful parser that was developed to work with it caused the formalism has been taken into consideration. MICA (stand for Marseille-INRIA-Columbia-AT&T) is a dependency parser that returns the deep dependency representations of the given sentence, have been created since 2009 [4]. In supertagging, MICA uses the set of Chen's automatically extracted supertags [9]. The output of supertagger then is given as an input to an Earley-like parser based on PCFG generated by SYNTAX [6].

Like the other automatically extracted LTAGs, MICA's grammar also suffers from the huge size of elementary tree set and sparse data problem [9]. The total number of XTAG single anchor elementary trees over the English Penn Treebank is about 500 whereas this number for the MICA's grammar is about 4700 elementary

[1] Updated version available at `http://www.cis.upenn.edu/~xtag`

trees. Enriching these huge size grammars with semantic representation needs a large humanly efforts. Nevertheless, this is an unavoidable task since this mapping enables it to be used in many areas of NLP and computational linguistic applications.

3 Related Works

Bridging between English XTAG and some other English grammars was considered by researchers in the last two decades. The significance of this work is due to the fact that English XTAG is a fairly comprehensive and ready to use grammar for English in compare to many of other grammars that are not as comprehensive as it is. The most important reasons that make it necessary to build such a bridge can be considered as follow: 1) sparseness and the lack of semantic representation of automatically extracted LTAGs, 2) obtaining a linguistically sound grammar for some efficient parsers such as HPSG parser [20] and MICA parser [4], 3) increasing the syntactic coverage of some lexicalized resources such as VerbNet [19, 10], and 4) finding the overlap between an automatically extracted LTAG and XTAG grammar [31].

Due to enriching an automatically extracted LTAG with semantic representation, in [9] a conversion method has been proposed that maps an individual XTAG elementary tree onto an automatically extracted elementary tree using a heuristic rule-based classification procedure. The outline of Chen's method [9] is based on two transformations, *node local transformation* and *structural transformation*. In *node local transformation* the minor changes between source and target formatting is alleviated. In *structural transformation* also the syntactic differences between automatically extracted elementary trees and the output of *node local transformation* phase was diminished. Using these transformations, many of most frequently used XTAG elementary trees, about 4% of overall tree frames and about 30% of tree frames anchored by verbs, were mapped onto the elementary trees of automatically extracted grammar.

Xia and Palmer [31] also used a similar rule-based method to calculate the overlap between XTAG and the LTAG that was automatically extracted from Penn Treebank using Xia's approach [30]. They defined two type of matching named *t-match* and *c-match*. Two elementary trees were considered to be *t-match* if they had the exactly same structure barring the type of information present in one grammar such as feature structure in XTAG. In *c-match* also an elementary tree was decomposed into three parts: *subcat frame*, *subcat chain* and *modification pair*. Two elementary trees were considered *c-match*, if they were decomposed into same tuple [31]. Regarding their work, two elementary trees of XTAG and the automatically extracted LTAG are considered match if they satisfy one of *t-match* or *c-match* rules. They reported 82.1% of accuracy in the matching procedure regarding *t-match* and *c-match* types.

In [26] the syntactic coverage of VerbNet [19, 10], a domain independent of verb lexicon with explicit syntactic and semantic information for English, was extended by incorporating the coverage of XTAG grammar, accounting for possible transformation of each declarative frame. Due to the limitation of the VerbNet's syntactic coverage that describes only declarative frame for each syntactic construction or alternation, mapping syntactic information of this resource to XTAG elementary trees could increase the robustness of it by capturing possible transformations of each frame [26]. Moreover, the semantic predicates presented in the VerbNet could be used to disambiguate the XTAG verb senses regarding this mapping [26]. From 196 VerbNet's frame, all but 18 correspond exactly to 16 of 57 XTAG elementary trees by doing manually mapping between XTAG trees and VerbNet's frames, they reported.

In [14] also a statistical approach was proposed that the automatically extracted grammar used by MICA parser [4] was enriched by the semantic representation of XTAG grammar. The mentioned mapping problem was reformulated as a sequence tagging problem and modeled by an HMM. By manually checking a test corpus randomly selected from Penn Treebank, about 82% of accuracy in which XTAG elementary trees were correctly assigned to MICA elementary trees given a sequence of MICA elementary trees was reported.

Here, by modeling the problem at issue as a statistical problem we have closely followed the proposed approach in [14]. Making some modifications in training algorithm as well as using comprehensive data for training phase makes the model to be more accurate and also simpler than the proposed model in [14]. The simplicity of the model is due to skipping the complicated smoothing method and embedding it in training algorithm. Another important difference between this solution and the solution available in [14] is in the evaluation function used in training algorithm that we will express it in detail in section 5.

It is expected that the significant improving in both accuracy (about 88% against 82% over WSJ) and complexity of the model in compare to [14] can enable the model to be used by researchers and developers.

4 HMM-Based LTAGs Mapping

As originally was introduced in [14], the mapping problem can be reformulated as a sequence tagging in which any target elementary tree is strictly depends on the local and non-local information of a sequence of source elementary tree. For the sake of simplicity we will use P as the MICA parser, G as the MICA grammar, P' as the XTAG parser and G' as the XTAG grammar. The statistical formulation of the problem could be as below:

Given a sequence of elementary trees $T=(t_1,t_2,\ldots,t_n)$ $t_i \in G$ assigned to sentence $S=(w_1,w_2,\ldots,w_n)$ by P, tag each member of T with an isolated elementary tree $t_i' \in G'$ such that the likelihood of $T' = (t_1', t_2', \cdots, t_n')$ given T and S be maximized.

The sequence classification problem can statistically be modeled using a Hidden Markov Model [14]. HMM is a statistical model that is used in modeling a Markov process with unobserved state. Here we have used a first order HMM to model the mentioned problem. The initializing and training phases of the HMM also have been done in an unsupervised fashion

Figure 1 shows the basic architecture of the model to be used as XTAG-supertagger or in desired state as XTAG-parser given the trained HMM λ and MICA parser. Regarding this graph, creation a XTAG derivation tree[2] for a given sentence is

[2] A derivation tree represents the process of combining elementary trees to yield a parse for the sentence.

decomposed into two main steps. At first the sentence will be supertagged using *MICA* and the *HMM-Based Tagger* to get a XTAG elementary tree sequence and then by combining the associated supertags regarding the dependency of the sentence's lexemes the XTAG based derivation tree is generated. Here the *HMM-Based Tagger* is an implementation of Viterbi's algorithm; the standard solution for the second issue raised in literature that deals with finding most likely hidden state path in which maximizes the probability of the observation sequence given the hidden state path and HMM.

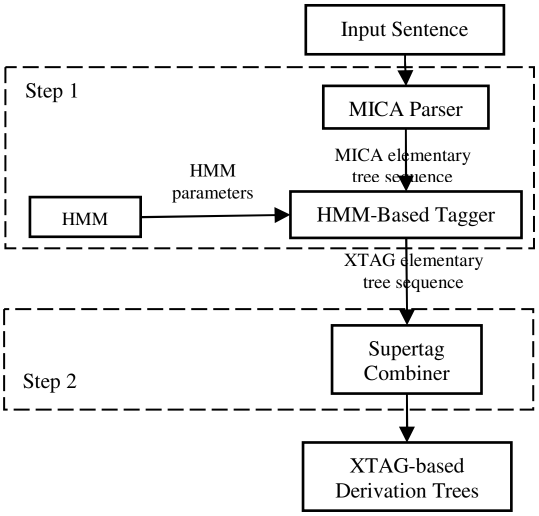


Fig. 1. The basic architecture of model to be used as XTAG supertagger or XTAG parser

5 Problem Modeling Using HMM

HMM (Hidden Markov Model) as a statistical model for modeling a Markov process with unobserved states is widely getting used in many applications of temporal pattern recognitions such as speech recognition, hand written recognition and part of speech tagging. Like the other corpus based methods, HMM also is a successful classifier while it doesn't use the complexity of classification problem. The success of the corpus based taggers like HMM, is due to this fact that linguistic phenomena can often be observed trough epiphenomena [7]. In HMM states are not visible while the output depends on the states are visible.

Regarding the HMM formulation of the problem presented in [14], the problem can be modeled by considering each XTAG elementary tree as a hidden state of the model and each MICA elementary tree as an observation symbol. The observation and the state transition probabilities also were considered as the probability of observing each MICA elementary tree given a XTAG elementary tree and the probability of seeing each XTAG elementary tree after the other XTAG elementary trees, respectively.

5.1 Training the HMM

Training as the adjusting the model parameters (A, B and ∏) to maximize the probability of the observation sequence is the most difficult problem arises in the literature. The current standard solutions, *Baum-Welch* or *Baldi-Chauvin* [1] algorithms, can efficiently find a local optimum λ= (A, B and ∏) that maximize P(O| λ) [25]. These algorithms inherit this weakness from the HMM in which does not provide any clear solution to use the extra information of problem context. In this case, the initial state of training algorithm provides a way to use a part of environment's knowledge that can largely enhance the mentioned weakness [25].

At issue, here, after initializing the model by using an innovative EM-based approach, the HMM was trained by Baum-Welch algorithm. The unsupervised EM-based initializing method was used for incorporating the available knowledge about the mapping between MICA and XTAG elementary trees and for avoiding the Baum-Welch algorithm of converging to a local optimal solution. In the next part, an outline of data preparation for initialization and training the model is presented. The introduced EM-based initialization procedure is expressed in detail too.

Data Preparation. Like the other statistical models, HMM also needs a large-scale training corpus for both training and initialization. In initialization phase, as will be given later, having a set of English sentences that can be parsed by both of MICA and XTAG parsers is required; let us name this corpus as *Initialization Database (IDB).* Training also must be done over an enough large MICA elementary tree sequence data set, namely *Train Database (TRDB)*, achieved from parsing some English sentences using MICA parser;

Let C and C' be two sets of elementary tree sequences were utilized in a derivation tree obtained from parsing IDB using MICA and XTAG parsers respectively. The order of elementary trees in each of C or C' sequences are same as the order of their anchors in corresponding sentence in IDB. Since neither of these parsers produces a unique derivation tree for a given sentence, we applied some restrictions over their output.

While due to its probabilistic architecture, MICA has the ability to score its resulted derivation trees; XTAG does not provide any solution for the ambiguity in parser results. Among the whole elementary tree sequences achieved from parsing a sentence in IDB, C contains the most probable ones. But, C' contains all but some XTAG elementary tree sequences produced by the parser that contains at least one multi anchor elementary trees regarding this fact that MICA's grammar does not contain any multi anchor elementary trees.

To summarize, given a sentence S_i in IDB, C contains its related most probable MICA elementary trees sequence $T_i = (t_{i,1}, \cdots, t_{i,n})\ t_{i,k} \in G$ and C' contains a set of XTAG elementary tree sequences $\xi_i = \{T'_{i,j} | T'_{i,j} \in C'\}$ in which each member of ξ_i is a sequence of XTAG elementary trees $T'_{i,j} = (t'_{i,j,1}, \cdots, t'_{i,j,n})\ t'_{i,j,k} \in G'$.

Initializing the HMM Parameters. The simplest and most intuitive way to estimate the mentioned HMM parameters is using Maximum Likelihood Estimation (MLE) method on the annotated corpora C and C'. Despite the simplicity of the MLE, the existence ambiguity in C' would mislead the MLE method. To reduce the ambiguity size of C', we have used a specialized kind of expectation maximization (EM) algorithm [11] to score each XTAG elementary tree sequence indicating its compatibility with the related MICA elementary tree and other sequences of C'.

Here, we are going to estimate the missed HMM parameters A, B and ∏ that maximizes the likelihood of observing C regarding the implicit knowledge of the HMM about the given mapping between XTAG and MICA sequences in C and C'. Formally, we are going to estimate HMM parameters λ such that the probability

$$P(C|\lambda) = P(T_1, \cdots, T_m\, |\lambda) \tag{1}$$

is being maximized. The probability (1) can easily be estimated using forward algorithm, the standard algorithm for the first problem raised in the literature.

In the EM formulation, E-step was defined as the HMM parameters estimation and were approximated by MLE with some consideration. Taking count in the MLE has been done by considering the score of each sequence of C'. The mentioned score shows how well a XTAG elementary tree sequence is compatible with its related sequence in C and other sequences in C'. Therefore, the score of a sequence $T'_{i,j} \in \xi_i$ could be defined as the joint probability of observing it and its related sequence $T_i \in C$ given the current values of HMM parameters (λ) normalized by the number of sequences in ξ_i ($|\xi_i|$). Due to the variation of HMM parameters in each iteration of EM-based initialization, the scores of each sequences in C' may vary too. The initial score of each sequence in C' is first considered to be uniform distribution of probability over each member of ξ_i. Then, regarding the variation of HMM parameters, these scores will be vary. Eq. (2) shows the scoring function:

$$\Im\left(T'_{i,j}\right) = \begin{cases} \frac{1}{|\xi_i|} & iter = 0 \\ \frac{P\left(T'_{i,j}, T_i|\lambda\right)}{|\xi_i|} & iter \neq 0 \end{cases}. \tag{2}$$

Given the *ith* MICA elementary tree sequence $T_i = \left(t_{i,1}, t_{i,2}, \cdots, t_{i,n}\right)$ in C and its *jth* related XTAG elementary tree sequence $T'_{i,j} = \left(t'_{i,j,1}, t'_{i,j,2}, \cdots, t'_{i,j,n}\right)$ in ξ_i, the observation probabilities could be estimated by taking weighted-count from the set:

$$\tau = \left\{\left(t_{i,k}, t'_{i,j,k}\right) | i \leq |C|, j \leq |\xi_i|, k \leq |T_i|\right\} \tag{3}$$

and normalizing them by the weighted sum of all observed pair $\left(t_x, t'_{i,j,k}\right) \in \tau$ that share the same second XTAG elementary tree as shown in eq. (4):

$$P\left(t_{i,j}|t'_{i,k,j}\right) = \frac{C^w\left(t_{i,j}, t'_{i,k,j}\right)}{\sum_x C^w\left(t_{x,j}, t'_{i,k,j}\right)} \tag{4}$$

Here $C^w\left(t_{i,j}, t'_{i,k,j}\right)$, weighted-count, is the count of pair $\left(t_{i,j}, t'_{i,k,j}\right)$ in τ, weighted by the score of the related XTAG elementary tree sequence that $t'_{i,k,j}$ is one of its constituents, $\Im\left(T'_{i,j}\right)$.

The state transition probabilities also could be estimated by computing the weighted-count of bigram $C^w\left(t'_{i-1}, t'_i\right)$ from any XTAG elementary tree sequence $T' = \left(t'_1, t'_2, \cdots, t'_n\right)$ member of C' and normalizing by the sum of all bigrams that share the same first elementary tree, as shown in eq. (5).

$$P(t_i'|t_{i-1}') = \frac{c^w(t_{i-1}', t_i')}{\sum_k c^w(t_{i-1}', t_k')} . \quad (5)$$

The prior probabilities also could be estimated more easily. The prior probability matrix elements could be estimated by taking weighted-count from the elementary tree t_1' at beginning of any sequence $T' = (t_1', t_2', \cdots, t_n')$ in C' and normalizing them by the sum of all scores related to C' sequences. Eq. (6) shows how to compute the prior probability of a given XTAG elementary tree.

$$P(t_1') = \frac{c^w(t_1')}{\sum_{i \leq |C'|} \Im(T_i')} \quad (6)$$

Another constituent step of the EM formulation, M-step, was considered as updating the model parameters such that the probability value in eq. (1) increases. Regarding this fact that HMM parameters are strongly depend on the values of scoring function, the only thing that should be done to increase the value of eq. (1) is estimating the new score values according to the earlier HMM.

Fig. 2 summarizes the main steps of initialization phase. In each iteration it estimates the HMM parameters using weighted-MLE (E-step) and re-estimate the new score values of C' sequences regarding the earlier HMM (M-step). It iterates until the score values of C' converge to a stable state or it exceeded predefined maximum number of iterations.

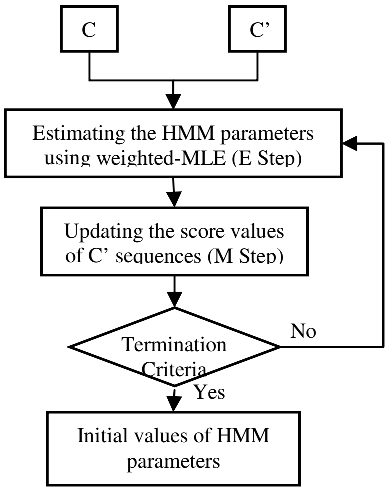

C
C'
Estimating the HMM parameters using weighted-MLE (E Step)
Updating the score values of C' sequences (M Step)
Termination Criteria
No
Yes
Initial values of HMM parameters

Fig. 2. The EM-based Initialization process for initializing the HMM parameters

6 Numerical Illustration

To evaluate the mapping accuracy of the proposed system, the model were initialized and trained with three real world data sets include ATIS [16], IBM Manual and WSJ [21] corpora. Table 1 summarizes some statistical information related to each randomly selected subset of mentioned corpora used in initialization, training and testing the HMM. To guarantee valid results for making prediction regarding new data, the test data sets, namely TSDBs, were further randomly selected from each related corpus independent of the IDBs and TRDBs. However, in all of these cases, IDBs were selected as a subset of their related TRDBs.

Our implementation was carried out on the MATLAB 7.6 development environment by extending the HMM toolbox written by Kevin Murphy [22]. The empirical evaluation was performed on the Intel Core™ 2 Duo running at 2.2 GHz and 3 GB RAM.

Table 1. Some statistical information about initialization, training and testing dataset

	No. Sentences			No. Lexemes			Avg. Sen. Length			Max. Sen. Length		
	IDB	TRDB	TSDB	IDB	TRDB	TSDB	IDB	TRDB	TSDB	IDB	TRDB	TSDB
ATIS	904	1291	12	8612	17020	134	9	13	11.16	20	86	18
IBM	3463	9754	71	36387	154917	873	10	15	12.2	17	58	19
WSJ	11913	21709	198	113883	221337	2039	9	10	10.3	14	21	16

6.1 The Initializing Effect

The aforementioned initialization method was applied over three IDB's, ATIS-IDB, IBM-IDB and WSJ-IDB. Fig. 3 shows how well the model enhanced itself while running the HMM initializing method. It represents the logarithm value of eq. 3, normalized by the number of sentences in each IDB during running the initialization method.

Reviewing the effect of initial state on the Baum-Welch training algorithm as another way for showing the quality of the HMM initial state is summarized in table 2. It shows how well the initial state can converge the HMM to a better solution than when it is trained with other randomly selected initial states. Due to this fact that Baum-Welch is strongly dependent on the HMM initial's state, a good initial state can avoid of stopping algorithm in local optimum state. As it shows from the five experiments that have been carried out for training the HMM over each mentioned TRDB, the training process that is started with the proposed initial state, the *EM* column of each TRDB categories, gets a better response than when it was started with other randomly selected initial states, the four *random* columns of each TRDB categories.

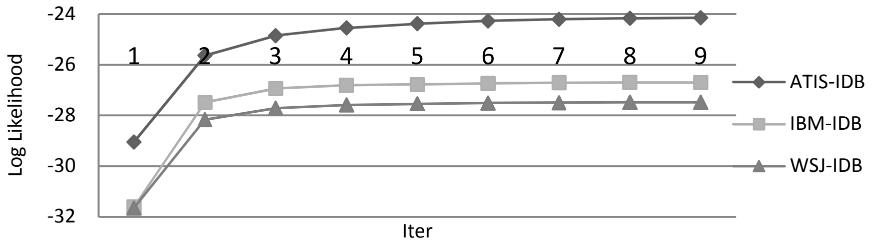


Fig. 3. Increasing the normalized value of P(C|λ) while running the HMM initialization method

Table 2. The negative value of logarithm likelihood of observing TRDBs while training the HMMs regarding their relevant initial states

×-1	**ATIS-TRDB**					**IBM-TRDB**					**WSJ-TRDB**				
Iter.	Random				EM	Random				EM	Random				EM
1	143228	143071	143167	143257	**52397**	1310790	1310849	1310790	1310790	**478280**	1862464	1863074	1864440	1863123	**673492**
2	56290	56280	56303	56282	**42301**	545387	545396	545387	545387	**404000**	804452	804216	804219	804413	**606253**
3	56069	56046	56078	56054	**41606**	544138	544153	544138	544138	**399605**	802300	802008	801912	802204	**596898**
4	55707	55675	55684	55694	**41269**	542331	542389	542331	542331	**397236**	799048	798804	798525	798905	**591915**
5	55131	55122	55056	55136	**40974**	539670	539843	539670	539670	**395411**	794170	793902	793560	794054	**588348**
6	54396	54441	54329	54422	**40835**	536080	536409	536080	536080	**394075**	788027	787213	787259	787858	**585622**
7	53564	53627	53506	53603	**40698**	531432	531974	531432	531432	**392958**	781647	780926	780308	780926	**583566**
8	52410	52429	52333	52451	**40445**	524274	525397	524274	524274	**392111**	774719	771456	771903	772534	**582135**
9	50671	50681	50700	50773	**40322**	510952	513470	510952	510952	**391400**	764814	759079	758881	759539	**580955**
10	48242	48274	48326	48398	**40230**	488795	492899	488795	488795	**390925**	747794	737006	737136	737756	**579861**

6.2 Model's Evaluation

We have defined two type of evaluation for evaluating the model at issue, namely, *lexical agreement, and tagging accuracy*. In lexical agreement, it was measured how well the model observes the agreement between the anchor of both source and target elementary trees. In the other words, linguistically, can the target elementary tree be assigned to the anchor of the source elementary tree independent of the contextual environment of the anchor? In *tagging accuracy* also the model's response over a test corpus was evaluated.

The *lexical agreement* evaluation was done over the all of mentioned test datasets. Table 3 shows the result of the evaluation over the prepared test sets. Certainly, by considering the contextual environment, lexical agreement doesn't guarantee that the assigned XTAG elementary tree to a MICA elementary tree (consequently to its related word in input sentence) is correct. For instance, while it gives a considerable value over ATIS-TSDB, as we will say later, the next evaluation criterion is not as acceptable as it is.

As it shows, regarding this fact that the values of *lexical agreement* over each test sets are near to each other, it can be concluded that the model's response about this evaluation measurement is not depend on the size of training set nor the nature training sentences.

Tagging accuracy shows the model's accuracy comparing with a gold tagged corpus. In *tagging accuracy* the model is considered as a supertagger that tags each words of an input sentence with its proper XTAG elementary tree (supertag). To give more statistical information about the evaluation, depend on the average length of sentences in each datasets, TSDBs was divided into two parts. P1 includes all sentences of a test dataset with length smaller than the average, and P2 includes the other sentences. Table 4 gives the result of manually checking the models response over the mentioned test sets, *tagging accuracy*. As it shows, the model's response is not that much depends on the sentence length. Because, as can be seen, the achieved accuracies in both of P1 and P2 are very close to each other regarding the test sets. Moreover, the relatively large distance between the *ATIS-TSDB tagging accuracy* and the others can be interpreted by this assumption that the measurement is strictly dependent on the size of training data. Since, though figure 3 shows that *ATIS-TRDB* is a better training history in compare to both of *IBM-TRDB* and *WSJ-TRDB*, it gets some lower values for *tagging accuracy*. The assumption about the invert dependency between size of training data and *tagging accuracy* is observable in the model's response for *IBM* and *WSJ*. However, as it was mentioned before, the assumption is not true for the *lexical agreement.*

Figure 4 shows the distribution of error counts in a given sentence in TSDBs. It shows how a sentence is susceptible to issue errors regarding its dataset. For instance, the two most likely error counts that a *WSJ-TSDB* sentence may contains are *zero* or *one*. And in *ATIS-TSDB*, the most likely error count that may issue is two. As it shows most of the errors counts that may generated by the model for a sentences are 1, 2, or 3.

Table 3. Lexical agreement evaluation over test sets (TSDBs)

	ATIS-TSDB	IBM-TSDB	WSJ-TSDB
Lexical Agreement	89.55%	91.7%	90.66%

Table 4. The result of the accuracy tagging evaluation over test sets

	ATIS-TSDB			IBM-TSDB			WSJ-TSDB		
	P1	**P2**	**All**	**P1**	**P2**	**All**	**P1**	**P2**	**All**
Incorrect	12	17	29	29	70	99	86	139	225
Correct	45	60	105	220	554	774	711	1103	1814
Err Rate	21%	22%	21.6%	11.6%	11.2%	11.3%	10.7%	11.1%	11.03%
Accuracy	79%	78%	**78.3%**	88.3%	88.8%	**88.7%**	88.3%	88.9%	**88.96%**
Total	57	77	134	249	624	873	797	1242	2039

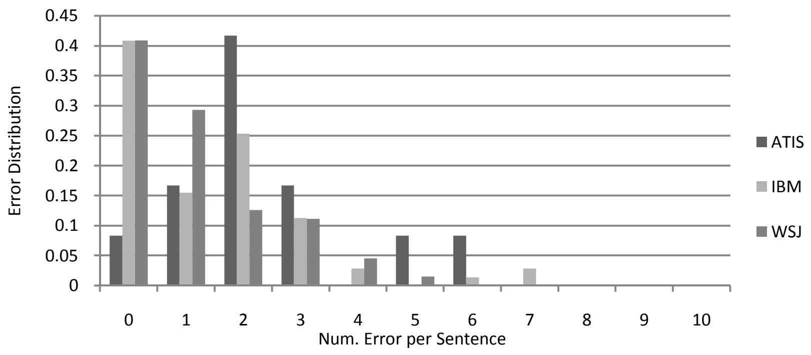


Fig. 4. Distribution of error counts in a given sentence of TSDBs

7 Conclusion

In this paper, we have tackled the problem of mapping MICA's [4] output to XTAG [32] elementary tree sequence as a sequence-tagging problem. An unsupervised sequence tagger based on Hidden Markov Model has been illustrated that it's initial values were trained by an innovative EM-based unsupervised method and then the whole parameters also were trained by forward-backward algorithm. It is expected that the model can provide way for using MICA's [4] parser with XTAG grammar [32].

References

1. Baldi, P., Chauvin, Y.: Smooth On-Line Learning Algorithms for Hidden Markov Models. Neural Computation 6(2), 307–318 (1994)
2. Bangalore, S., Joshi, A.: Supertagging: An approach to almost parsing. Computational Linguistics 25(2), 237–266 (1999)
3. Bangalore, S., Haffner, P., Emami, G.: Factoring global inference by enriching local representations. Technical report, AT&T Labs – Reserach (2005)
4. Bangalore, S., Boulllier, P., Nasr, A., Rambow, O., Sagot, B.: MICA: A probabilistic dependency parser based on Tree Insertion Grammar. In: North American Chapter of the Association for Computational Linguistics, NAACL (2009)
5. Baum, L.E.: An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. In: Shisha, O. (ed.) Inequalities III: Proceedings of the Third Symposium on Inequalities, University of California, Los Angeles, pp. 1–8. Academic Press, London (1972)
6. Boullier, P., Deschamp, P.: Le système SYNTAX^{TM} – manuel d'utilisation et de mise en oeuvre sous UNIX^{TM} (1972), `http://syntax.gforge.inria.fr/syntax3.8-manual.pdf`
7. Brill, E.: Transformation-based error-driven learning and natural language processing: A case study in part of-speech-tagging. Computational Linguistics 21(4), 543–566 (1995)
8. Chen, J., Bangalore, S., Vijay-Shanker, K.: New models for improving supertags disambiguation. In: Proc. EACL 1999, Bergen, pp. 188–195 (1999)
9. Chen, J.: Towards Efficient Statistical Parsing Using Lexicalized Grammatical Information. Ph.D. thesis, University of Delaware (2001)
10. Dang, H., Kipper, K., Palmer, M.: Integrating Compositional Semantic into a Verb Lexicon. In: Proceedings of the Eighteenth International Conference on Computational Linguistic, COLING 2000 (2000)
11. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from in- complete data via the EM algorithm. Journal of the Royal Statistical Society 39(1), 1–38 (1977)
12. Faili, H., Ghassem-Sani, G.: An application of Lexicalized Grammars in English-Persian Translation. In: Proceedings of 16^{TM} European Conference on Artificial Intelligence (ECAI), pp. 596–600 (2004)
13. Faili, H.: From partial toward full parsing. In: Recent Advance In Natural Language Processing (RANLP), pp. 71–75 (2009)
14. Faili, H., Basirat, A.: Augmenting the Automated Extracted Tree Adjoining Grammars by Semantic Representation. In: The 6th IEEE International Conference on Natural Language Processing and Knowledge Engineering (IEEE NLP-KE 2010), pp. 584–590 (2010)
15. Habash, N., Rambow, O.: Extracting a Tree Adjoining Grammar from the Penn Arabic Treebank. In: Proceedings of Traitement Automatique du Langage Naturel (TALN 2004), Fez, Morocco (2004)
16. Hemphill, C.T., Godfrey, J.J., Doddington, G.R.: The atis spoken language systems pilot corpus. In: DARPA Speech and Natural Language Workshop, Hidden Valley (1990)
17. Joshi, A.K., Levy, L., Takahashi, M.: Tree Adjunct Grammars. Journal of Computer and System Sciences 10(1), 136–163 (1975)

18. Joshi, A.K.: How much context-sensitivity is necessary for characterizing structural descriptions? In: Natural Language Processing: Theoretical, Computational, and Psychological Perspectives, pp. 206–250. Cambridge University Press, New York (1985)
19. Kipper, K., Dang, H., Palmer, M.: Class-based Construction of Verb Lexicon. In: Proceedings of Seventh Nation Conference on Artificial Intelligence, AAAI 2000 (2000)
20. Makino, T., Yoshida, M., Torisawa, K., Tsujii, J.: LiLFeS-Toward a Practical HPSG Parser. In: Proceeding of COLING-ACL 1998 (1998)
21. Marcus, M., Santorini, B., Marcinkiewicz, M.: Building a large annotated corpus of English: the Penn Treebank. Computational Linguistics 19(2) (1993)
22. Murphy, K.: A Hidden Markov Model (HMM) Toolbox for Matlab (1998), `http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html`
23. Neumann, G.: Automatic extraction of stochastic lexicalized tree grammars from tree banks. In: Forth International Workshop on TAG and Related Frameworks, TAG+4 (1998)
24. Park, J.: Extraction of tree adjoining grammar from a tree bank for Korean. In: Proceedings of the COLING/ACL 2006 Student Research Workshop, pp. 73–78 (2006)
25. Rabiner, L.R.: A tutorial on Hidden Markov Models and selected applications in speech recognition. Proceedings of the IEEE 77(2), 257–286 (1989)
26. Ryant, N., Kipper, K.: Assigning XTAG trees to VerbNet. In: Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+7), pp. 194–198 (May 2004)
27. Schabes, Y., Waters, R.C.: Tree Insertion Grammar. Computational Linguistics 21(4) (1995)
28. Shen, L., Joshi Aravind, K.: Incremental ltag parsing. In: HLT-EMNLP 2005 (2005)
29. Van Noord, G.: Head-corner parsing for TAG. Computational Intelligence 10(4), 525–534 (1994)
30. Xia, F.: Automatic grammar generation from two different perspectives. Ph D. thesis, University of Pennsylvania (2001)
31. Xia, F., Palmer, M.: Evaluating the Coverage of LTAGs on Annotated Corpora. In: Proceeding of Workshop on Using Evaluation within HLD Programs. Results and Trends at Second International Conference on Language Resources and Evaluation, pp. 1–6 (2001)
32. XTAG-group. A Lexicalized Tree Adjoining Grammar for English, Technical Report IRCS 98-18, Institute for Research in Cognitive Science, University of Pennsylvania, pp. 5–10 (1998)

Tamil Dependency Parsing: Results Using Rule Based and Corpus Based Approaches

Loganathan Ramasamy and Zdeněk Žabokrtský

Institute of Formal and Applied Linguistics (ÚFAL)
Faculty of Mathematics and Physics, Charles University in Prague
{ramasamy,zabokrtsky}@ufal.mff.cuni.cz

Abstract. Very few attempts have been reported in the literature on dependency parsing for Tamil. In this paper, we report results obtained for Tamil dependency parsing with rule-based and corpus-based approaches. We designed annotation scheme partially based on Prague Dependency Treebank (PDT) and manually annotated Tamil data (about 3000 words) with dependency relations. For corpus-based approach, we used two well known parsers MaltParser and MSTParser, and for the rule-based approach, we implemented series of linguistic rules (for resolving coordination, complementation, predicate identification and so on) to build dependency structure for Tamil sentences. Our initial results show that, both rule-based and corpus-based approaches achieved the accuracy of more than 74% for the unlabeled task and more than 65% for the labeled tasks. Rule-based parsing accuracy dropped considerably when the input was tagged automatically.

Keywords: Tamil, Dependency parsing, Syntax, Clause boundaries.

1 Introduction

The most important thing in Natural Language Processing (NLP) research is data, importantly the data annotated with linguistic descriptions. Much of the success in NLP in the present decade can be attributed to data driven approaches to linguistic challenges, which discover rules from data as opposed to traditional rule based paradigms. The availability of annotated data such as Penn Treebank [1] and parallel corpora such as Europarl [2] had spurred the application of statistical techiniques [4], [5], [3] to various tasks such as Part Of Speech (POS) tagging, syntactic parsing and Machine Translation (MT) and so on. They produced state of the art results compared to their rule based counterparts. Unfortunately, only English and very few other languages have the privilege of having such rich annotated data due to various factors.

In this paper, we take up the case of dependency parsing task for Tamil language for which no annotated data is available. We report our initial results of applying rule based and corpus based techniques to this task. For rule-based approach, the rules (such as rules for coordination and main predicate identification) have been crafted after stuying the Tamil data in general. The rules often rely on morphological cues to identify governing or dependent nodes. For corpus-based approach, we used two popular dependency parsers: Malt and MSTParser. For the purpose of experimentation, both the approaches have been tested on small syntactically annotated corpus. The annotation is partially based on popular Prague Dependency Treebank (PDT) [15].

In Section 2, we introduce previous attempts on this topic and similar work that is happening for other Indian languages. Section 3 introduces some important aspects of Tamil language. Section 4 proposes the annotation scheme, Section 5 introduces the rule-based method and corpus-based techniques and the remaining sections introduce experimental evaluations and discussions.

2 Related Work

Tamil syntactic parsing is less discussed in the literature though there are some recent work on developing morphological analyzers and POS taggers for Tamil. This is evident from the scarce number of publications that have appeared on this topic. The situation is better for other major Indian languages such as Hindi and Telugu. There is an active research on dependency parsing ([7], [8], [9]) and developing annotated treebanks for Hindi and Telugu. One such effort is, developing a large scale dependency treebank [10] (aimed at 1 million words) for Telugu, as of now the development for which stands [11] at around 1500 annotated sentences. Recently in year 2009, there was an NLP tools contest dedicated for parsing Indian languages (Hindi, Telugu and Bangla) as part of the ICON 2009 conference. That unfortunately didn't include Tamil.

There were two prior works [12], [13] that discussed about Tamil syntactic parsing. [12] used morphological analyzer and heuristic rules to identify phrase structures. [13] built a dependency analyzer for spoken language Tamil utterences. The work [13] used relative position of words to identify semantic relations. The current paper is different from [12] with respect to building dependency trees rather than phrase structure trees. The current paper gives a comprehensive treatment (in terms annotation scheme, parsing approaches and experimentation) to Tamil dependency parsing than the other two papers. There is a recent paper [14] which used machine learning approach to dependency parsing of Tamil. Since no results were reported, we are not able to compare our results with theirs.

3 General Aspects of Tamil Language

Tamil is an south Indian language that belongs to Dravidian family of languages. Other major languages in the Dravidian family include Telugu, Malayalam and Kannada. The main features of the Tamil language include agglutination, relatively free word order, head final and the subject-verb agreement. Below we touch briefly on these features.

Morphology. Tamil is an agglutinative language [6] and has rich set of morphological suffixes which can be added one after another to noun and verb stems (mainly) as suffixes. Tamil morphology is mainly concatenative and derivations are also possible by means of *adjectivalization, adverbialization* and *nominalization.* In general, Tamil morphology can be represented [6] as [*stem (+affix)*^{n}]. Though there are only eight basic POS categories, with no such restrictions placed on as to how many words can be glued together, Tamil morphology pose significant challenges to POS tagging and parsing.

Head Final and Relatively Free Word Order. Tamil is a head final language, meaning the head of the phrasal categories always occur at the end of a phrase or constituent. Modifiers and other co-constituents always precede the phrasal head. For example, *postposition* is the head of the postpositional phrase, and will be modified by noun phrases. There are very few exceptions (identifiable) such as the subject of a sentence occuring after the finite verb (head). In most cases, head final rule is preserved.

Tamil is a Subject Object Verb (SOV) language and the word order is relatively free. Within a clause, phrases can be moved to almost any position except to the postion of clause head which should always be a verb. Consider the Tamil sentence *'appA enakku puTTakam kotuTTAr'*[1] (Father gave me a book). The free word order nature of Tamil for this sentence is given below,

appA enakku puTTakam kotuTTAr	S-O2-O1-V
enakku appA puTTakam kotuTTAr	O2-S-O1-V
puTTakam enakku appA kotuTTAr	O1-O2-S-V
puTTakam appA enakku kotuTTAr	O1-S-O2-V
....	...
enakku puTTakam kotuTTAr appA	O2-O1-V-S

Agreement. There are two kinds of verbs in Tamil: finite verbs and non finite verbs. Finite verbs usually occur as sentence final and will act as a main verb of the sentence. Finite verbs agree with their subject in number and gender. This is accomplished via coding the number and gender of subject in the finite verbs as suffixes. Both finite and non-finite verbs can take subjects, but only finite verbs can code their agreement with subjects. The explicit coding of subject-verb agreement in finite verbs make the presence of subjects optional in certain situations, and the subject can be partially inferred from finite verb affixes. Table 1 shows that the total number of subjects are less than the total number of finite and non-finite verbs. This implies that some of the subjects are shared between finite and non finite verbs in sentences or some verbs may not take subjects at all. This point is to illustrate that it is not always possible to identify the subjects just by knowing agreement markers.

4 Annotation Scheme

In this section, we describe the data and propose annotation scheme for Tamil dependency structures.

[1] We have transliterated the Tamil script into Latin script. Transliterated form is used throughout the paper.

Table 1. Counts of subjects and verbs

Category	#Num
Sentences	204
Subjects	332
Finite verbs	203
Non finite verbs	220

We collected Tamil data from a news website[2]. We picked news articles from the website randomly, so to have our data as diverse as possible. The raw Tamil data was in `utf-8` encoding, we transliterated that into Latin script for the ease of processing during annotation work and as well as to handle the data programmatically. As of now, the corpus size we have taken for annotation is 2961 words.

Annotation is divided into two parts: (i) POS tagging of the data and (ii) assigning relations and dependency structure to words. Tagging the data with POS is necessary as our rule based dependency parser often use POS tags of words to predict dependency relations.

4.1 POS Tagging

As is the case for any morphologically rich language, providing a fixed tagset for languages such as Tamil is a complex task. Many tagsets were proposed in the literature and many methodologies based on finite state machines, Hidden Markov Models (HMM) and Support Vector Machines (SVM) were proposed for tackling morphology and tagging of Tamil. The main question here is, "What kind of POS tagset is required for our task? Whether simple POS tagset or fine grained morphological pos tagset". Considering the rich suffix base Tamil have [6], we decided to use detailed morphological tags instead of just categorical. For morphologically rich languages, morphological cues can to some extent help identifying syntactic relationship between words. Unfortunately, no standard exist for morphological tagging of Tamil.

Our manual morphological tagging is not based on marking every linguistic aspect within a word. Rather, our aim is to add more information to the tag which are needed for identifying syntactic relationships. For example, we tag the word *pUkkatai* (flower shop) as a single noun rather than having composed of two separate nouns *pU* (flower) and *katai* (shop). Thus the tag of that word will be "3n_nn", where *nn* would indicate the word is a *noun* and *3n* would indicate the noun is a *3rd person neuter gender*. For tagging verbs, to distinguish lexical and auxiliary verbs, we add prefixes "mv" and "aux" respectively. For example, the lexical verb *patiTTAn* (read he) is tagged as "mv_pst_3sm_f". In the tag, "pst_3sm" indicates the verb is a past tense verb and the subject is 3rd person singular masculine. "f" indicates the verb is a finite verb. Knowing whether the verb is *finite* or *non finite* will greatly help in identifying the main predicate of the sentence.

[2] `http://www.dinamani.com`

In the Table 2, we have `Corpus 1` with 2961 words tagged and syntactically annotated, and `Corpus 2` which also contains `Corpus 1` and the remaining words in the corpus are tagged, but not syntactically annotated. The table also provides some insights into how many tags a single token can take. From the Table 2, it is clear that, more than 90% of the tokens take only a single tag. Tokens with 2 tags vary between 5-8%, and the remaining count is almost negligible.

Table 2. Number of tags assigned to tokens

	Corpus 1	**Corpus 2**
Tagset size	296	459
Lexical verb tags	120	194
Auxliary verb tags	31	44
# of words	2961	8421
Unique tokens	1634	3747
1 tag count	1534/(93.88%)	3427/(91.46%)
2 tag count	92/(05.63%)	284/(07.58%)
3 tag count	8/(00.49%)	33/(00.88%)
4 tag count	0/(00.00%)	3/(00.08%)

4.2 Dependency Annotation

Our annotation scheme is partially based on Prague Dependency Treebank (PDT) [15], [16] , a large scale dependency annotation project for Czech language. The PDT was annotated on three layers: *morphological layer (m-layer)*, *analytical layer (a-layer)* and *tectogrammatical layer (t-layer)*. In *m-layer*, a sentence is annotated at the word level, marking their POS and identifying their lemmas. This layer roughly corresponds to morphological tagging of a sentence. The sentence is represented as a rooted tree in *a-layer*, where each node correspond to a word in the sentence. Each dependent node is connected with a governing node via an edge which hold a relationship between the two nodes. In the PDT style annotation, the edges are not labeled, rather each node (analytical node) is assigned an analytical function (*afun*) which acts as relation with which it connect to its governing node. In *tectogrammatical* layer, a tree represents the deep syntactic structure of a sentence. In *t-layer* tree, the nodes correspond to auto- semantic words which have the lexcical meaning. Words that correspond to prepositions, determiners and other function words will not be represented in *t-layer*. There is also one more layer called *w-layer* which simply represents the input sentence or tokens (prior to *m-layer*).

Our annotation involves only *m-layer* and *a-layer*. In *m-layer*, words are annotated with morphological tags as mentioned in the previous sub section (POS tagging). As of now, for *a-layer* we have defined 19 analytical functions, 13 of them comes from PDT. The list of analytical functions is given in Table 3. The *a-layer* annotation is performed using *TrEd* annotation tool [17] after parsing the text using rule-based parser (Section 5.1). The sample annotation is shown in Figure 1.

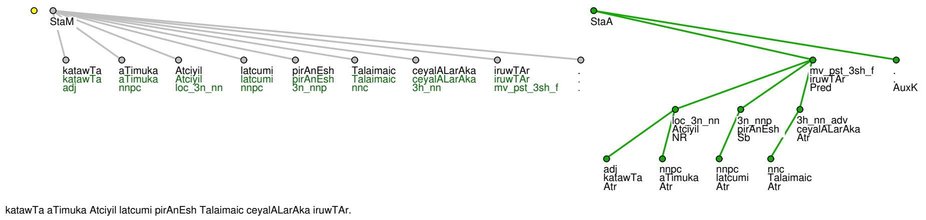


Fig. 1. Sample annotation using TrEd tool

Table 3. Analytical functions for a-layer annotation

AFUN	Description	Example
AdjCl	Adjectival clauses	inRu *wataipeRRa* pOttiyil
		today *take-place-which* match-loc
Adv	Adverbial	*inimEl* watakkATu
		hereafter happen-will-not-it
Atr	Attribute	*iwTiya* aracu
		Indian government
AuxA	Determiners	*iwTap* paiyan
		this boy
AuxC	Conjuncts	*rAman* maRRum *cITA*
		Ram and *Sita*
AuxK	Terminal punctuation	–
AuxP	Postposition	pUmikkuk *kIzE*
		earth-dat *under*
AuxV	Auxiliary	cAppAtu cAppittu *vittAn*
		food eat-pst *leave-pst-3sm*
AuxX	Comma	–
CondCl	Conditional clause	avar ennai *azaiTTAl*
		he-honorific me-acc *call-if*
Coord	Coordination node	rAman *maRRum* cITA
		Ram *and* Sita
InfCl	Infinitive clause	puTTakam *kotukka* rAman ...
		book *give-inf* Ram ...
NR	Dependency not defined	–
Obj	Direct object	rAman oru *puTTakam* kotuTTAr
		Ram a *book* give-3s-honorific
PObj	Postpositional object	*rAmanaip* paRRi
		Ram-acc about
Pred	Predicate of the sentence	ramEsh paNam *kotuTTAn*
		Ramesh money *give-pst-3sm*
Sb	Subject	*ramEsh* paNam kotuTTAn
		Ramesh money give-pst-3sm
VbpCl	Verbal participle clause	kAcu *kotuTTu* mittAy
		money *give-pst candy*
VFin	Finite verb (not predicate)	paricu *kotuTTAn* enRu kURinAn
		gift *give-pst-3sm* that say-pst-3sm

Table 4. Coordination pattern

S.No	Pattern	Coordination type	Coordination head
1	... A*um* ... B*um* ...	and(*um*)	B
2	... A*um* , ... B*um* ...	and(*um*)	,
3	... A, ...B, ...C *maRRum* D	and(*maRRum*)	*maRRum*
4	... A*O* ... B*O* ...	or(*O*)	B
5	... A*O* *allaTu* ...B*O* ...	or(*O*)	*allaTu*

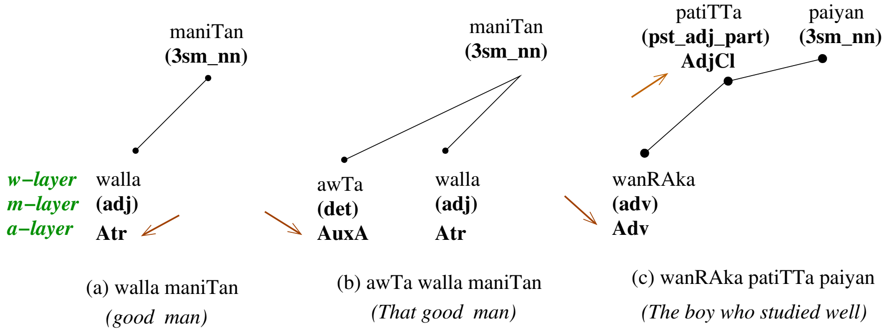


Fig. 2. Illustration of **Atr, Adv, AuxA, AdjCl** analytical functions

Atr, Adv, AuxA, AdjCl. *Atr* is used to mark the nodes in an attribute relationship to their governing nodes. In Figure 2a) the word *walla* (good) is in an attribute relation to it's governing node *maniTan* (human). In noun-noun combinations, the first noun will be in attribute relationship to the second noun (head). Similarly *Adv* and *AuxA* are used to mark adverbials and determiners, and they modify verbs and nouns. Adjectival clauses in Tamil are equivalent to relative clauses in English. Though adjectival heads act as modifiers to the following nouns, they still retain the verbal properties and take left side arguments. Adjectival clause heads are marked with *AdjCl.*

Coord, AuxC,AuxX. Coordination is one of the complex phenomena in Tamil. [6] talks about only one type coordination which uses morphological suffix as a coordination device. But coordination can be marked either morphologically with -*um* ("and") or with a separate word *maRRum* ("and") or with commas or in combination of any of the previous three. They can coordinate any type of individual words, phrases and clauses with same categorical status. The type of role which the coordination node can take depends on the type of conjuncts they coordinate. Here we list the most common type of coordination patterns we identified from the corpus. If A, B, C and D are elements to be conjoined via coordination, then the pattern (also in Figure 3) for coordination (*and, or*) is listed in the Table 4.

The main issue in coordination conjunction is, when elements are conjoined, the coordinating node should assume the role of the conjoined elements and should attach to the appropriate node where the conjoined elements would have

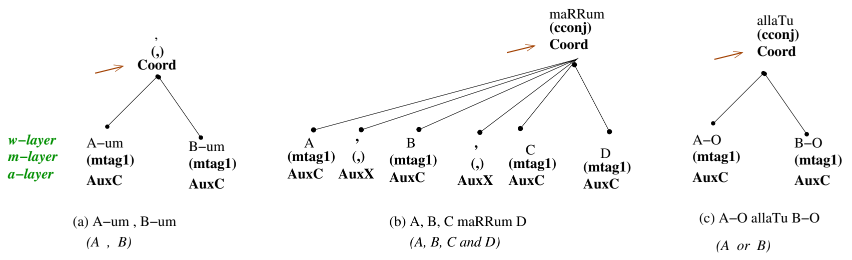


Fig. 3. Illustration of coordination conjunction

attached. At present, during annotation, the category of the conjoined elements are not reflected in the coordinating node.

5 Approaches to Parsing

In this section, we define two approaches (rule based & corpus based) we have used to parse Tamil sentences.

5.1 Rule Based Parsing

The algorithm 5.1 simply returns the parents (in array) of each word token which is equivalent to unlabeled edges in the dependency tree. Each element of this array is an integer, representing the position of the parent word in the sentence. Array index correspond to word position of a child node .

Algorithm 5.1: GetUnlabeledEdges(*words*, *mtags*)

```
comment: Set parents (p) and is_parent_available (ipa) array to 0
comment: size(p) = size(ipa) = size(words)
$p \leftarrow 0$
$ipa \leftarrow 0$
IDENTIFY_MAIN_PREDICATE($p, ipa, words, tags$)
RESOLVE_COORDINATION($p, ipa, words, tags$)
IDENTIFY_TRIVIAL_PARENTS($p, ipa, words, tags$)
PROCESS_COMPLEMENTS($p, ipa, words, tags$)
return ($p$)

procedure SET_PARENT($node, parent$)
 $acyclicity \leftarrow$ CHECK_ACYCLICITY()
 if not $acyclicity$ and $ipa[node] \neq 1$
   then $\begin{cases} p[node] \leftarrow parent \\ ipa[node] \leftarrow 1 \end{cases}$
 return
```

We have mainly defined four procedures inside the algorithm, which update the *parent (p)* and *is_parent_available (ipa)* array whenever the linguistic rules defined inside them are satisfied. For instance, the procedure *identify_main_predicate* will look for the main predicate of the sentence. If found, then the parent of that node will be updated (0 in this case. 0 is a technical root of the tree). Once the parent is found, finding the parent again for this node is prohibited via the *ipa* array, which sets the boolean variable, once the parent is found for that node. Each procedure implements a set of linguistic rules which assign parents to words in the sentence. For instance, predicate finding rule is very simple. It says, *if the last word of the sentence is a 'finite verb', and it's morphological tag ends with '_f'* then the parent of that node is 0 which is the technical root of the tree.

The procedure *resolve_coordination* will try to locate coordination head and the conjuncts. If the coordination head is found, then the conjuncts will be located and their parent will be set to the coordination head. Setting the parent of the coordination head is the complicated task, the procedure will try to assign the parent based on the conjoined elements. This procedure implements the coordination rules specified in Table 4.

The procedure *identify_trivial_parents* will try to locate modifiers such as adjectives, determiners, cardinals, ordinals and so on, and set their parents which would be the immediate phrasal head following the modifiers. In the case of postpositions which act as the PP head, the task is to identify the noun phrase preceding them, and attach them as children of the postpositional head.

The procedure *process_complements* takes care of complementation in Tamil. Tamil uses various devices to perform complementation. Of which, the most common types of complementations are non finite clauses, using certain lexical verbs such as *'en'* (say), and few nouns and postpositions as complementizers. Presence of a complementation would indicate that there is some clause ending at that point. If there is an explicit presence of complementizer, then the complementizer would act as a head of the clausal predicate preceding it. Then the complementizer would attach itself to the appropriate node that follow the complementizer.

In the same procedure, we also perform clause boundary identification and attach arguments to the clausal predicates. Clausal predicate would signal the end of a clause. Based on morphological tags, we identify these clause boundaries and attach arguments of that clause to the clausal predicate. Let us assume, there are three clauses inside a sentence. This procedure would identify the 3 clausal predicates and attach arguments that belong to them. In Tamil, clauses usually occur in linear order, so the arguments always appear to the left of a clausal predicate. Thus the arguments for clause 2 would immediately start from clause 1 boundary and end before the second clausal predicate. Once the clausal predicates and their arguments are attached, the clausal predicates would be attached to either complementizers, or nouns (in the case of adjectival clauses) or other clauses. At present, the procedure works only if the clauses are in linear order. If there is an embedded clause, then this procedure often attaches wrong arguments to the clausal predicates.

Label Assignment. Once the unlabeled edges are identified, this procedure assigns labels to nodes. The procedure again makes use of morphological tags of parents and their children to assign labels to them. For example, the children of *postpositional* node is assigned *PObj* label and if the case of a node is *accusative* and its parent is verb, then the node is assigned *Obj*. This procedure labels all the nodes that are present in the sentence.

The rule-based parsing system is implemented within TectoMT framework [18], a highly modular NLP system for Machine Translation and other related NLP tasks. All tasks of the rule-based parser have been implemented as modules inside TectoMT system. Evaluation of this parser is given in the next section.

5.2 Corpus Based Approach

For corpus based approach we are using two well known parsers namely MaltParser [20] and MSTParser [19]. Evaluation of these parsers for our data is explained in the next section.

6 Experiments and Results

We did manual morphological tagging on 8421 tokens. Out of them, 2961 tokens are both morphlogically tagged as well as dependency annotated. Refer Section 4 and Table 2 for more information about the nature of the data. We name the corpus as follows: 2961 tokens corpus is called C1, 8421 tokens corpus as C2 , and 5500 tokens (C2-C1) as C3.

Experiments for Rule Based (RB) parsing and Corpus Based (CB) parsing is done with different settings. Though direct comparison cannot be made, we can find some similarities in the individual label performance.

The main input for the RB parsing is an input sentence and its morphological tags. Two experiments have been conducted for RB parsing. For both the experiments, RB approach was tested against the whole dataset (2961) tokens. In the first experiment, morphological tags for input tokens have been automatically tagged by TnT tagger [21] trained on C3 (tagged the C1 with 72.34% accuracy). For the second experiment, input tokens are provided to RB parser with gold standard morphological tags. The Table 5 (left) shows the accuracy of RB parser. The column *Auto POS* corresponds to the first experiment and the *Manual POS* corresponds to the second experiment. We can see the sharp decline (for both unlabeled and labeled) in performance when the input tokens are not given gold standard tags. Individual label performance is given in Table 6. The only label which achieved a higher performance in both the experiments is the prediction of *Pred* which is the main predicate of the sentence. Since *AuxK* was a sentence termination, it was not included. The worst performed label in both the experiments is *Coord*. The precision of subject *Sb* is also low due to difficulties in identifying the right subject when more than one nominatives qualify for subject status. The performance of *Sb* may also decrease if the subject of one clause is occurring outside of its boundary in the case of embedded clauses.

Table 5. Parsing Accuracy a) Rule based b) Corpus based

	Auto POS	Manual POS
Unlabeled	71.94	84.73
Labeled	61.70	79.13

	MaltParser	MST Parser
Unlabeled	75.03	74.92
Labeled	65.69	65.69

Table 6. Precision/Recall of Rule Based Parser: a) Auto POS (left) b) Manual POS (right)

Afun	**Precision**	**Recall**
AdjCl	48.31	78.18
Adv	43.75	72.41
Atr	68.26	79.93
AuxA	78.95	100.00
AuxC	68.89	48.44
AuxK	100.00	99.03
AuxP	50.88	61.70
AuxV	50.00	39.39
AuxX	44.92	100.00
CondCl	50.00	16.67
Coord	**33.33**	**26.92**
InfCl	75.68	66.67
NR	57.17	53.99
Obj	48.28	65.12
PObj	49.06	66.67
Pred	**96.06**	**96.06**
Sb	**44.73**	74.55
VFin	48.00	80.00
VbpCl	53.52	61.29

Afun	**Precision**	**Recall**
AdjCl	81.94	98.33
Adv	87.27	97.96
Atr	87.08	97.18
AuxA	100.00	100.00
AuxC	87.50	67.74
AuxK	99.03	100.00
AuxP	82.54	100.00
AuxV	69.44	100.00
AuxX	52.54	100.00
CondCl	57.14	100.00
Coord	**59.09**	**41.94**
InfCl	93.33	100.00
NR	73.64	73.37
Obj	81.05	73.33
PObj	70.37	97.44
Pred	**97.07**	**98.03**
Sb	**58.91**	94.93
VFin	91.18	100.00
VbpCl	81.08	100.00

For CB parsing, we have divided the corpus C1 into two parts: training set (2008 tokens) and test set (953 tokens). Both the Malt and MST parsers were trained on the same training set and evaluated on the same test set. Table 5 (right) shows the accuracy of both Malt and MST parser. They perform almost similarly for both labeled and unlabeled tasks. In the case of individual label performance (Table 7), labeled precision for *Sb* is high and low for *Coord.* Some individual labels from both RB parsing and CB parsing show same level of performance, for instance *Sb* and *Coord.*

As a general remark, the current experiments for both RB & CB tasks have been done with very small amount of data. Yet we were able to correlate some of the labels' individual performance. As observed from the labeled precision of both RB and CB tasks, some labels are easily predictable (*Pred*) and some labels are not, such as *Coord, Sb.* As far as the non-projective structures are concerned, for Hindi, the study [22] has found that 14% of Hindi treebank structures are non-projective. For Tamil, non-projectivity may arise in embedded clauses.

Table 7. Precision/Recall of corpus based parsing: a) MaltParser (left) and b) MST Parser (right)

Afun	Precision	Recall
AdjCl	71.43	71.43
Adv	57.89	100.00
Atr	91.46	90.55
AuxA	91.67	84.62
AuxC	100.00	84.62
AuxK	100.00	96.72
AuxP	53.57	100.00
AuxV	100.00	26.67
AuxX	40.74	100.00
CondCl	–	0.00
Coord	**50.00**	**25.00**
InfCl	82.35	70.00
NR	41.28	75.32
Obj	38.46	66.67
PObj	56.25	52.94
Pred	**96.61**	**96.61**
Sb	59.38	55.88
VFin	77.78	87.50
VbpCl	68.42	56.52

Afun	Precision	Recall
AdjCl	76.00	90.48
Adv	50.00	100.00
Atr	84.79	93.40
AuxA	92.31	92.31
AuxC	46.67	58.33
AuxK	100.00	96.72
AuxP	50.00	100.00
AuxV	66.67	28.57
AuxX	50.00	100.00
CondCl	–	0.00
Coord	**20.00**	**16.67**
InfCl	83.33	75.00
NR	42.86	73.94
Obj	51.72	78.95
PObj	44.44	47.06
Pred	**80.60**	**91.53**
Sb	70.73	56.31
VFin	71.43	62.50
VbpCl	78.26	75.00

7 Conclusion and Future Work

In this paper, we reported our initial experiments with dependency parsing for Tamil using both rule based and corpus based approaches. For the rule based approach, the labeled accuracy achieved 79% when input tokens are provided with morphological tags and declined to 61% when tested in a real world scenario. For the corpus based approach, the labeled accuracy stood at around 65% (for both Malt and MST) and unlabeled accuracy stood at around 75%. From the experiments, we observed that, both the rule based and corpus approaches found difficulty in identifying coordination nodes (*Coord*) while they performed well in identifying main predicate of the sentence for unseen cases. Tagging accurately also had an impact on the performance (as shown by the rule based parser). The current experiments were done with very small data, more insights can be gained and accuracy can be improved if we have more data. In the future, we are planning to add more annotated corpora for our experiments.

Acknowledgement

The research leading to these results has received funding from the European Commission's 7th Framework Program (FP7) under grant agreement n° 238405 (CLARA), and from grant MSM 0021620838. We also would like to thank anonymous reviewers for their useful comments.

References

1. Mitchell, P.M., Mary Ann, M., Beatrice, S.: Building a Large Annotated Corpus of English: the Penn Treebank. Comput. Linguist. 9, 313–330 (1993)
2. Koehn, P.: Europarl: A Parallel Corpus for Statistical Machine Translation. In: MT Summit (2005)
3. Koehn, P., Och, F.J., Marcu, D.: Statistical Phrase-Based Translation. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, pp. 48–54. Association for Computational Linguistics (2003)
4. Ratnaparkhi, A.: A Maximum Entropy Model for Part-Of-Speech Tagging. In: Proceedings of the Empirical Methods in Natural Language Processing, pp. 133–142 (1996)
5. Collins, M.: Head-Driven Statistical Models for Natural Language Parsing. Comput. Linguist. 29, 589–637 (2003)
6. Lehmann, T.: A Grammar of Modern Tamil. Pondicherry Institute of Linguistics and Culture (PILC), Pondicherry, India (1989)
7. Bharati, A., Sangal, R.: Parsing Free Word Order Languages in the Paninian Framework. In: Proceedings of the 31st annual meeting on Association for Computational Linguistics, pp. 105–111. Association for Computational Linguistics (1993)
8. Bharati, A., Gupta, M., Yadav, V., Gali, K., Sharma, D.M.: Simple Parser for Indian Languages in a Dependency Framework. In: Proceedings of the Third Linguistic Annotation Workshop (ACL-IJCNLP 2009), pp. 162–165. Association for Computational Linguistics (2009)
9. Nivre, J.: Parsing Indian Languages with MaltParser. In: Proceedings of the ICON 2009 NLP Tools Contest: Indian Language Dependency Parsing, pp. 12–18 (2009)
10. Begum, R., Husain, S., Dhwaj, A., Sharma, D., Bai, L., Sangal, R.: Dependency Annotation Scheme for Indian Languages. In: Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP), Hyderabad, India (2008)
11. Vempaty, C., Naidu, V., Husain, S., Kiran, R., Bai, L., Sharma, D., Sangal, R.: Issues in Analyzing Telugu Sentences towards Building a Telugu Treebank. In: Gelbukh, A. (ed.) CICLing 2010. LNCS, vol. 6008, pp. 50–59. Springer, Heidelberg (2010)
12. Saravanan, K., Ranjani, P., Geetha, T.V.: Syntactic Parser for Tamil. In: Proceedings of Sixth Tamil Internet 2003 Conference, Chennai, India (2003)
13. Janarthanam, S., Nallasamy, U., Ramasamy, L., Santhoshkumar, C.: Robust Dependency Parser for Natural Language Dialog Systems in Tamil. In: Proceedings of 5th Workshop on Knowledge and Reasoning in Practical Dialogue Systems (IJCAI KRPDS 2007), Hyderabad, India, pp. 1–6 (2007)
14. Dhanalakshmi, V., Anand Kumar, M., Rekha, R.U., Soman, K.P., Rajendran, S.: Grammar Teaching Tools for Tamil Language. In: Technology for Education Conference (T4E 2010), India, pp. 85–88 (2010)
15. Hajic, J.: Building a Syntacticly Annotated Corpus: The Prague Dependency Treebank. In: Issues of Valency and Meaning, Karolinum, Prague, pp. 106–132 (1998)
16. The Prague Dependency Treebank 2.0, `http://ufal.mff.cuni.cz/pdt2.0/`
17. Tree Editor TrEd, `http://ufal.mff.cuni.cz/~pajas/tred/`
18. Žabokrtský, Z., Ptáček, J., Pajas, P.: TectoMT: Highly Modular MT System with Tectogrammatics Used as Transfer Layer. In: Proceedings of the Third Workshop on Statistical Machine Translation (StatMT 2008), pp. 167–170. ACL (2008)

19. McDonald, R., Crammer, K., Pereira, F.: Online Large-margin Training of Dependency Parsers. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pp. 91–98. ACL (2005)
20. Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kubler, S., Marinov, S., Marsi, E.: MaltParser: A Language-Independent System for Data-Driven Dependency Parsing. Natural Language Engineering 13, 95–135 (2007)
21. Brants, T.: TnT - A Statistical Part-of-Speech Tagger. In: Proceedings of the Sixth Conferenceon Applied Natural Language Processing ANLP 2000, Seattle (2000)
22. Mannem, P., Chaudhry, H., Bharati, A.: Insights into non-projectivity in Hindi. In: Proceedings of the ACL-IJCNLP 2009 Student Research Workshop, pp. 10–17. ACL (2009)

Incremental Combinatory Categorial Grammar and Its Derivations*

Ahmed Hefny[1], Hany Hassan[2], and Mohamed Bahgat[3]

[1] Computer Engineering Department, Faculty of Engineering, Cairo University, Giza, Egypt
ahefny@eng.cu.edu.eg
[2] Microsoft Research, Redmond, WA 98052, USA
hanyh@microsoft.com
[3] IBM Cairo Technology Development Center, P.O. 166 El-Ahram, Giza, Egypt
mbahgat@eg.ibm.com

Abstract. Incremental parsing is appealing for applications such as speech recognition and machine translation due to its inherent efficiency as well as being a natural match for the language models commonly used in such systems. In this paper we introduce an Incremental Combinatory Categorical Grammar (ICCG) that extends the standard CCG grammar to enable fully incremental left-to-right parsing. Furthermore, we introduce a novel dynamic programming algorithm to convert CCGbank normal form derivations to incremental left-to-right derivations and show that our incremental CCG derivations can recover the unlabeled predicate-argument dependency structures with more than 96% F-measure. The introduced CCG incremental derivations can be used to train an incremental CCG parser.

1 Introduction

An incremental parser is able to processes an input sentence left-to-right, word-by-word, and build for each prefix of the input sentence a partial parse that is a sub-graph of the partial parse that it builds for a longer prefix. Besides being cognitively plausible, an incremental parser is more appealing for applications since its time and space complexities are close to linear in input length. It should, for example, constitute a natural match for the word-by-word decoding and pruning schemes used within statistical machine translation and speech recognition.

Combinatory Categorial Grammar (CCG) [10] is a lexicalized grammar formalism that has very strong potential for incremental parsing, mainly due to its ability to represent an arbitrary subsequence of a valid sentence by a single category even if they do not form a complete phrase. CCG [10] extends the categorial grammar by adding new combinatory rules such as type raising and composition. Not only do these extensions increase the grammar coverage and ability to recover long-range dependencies but also they allow incremental parsing. However, there are still difficulties in handling some linguistic constituents such as back modifiers (e.g. adverbs) in an efficient and deterministic, yet incremental manner. Although standard CCG rules may be able to handle

* This work was conducted while the first two authors were at IBM Cairo Technology Development Center.

such constituents, such handling is heavily dependent on type raising in a way that can greatly increase the search space of an incremental parser.

CCGbank [4] is a set of CCG normal form derivations that was obtained by transforming the parse trees in the Penn Wall Street Journal (WSJ) Treebank into normal form CCG derivations, namely head dependency annotations over a wide-coverage lexicon of supertags. CCGbank has been used to train a number of wide-coverage CCG parsers, e.g. [1].

The contribution of this paper is twofold. First, extending Steedman's CCG by introducing combinatory rules to support efficient incremental left-to-right parsing. Second, introducing an incremental version of CCGbank by transforming the normal form CCG derivations into head dependency left-to-right derivations that can be used to train incremental CCG parsers.

The rest of the paper proceeds as follows. In section 2, we review previous works on incremental parsing. In section 3, we review the standard CCG. In section 4, we describe our Incremental CCG extension. In section 5, we introduce the algorithm to transform CCG normal for derivations into incremental derivation. In section 6, we evaluate the incremental CCG and its derivations. Finally, we conclude and discuss the future work.

2 Related Work

Yamada and Matsumoto [11] proposed a bottom-up shift-reduce dependency parser that processes the sentence word by word and uses an SVM classifier to decide the actions to take for each word. These actions can push or pop tokens to or from a parsing stack. They can also create dependency links in the dependency graph, which represents the syntactic dependencies between the words of the input sentence. That parser is, however, not fully incremental because it allowed multiple passes on the sentence to recover the full dependency graph. Nivre [5],[6] proposed a dependency parser based on the framework of Yamada and Matsumoto. The parser proposed by Nivre incorporated a modified set of actions that allowed for single-pass parsing that is incremental in most of the sentences. Nivre defines an incremental parse as a parse in which the dependency graph is always connected in all parsing steps.

The parsers of Yamada and Matsumoto and Nivre are examples of *transition-based deterministic parsers*. An incremental transition-based parser maintains a *parsing state* that represents a partial parse of the words processed so far. As a new word is processed, the parsing action(s) modifies this state. The state in the previous parsers is determined by the contents of the parsing stack and the dependency graph constructed so far. The transition-based parser is deterministic if it maintains a single parsing state. A *statistical parser*, on the other hand, maintains a set of n-best parsing states.

The parsers of Yamada and Matsumoto and Nivre are also examples of dependency grammar parsers, where a span of words is solely represented by a dependency subgraph. Dependency grammar does not have the notion of a phrasal node that represents a span of words.

Sagae and Lavie [7] propose a statistical shift-reduce parser. They show that incorporating a best-first search strategy by replacing the single parsing state with a heap of n-best states results in significant improvements in dependency accuracy.

Instead of dependency grammar, we consider incremental parsing based on a variant of CCG grammar formalism. Specifically, we focus on converting normal form derivations in CCGbank to incremental derivation to facilitate the training of incremental CCG parsers.

Hassan et al. [2] recently introduced an approach to obtain incremental CCG derivations by assigning a CCG operator to each word in the sentence, based on the gold dependency structure. However, because each word was tagged individually, the resulting operator sequence is not guaranteed to be applicable without heavy modifications to CCG operators allowing many non-standard exceptions. The dynamic programming approach we develop guarantees the generation of a set of combinatory rules that will fully parse the sentence without deviations from the standard CCG rules.

In a related work, Schuler et al. [9] introduced the right-corner transform that transforms context-free grammar (CFG) parsing trees to incremental trees that can be recognized by a bounded stack parser. This transformation utilized notions similar to those in CCG grammar, mainly the notion of incomplete constituents that contain subcategories to be satisfied. Schuler [8] also introduced a model-based transformation that transforms an entire probabilistic CFG (PCFG) grammar into a hierarchical hidden Markov model (HHMM) recognizer.

The approach we follow in this paper is different from that followed by Hassan et al. [2] and Schuler et al. [9]. Instead of specifying conversion rules to convert gold derivations to incremental derivations, we augment CCG with additional combinatory rules to facilitate efficient incremental parsing. Then, a dynamic programming algorithm searches for incremental derivations of the non-incremental sentences in CCGbank without prior knowledge or conversion rules. We show that, with little guidance from gold derivations, an incremental version of CCGbank can be obtained with the dependency information largely preserved.

3 Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG) [10] is a theory that assumes a lexicalized grammar consisting of a lexicon and a small set of combinatory rules. The combinatory rules assemble lexical entries together into parse-trees. The lexical entries consist of syntactic constructs (called categories or supertags) that describe such lexical information as the POS tag of the word, its sub-categorization information and the hierarchy of phrase categories that the word may project upwards in the parse-tree.

As a grammar formalism, CCG has its advantages over dependency grammar. First, due to the flexible and lexicalized nature of CCG, having an incremental CCG derivation provides a compact and at the same time rich representation of the parsing state that can be used as a feature in other systems [3], specifically, the parsing state is represented by at most two CCG categories. Second, CCG combinatory rules have direct semantic interpretations which can be used to implement semantic parsing in addition to syntactic parsing as in [12].

CCG Grammar is an extension of categorial grammar, which assumes that syntactic constituents can combine in a function-argument relationship. A word or a span of words is represented by a *category* that can be either *atomic* such as NP (noun phrase) or *composite* such as $\mathsf{S\backslash NP_1}$ (intransitive verb phrase) or $\mathsf{(S\backslash NP_1)/NP_2}$ (transitive verb phrase). A composite category $\mathsf{X\backslash Y}$ represents a constituent X that seeks an *argument* of type Y to the left in order to form a constituent. A composite category $\mathsf{X/Y}$ represents a constituent X that needs, to be complete, an argument of type Y to the right, where X and Y can be atomic or composite categories.

Two CCG categories can be combined through *combinatory rules*. The most important rules defined by CCG grammar are listed below:

Forward Application

$$\mathsf{X/Y} \quad \mathsf{Y} \Rightarrow_{>} \mathsf{X}$$

Forward Composition

$$\mathsf{X/Y} \quad \mathsf{Y/Z} \Rightarrow_{>\mathbf{B}} \mathsf{X/Z}$$

Backward Application

$$\mathsf{Y} \quad \mathsf{X\backslash Y} \Rightarrow_{<} \mathsf{X}$$

Backward Composition

$$\mathsf{X\backslash Z} \quad \mathsf{Y\backslash X} \Rightarrow_{<\mathbf{B}} \mathsf{Y\backslash Z}$$

Backward Cross-composition

$$\mathsf{X/Z} \quad \mathsf{Y\backslash X} \Rightarrow_{<\mathbf{B}_\times} \mathsf{Y/Z}$$

An important unary rule is the type raising rule, defined as follows

Forward type raising

$$\mathsf{T} \quad \Rightarrow_{>\mathbf{T}} \mathsf{T/(T\backslash X)}$$

Backward type raising

$$\mathsf{T} \quad \Rightarrow_{<\mathbf{T}} \mathsf{T\backslash(T/X)}$$

A CCG derivation starts by assigning a category to each word in the input sentence. Combinatory rules are then used to combine these categories until a single root category is reached. In a CCG derivation, a CCG category represents a continuous span of words in the sentence. Categories in the leaves, called lexical categories, represent single words while the root category represents the whole sentence. The following example shows a CCG derivation of the sentence "*I met the manager*":

$$\begin{array}{ccccl}
I & met & the & manager & \\
\mathsf{NP}_{I} & (\mathsf{S[dcl]}_{met}\backslash\mathsf{NP_1})/\mathsf{NP_2} & \mathsf{NP}_X/\mathsf{NP}_X & \mathsf{NP}_{manager} & \\
\cline{3-4}
 & & \multicolumn{2}{c}{\mathsf{NP}_{manager}} & > \\
\cline{2-4}
 & \multicolumn{3}{c}{\mathsf{S[dcl]}_{met}\backslash\mathsf{NP_1}} & > \\
\cline{1-4}
\multicolumn{4}{c}{\mathsf{S[dcl]}_{met}} & <
\end{array}$$

The numeric subscripts in the composite category indicate *argument slots*. A category is also augmented with a *head reference* which refers to a word in the sentence. For example, the verb *met* in a sentence is represented associated with a category $(S_{met}\backslash NP_1)/NP_2$. This enables inference of dependency links from a derivation. Basically, if a category X fills an argument slot in another category Y, a predicate-argument dependency is induced between the head reference of X to the head reference of Y.

The head reference of a category might be a variable. Variables enable the recovery of long-range dependencies through unification. An example of a category containing variables is the relative pronoun $(NP_X\backslash NP_{1,X})/(S_2\backslash NP_X)$, in which the variable X indicates that the subject of the relative clause is the noun phrase that precedes the relative pronoun.

The set of induced predicate-argument dependencies form a *predicate-argument dependency structure*, which is a directed acyclic graph where each node represents a word in the input sentence and there is a link to each node from each argument of the corresponding lexical categories. An example of that structure is shown in figure 1.

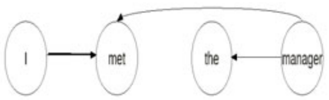


Fig. 1. Sample CCG dependency structure

Type-raising and composition introduce *spurious ambiguity* to CCG grammar; for a given sequence of words, multiple derivations can lead to equivalent results. For example, another derivation for the sentence "*I met the manager*" is shown here:

$$\begin{array}{cccc} \textit{I} & \textit{met} & \textit{the} & \textit{manager} \\ \mathsf{NP}_{I} & (\mathsf{S[dcl]}_{met}\backslash \mathsf{NP}_1)/\mathsf{NP}_2 & \mathsf{NP}_X/\mathsf{NP}_X & \mathsf{NP}_{manager} \end{array}$$

$$\overline{}{>}\mathbf{T}$$
$$\mathsf{S[dcl]/(S[dcl]\backslash NP)}$$
$$\overline{}{>}\mathbf{B}$$
$$\mathsf{S[dcl]}_{met}/\mathsf{NP}$$
$$\overline{}{>}\mathbf{B}$$
$$\mathsf{S[dcl]}_{met}/\mathsf{NP}$$
$$\overline{}{>}$$
$$\mathsf{S[dcl]}_{met}$$

Equivalent multiple derivations can be reduced to a normal form, where type raising and forward composition are performed only if needed. This notion of normal form, however, discourages incremental parsing because as shown in the previous derivation, type raising and forward composition can be used instead of application to obtain a left-to-right incremental parse. Therefore, to train incremental CCG parsers, the normal form derivations in CCGbank have to be transformed into incremental derivations.

Finally, an interesting property of CCG formalism is that it has direct semantic interpretation, based on Lambda calculus. For example, the forward application rule can be written as

$\mathsf{X/Y} : \lambda z.\, \mathsf{f(z)} \quad \mathsf{Y} : \mathsf{y} \Rightarrow \mathsf{X} : \mathsf{f(y)}$

This makes it simple to extend a CCG parser to a semantic parser.

4 Incremental CCG

4.1 Incremental Combinatory Rules

In this section, we describe our extension to Steedman's CCG by introducing combinatory rules to support efficient incremental left-to-right parsing. We propose an extended set of combinatory rules to increase the grammar coverage and the ability to recover long-range dependencies for a fully incremental left-to-right parsing. The set of CCG binary combinatory rules described in the previous sections are extended with the following rules to support incrementality:

CCGbank Binary Rules

CCGbank utilizes a set of non-CCG binary rules. We use this set of rules. An example of such rules is:

$$\mathsf{S[dcl]/S[dcl]} \quad , \Rightarrow \mathsf{S\backslash S}$$

Type-raising and CCGbank Unary Rules

CCGbank utilizes a set of unary type-changing rules and type-raising rules. We use this set of rules. An example of such rules is:

$$\mathsf{S[adj]\backslash NP} \Rightarrow \mathsf{NP\backslash NP},$$

which states that an adjective phrase can act as a noun phrase modifier. We permitted additional type-raising rules that are appropriate for incremental processing such as

$$\mathsf{NP} \Rightarrow \mathsf{S/(S\backslash NP)},$$

which is useful when parsing sentences that start with a sentence modifier $\mathsf{S/S}$.

Argument Swapping (AS)

This is a unary type changing rule that uses Lambek associativity axioms to change a category such that the direction of its outermost argument is to the left [1].

$$\mathsf{((X|Y|...|Z)\backslash W)/V/U.../T} \Rightarrow_{>\mathbf{AS}} \mathsf{((X|Y|...|Z)/V/U.../T)\backslash W}$$

This allows for combining two categories even if the right category has unfilled arguments to the right without the need to type-raise the left category. This is necessary

[1] The notation $\mathsf{X|Y}$ indicates that the rule applies to $\mathsf{X\backslash Y}$ and $\mathsf{X/Y}$.

to minimize the allowed type raising options in order not to expand the search space during parsing. The following derivation demonstrates argument swapping

$$\begin{array}{cccl}
\textit{He} & \textit{ate} & \textit{dinner} & \\
\mathsf{NP}_{He} & (\mathsf{S[dcl]}_{\mathsf{ate}}\backslash\mathsf{NP}_1)/\mathsf{NP}_2 & \mathsf{NP}_{dinner} & \\
\cline{2-2}
 & (\mathsf{S[dcl]}_{ate}/\mathsf{NP}_2)\backslash\mathsf{NP}_1 & & >\mathbf{AS} \\
\cline{1-2}
\multicolumn{2}{c}{(\mathsf{S[dcl]}_{ate}/\mathsf{NP}_2)} & & < \\
\cline{1-3}
\multicolumn{3}{c}{\mathsf{S[dcl]}_{ate}} & >
\end{array}$$

Backward Modification (BM)

Backward modification is introduced to allow incremental parsing of sentences containing back-modifiers such as adverbs. The problem with back-modifiers is that an explicit representation of their arguments may be lost in the incremental parsing process. Consider the sentence "*He ran quickly*". The phrase "*He ran*" resolves to a sentence category S. It is now required to combine it with the verb modifier "*quickly*", whose category is $(S\backslash NP)\backslash(S\backslash NP)$. Such combination is not supported by standard CCG. To handle this problem we define backward modification as follows: if $History(X, Y)$ exists then

$$\mathsf{X} \quad \mathsf{Y}\backslash\mathsf{Y} \Rightarrow_{>\mathbf{BM}} X\,,$$

where $History(\mathsf{X}, \mathsf{Y}) = \mathsf{Z}$ if $\mathsf{Z} = (...((\mathsf{Y}|\mathsf{Z}_1)|\mathsf{Z}_2)...)|\mathsf{Z}_\mathsf{N}$ for some $N \geq 0$ and Z is the lexical category of a word in the span of X. If there are multiple lexical categories in the span of X that satisfy these conditions, the right-most one is considered.

The effect of back-modification can be obtained using type-raising. By type-raising Z to $(...((\mathsf{Y}/(\mathsf{Y}\backslash\mathsf{Y})|\mathsf{Z}_1)|\mathsf{Z}_2)...)|\mathsf{Z}_\mathsf{N}$ and assuming, without loss of generality, that the arguments Z_1 through Z_N will be satisfied by words preceding the modifier, the left hand side would then be $X/(Y\backslash Y)$ instead of X. That new category can be combined with the modifier using standard forward application.

To illustrate this, consider the sentence "*He ate dinner quickly*". The backward modification rule can be used to parse this sentence as follows

$$\begin{array}{ccccl}
\textit{He} & \textit{ate} & \textit{dinner} & \textit{quickly} & \\
\mathsf{NP}_{He} & (\mathsf{S[dcl]}_{ate}\backslash\mathsf{NP}_1)/\mathsf{NP}_2 & \mathsf{NP}_{dinner} & (\mathsf{S}_X\backslash\mathsf{NP})\backslash(\mathsf{S}_{X,1}\backslash\mathsf{NP}) & \\
\cline{2-2}
 & (\mathsf{S[dcl]}_{\mathsf{ate}}/\mathsf{NP}_2)\backslash\mathsf{NP}_1 & & & >\mathbf{AS} \\
\cline{1-2}
\multicolumn{2}{c}{(\mathsf{S[dcl]}_{\mathsf{ate}}/\mathsf{NP}_2)} & & & < \\
\cline{1-3}
\multicolumn{3}{c}{\mathsf{S[dcl]}_{\mathsf{ate}}} & & > \\
\cline{1-4}
\multicolumn{4}{c}{\mathsf{S[dcl]}_{\mathsf{ate}} : \mathsf{S[dcl]}_{ate}\backslash\mathsf{NP}_1 \text{ satisfies } \mathsf{S}_{X,1}\backslash\mathsf{NP}} & >\mathbf{BM}
\end{array}$$

An equivalent result can be achieved using type-raising as follows

$$\begin{array}{cccc}
\textit{He} & \textit{ate} & \textit{dinner} & \textit{quickly} \\
\mathsf{NP}_{He} & (\mathsf{S[dcl]}_{ate}\backslash\mathsf{NP}_1)/\mathsf{NP}_2 & \mathsf{NP}_{dinner} & (\mathsf{S}_X\backslash\mathsf{NP})\backslash(\mathsf{S}_{X,1}\backslash\mathsf{NP}) \\
\cline{2-2}
 & \multicolumn{1}{r}{>\mathbf{T}} & & \\
 & ((\mathsf{S[dcl]}_{ate}\backslash\mathsf{NP}_1)/((\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP})))/\mathsf{NP}_2 & & \\
\cline{2-2}
 & \multicolumn{1}{r}{>\mathbf{AS}} & & \\
 & ((\mathsf{S[dcl]}_{ate}/((\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP})))/\mathsf{NP}_2)\backslash\mathsf{NP}_1 & & \\
\cline{1-2}
\multicolumn{2}{r}{<} & & \\
\multicolumn{2}{c}{(\mathsf{S[dcl]}_{ate}/((\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP})))/\mathsf{NP}_2} & & \\
\cline{1-3}
\multicolumn{3}{r}{>} & \\
\multicolumn{3}{c}{\mathsf{S[dcl]}_{ate}/((\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP}))} & \\
\cline{1-4}
\multicolumn{4}{r}{>} \\
\multicolumn{4}{c}{\mathsf{S[dcl]}_{ate}}
\end{array}$$

The problem with type-raising, however, is that it requires the parser to type-raise the modified category to the type that matches the modifier, which could be several words beyond. This would require the parser to have unbounded look-ahead memory or an exponentially growing beam that can maintain all type-raising options. Back-modification results in the same dependency structure obtained by type-raising while alleviating these difficulties.

4.2 Recursive Extensions

To facilitate incremental parsing, forward application and backward modification are recursively extended as follows:

$$\begin{array}{ll}
\textbf{if} & \mathsf{X} \quad \mathsf{Y} \Rightarrow \mathsf{Z} \\
\textbf{then} & \mathsf{X} \quad \mathsf{Y|W} \Rightarrow \mathsf{Z|W}
\end{array}$$

Similarly, unary rules are recursively extended as follows:

$$\begin{array}{ll}
\textbf{if} & \mathsf{X} \Rightarrow \mathsf{Z} \\
\textbf{then} & \mathsf{X|W} \Rightarrow \mathsf{Z|W}
\end{array}$$

4.3 Backward Modification with Argument Replacement

In incremental parsing, the argument fillers may change as the parse continues. For example, in the incremental parse of the sentence "*I met John's friend*", when parsing the subsequence "*I met John*", the noun $John$ will, incorrectly, fill the object argument of the verb met, creating an erroneous dependency. To handle such a situation, the apostrophe is tagged as $(\mathsf{NP}_X\backslash\mathsf{NP}_{Y:Y\rightarrow X})/\mathsf{NP}_X$. The $Y \rightarrow X$ notation means that any dependency whose argument is Y will have it changed to X. Thus, when the apostrophe back modifies the noun $John$, the dependency between met and $John$ will be replaced by a dependency between met and $friend$, correcting the dependency structure. An argument may be replaced by a set of arguments, thus replacing a single dependency with multiple dependencies. This happens in the case of conjunction, where a conjunction category $conj$ is changed to $(\mathsf{Z}_X\backslash\mathsf{Z}_{X:Y\to\{X,Y\}})/\mathsf{Z}_X$, indicating that for any dependency involving X as an argument, an additional dependency involving Y as the argument is created.

Some dependencies, however, are not affected by argument replacement such as the dependency between a definite article and its noun argument, as in "*I met the manager's friend*" where the dependency between the article the and its noun argument $manager$ is correct regardless of the presence of the possessive apostrophe. Whether a dependency is to be affected by argument replacement is encoded in the category. For example the category corresponding to the definite article is $\mathsf{NP}_X/\mathsf{N}_{X,1*}$, where $1*$ indicates that the dependency between argument slot 1 and its filler will not be affected by argument replacement.

5 Transforming CCG Normal Form Derivations into Incremental Derivations

In this section we describe our novel dynamic programming technique for transforming CCG normal form derivations into incremental derivations. The result of the transformation process is an incremental CCGbank where sentences are annotated with lexical categories as well as combinatory rules that allow left-to-right, incremental parses satisfying the same dependency relations specified in the normal form derivations of the CCGbank.

We introduce a dynamic programming transformation algorithm based on a constrained variation of CYK chart parsing. CYK operates on a chart in which the cell (i, j) corresponds to the span covering words i through j. Each cell in the chart stores possible CCG categories for the corresponding span. Each category holds pointers to parent categories and the rule that combines them to obtain that category. Thus, each category member is the root of the derivation sub-tree covering the corresponding span. The parsing algorithm starts by assigning CCG categories to diagonal cells, which correspond to single word spans. Then, bottom-up parsing is carried out to try all possible combinatory rules to reach the cell that spans the whole sentence. The conversion process is outlined in algorithm 1.

Step 6 in the algorithm accounts for incrementality constraints. To ensure incrementality, the bottom-up parsing is subject to the following constraints:

1. The only permissible combinations are:
   - $(1, i)(i + 1, j) \to (1, j)$
   - $(i, j - 1)(j, j) \to (i, j)$
2. The combination $(i, k)(k + 1, j) \to (i, j)$ is allowed only if there is no dependency arc between words x and y such that $x < k + 1$ and $k + 1 \leq y < j$.

The first constraint allows the parser to independently process a segment of the sentence starting from $i > 1$ (we call this a *fork* operation) until it reaches a category that spans the range (i, j) then combines the spans $(1, i - 1)$ and (i, j) (we call this a *join* operation). The constraint, however, prevents nested forks, thus facilitating parsing with a bounded stack. The constraint ensures that the parsing state can be represented by at most two categories, the category obtained before the fork (if there is) and the category

Algorithm 1. Linear parse generation

```
1: for $i = 1$ to $N$ do
2:     Add CCG category corresponding to word $i$ to $(i, i)$
3: for $j = 2$ to $N$ do
4:     for $i = j - 1$ to $1$ do
5:         for $k = i + 1$ to $j$ do
6:             if $(i, k - 1)(k, j) \rightarrow (i, j)$ is allowed then
7:                 for all category pairs $(c_1, c_2)$ in $(i, k - 1) \times (k, j)$ do
8:                     for all possible combinatory rules $r$ do
9:                         Add $r(c_1, c_2)$ to $(i, j)$
10: Find a category $c$ in $(1, N)$ matching the root category of the non-incremental parse
11: if $c$ does not exist then
12:     print "Conversion Failed"
13: else
14:     Traverse tree rooted by $c$, printing combinatory rules
```

obtained from the segment processed so far. The fork/join operation is analogous to the interrupt operator used by [2] to handle appositions and interruptions.

The second constraint states that if a segment of the sentence is to be processed by a fork/join operation, processing cannot exceed a dependency link between this segment and the part of the sentence before it. The constraint thus prevents forks that cause the intermediate dependency structure to have multiple disconnected components while the corresponding true dependencies form a connected graph.

Figure 2 visualizes how the conversion algorithm obtains an incremental parse for the sentence "*I met the manager*". To simplify the chart, only combinations that correspond to forward application/composition and backward application operators are shown. For each entry, the figure shows the corresponding category, the operator that resulted in this category, pointers to parent entries that were combined by that operator and a time step (to visualize the order of obtaining these entries).

An example for a violation for each constraint is seen in the figure. The combination at time step 5 violates the first incrementality constraint; it would require combining the verb met, whose span does not start at the beginning of the sentence, with the phrase "*the manager*" that spans multiple words. The combination at time step 2 is also invalid, since it violates the second incrementality constraint; there is a dependency link between the verb met and its subject I which prevents combining the verb alone with words to the right. Otherwise, until the dependency between met and I is resolved, the intermediate dependency graph will contain at least two disconnected components, the node I and the verb phrase starting with met. Thus, the phrase "*met the manager*" cannot be processed independently by a fork/join operation.

The figure shows that there are two identical entries in the cell (1,4). Thus, we have two possible incremental parses. The first one, whose time step is 6, is selected since it does not involve any fork/join operations. In general, different sub-derivations may lead to identical categories in some cells. In such case, only one of them is kept; we keep the sub-derivation which contains the least number of forks. Tie breaking is done by selecting the derivation in which the length of the latest fork is shorter. This way, generated parses use forking only as necessary.

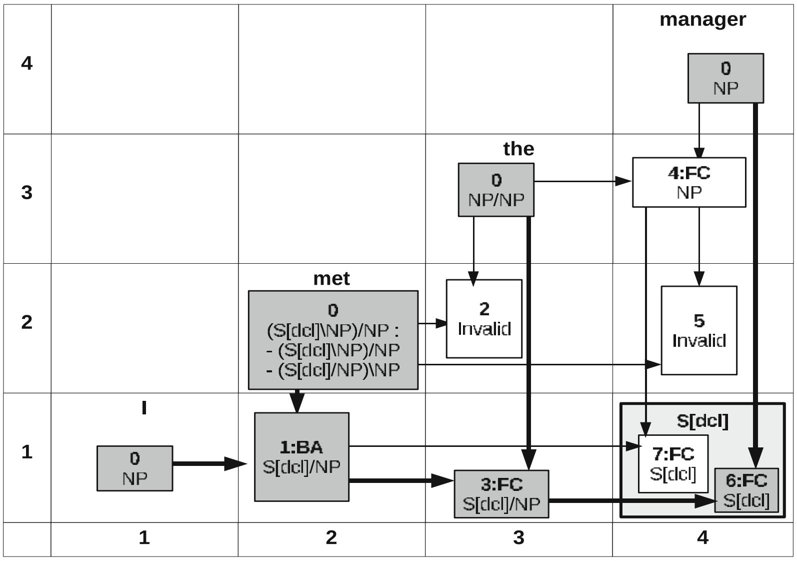


Fig. 2. Linear parse example. Entries that will be included in the selected linear parse tree are shaded.

By traversing the tree whose root is the selected entry, the incremental parse can be expressed as the following sequence of operations: (START, Swap arguments of the next input word, BA, FC, FC). In general, any linear parse tree can be expressed as a sequence of the following operations:

– Binary combinatory rule
– Unary rule on the parsing state category
– Unary rule on the lexical category of the next word
– Fork
– Join by a binary combinatory rule

There is one case in which we allow the violation of the second constraint, which is the case of relative clauses that result in a S/NP such as "*The gift that* ***he has given***". In this case the relative clause can be processed to end up with the S/NP even though there may be dependencies between the relative pronoun, *that*, and words within the clause. We will refer to this exception as *relative clause exception*.

6 Experimental Results

To evaluate the conversion algorithm, we converted CCG derivations in sections 00 through 21 of the CCGbank to incremental derivations [2]. To keep the conversion process manageable, CCG categories corresponding to commas and conjunction words

[2] The resulting incremental CCG derivations together with the inferred dependency structures can be downloaded from `http://www.CICLing.org/2011/software/128`

where determined from CCGbank derivations. Dependency structures were then inferred from incremental derivations, according to resolution rules defined in section 4, and compared to gold dependency structures. In our evaluation, we are interested in coverage (the percentage of sentences for which an incremental derivation could be obtained), dependency precision , recall and f-measure, and the percentage of correctly parsed sentences (sentences for which the inferred dependency structure is identical to the gold dependency structure). Table 1 summarizes the results for three configurations. The first configuration ,ICCG, uses the proposed extended combinatory rules. However, forking is not allowed and hence strict incremental parsing is enforced. The second configuration, ICCG+forks, allows forking as described in section 5. The third configuration, ICCG+fork+RC exceptions, allows for the relative clause exception described in section 5.

The results show that our proposed rules achieve very high coverage when forking is allowed. As coverage increases, dependency accuracy in terms of precision, recall, and f-measure decreases a little. This is expected because by relaxing incrementality constraints, longer and more complicated sentences can be incrementally parsed. It is, however, more difficult to correctly resolve dependencies in those sentences due to ambiguities and use of complex constructs. By examining the resulting dependency structures, we have found that most of the disagreements with gold dependencies are due to the assumption in section 4.3 that whether a dependency is affected by argument replacement depends on the involved categories. In fact, it is tightly related to the actual semantics of the sentence, which can be ambiguous and require background knowledge to resolve. Using word-word dependency statistics or additional semantic resources can reduce the problem.

With more than 96% f-measure, the results indicate clearly that our framework can provide an incremental CCG for efficient incremental parsing with a small loss of accuracy.

Table 1. Experimental Results

	Coverage	Precision	Recall	F-measure	Correct Sentences
ICCG	0.8023	0.9594	0.9681	0.9637	0.5122
ICCG+forks	0.9494	0.9588	0.9675	0.9632	0.4931
ICCG+forks+RC exceptions	0.959	0.9585	0.9670	0.9628	0.4894

7 Conclusion and Future Work

We have extended the standard CCG to support incremental parsing by introducing a number of combinatory rules to support efficient incremental left-to-right parsing. We have introduced extensions to CCG combinatory rules as well as additional category information to facilitate incremental parsing by overcoming the incrementality limitations of standard CCG such as back modifiers and argument replacement. Moreover, we have introduced an incremental version of CCGbank by transforming the normal form CCG derivations into left-to-right derivations and have shown that the incremental derivations can recover predicate-argument dependency structures with more than 96% f-measure. These incremental CCG derivations should facilitate the training of efficient incremental CCG parsers.

References

1. Clark, S., Curran, J.R.: Wide-coverage efficient statistical parsing with CCG and log-linear models. Comput. Linguist. 33(4), 493–552 (2007)
2. Hassan, H., Sima'an, K., Way, A.: Lexicalized semi-incremental dependency parsing. In: Proceedings of RANLP 2009 (2009)
3. Hassan, H., Sima'an, K., Way, A.: A syntactified direct translation model with linear-time decoding. In: EMNLP 2009: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pp. 1182–1191. Association for Computational Linguistics, Morristown (2009)
4. Hockenmaier, J., Steedman, M.: CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. Comput. Linguist. 33(3), 355–396 (2007)
5. Nivre, J.: Incrementality in deterministic dependency parsing. In: IncrementParsing 2004: Proceedings of the Workshop on Incremental Parsing, pp. 50–57. Association for Computational Linguistics, Morristown (2004)
6. Nivre, J.: Algorithms for deterministic incremental dependency parsing. Comput. Linguist. 34(4), 513–553 (2008)
7. Sagae, K., Lavie, A.: A best-first probabilistic shift-reduce parser. In: Proceedings of the COLING/ACL on Main Conference Poster Sessions, pp. 691–698. Association for Computational Linguistics, Morristown (2006)
8. Schuler, W.: Positive results for parsing with a bounded stack using a model-based right-corner transform. In: NAACL 2009: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 344–352. Association for Computational Linguistics, Morristown (2009)
9. Schuler, W., Miller, T., AbdelRahman, S., Schwartz, L.: Toward a psycholinguistically-motivated model of language processing. In: COLING 2008: Proceedings of the 22nd International Conference on Computational Linguistics, pp. 785–792. Association for Computational Linguistics, Morristown (2008)
10. Steedman, M.: The Syntactic Process. MIT Press, Cambridge (2000)
11. Yamada, H., Matsumoto, Y.: Statistical dependency analysis with support vector machines. In: Proceedings of IWPT, pp. 195–206 (2003)
12. Zettlemoyer, L.S., Collins, M.: Online learning of relaxed CCG grammars for parsing to logical form. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007), pp. 678–687 (2007)

Dependency Syntax Analysis Using Grammar Induction and a Lexical Categories Precedence System[*]

Hiram Calvo[1,2], Omar J. Gambino[1], Alexander Gelbukh[1,3], and Kentaro Inui[2]

[1] Center for Computing Research, IPN,
Av. J. D. Bátiz e/ M.O. de Mendizábal, México, D.F., 07738. México
[2] Computational Linguistics, Nara Institute of Science and Technology,
Takayama, Ikoma, Nara 630-0192, Japan
[3] Faculty of Law, Waseda University,
Nishi-Waseda 1-6-1, Shinjuku-ku, Tokyo 169-8050, Japan
hcalvo@cic.ipn.mx, omarjg82@gmail.com,
www.gelbukh.com, inui@is.naist.jp.
http://www.diluct.com

Abstract. The unsupervised approach for syntactic analysis tries to discover the structure of the text using only raw text. In this paper we explore this approach using Grammar Inference Algorithms. Despite of still having room for improvement, our approach tries to minimize the effect of the current limitations of some grammar inductors by adding morphological information before the grammar induction process, and a novel system for converting a shallow parse to dependencies, which reconstructs information about inductor's undiscovered heads by means of a lexical categories precedence system. The performance of our parser, which needs no syntactic tagged resources or rules, trained with a small corpus, is 10% below to that of commercial semi-supervised dependency analyzers for Spanish, and comparable to the state of the art for English.

1 Introduction

There are mainly two approaches for creating syntactic dependencies analyzers: supervised and unsupervised. The main goal of the first approach is to attain the best possible performance for a single language. For this purpose, a great collection of resources is collected (manually annotated corpora with part of speech annotation, and syntactic and structure tags) which requires a great effort and years to be collected. For this approach the state of the art is around of 85% of syntactic annotation of full sentences in several languages (Sabine and Marsi, 2006), getting over 90% for English. On the other hand, the unsupervised approach tries to discover the structure of the text using only raw text. This would allow creating a dependency

[*] We thank the support of SNI, SIP-IPN, COFAA-IPN, and PIFI-IPN, CONACYT; and the Japanese Government; the first author is a JSPS fellow. The third author is a Visiting Scholar at Waseda University and specifically acknowledges support of SIP-20100773 grant, CONACYT 50206-H grant, and CONACYT scholarship for Sabbatical stay 2010.

Table 1. The NAIST Supervised Dependency Syntax Analyzer's precision

Language	Precision	Language	Precision
Arab	65.19	Portuguese	85.07
Chinese	84.27	Slovene	71.42
Czech	76.24	Spanish	80.46
Danish	81.72	Swedish	81.08
Dutch	71.77	Turkish	61.22
German	84.11	Bulgarian	86.34
Japanese	89.91		

analyzer for virtually any language. In this paper we explore this second approach. We present the model of an unsupervised dependency analyzer, named DILUCT-GI (GI from Grammar Inference). We propose adding morphological information before the grammar induction process, and after the grammar induction process, converting the shallow parsing to dependencies by reconstructing unavailable dependency information from the grammar inductors by means of a lexical categories precedence system, in a simpler fashion compared to previous works that use complex rule systems (Robinson, 1967; Gelbukh *et al.* 2005; Civit *et al.*, 2006).

In the following sections we present an overview of Syntactic Analyzers (Section 2), the implementation of our System (Section 3), description of the method for converting from Constituent Chunks to Dependencies (Section 4) introducing the Lexical Categories Precedence Hierarchy (Section 4.1); evaluation for Spanish (Section 5), and for English (Section 5.1) and finally our Conclusions and Future Work (Section 6).

2 Overview of Syntactic Analyzers

Recent syntactic analyzers use a manually tagged corpus as training for the learning grammar structures similar as those found in such corpus (Charniak, 1997). Some analyzers try to learn different grammar rules than those used in the training corpus; however, it has been shown that usually these analyzers are less successful than the previous mentioned ones (Briscoe and Waegner, 1993; Pereira and Schabes, 1992).

Most of the mentioned analyzers use statistical techniques. Different probabilities are assigned to all the possible representation of a sentence; then they select the most probable and present it as the correct representation. In the following sections we present a state of the art of supervised, semi-supervised and unsupervised syntactic analyzers.

2.1 Supervised Syntactic Analysis

Most of the state of the art for supervised syntactic analyzers was established in the shared task of CONLL-X (Buchholz and Marsi, 2006) where 19 analyzers for 13 different languages were presented. The vast majority of these analyzers was based on treebanks, and used different machine learning methods amongst which we can mention Support Vector Machines and EM algorithms. As an example we can mention the NAIST multilingual dependency analyzer from NAIST (Cheng *et al.*, 2006), which had the following results.

The state of the art for English (Penn Treebank converted to dependencies) corresponds to transition based systems, such as NAIST's (Yamada and Matsumoto, 2003), graph-based algorithms (McDonald *et al.*, 2005b, 2006), ensemble parsers (Sagae and Lavie, 2006; McDonald *et al.* 2005a), and phrase structure based analyzers, such as those by Collins *et al.*, (1999) and Charniak (2000). Semi-Supervised Dependency Syntax Analysis.

Connexor is a Semi-Supervised Syntactic Dependency Analyzer commercially available for several languages (English, Spanish, French, German, Swedish and Finnish). This analyzer is based on the functional dependency grammar approach developed by Tapanainen and Järvinen (1997). This analyzer is composed of three elements: Lexicon; Morphologic disambiguation module focused on sub-categorical information such as person, gender and number; and FDG.

Another semi-supervised syntactic analyzer is DILUCT (Calvo and Gelbukh, 2006). This algorithm uses heuristics for discovering relationships between words, as well as co-occurrence statistics learnt in an unsupervised way for PP-attachment resolution.

2.2 Unsupervised Syntax Analysis

This approach is relatively new, being one of the seminal works the one of Yuret (1998). Subsequently several works have been presented, such as grammar bigrams (Paskin, 2001), Top-down generative models (Klein and Manning, 2004), contrastive estimation (Smith and Eisner, 2005) and non-projective examples (McDonald *et al.*, 2007) applied to certain phenomena of adjunction for syntax analysis.

Gorla *et al.* (2007) describe two proposals for elaborating an unsupervised dependency analysis system; however, to our knowledge, this system is still under development. Our proposal differs completely from theirs. Cohen *et al.* (2008) use Bayesian parameter estimation for unsupervised dependency analysis, obtaining a precision of adjunction of 59.4% for sentences shorter than 10 words, 45.9% for sentences shorter than 20 words, and 40.5 for all sentences, using Minimum Bayes Risk for English. The grammar they obtain is a probabilistic grammar trained and tested with Part of Speech Tags only.

3 Implementation

Several steps are needed for creating a Dependency Analyzer using the Grammar Inference algorithms described previously. We describe this process in four stages:

1. Adding morphological tags to improve the performance of the inductors. (Section 4.1)
2. Grammar Induction. We used the grammar induction tools ABL (Section 4.2) and EMILE (Section 4.3), both open source and freely available from their author's page[1].
3. Parameter tuning of grammar inductors. (Section 4.4)
4. Convert the output of the grammar inductor (a shallow parse) in dependency relations. We propose doing this conversion using a simple algorithm based on lexical categories precedence. (see Section 5)

[1] `ilk.uvt.nl/~menno/research/software/abl` and `staff.science.uva.nl/~pietera/Emile/`

In the following subsections we describe details of each one of this stages. We will illustrate our procedure using the Spanish CAST-3LB corpus (Civit *et al.*, 2003) as a means of exemplification, however our approach should work for any language requiring only a PoS tagger as an external resource.

3.1 PoS Tagging of Raw-Text

CAST-3LB is a Spanish Dependency Tagged corpus. We use this corpus to be able to compare afterwards with the annotated version as a gold standard; however, to simulate a real situation, our algorithm was given access only to the raw text of this corpus.

We generated a PoS-tagged version of raw CAST-3LB corpus using the TnT Tagger (Brants, 2000) trained on the Spanish CLiC-TALP Corpus[2]. The TnT Tagger has been shown to have a performance of 94% for these settings (Morales-Carrasco and Gelbukh, 2003). The benefits of adding morphological tags information prior to Grammar Induction was shown in Calvo and Gambino (2007).

3.2 Grammar Inductor's Parameter Selection

In order to find the best parameters of these inductors, we tested the similarity of the output of the inductors with regard to the gold standard CAST-3LB. We compared the location of opening and closing parentheses. For example, the following sentence ("we cannot remember either why they came") shows the original CAST-3LB chunking and a sample output after grammar induction.

CAST-3LB:

(tampoco recordamos (**(**por qué**)** llegaron).**)**

Grammar Inductor:

(tampoco (recordamos **(**por qué**)** llegaron.)**)**

The first and third opening parentheses are in the same position, as well as the first and third closing parentheses (shown in bold). From here we computed the Recall, Precision and F-score measures as follows. Note that these measures were used only for parameter selection, in this middle stage.

$$\text{Recall} = \frac{\text{\# of coincident parenthesis}}{\text{Total \# of parenthesis in Gold Standard}}$$

$$\text{Precision} = \frac{\text{\# of coincident parenthesis}}{\text{Total \# of parenthesis in Induction}}$$

F-Score combines recall and precision into one score. We selected so β=1 so that recall and precision are equally weighed.

$$F_\beta = \frac{(\beta^2+1) * \text{Precision} * \text{Recall}}{(\beta^2 * \text{Precision}) + \text{Recall}}$$

[2] `http://clic.fil.ub.es`

Table 2. ABL with different parameters test

Corpus	Recall	Precision	F-score
Parameters	Alignment method: **Biased,** Selection method: **Branch**		
Raw	17.58%	21.19%	19.22%
Raw+PoS	**17.60%**	21.27%	19.26%
Parameters	Alignment method: **Biased,** Selection method: **Leaf**		
Raw	14.27%	26.21%	18.48%
Raw+PoS	14.56%	26.63%	18.82%
Parameters	Alignment method: **Default,** Selection method: **Branch**		
Raw	16.88%	23.64%	**19.69%**
Raw+PoS	16.96%	23.50%	19.70%
Parameters	Alignment method: **Default,** Selection method: **Leaf**		
Raw	11.69%	**31.24%**	17.01%
Raw+PoS	12.39%	**31.24%**	17.74%

EMILE provides the following selection of parameters:

1. **total_support_percentage** of context / expression of a particular kind.
2. **expression_support_percentage** for an expression in a determined context
3. **context_support**_percentage of appearances in a context along with expression of certain kind.
4. **rule_support_percentage** of characteristic expressions for a type that can be substituted by one of the referred types in the rule. A rule will be incorporated to the grammar only if this percentage is exceeded.

Table 3 and Table 4 show the performance obtained with different parameters. We show the best 4 F scores, the default (in italics), and the worse 4 F-scores—note that however, precision is the highest in one of the cases.

ABL provides the following alignment methods: default, biased, all; and the following selection methods: first, leaf, branch. Table 2 shows the results of testing with different parameters.

Table 3. Parameter selection for EMILE (no PoS Tags, i.e., raw text only)

1	2	3	4	Recall	Prec.	F
50	20	20	25	**9.75%**	53.72%	**16.51%**
60	30	30	30	9.72%	54.17%	16.49%
40	40	40	20	9.71%	53.95%	16.46%
50	40	40	25	9.68%	54.39%	16.44%
75	*50*	*50*	*50*	*9.53%*	*55.06%*	*16.25%*
50	30	30	25	9.47%	**54.90%**	16.16%
70	30	30	35	7.50%	42.71%	12.76%
80	50	50	40	7.47%	42.84%	12.72%
70	50	50	35	7.46%	42.91%	12.71%

Table 4. Parameter selection for EMILE (using PoS Tags, *i.e.,* raw + PoS)

1	2	3	4	Recall	Prec.	F
70	70	70	35	9.55%	**54.96%**	**18.91%**
50	20	20	25	**9.80%**	53.78%	16.57%
60	20	20	30	9.69%	54.43%	16.45%
70	20	20	35	9.67%	54.58%	16.42%
70	60	60	35	9.45%	54.75%	16.11%
75	*50*	*50*	*50*	*9.40%*	*53.06%*	*15.97%*
70	30	30	35	7.40%	42.61%	12.61%
70	50	50	35	7.30%	42.55%	12.46%

4 From Chunks to Dependency Relations

The CAST-3LB, as well as the output of the grammar inductors can be regarded as chunks of constituents. In this section we explore a simple mechanism for transforming these constituent chunks to dependencies. Let us review first some considerations concerning this conversion.

Ninio (1996) points out that the relation between constituents and dependencies is formally weak. Grammar relations are primary to a dependency grammar and as such, they do not have a role within the dependency approach. However, relations can be derived from one representation to the other one. Marneffe *et al.* (2006) generate typed dependency trees from constituent trees using a constituent grammar analyzer. They identify the constituent heads afterwards following the rules proposed by Collins *et al.* (1999).

Robinson (1967) points that one important difference between both representations is that the dependency approach uses only terminal categories, while the constituents approach uses categories of a higher degree. Despite of this, there is a systematic correspondence between the trees produced by each approach. The author proposes a series of rules by which it is possible to convert from one representation to the other.

Gelbukh *et al.* (2005) propose a procedure based in heuristics coded as 15 rules for marking the head, to convert a constituent corpus in a dependency corpus estimating an accuracy of 95%. Civit *et al.* (2006) propose a similar method based on rules linguistically motivated, which are encoded in a *head table*, but they do not provide an evaluation.

In this work we look for a simple yet effective way for doing such conversion, given that we do not have constituent tags available, but only PoS tags.

4.1 Lexical Categories Precedence

Hengeveld (1992) suggests that there exist common lexical hierarchies amongst the majority of languages, including the flexible and not flexible ones, referring to the linguistic regularity of such languages; however, he does not mention its application for syntactic analysis. On the other hand, Genthial *et al.* (1990) suggest the existence of a Lexical Categories (LC) hierarchy for the construction of syntactic structures. When applying them, however, they code them into rules, in a similar way as Gelbukh *et al.* (2005) and Civit *et al.* (2006).

We propose using a LC hierarchy to determine the head for dependency analysis starting from a shallow parse. Our procedure is described in the following pseudo-algorithm.

```
 function convert (syntactic groups, head)
1. get the most deep-nested syntactic group
2. obtain words and LC from this syntactic group
3. compare the LC of the group
4. mark the word with highest LC precedence as head of
   this group
5. mark other words as dependent
6. convert(rest of syntactic groups, head of the group)
 end function.
```

In order to obtain a correct LC hierarchy, we used the original syntactic groups found in the CAST-3LB as a gold standard. Iteratively we adjusted the LC hierarchy until a representative sample group of sentences of the gold standard were parsed correctly. The LC hierarchy we obtained follows, listed by highest (1) to lowest (12) precedence. The symbols in parenthesis correspond to the 3LB tagging system. However, as we will show later, this hierarchy can be easily adapted to a different tagging system.

Table 5. Comparison of our system (DILUCT-GI) with other systems

	Measures			Resources			
System	Recall	Precision	F-meas.	Dictio naries	Rules	Syntactic annotations	Morphol. annotatio ns
DILUCT	55.0%	47.0%	51.0%	✓	✓	✓	✓
Connexor	42.1%	39.6%	40.8%	✓	✓	✓	✓
DILUCT-GI	31.8%	32.3%	32.1%				✓
TACAT	30.0%	—	—	✓	✓		

Verb (v), Adverb (rg), Noun (n), Adjective (a), Pronoun (p), Negation (rn), Subordinated conjunction (cs), Preposition (s), Determiner (d), Coordinated conjunction (c), Interjection (i), Punctuation symbols (f).

We found that additional information conveyed by the tags was not necessary for the correct identification of the position in the hierarchy. *i.e.*, `vmis3s0` is simplified to `v`. Additional information such as person, gender, number or tense is discarded for verbs. This simplification is done for every PoS.

5 Evaluation of Dependencies

Briscoe *et al.* (2002) suggest evaluating the accuracy of syntactic analyzers based on the grammar relationships between lemmatized lexical heads. Each tree can be represented as a *n*-ary representation with *n* triplets: each dependency relationship has

Table 6. DILUCT-GI dependency analyzer results for the Susanne corpus

Corpus	Recall	Precision	Fscore
Parameters	Alignment method: Biased		
	Selection method: Branch		
Raw+PoS			**40.37%**
Parameters	Alignment method: Default		
	Selection method: Branch		
Raw+PoS		7	39.00%

a head, a dependant, and the kind of relationship between them. Following this, we compare two dependency analyses by comparing every triple in them. When we generated extra triplets, we counted them as errors, since they are outside of the gold standard.

CAST-3LB is a Spanish corpus with 3,700 tagged sentences. The best scores we obtained were 31.83% recall, 32.36% precision and 32.09% F-measure, using ABL, Text and PoS tags using Default alignment mode and Branch selection method. Table 5 compares these results with other semi-supervised dependency analyzers: DILUCT (Calvo and Gelbukh, 2008) and TACAT (Atserias and Rodríguez, 1998). TACAT is a shallow syntactic parser for Spanish and results were converted to dependencies for comparison[3]. The compared analyzers use resources such as dictionaries, rules, and syntactic annotations, whereas our proposal uses only morphological annotations, so that this comparison might be unfair, however, at our knowledge, presently there is no unsupervised dependency analyzer for Spanish available for comparison.

5.1 Building an English Parser in a Few Days

In order to perform a fair comparison, we need to compare with an unsupervised method. For example, for English, the syntactic analyzer using unsupervised Bayesian parameter estimation by Cohen *et al.* (2008) obtains an adjunction precision of 59.4% for sentences shorter than 10 words, 45.9% for sentences shorter than 20 words, and 40.5% for all sentences. They infer a grammar based on PoS tags with no words and output is not a dependency tree. Gorla *et al.* (2007) did not report results of their unsupervised dependency analyzer.

We did an implementation of a parser for English based on the Susanne corpus (Sampson, 1995) consisting of 7,500 annotated sentences for English. Same as before, we used only the raw text and morphological tags of this corpus, ignoring all syntactic information as input for our syntactic analyzer, but used the annotation as gold standard.

The Susanne corpus is annotated following the approach called Susanne Analytic Scheme. Genabith *et al.* (2001) recommend converting a corpus to the Xbar notation to minimize the creation of CFG rules for grammar induction; therefore, we used the

[3] Results kindly provided by Jordi Atserias, Technical University of Catalonia.

Xbar annotated version of the Susanne corpus by Nick Cercone[4]. This corpus' morphological tags are different to other corpora; for example, pronouns are tagged as nouns and adjectives and adverbs are classified together. Based on the previous LC hierarchy we found, by testing again over sample sentences, we determined in short time the Susanne English LC Hierarchy as follows.

> Verb (V), Auxiliar verb (have, be, being), Auxiliar (Aux), Noun (N), Adjective_1 (Aeasy), Adjective_2 (A), Preposition (P), Determiner (D), Predeterminer (PreDet), Conjunction (C)

Results are shown in Table 6 for the best parameters found in Section 3.4.

6 Conclusions and Future Work

Although not directly comparable, the performance of our system obtained for English suggests that by using a bigger corpus for Spanish we may achieve better performance—the corpus for English has 7500 sentences, whereas 3LB for Spanish has only 3500.

For the purpose of Dependency Analysis, ABL had better performance. EMILE stores all context-expression pairs during the induction process in order to create a new non-terminal as part of the grammar. Because of this, van Zaanen and Adriaans (2001) consider that this inductor will obtain better results with a big corpus (more than 100,000 sentences.) On the other hand, ABL uses a greedy algorithm that stores all possible constituents found before selecting the best. This property allows ABL to have a better performance with small corpora.

We obtained better results using a combined corpus of words and tags. It was a relatively small, but constant difference in all configurations tested. The information provided by these tags helps during the alignment process to disambiguate some constituents that belong to several lexical categories.

We found that the ABL Grammar Inductor had better performance than the one reported by their authors, who tested with the Wall Street Journal Corpus in English (van Zaanen, 2002), obtaining 12% of recall. The Biased-Branch configuration for ABL obtained the highest recall (17.60%), while the configuration Default-Leaf obtained the highest precision (31.24%). As expected, the higher amount of syntactic groups found, the less precision they have.

We presented a model for dependency analysis, which can be reasonably easy adapted to other languages, based on unsupervised learning on raw text annotated with morphological tags. As far as we know, for Spanish this would be the first unsupervised (after adding PoS tags) dependency analyzer, whereas for English we achieved results within the state of the art. Despite having still margin for improvement, our proposed model alleviates some intrinsic limitations such as those taking place when learning with Grammar Inductors (Gold, 1967), by adding morphologic information before the induction process, and a novel system for converting a shallow parse into a dependency analysis by means of a Lexical Category Precedence Hierarchy. Our method can be used for languages where

[4] Available at `www.student.cs.uwaterloo.ca/~cs786s/susanne/`

linguistic resources are scarce, given that morphologic tags are available. We believe that at least romance languages share a similar lexical precedence hierarchy; however, proving this, as well as testing with other corpora, is left as future work.

References

Atserias, J., y Rodríguez, H.: TACAT: TAgged Corpus Text Analyzer, Technical Report LSI-UPC RT-2-98 (1998)

Brants, T.: TNT — A Statistical Part-of-Speech Tagger. In: ANLP 2000, 6th Applied NLP Conference, Seattle, Washington, USA (2000)

Briscoe, T., Carroll, J., Graham, J., Copestake, A.: Relational evaluation schemes. In: Procs. of the Beyond PARSEVAL Workshop at the 3rd International Conf. on Language Resources and Evaluation, Gran Canaria, pp. 4–8 (2002)

Briscoe, T., Waegner, N.: Generalized probabilistic LR parsing of natural language (corpora) with unification-based grammars. Computational Linguistics 19, 25–69 (1993)

Brooks, D.J.: Unsupervised Grammar Induction by Distribution and Attachment. In: Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X), pp. 117–124. Association for Computational Linguistics, New York City (2006)

Calvo, H., Gelbukh, A.: Automatic Semantic Role Labeling using Selectional Preferences with Very Large Corpora. Computación y Sistemas 12(1), 128–150 (2008)

Calvo, H., Gelbukh, A.: DILUCT: An Open-Source Spanish Dependency Parser based on Rules, Heuristics, and Selectional Preferences. In: NLDB 2006, pp. 164–175 (2006)

Charniak, E.: A Maximum-Entropy-Inspired Parser. In: NAACL 2000, pp. 132–139 (2000)

Charniak, E.: Statistical techniques for natural language parsing. AI Magazine 18, 33–43 (1997)

Civit, M., Antònia Martí, M., Bufí, N.: From Constituents to Dependencies. In: Salakoski, T., Ginter, F., Pyysalo, S., Pahikkala, T. (eds.) FinTAL 2006. LNCS (LNAI), vol. 4139, pp. 141–152. Springer, Heidelberg (2006)

Cohen, S.B., Gimpel, K., Smith, N.A.: Unsupervised Bayesian Parameter Estimation for Dependency Parsing. In: Advances in NIPS 22 (2008)

Collins, M.: Head-Driven Statistical Models for Natural Language Parsing, Ph.D. thesis, University of Pennsylvania (1999)

Dörnenburg, E.: Extension of the EMILE algorithm for inductive learning of context-free grammars for natural languages, Master's Thesis, University of Dortmund (1997)

Gambino, O.J., Calvo, H.: On the usage of morphological tags for grammar induction. In: Gelbukh, A., Kuri Morales, Á.F. (eds.) MICAI 2007. LNCS (LNAI), vol. 4827, pp. 912–921. Springer, Heidelberg (2007)

Gelbukh, A., Calvo, H., Torres, S.: Transforming a constituency Treebank into a dependency Treebank. Procesamiento Del Lenguaje Natural 35, 145–152 (2005)

Genabith, J., van, A., Frank, A.: Way. Treebank vs. Xbar-based Automatic F-Structure Anotation. In: Proceedings of the LFG 2001 Conference. University of Hong Kong, CSLI Publications, Hong Kong (2001)

Genthial, D., Courtin, J., Kowarski, I.: Contribution of a Category Hierarchy to the Robustness of Syntactic Parsing. In: COLING 1990, pp. 139–144 (1990)

Gold, E.M.: Language Identification in the Limit. Information and Control 10(5), 447–474 (1967)

Gorla, J., Goyal, A., Sangal, R.: Two Approaches for Building an Unsupervised Dependency Parser and their Other Applications. In: AAAI 2007, pp. 1860–1861 (2007)

Hengeveld, K.: Parts of speech. In: Fortescue, M., Harder, P., Kristoffersen, L. (eds.) Layered Structure and Reference in a Functional Perspective, Benjamins, Amsterdam, pp. 29–56 (1992)

Yamada, H., Matsumoto, Y.: Statistical dependency analysis with support vector machines. In: Procs. of the 8th International Workshop on Parsing Technologies (IWPT), pp. 195–206 (2003)

Klein, D., Manning, C.: Corpus-based induction of syntactic structure: Models of dependency and constituency. In: Proceedings of the ACL (2004)

de Marneffe, M.-C., MacCartney, B., Manning, C.D.: Generating Typed Dependency Parses from Phrase Structure Parses. In: Proceedings of LREC 2006 (2006)

McDonald, R., Lerman, K., Pereira, F.: Multilingual dependency analysis with a two-stage discriminative parser. In: Proceedings of the CoNLL (2006)

McDonald, R., Pereira, F., Ribarov, K., Hajič, J.: Non-projective dependency parsing using spanning tree algorithms. In: Proceedings of the HLT/EMNLP (2005a)

McDonald, R., Crammer, K., Pereira, F.: Online large-margin training of dependency parsers. In: Proceedings of the ACL (2005b)

McDonald, R., Satta, G.: On the complexity of non-projective data-driven dependency parsing. In: Proceedings of the IWPT (2007)

van Zaanen, M., Adriaans, P.: Alignment-Based Learning versus EMILE: A Comparison. In: Proceedings of the Belgian-Dutch Conference on Artificial Intelligence (BNAIC), Amsterdam, The Netherlands, pp. 315–322 (2001)

Morales-Carrasco, R., Gelbukh, A.: Evaluation of TnT Tagger for Spanish. In: Fourth Mexican International Conference on Computer Science ENC 2003, Tlaxcala, México, pp. 18–28 (2003)

Navarro, B., Civit, M., Antonia Martí, M., Marcos, R., Fernández, B.: Syntactic, semantic and pragmatic annotation in Cast3LB. In: Shallow Processing of Large Corpora (SProLaC), a Workshop of Corpus Linguistics, Lancaster, UK (2003)

Ninio, A.: A proposal for the adoption of dependency grammar as the framework for the study of language acquisition. Honor of Shlomo Kugelmass, pp. 85–103 (1996)

Paskin, M.A.: Cubic-time parsing and learning algorithms for grammatical bigram models, Technical Report, UCB/CSD-01-1148, Computer Science Division, University of California Berkeley (2001)

Paskin, M.A.: Grammatical Bigrams. In: Advances in Neural Information Processing Systems, vol. 14, MIT Press, Cambridge (2001)

Pereira, F., Schabes, Y.: Inside-outside reestimation from partially bracketed corpora. In: 27th Annual Meeting of the Association for Computational Linguistics, ACL, pp. 128–135 (1992)

Robinson, J.J.: Methods for obtaining corresponding phrase structure and dependency grammars. In: Proceedings of the 1967 Conference on Computational Linguistics, pp. 1–25 (1967)

Buchholz, S., Marsi, E.: CoNLL-X shared task on multilingual dependency parsing. In: Proceedings of the Tenth Conference on Computational Natural Language Learning, pp. 149–164 (2006)

Sagae, K., Lavie, A.: Parser combination by reparsing. In: Proceedings of the HLT/NAACL (2006)

Sampson, G.: English for the Computer, The SUSANNE Corpus and analytic scheme. Clarendon Press, Oxford (1995)

Smith, N., Eisner, J.: Guiding unsupervised grammar induction using contrastive estimation. In: Working Notes of the International Joint Conference on Artificial Intelligence Workshop on Grammatical Inference Applications (2005)

Tapanainen, P., Järvinen, T.: A non-projective dependency parser. In: Proceedings of the 5th Conference on Applied Natural Language Processing, Washington, D.C., pp. 64–71 (1997)

van Zaanen, M.: Bootstrapping Structure into Language: Alignment-Based Learning, PhD Thesis, School of Computing, University of Leeds (2002)

Cheng, Y., Asahara, M., Matsumoto, Y.: Multi-lingual Dependency Parsing at NAIST, CONLL-X, Nara Institute of Science and Technology (2006)

Yuret, D.: Discovery of linguistic relations using lexical attraction, Ph.D. thesis, MIT (1998)

Labelwise Margin Maximization for Sequence Labeling

Wenjun Gao, Xipeng Qiu, and Xuanjing Huang

School of Computer Science, Fudan University, China
{082024008,xpqiu,xjhuang}@fudan.edu.cn

Abstract. In sequence labeling problems, the objective functions of most learning algorithms are usually inconsistent with evaluation measures, such as Hamming loss. In this paper, we propose an online learning algorithm that addresses the problem of labelwise margin maximization for sequence labeling. We decompose the sequence margin to per-label margins and maximize these per-label margins individually, which can result to minimize the Hamming loss of sequence. We compare our algorithm with three state-of-art methods on three tasks, and the experimental results show our algorithm outperforms the others.

1 Introduction

In recent years, the sequence labeling problems have obtained much attention especially in the machine learning, computational biology and natural language processing communities such as part-of-speech tagging [16], chunking [17], named entity recognition [18] and Chinese word segmentation [26,15]. The goal of sequence labeling is to assign labels to all elements of a sequence. Due to the exponential size of the output space, sequence labeling problems tend to be more challenging than the conventional classification problems.

Recently, many algorithms have been applied for sequence labeling and the progress has been encouraging. These algorithms usually have the different objective functions. For example, average perceptron algorithm [1] aims to minimize the 0-1 loss of sequence. Maximum entropy Markov models (MEMM) [13] and conditional random fields (CRF) [11] aims to maximize the conditional likelihood. SVM^{struct} [23] and maximum margin Markov networks (M3N) [22] aims to maximize the margin or minimize hinge loss [5].

However, most of these objective functions often calculate the conditional probability or margin on the whole sequence, which are usually inconsistent with conventional evaluation measures of sequence labeling, such as Hamming loss. The probability and margin cannot response the Hamming loss directly.

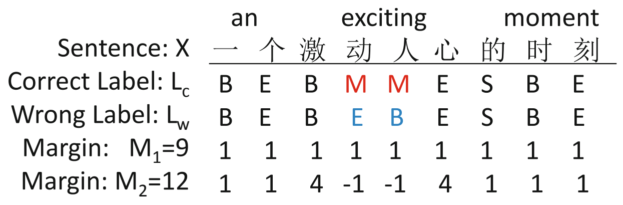


Fig. 1. Per-label decomposition of margin for Chinese word segmentation. “B”, “M”, “E” and “S”, which represent the beginning, middle, end or single character of a word respectively.

For example, in maximum margin algorithm for Chinese word segmentation (shown in Figure 1), given a sentence X and its label L_c, the margin is calculated between the correct label L_c and the wrong label L_w which has maximum score besides L_c. Assuming that we have two parameters $\mathbf{w}_1$ and $\mathbf{w}_2$, which can result in two different margins $M_1 = 9$ and $M_2 = 12$. M_1 and M_2 are the sum of per-label margins in each position. Although $\mathbf{w}_2$ is apparently better than $\mathbf{w}_1$ since $\mathbf{w}_2$ have larger margin, we claim that $\mathbf{w}_2$ is worse than $\mathbf{w}_1$. If we decompose the margin into individual positions, we find that M_2 contains negative value in per-label margin. The negative margin indicate a wrong predict, so $\mathbf{w}_1$ should be better than $\mathbf{w}_2$ since that $\mathbf{w}_1$ can give higher accuracy.

So we wish to find an objective function to approximate the evaluation measures of sequence labeling. However, it is difficult to find parameters that achieve the highest possible accuracy even on the training data. In particular, if we wish to minimize Hamming loss, which measures the number of incorrect labels, gradient-based optimization methods cannot be applied directly.

In this paper, we define the labelwise hinge loss to approximate the Hamming loss, which is the minimal differences between the score of the correct label and the closest negative. Then we propose an online learning algorithm that addresses the problem of per-label margin maximization for sequence labeling. Our learning algorithm is based on Passive-Aggressive (PA) algorithm [4], which passively accepts a solution whose loss is zero, while it aggressively forces the new prototype vector to stay as close as possible to the one previously learned. Our method maximize the labelwise margin instead of the separation margin of whole sequence.

The rest of the paper is organized as follows. Section 2 gives a brief introduction to sequence labeling models. Then we propose our algorithm in section 3 and compare it with CRF, M3N and PA in section 4. Section 5 gives the analysis and related words. Section 6 concludes the paper.

2 Sequence Labeling Models

Sequence labeling is the task of assigning labels $\mathbf{y} = y_1, \ldots, y_L$ to an input sequence $\mathbf{x} = x_1, \ldots, x_L$.

Give a sample $\mathbf{x}$, we define the feature is $\Phi(\mathbf{x}, \mathbf{y})$. Thus, we can label $\mathbf{x}$ with a score function,

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} F(\mathbf{w}, \Phi(\mathbf{x}, \mathbf{y})), \tag{1}$$

where $\mathbf{w}$ is the parameter of function $F(\cdot)$. The feature vector $\Phi(\mathbf{x}, \mathbf{y})$ consists of lots of overlapping features, which is the chief benefit of discriminative model.

For example, in first-order Markov sequence labeling, the feature can be denoted as $\phi_k(y_{i-1}, y_i, \mathbf{x}, i)$, where i is the position in the sequence. Then the score function can be rewritten as

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} F(\sum_{i=1}^{L} \sum_{k} w_k \phi_k(y_{i-1}, y_i, \mathbf{x}, i)), \tag{2}$$

where L is length of $\mathbf{x}$.

Different algorithms vary in the definition of $F(\cdot)$ and the corresponding objective function. $F(\cdot)$ is usually defined as linear or exponential family function.

In this section, we introduce several related sequence labeling methods.

2.1 Averaged Perceptron

The average perceptron algorithm for structured learning is described in [1,10]. In the training phase, given training examples with a random initial weight vector $\mathbf{w}$, the examples are iteratively processed. Let $\mathbf{y}$ be the true label sequence for input $\mathbf{x}$ and $\hat{\mathbf{y}}$ be the predicted label sequence. If a mistake is made, i.e, $\mathbf{y} \neq \hat{\mathbf{y}}$, the weight vector is updated as $w = \mathbf{w} + \Phi(\mathbf{x}, \mathbf{y})$. In order to avoiding overfitting, the averaging technology is employed.

2.2 Conditional Random Fields

Conditional random fields (CRF) are undirected graphical models trained to maximize a conditional probability [11]. When used for sequential labeling problems, a common special-case graph structure used is a linear chain. A linear-chain CRF with parameters $\mathbf{w}$ defines a conditional probability for a state sequence $\mathbf{y}$ given an input sequence $\mathbf{x}$ to be

$$P_{\mathbf{w}}(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_{\mathbf{w}}} \exp(\mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y})), \tag{3}$$

where $Z_{\mathbf{w}}$ is the normalization constant such that the sum of all the terms is one.

2.3 Maximum Margin Markov Networks

Maximum margin Markov networks (M3N) [22], which defines a log-linear Markov network over a set of label variables, represents the correlations between these label variables. They define a margin-based optimization problem for the parameters of this model.

$$\begin{aligned} \min \quad & \tfrac{1}{2}||\mathbf{w}||^2 + \mathcal{C} \textstyle\sum_{\mathbf{x}} \xi_{\mathbf{x}} \\ \text{s.t.} \quad & \mathbf{w}^T \Delta \mathrm{f}_{\mathbf{x}}(\mathbf{y}) >= \Delta \mathrm{t}_{\mathbf{x}}(\mathbf{y}) - \xi_{\mathbf{x}}, \forall \mathbf{x}, \mathbf{y} \end{aligned} \tag{4}$$

where ξ is non-negative slack variable.

For Markov networks that can be triangulated tractably, the resulting quadratic program (QP) has an equivalent polynomial-size formulation (e.g., linear for sequences) that allows a very effective solution.

2.4 Online Passive-Aggressive Algorithm

Passive-aggressive (PA) algorithm [3,4] was proposed for normal multi-class classification and can be easily extended to structure learning [2]. Like perceptron, PA is an online learning algorithm. Given an example $(\mathbf{x}, \mathbf{y})$, $\hat{\mathbf{y}}$ is denoted as the incorrect label with the highest score,

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{z} \neq \mathbf{y}} w^T \Phi(\mathbf{x}, \mathbf{z}). \tag{5}$$

The **margin** $\gamma(\mathbf{w}; (\mathbf{x}, \mathbf{y}))$ is defined as

$$\gamma(\mathbf{w}; (\mathbf{x}, \mathbf{y})) = \mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y}) - \mathbf{w}^T \Phi(\mathbf{x}, \hat{\mathbf{y}}). \tag{6}$$

Thus, we calculate the **hinge loss**.

$$\ell(\mathbf{w}; (\mathbf{x}, \mathbf{y}) = \begin{cases} 0, & \gamma(\mathbf{w}; (\mathbf{x}, \mathbf{y})) > 1 \\ 1 - \gamma(\mathbf{w}; (\mathbf{x}, \mathbf{y})), & \text{otherwise} \end{cases} \tag{7}$$

In round t, the new weight vector $\mathbf{w}_{t+1}$ is calculated by

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w} - \mathbf{w}_t||^2 + \mathcal{C} \cdot \xi, \qquad \textbf{s.t. } \ell(\mathbf{w}; (\mathbf{x}_t, \mathbf{y}_t)) <= \xi \textbf{ and } \xi >= 0 \quad (8)$$

where ξ is non-negative slack variable, and $\mathcal{C}$ is a positive parameter which controls the influence of the slack term on the objective function.

3 Online Labelwise Margin Maximization Algorithm

These above algorithm often calculate the conditional probability or margin on the whole sequence, which are usually inconsistent with conventional evaluation measures of sequence labeling, such as Hamming loss.

In this section we propose a per-label margin maximization algorithm based on online PA algorithm, which can approximate the Hamming loss.

To decompose the margin into each label, we need give a definition to labelwise margin firstly.

The score function of our algorithm is linear function.

Given an example $(\mathbf{x}, \mathbf{y})$, $\hat{\mathbf{y}}$ is denoted as the incorrect label with the highest score[1],

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{z} \neq \mathbf{y}} w^T \Phi(\mathbf{x}, \mathbf{z}). \quad (9)$$

Then the **labelwise margin** $\gamma_i(\mathbf{w}; (\mathbf{x}, \mathbf{y}))$ in position i is defined as

$$\gamma_i(\mathbf{w}; (\mathbf{x}, \mathbf{y})) = \mathbf{w}^T \Phi_i(\mathbf{x}, \mathbf{y}) - \mathbf{w}^T \Phi_i(\mathbf{x}, \hat{\mathbf{y}}), \quad (10)$$

where $\Phi_i(\mathbf{x}, \mathbf{y})$ is the local feature vector drawn from position i.

Thus, we calculate the **labelwise hinge loss** for each position i.

$$\ell_i(\mathbf{w}; (\mathbf{x}, \mathbf{y}) = \begin{cases} 0, & \gamma_i(\mathbf{w}; (\mathbf{x}, \mathbf{y})) > 1 \\ 1 - \gamma_i(\mathbf{w}; (\mathbf{x}, \mathbf{y})), & \text{otherwise} \end{cases} \quad (11)$$

The labelwise hinge loss can be regarded as an approximation of Hamming loss, but it gives more penalty for the wrong label with larger score.

We use online learning algorithm to calculate the parameters $\mathbf{w}$.

In round t, we find new weight vector $\mathbf{w}_{t+1}$ by

$$\mathbf{w}_{t+1} = \arg\min_{\mathbf{w} \in \mathbb{R}^n} \frac{1}{2}||\mathbf{w} - \mathbf{w}_t||^2 + \mathcal{C} \cdot L \cdot \xi, \qquad \textbf{s.t. } \ell_i(\mathbf{w}; (\mathbf{x}_t, \mathbf{y}_t)) <= \xi \textbf{ and } \xi >= 0 \quad (12)$$

where ξ is non-negative slack variable, L is the sentence length and $\mathcal{C}$ is a positive parameter which controls the influence of the slack term on the objective function.

[1] We use Viterbi algorithm in inference phase.

The goal of our algorithm is to achieve a margin per label at least 1 as often as possible, thus the Hamming loss is also reduced indirectly. On rounds where the algorithm attains a margin less than 1 it suffers an instantaneous loss.

We abbreviate $\ell_i(\mathbf{w_t}; (x, y))$ to $\ell_{t,i}$. If $\ell_{t,i} = 0$ then w_t itself satisfies the constraint in Eq. (12) and is clearly the optimal solution. We therefore concentrate on the case where $\ell_{t,i} > 0$.

First, we define the Lagrangian of the optimization problem in Eq. (12) to be,

$$\mathcal{L}((w), \xi, \alpha, \beta) = \frac{1}{2}||\mathbf{w} - \mathbf{w}_t||^2 + \mathcal{C} \cdot L \cdot \xi + \sum_{i=1}^{L} \alpha_i(\ell_{t,i} - \xi) - \beta\xi$$
$$\textbf{s.t. } \alpha_i >= 0 (\forall i), \beta >= 0. \quad (13)$$

where α, β is a Lagrange multiplier.

Setting the partial derivatives of $\mathcal{L}$ with respect to the elements of ξ to zero gives

$$\sum_i \alpha_i + \beta = \mathcal{C} \cdot L. \quad (14)$$

The gradient of $\mathbf{w}$ should be zero,

$$\mathbf{w} - \mathbf{w}_t - \sum_{i=1}^{L} \alpha_i(\Phi_i(\mathbf{x}, \mathbf{y}) - \Phi_i(\mathbf{x}, \hat{\mathbf{y}})) = 0, \quad (15)$$

we get

$$\mathbf{w} = \mathbf{w}_t + \sum_{i=1}^{L} \alpha_i(\Phi_i(\mathbf{x}, \mathbf{y}) - \Phi_i(\mathbf{x}, \hat{\mathbf{y}})). \quad (16)$$

Substitute Eq. (14) and Eq. (16) to dual objective function Eq. (13), we get

$$\mathcal{L}(\alpha) = -\frac{1}{2}||\sum_{i=1}^{L} \alpha_i(\Phi_i(\mathbf{x}, \mathbf{y}) - \Phi_i(\mathbf{x}, \hat{\mathbf{y}}))||^2$$
$$- \sum_{i=1}^{L} \alpha_i \mathbf{w_t}^T (\Phi_i(\mathbf{x}, \mathbf{y}) - \Phi_i(\mathbf{x}, \hat{\mathbf{y}})) + \sum_{i=1}^{L} \alpha_i \quad (17)$$

Differentiate with each α_i, and set it to zero, we get

$$\left(\sum_{j=1}^{L} \alpha_j \left(\Phi_j(\mathbf{x}, \mathbf{y}) - \Phi_j(\mathbf{x}, \hat{\mathbf{y}})\right) + \mathbf{w_t}^T \right)^T \cdot (\Phi_i(\mathbf{x}, \mathbf{y}) - \Phi_i(\mathbf{x}, \hat{\mathbf{y}})) - 1 = 0. \quad (18)$$

So we can get a linear system about α_i with L linear equations,

$$\sum_j A_{ij}\alpha_j = B_i, (i = 1, \cdots, L) \quad (19)$$

where $A_{ij} = (\Phi_j(\mathbf{x}, \mathbf{y}) - \Phi_j(\mathbf{x}, \hat{\mathbf{y}}))^T \ (\Phi_i(\mathbf{x}, \mathbf{y}) - \Phi_i(\mathbf{x}, \hat{\mathbf{y}}))$ and $B_i = 1 - \mathbf{w_t}^T (\Phi_i(\mathbf{x}, \mathbf{y}) - \Phi_i(\mathbf{x}, \hat{\mathbf{y}}))$.

We reformulate the Eq. (19, and the equivalent matrix equation is

$$A\bar{\alpha} = B, \tag{20}$$

where A is $L \times L$ symmetric matrix with elements A_{ij}, $B = [B_1, \cdots, B_L]^T$ is vector and $\bar{\alpha} = [\alpha_1, \cdots, \alpha_L]^T$.

Here, A is usually singular and sparse. We can solve it by singular value decomposition (SVD) [7].

$$A = UDV^T, \tag{21}$$

where U, V is orthogonal.

We can calculate $\bar{\alpha}$ by

$$\bar{\alpha} = VD^{-1}U^T B, \tag{22}$$

where D^{-1} is defined as

$$D^{-1} = \begin{cases} 1/\sigma_i & \text{if} \sigma_i > 0 \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

From $\sum_i \alpha_i + \beta = \mathcal{C} \cdot L$, we know that $\sum_i \alpha_i < \mathcal{C} \cdot L$. We give the same treatment to each α_i, so

$$\bar{\alpha}_i^* = \min(\mathcal{C}, \bar{\alpha}_i). \tag{24}$$

Finally, we get update strategy,

$$\mathbf{w_{t+1}} = \mathbf{w}_t + \sum_{i=1}^{L} \bar{\alpha}_i^* (\Phi_i(\mathbf{x}, \mathbf{y}) - \Phi_i(\mathbf{x}, \hat{\mathbf{y}})). \tag{25}$$

```
input : training data set: $(x_n, y_n), n = 1, \cdots, N$,
        aggressive parameters: $\mathcal{C}$,
        average number: $K$
        maximum iterative number: $T$
output: $\mathbf{w}$

Initialize: $\mathbf{cw} \leftarrow 0$,;
for $k = 0 \cdots K - 1$ do
    $\mathbf{w}_0 \leftarrow 0$ ;
    for $t = 0 \cdots T - 1$ do
        receive an example $(\mathbf{x}_t, \mathbf{y}_t)$;
        predict: $\hat{\mathbf{y}}_t = \arg\max_{\mathbf{z} \neq \mathbf{y}_t} \langle \mathbf{w}_t, \Phi(\mathbf{x}_t, \mathbf{z}) \rangle$;
        calculate $\ell_i(\mathbf{w}; (\mathbf{x}, \mathbf{y}))$;
        for $i = 0 \cdots L$ do
            if $\ell_i(\mathbf{w}; (\mathbf{x}, \mathbf{y})) \leq 1$ then
                calculate $\alpha_i^*$ by Eq. (24);
            end
        end
        update $\mathbf{w}_{t+1}$ with Eq.(25);
    end
    $\mathbf{cw} = \mathbf{cw} + \mathbf{w}_T$ ;
end
$\mathbf{w} = \mathbf{cw}/K$ ;
```

Algorithm 1. Labelwise Margin Maximization Algorithm

Our final algorithm is shown in Algorithm 1.

In order to avoid overfitting, the averaging technology is employed. We set the repeated number K to 2 for average parameters in this paper.

4 Experiment

To evaluate our algorithm, denoted as LPA (Labelwise PA), we compare it with CRF, M3N and PA. We used CRF++[2] and Pocket M3N[3] packages as the CRF and M3N implementations. In our experiments, we use linear kernel for M3N and set $\mathcal{C} = 1$ for PA and LPA. We also try to use the different values of $\mathcal{C}$, and found that larger values of $\mathcal{C}$ imply a more aggressive update step and result to fast convergence, but it has little influence on the final accuracy. The maximum iterative number T is set to 20 for PA and LPA.

4.1 Chunking

Text chunking consists of dividing a text in syntactically correlated parts of words. We use the training and test data from CoNLL-2000[4]. This data consists of the same partitions of the Wall Street Journal corpus (WSJ) as the widely used data for noun phrase chunking: sections 15-18 as training data (211727 tokens) and section 20 as test data (47377 tokens). The annotation of the data has been derived from the WSJ corpus by a program written by Sabine Buchholz from Tilburg University, The Netherlands.

The feature templates are shown in Table 1(a). W represents a word, P represent the part-of-speech tag, and T represents the chunking tag. The subscript of W indicates its position relative to the current character, whose subscript is 0.

Table 1. Chunking

(a) Feature templates for chunking

$W_i T_0, i = -2, -1, 0, 1, 2$
$W_{-1,0} T_0, W_{0,1} T_0$
$P_i T_0, i = -2, -1, 0, 1, 2$
$P_{-2,-1} T_0, P_{-1,0} T_0, P_{0,1} T_0, P_{1,2} T_0$
$P_{-2,-1,0} T_0, P_{-1,0,1} T_0, p_{0,1,2} T_0$
$T_{-1,0}$

(b) Results of Chunking

	Precision	Recall	F_1
CRF	93.91%	93.67%	93.79
M3N	93.59%	92.99%	93.29
PA	93.69%	92.95%	93.32
LPA	**94.59%**	**93.63%**	**94.11**

Evaluations are done using precision and recall on the extracted chunks, and we report $F_1 = 2PR/P + R$. The results are shown in Table 1(b). We can see that our algorithm gives best performance.

[2] Available from `http://crfpp.sourceforge.net`

[3] Available from `http://code.google.com/p/pocketcrf/`

[4] `http://www.cnts.ua.ac.be/conll2000/chunking/`

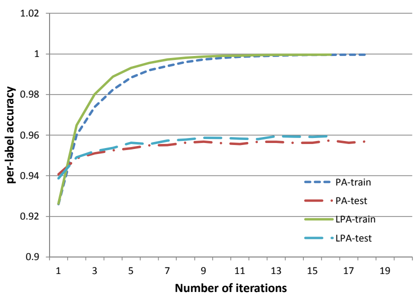


Fig. 2. Per-label accuracy

4.2 Chinese Word Segmentation

Chinese word segmentation is to segment Chinese sentence (not space-separated) into a sequence of words. Chinese word segmentation can be regarded as sequence labeling problem. We use "B", "M", "E" and "S" to represent the beginning, middle, end or single character of a word respectively.

We use four corpora[5] in SIGHAN Bakeoff 2 [6] to evaluate our algorithm, which includes the Academia Sinica Corpus (AS), the Hong Kong City University Corpus (CityU), the Peking University Corpus (PKU) and the Microsoft Research Corpus (MSR).

The feature templates are shown in Table 2. C represents a Chinese character, and the subscript of C indicates its position relative to the current character, whose subscript is 0. T represents the character-based tag.

The evaluation measure are reported are precision, recall, and an evenly-weighted F_1.

Table 2. Feature templates for Chinese word segmentation and named entity recongition

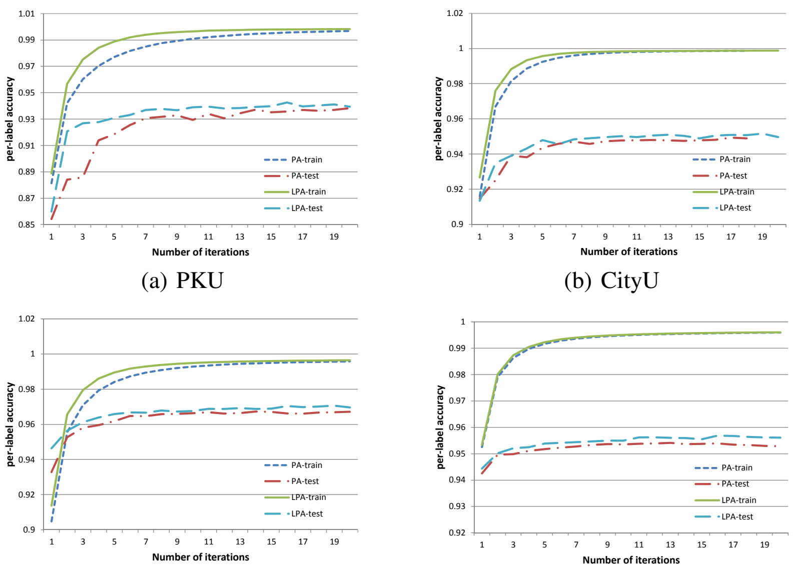
$C_iT_0, (i = -1, 0, 1)$
$C_{-1,0}T_0, C_{0,1}T_0, C_{-1,1}T_0$
$T_{-1,0}$

The results are shown in Table 3, which show LPA outperforms the others. PA gives the poorest performance. Although the result of LPA is almost the same as CRF and M3N, it has about 2% percent boost on PA in average by adding the constraints for each label.

Table 3. F-measure on SIGHAN Bakeoff 2

	PKU	MSR	CityU	AS
CRF	93.2	96.5	9.41	95.1
M3N	93.4	96.3	94.5	94.9
PA	93.9	92.6	93.5	94.9
LPA	**94.0**	**96.7**	**94.8**	**95.2**

[5] `http://www.sighan.org/bakeoff2005/`



(a) PKU (b) CityU

(c) MSR (d) AS

Fig. 3. Per-label accuracy

4.3 Chinese Named Entity Recognition

Chinese named entity recognition is a task to find the named entity in Chinese sentences. The entities often includes person, organization, location and geopolitical. We use the MSRA and CityU corpora from Bakeoff 2006[6] [12]. The feature templates are same to Chinese word segmentation. The results are shown in Table 5(a) and 5(b). Chinese named entity recognition can also be regarded as sequence labeling problem, in which each Chinese character of sentence is to be assigned with one of 9 tags (see Table 4).

Table 4. Tags of Chinese Named Entity Recognition

Tag	Meaning
0 (zero)	Not part of a named entity
B-PER	Beginning character of a person name
I-PER	Non-beginning character of a person name
B-ORG	Beginning character of an organization name
I-ORG	Non-beginning character of an organization name
B-LOC	Beginning character of a location name
I-LOC	Non-beginning character of a location name
B-GPE	Beginning character of a geopolitical entity
I-GPE	Non-beginning character of a geopolitical entity

The evaluation measure are reported are precision, recall, and an evenly-weighted F_1. The results are shown in Table 5(a) and 5(b). LPA gives best performance on F_1 measure.

[6] `http://www.sighan.org/bakeoff2006/`

Table 5. Results of Chinese Named Entity Recognition

(a) CityU

	Precision	Recall	F_1
CRF	74.43%	58.81%	65.71
M3N	71.93%	63.65%	67.54
PA	74.47%	60.03%	66.48
LPA	**75.37%**	62.41%	**68.28**

(b) MSRA

	Precision	Recall	F_1
CRF	83.18%	76.28%	79.58
M3N	79.94%	78.95%	79.44
PA	82.13%	79.83%	80.96
LPA	**83.59%**	**80.05%**	**81.78**

5 Discussion and Related Works

From the experimental results, we can see that the labelwise decomposition of the margin gives a boost on performances of three sequence labeling tasks. A major reason is that the labelwise hinge loss is closer to Hamming loss than the whole loss.

A few works have also dealt with the problem of minimizing the loss of sequence directly.

Gross et al. [8] gave a gradient-based procedure for minimizing an arbitrarily accurate approximation of the empirical risk [24] under a Hamming loss function. Our work is to minimize the structural risk [24] with maximizing the labelwise margin. Kakade et al [9] proposed a method to minimize the loss incurred by maximum a posteriori, rather than maximum expected accuracy, parsing on the training set. Xiong et al [25] also proposed a criterion called minimum tag error (MTE) for discriminative training of conditional random fields (CRFs) and applied it to Chinese word segmentation. The MTE criterion is an average of the raw tag accuracy over all possible label sequences (weighted by their likelihood).

More generally, our work is related to piecewise decomposition [14] [19] [20] [21] for factor graph in graphical model, which have recently been the subject of much interest. Piecewise decomposition is proposed for fast learning and inference of structure learning.

6 Conclusion

In this paper, we propose an online learning algorithm that addresses the problem of per-label margin maximization for sequence labeling. In the future, we will investigate our algorithm in other sequence labeling problems, such as part-of-speech tagging. Then we will extend our algorithm to more complex structure learning problems, such as parsing.

Moreover, we also wish to extend labelwise margin to piecewise margin for parallel learning and inference.

Acknowledgments

This work was (partially) funded by NSFC (No. 61003091), Shanghai Leading Academic Discipline Project (No. B114), Shanghai Committee of Science and Technology (No. 10dz1500102) and 863 Program (No. 2009AA01A346).

References

1. Collins, M.: Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In: Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (2002)
2. Crammer, K., McDonald, R., Pereira, F.: Scalable large-margin online learning for structured classification. In: NIPS Workshop on Learning With Structured Outputs, Citeseer (2005)
3. Crammer, K., Singer, Y.: Ultraconservative online algorithms for multiclass problems. Journal of Machine Learning Research 3, 951–991 (2003)
4. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y.: Online passive-aggressive algorithms. Journal of Machine Learning Research 7, 551–585 (2006)
5. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines. Cambridge University Press, Cambridge (2001)
6. Emerson, T.: The second international chinese word segmentation bakeoff. In: Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing, Jeju Island, Korea, pp. 123–133 (2005)
7. Golub, G., Van Loan, C.: Matrix computations. Johns Hopkins Univ. Pr., Baltimore (1996)
8. Gross, S., Russakovsky, O., Do, C., Batzoglou, S.: Training conditional random fields for maximum labelwise accuracy. Advances in Neural Information Processing Systems 19, 529 (2007)
9. Kakade, S., Teh, Y., Roweis, S.: An alternate objective function for markovian fields. In: Proceedings of International Conference on Machine Learning, vol. 19, pp. 275–282 (2002)
10. Kazama, J., Torisawa, K.: A new perceptron algorithm for sequence labeling with non-local features. In: Proceedings of Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL (2007)
11. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: ICML 2001: Proceedings of the Eighteenth International Conference on Machine Learning (2001)
12. Levow, G.: The third international chinese language processing bakeoff: Word segmentation and named entity recognition. In: Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing, Sydney, pp. 108–117 (2006)
13. McCallum, A., Freitag, D., Pereira, F.: Maximum entropy markov models for information extraction and segmentation. In: Proceedings of the Seventeenth International Conference on Machine Learning, Citsseer, pp. 591–598 (2000)
14. McCallum, A., Sutton, C.: Piecewise training with parameter independence diagrams: Comparing globally-and locally-trained linear-chain crfs. In: NIPS 2004 Workshop on Learning with Structured Outputs (2004)
15. Peng, F., Feng, F., McCallum, A.: Chinese segmentation and new word detection using conditional random fields. In: Proceedings of the 20th International Conference on Computational Linguistics (2004)
16. Ramshaw, L., Marcus, M.: Exploring the statistical derivation of transformational rule sequences for part-of-speech tagging. In: Proceedings of the ACL Workshop on Combining Symbolic and Statistical Approaches to Language, pp. 128–135 (1994)
17. Sang, E., Veenstra, J.: Representing text chunks. In: Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics, pp. 173–179. Association for Computational Linguistics (1999)
18. Settles, B.: Biomedical named entity recognition using conditional random fields and rich feature sets. In: Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications, NLPBA (2004)

19. Sutton, C., McCallum, A.: Piecewise training of undirected models. In: 21st Conference on Uncertainty in Artificial Intelligence. Citeseer (2005)
20. Sutton, C., McCallum, A.: Piecewise pseudolikelihood for efficient training of conditional random fields. In: Proceedings of the 24th International Conference on Machine Learning, p. 870. ACM, New York (2007)
21. Sutton, C., McCallum, A.: Piecewise training for structured prediction. Machine learning 77(2), 165–194 (2009)
22. Taskar, B., Guestrin, C., Koller, D.: Max-margin markov networks. In: Proceedings of Neural Information Processing Systems (2003)
23. Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y.: Support vector machine learning for interdependent and structured output spaces. In: Proceedings of the International Conference on Machine Learning, ICML (2004)
24. Vapnik, V.: Statistical Learning Theory. Wiley, Chichester (1998)
25. Xiong, Y., Zhu, J., Huang, H., Xu, H.: Minimum tag error for discriminative training of conditional random fields. Information Sciences 179(1-2), 169–179 (2009)
26. Xue, N.: Chinese word segmentation as character tagging. Computational Linguistics and Chinese Language Processing 8(1), 29–48 (2003)

Co-related Verb Argument Selectional Preferences*

Hiram Calvo[1], Kentaro Inui[2], and Yuji Matsumoto[3]

[1,3] Computational Linguistics, Nara Institute of Science and Technology,
Takayama, Ikoma, Nara 630-0192, Japan
[2] Communication Science Lab., Tohoku University
Aoba, Sendai, 980-8579, Japan
{calvo,matsu}@is.naist.jp, inui@ecei.tohoku.ac.jp

Abstract. Learning Selectional Preferences has been approached as a verb and argument problem, or at most as a tri-nary relationship between subject, verb and object. The correlation of all arguments in a sentence, however, has not been extensively studied for sentence plausibility measuring because of the increased number of potential combinations and data sparseness. We propose a unified model for machine learning using SVM (Support Vector Machines) with features based on topic-projected words from a PLSI (Probabilistic Latent Semantic Indexing) Model and PMI (Pointwise Mutual Information) as co-occurrence features, and WordNet top concept projected words as semantic classes. We perform tests using a pseudo-disambiguation task. We found that considering all arguments in a sentence improves the correct identification of plausible sentences with an increase of 10% in recall among other things.

1 Introduction

A sentence can be regarded as a verb with multiple arguments. The plausibility of each argument depends not only on the verb, but also on other arguments. Measuring the plausibility of verb arguments is required for several tasks such as Semantic Role Labeling, since grouping verb arguments and measuring their plausibility increases performance, as shown by Merlo and Van Der Plas (2009) and Deschacht and Moens (2009). Metaphora recognition requires this information too, since we are able to know common usages of arguments, and an uncommon usage would suggest its presence, or a coherence mistake (*v. gr. to drink the moon in a glass*). Malapropism detection can use the measure of the plausibility of an argument to determine misuses of words (Bolshakov, 2005) as in hysteric *center*, instead of *historic center*; density *has brought me to you*; *It looks like a* tattoo *subject*; and *Why you say that with* ironing*?* Anaphora resolution consists on finding referenced objects, thus, requiring among other things, to have information about the plausibility of arguments at hand, *i.e.*, what kind of fillers is more likely to satisfy the sentence's constraints, such as in: The boy plays with *it there*, *It* eats grass, I drank *it* in a glass.

This problem can be seen as collecting a large database of semantic frames with detailed categories and examples that fit these categories. For this purpose, recent

* This research is supported by SNI, SIP-IPN, COFAA-IPN, and PIFI-IPN, CONACYT; and the Japanese Government (JSPS).

works take advantage of existing manually crafted resources such as WordNet, Wikipedia, FrameNet, VerbNet or PropBank. The problem with the semantic frames approach for this task is that semantic frames are too general. For example Anna Korhonen (2000) considers the verbs *to fly, to sail* and *to slide* as similar and finds a single subcategorization frame. On the other hand, n-gram based approaches are too particular, and even using a very big corpus (such as using the web as corpus) has two problems: some combinations are unavailable, or counts are biased by some syntactic constructions. For example, solving the PP attachment for *extinguish fire with water* using Google[1] yields *fire with water*: 319,000 hits; *extinguish with water*: 32,100 hits. Resulting in the structure **(extinguish (fire with water))* instead of *(extinguish (fire) with water).* Thus, we need a way for smoothing these counts. This latter has been done by using Selectional Preferences since Resnik (1996) for verb to class preferences, and then generalized by Agirre and Martinez (2000) for verb class to noun class preferences. More recent work includes (McCarthy and Carroll, 2003), which disambiguate nouns, verbs and adjectives using automatically acquired selectional preferences as probability distributions over the WordNet noun hyponym hierarchy and evaluate with Senseval-2. However all these works have a common problem which is that they address separately each argument for a verb.

1.1 The Need for Co-occurrence

Calvo *et al.* (2009) show that considering simultaneously three arguments yields better precision than only two, with certain loss of recall. Kawahara and Kurohashi (2006) perform verb disambiguation for learning preferences by differentiating the main verb with the closest argument. For example *play a joke* and *play a guitar* will have different argument preferences; however, in some cases this is not enough, as it can be seen in the following example, where the verb has different meanings depending of a far argument:

Play a scene for friends in the theatre	(to act), and
Play a scene for friends in the VCR	(to reproduce.)

Recent works have proposed a discriminative approach for learning selectional preferences, starting with Bergsma *et al.* (2008). Ritter *et al.* (2010) and Ó Séaghdha (2010) propose a LinkLDA (Latent Dirichlet Allocation) model with linked topic hidden variables drawn from the same distribution to model <subject, verb, object> combinations, such as <*man*, *eats*, *ramen*> and <*cow*, *eats*, *grass*>. However, these works consider at most tri-nary relations. Motivated by the problem of considering as many arguments as possible for clustering verb preferences, we propose here a general model for learning all co-related preferences in a sentence, allowing us to measure the plausibility of its occurrence. In addition this model allows using both statistical resources as well as manual resources such as dictionaries or WordNet to improve the prediction. In this work we show an example of using Probabilistic Latent Semantic Indexing (PLSI), Pointwise Mutual Information (PMI) and WordNet for measuring this plausibility.

Furthermore, there are several particular questions that we seek to answer in this paper: (1) For automatic learning, building the co-occurrence table from real examples

[1] Google query as of April 20th, 2010.

can be achieved using different approaches (see Section 2). Which approach offers the best solution?; (2) Joining verb and nouns information in a single table is suitable for the model?; (3) Using a Support Vector Machine (SVM) trained only on PLSI information performs better than the PLSI model?; (4) How does this model perform when varying training information?; (5) Combining statistical information (PLSI and PMI) with manually crafted resources information such as WordNet improves results?

In Section 2 we describe our proposed model, and present in Section 3, the experiments we conducted to answer the previous questions. Finally in Section 4 we analyze the results and draw our conclusions.

2 Methodology

First we build the resource for counting co-occurrences. We do this by parsing the UKWaC corpus with MINIPAR (Lin, 1998) to obtain a lemmatized dependency representation. The UKWaC corpus (Ferraresi et al. 2008) is a large balanced corpus of English from the UK Web with more than 2 billion tokens. The sentence *Play a scene for friends in the theatre* becomes: play obj:scene for:friend in:theatre. Then we pre-calculate the mutual information statistics between each pair of words, i.e., (play, obj:scene), (play, for:friend), (play, in:theatre), (obj:scene, for:friend), (obj:scene, in:theatre), (for:friend, in:theatre). We then proceed to calculate the topic representation of each word using PLSI.

The Probabilistic Latent Semantic Indexing (PLSI) model was introduced in Hofmann (1999). It was derived from Latent Semantic Indexing (Deerwester et al., 1990). The model attempts to associate an unobserved class variable $z \in Z = \{z_1, ..., z_k\}$, with two sets of observable arguments. In terms of generative models, PLSI can be defined as follows: a document is selected with probability $P(d)$, then a latent class z is selected with probability $P(z|d)$ and finally a word w is selected with probability $P(w|z)$. This definition can also be represented as Eq. (1).

$$P(d,w) = \sum_{Z} P(z_i) \cdot P(d|z_i) \cdot P(w|z_i) \tag{1}$$

Given a set of sentences, there are several ways for considering what a word and what a document is. We can consider grouping the documents by verb or by noun. That is, a document *eat* will contain all the arguments co-occurring with the verb *eat*, or, a document *ball* contains the other arguments and verbs co-occurring with the noun *ball*, for instance *play, with:stripes, for:exercise*, etc. (see Table 1). On the other hand, documents can be nouns only, and the co-occurrences would be verbs plus functions. See Table 2.

Table 1. Co-occurrence table (verbs+nouns)

	with friend	in park
play	1	1
eat	1	
ball		1
yoyo	1	

Table 2. Co-occurrence table (nouns only)

	play with	play at
Yoyo	1	
ball	1	

In summary, the following different ways of building the sentence co-occurrence matrix for PLSI are listed below. fn means function:noun (*with:stripes*), v means verb (*play*), n noun (*ball*), vf means verb:function (*play:with*). Baroni and Lenci (2009) have performed experiments with similar matrices. Their nomenclature is indicated in square brackets.

a. (fn|v,fn|v)
bc. (fn,fn), (v,fn) [LCxLC, CxLC]
d. (v|n,fn)
ef. (v,fn), (n,fn) [CxLC,CxLC]
g. (n,vf|nf) [CxCL]
h,i. (n,vf) (n,nf)

Note that modes *a* and *bc* are the same, however *bc* considers building and training the PLSI model separately for nouns and verbs. The same happens for modes *d* and *ef* and *g* and *hi*. In the experiment section we detail results for each one of the different settings for building the sentence co-occurrence matrix for PLSI.

2.1 Assembling SVM Features for Training and Testing

Once the PLSI and PMI resources are built, the training and test sentences are parsed with MINIPAR. In this paper, only the first level shallow parse is used. We mapped features to positions in a vector. Every argument has a fixed offset, i.e., the subject will always be in the first position, the object in position 75, the arguments beginning with *in* at position 150, etc. In this way the co-relation can be captured by an SVM learner. We have chosen a second-degree polynomial kernel, to capture combinations of features. Each of the arguments is decomposed into several sub-features. These sub-features consist of a projection of each word in the PLSI topic space, the Pointwise Mutual Information (PMI) between target word and the feature word, and a projection of the feature word in the WordNet space.

Table 3. Simplified example representation for SVM training and testing (one long row), verb: play

subj							**obj** (target)						
z_1	z_2	z_3	PMI	wn_1	wn_2	wn_3	z_1	z_2	z_3	PMI	wn_1	wn_2	wn_3
							0.3	0.2	0.5	1	0.8	0.3	0.2
in							**on**						
z_1	z_2	z_3	PMI	wn_1	wn_2	wn_3	z_1	z_2	z_3	PMI	wn_1	wn_2	wn_3
0.4	0.3	0.8	0.4	0.2	0.4	0.3							
with							**for**						
z_1	z_2	z_3	PMI	wn_1	wn_2	wn_3	z_1	z_2	z_3	PMI	wn_1	wn_2	wn_3
							0.4	0.6	0.4	0.2	0.1	0.9	0.1

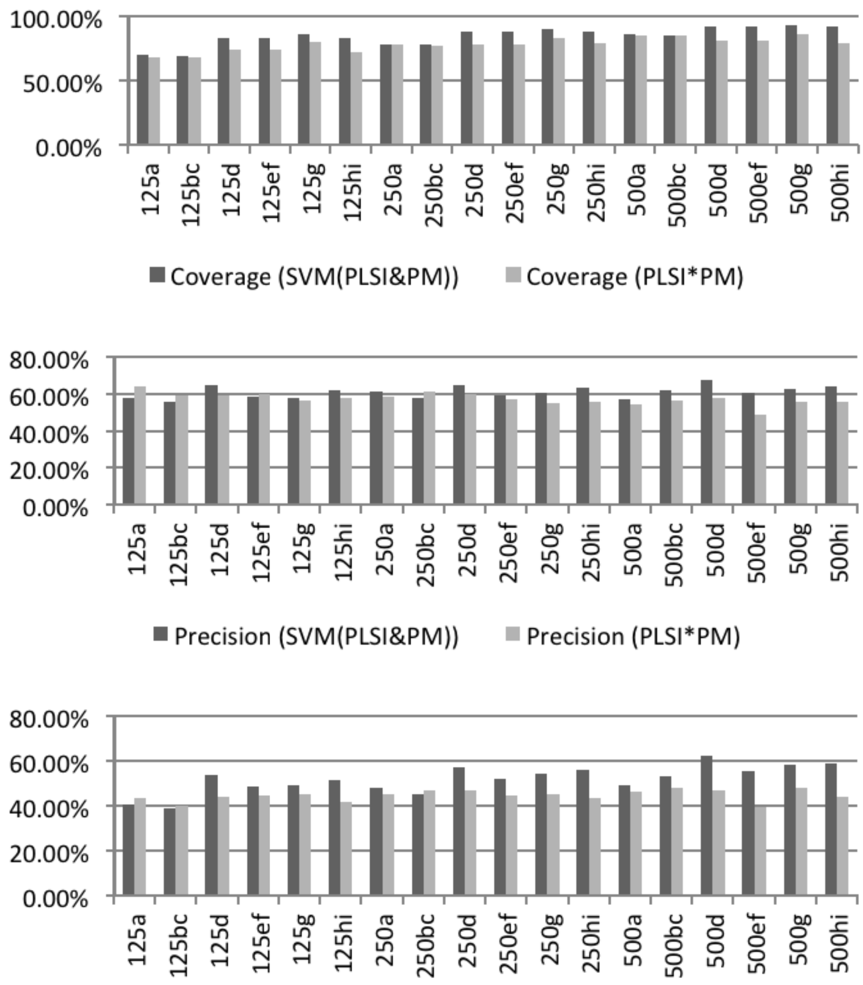


Fig. 1. Coverage, Precision and Recall of first experiment

The PMI was calculated as follows,

$$PMI(t_1, t_2) = \frac{\log P(t_1, t_2)}{P(t_1, t_2)} \tag{2}$$

Sample data for learning is shown in Table 3. This table represents the row corresponding to *play a scene for friends in theatre*. The word *scene* is projected in three topics z_1, z_2 and z_3 as 0.3, 0.2, 0.5 (note however that the experiments considered the projection on 38 topics.) Then, we calculate the projection in a WordNet space for each word. This is done by calculating the jcn distance (Jiang and

Table 4. Top concepts in WordNet

dry_land_1	writing_4	money_2	state_1
object_1	construction_4	garment_1	abstraction_1
being_1	worker_2	feeling_1	attribute_1
human_1	creation_3	change_of_state_1	relation_1
animal_1	food_1	motion_2	cognition_1
flora_1	beverage_1	effect_4	unit_6
artifact_1	location_1	phenomenon_1	relationship_3
instrument_2	symbol_2	activity_1	time_1
device_2	substance_1	act_1	fluid_2
product_2			

Table 5. Results of using different modes for building the co-occurrence matrix for PLSI, and using PLSI and PM with and without SVM learning

Train size	mode	SVM on PLSI&PM	PLSI* PM	SVM on PLSI&PM	PLSI* PM	SVM (PLSI&PM)	PLSI* PM
		Coverage		Precision		Recall	
125	a	0.70	0.68	0.58	**0.64**	0.40	0.44
125	bc	0.69	0.68	0.56	0.59	0.39	0.40
125	d	0.83	0.74	0.65	0.59	0.54	0.44
125	ef	0.83	0.74	0.59	0.60	0.48	0.44
125	g	0.86	0.80	0.58	0.56	0.49	0.45
125	hi	0.83	0.72	0.62	0.58	0.51	0.42
250	a	0.78	0.78	0.62	0.59	0.48	0.45
250	bc	0.78	0.77	0.58	0.61	0.45	0.47
250	d	0.88	0.78	0.65	0.60	0.57	0.47
250	ef	0.88	0.78	0.59	0.57	0.52	0.45
250	g	0.90	0.83	0.61	0.55	0.54	0.45
250	hi	0.88	0.79	0.64	0.55	0.56	0.44
500	a	0.86	0.85	0.57	0.54	0.49	0.46
500	bc	0.85	0.85	0.62	0.57	0.53	0.48
500	d	0.92	0.81	**0.68**	0.58	**0.62**	0.47
500	ef	0.92	0.81	0.60	0.49	0.55	0.39
500	g	**0.93**	**0.86**	0.62	0.56	0.58	**0.48**
500	hi	0.92	0.79	0.64	0.56	0.59	0.44

Conrath, 2007) with regard to 38 top concepts in WordNet shown in Table 4. The PMI value for the target word and every target word is also included: (*scene, scene*) 1, (*scene, in theatre*) 0.4, (*scene, for friends*) 0.2.

3 Experiments

Let us recall the questions we intend to answer for these experiments.

1. For automatic learning, building the co-occurrence table out from real examples can be done in several ways, as shown in Section 2. Which approach is preferable?
2. Joining verb and nouns information in a single table is better for the model?

3. Using an SVM trained only on PLSI information can perform better than the PLSI model itself?
4. How does this model perform with varying training information?
5. Combining statistical information (PLSI and PMI) with manually crafted resources information such as WordNet improves results?

Table 6. Results of pseudo-disambiguation task with different settings of PMI, PLSI and WordNet (WN)

Wordset 125							
PMI	PLSI	WN	Learning	Coverage	Precision	Recall	F
0	0	1	68.36%	89.44%	54.88%	49.09%	51.82%
0	1	0	89.59%	82.61%	66.96%	55.23%	60.53%
0	1	1	92.60%	**96.09%**	63.23%	60.76%	61.97%
1	0	0	93.63%	46.62%	**70.98%**	33.10%	45.15%
1	0	1	94.55%	94.88%	65.85%	**62.48%**	**64.12%**
1	1	0	97.14%	83.03%	66.09%	54.85%	59.95%
1	1	1	**98.01%**	**96.09%**	65.26%	62.71%	63.96%
Wordset 250							
0	0	1	67.85%	89.49%	53.87%	48.21%	50.88%
0	1	0	88.01%	87.02%	69.44%	60.43%	64.62%
0	1	1	90.82%	**96.28%**	68.22%	**65.69%**	**66.93%**
1	0	0	93.24%	55.18%	**70.34%**	38.81%	50.02%
1	0	1	93.78%	95.39%	64.86%	61.87%	63.33%
1	1	0	96.88%	87.12%	68.99%	60.11%	64.24%
1	1	1	**97.28%**	**96.28%**	66.10%	64.64%	65.36%
Wordset 500							
0	0	1	91.09%	89.49%	46.75%	41.84%	44.16%
0	1	0	86.75%	91.58%	68.32%	62.57%	65.32%
0	1	1	93.46%	96.79%	54.37%	52.63%	53.49%
1	0	0	92.95%	64.62%	65.11%	42.07%	51.11%
1	0	1	93.46%	95.72%	63.18%	60.48%	61.80%
1	1	0	96.65%	91.72%	**68.77%**	63.08%	**65.80%**
1	1	1	**96.68%**	**97.69%**	65.51%	**63.41%**	64.44%
Average							
0	0	1	91.09%	89.47%	51.83%	46.38%	48.96%
0	1	0	86.75%	87.07%	68.24%	59.41%	63.49%
0	1	1	93.46%	96.39%	61.94%	59.69%	60.80%
1	0	0	92.95%	55.47%	**68.81%**	37.99%	48.76%
1	0	1	93.46%	95.33%	64.63%	61.61%	63.08%
1	1	0	96.65%	87.29%	67.95%	59.35%	63.33%
1	1	1	**96.68%**	**96.69%**	65.62%	**63.59%**	**64.59%**

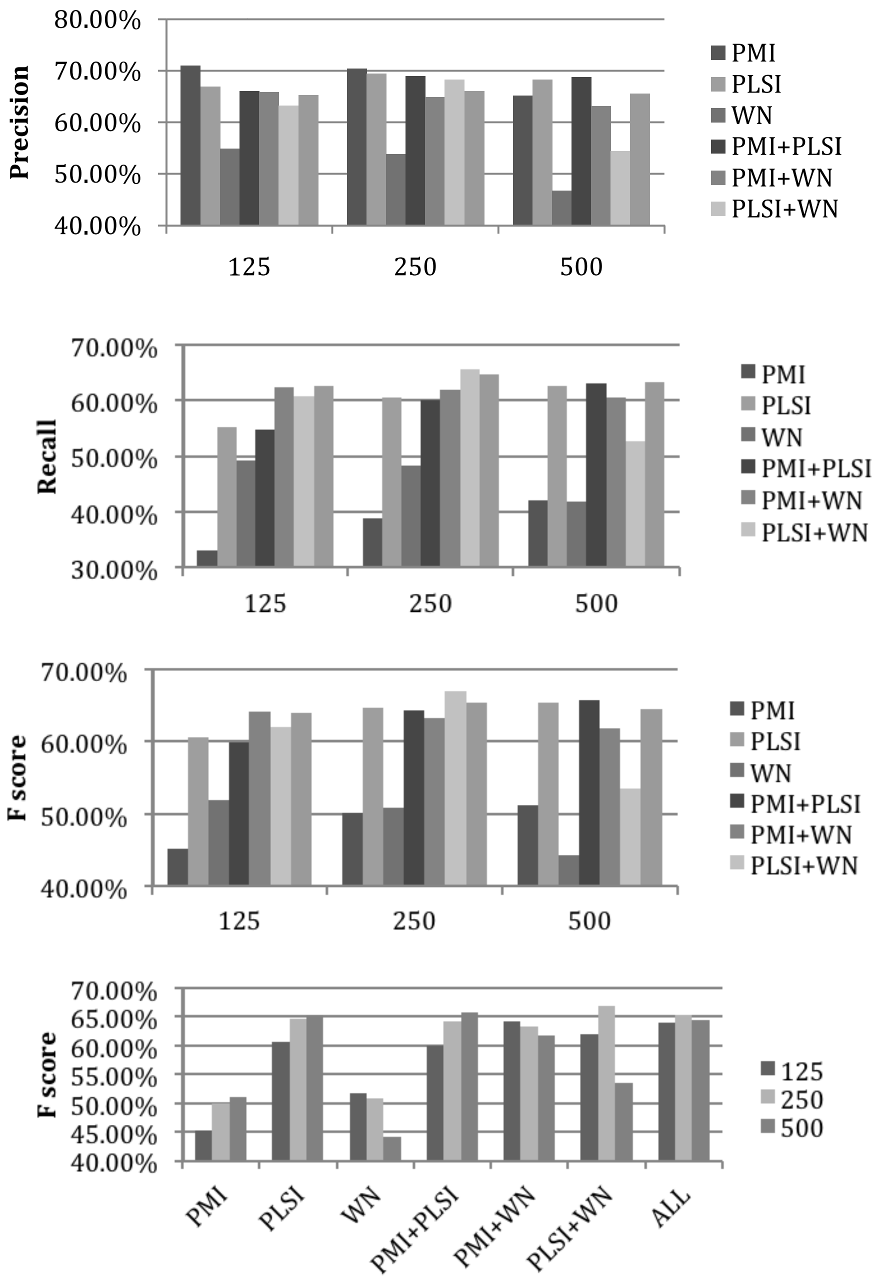


Fig. 2. Precision, Recall and F score (by training size and by method) for different training corpus' sizes (125, 250 and 500) and feature combinations (PMI, PLSI and WN)

Following Weeds and Weir (2003) we perform experiments for a pseudo-disambiguation task. This task consists of changing a target word (in this case, direct object) and identifying the most plausible sentence using the system, while considering the verb and all of its arguments. For example, for the sentences (i) *I eat* rice *with chopsticks at the cafeteria*; and (ii) *I eat* bag *with chopsticks at the cafeteria*, the system should be able to identify the first sentence as most plausible. We randomly selected 50 sentences from the WSJ corpus for the verbs: *play, eat, add, calculate, fix, read, write, have, learn, inspect, like, do, come, go, see, seem, give, take, keep, make, put, send, say, get, walk, run, study, need,* and *become*. These verbs were chosen as a sample of most frequent verbs, as well less frequent frequent verbs. They are also verbs that can support a great variety of arguments, such as *take* (i.e., ambiguity is high). For training we created wordsets for the same verbs. Each training wordset contains 125, 250 or 500 verb dependency triples per verb. Varying the size of training set provides answer to the fourth question. These wordsets were used for training the PLSI model, and also for creating the PMI database. Then, the same wordsets were used for training SVM. Each sentence was treated as a row, as described in Section 2.1, with each feature expanded in PLSI sub-features (topics). We generated two false examples randomly for each good example. For testing we generate a false for each existing test example.

At this point we have not included yet information about WordNet. This first experiment explores different ways of building the co-occurrence matrix, as described in Section 2 (questions number 1 and 2). We compare using PLSI and PM with and without SVM learning to answer question number 3. See results in Table 5.

From Table 5 and Figure 1 we observe that in all cases, the performance is improved when all arguments are considered (carried out by adding the SVM learning stage to the PLSI binary co-occurrences). In addition, whereas the *g* mode (n,vf|nf) for creating the co-occurrence matrix has greater coverage, by precision and recall mode *d* (v|n,fn) is always the best. We observe also an increasing performance while increasing the amount of data in the training wordset. Both the *g* and the *d* modes combine verbs and nouns, so that we can answer question two with a yes: it is better to join nouns and verbs in a single table.

3.1 Adding Manually Crafted Information

In this experiment we add manually crafted information into the model. As described in Section 2.1, we add information into the training and testing table about the distance to 38 common top concepts in WordNet. Table 6 shows the results obtained.

From Figure 2 it is possible to see that in most cases, combining the three sources of information improves the learning rate, although separately, PMI provides the highest learning rate. Coverage is always the best when combining the three resources. However, precision is better with PMI only for small amounts of training data, whereas PLSI gives better support when adding more data. Recall is greater for the cases involving the aid of WordNet information. On the average, except for precision, the best values are obtained when combining the three resources.

4 Analysis and Conclusions

Despite the minimal training data, we were able to obtain prediction rates above a trivial baseline of random selection between two options. With these experiments it was possible to determine the impact of using several resources, and also to measure the benefit of using an ensemble model for SVM with regard to a simple PLSI model. We found that considering all co-occurrences of arguments in a sentence increases recall by 10%. We also observed that, as expected, adding more data increases coverage; however, it increases recall in a greater extent using SVM over PLSI than in PLSI only. Using SVM increases coverage, precision and recall, even when trained with the same information available to PLSI. This suggests that generating negative examples randomly, and applying machine learning to this sample may improve performance of tasks using topic models.

We found also that the best mode to build the co-occurrence matrix for PLSI is mode *d* (v|n,fn), which corresponds in (Baroni and Lenci, 2009) work to the CxLC mode. We found also that building separately co-occurrence matrices for verbs and nouns does not improve the performance of the model; on the contrary, it worsens it. It is better to use a joint table of verbs and nouns because it is possible to share the information on the features between both groups of words. We proposed also that a single model to combine statistical information (PLSI and PMI) with manually crafted resources such as WordNet, and proved that performance increases in this way, however the increase was not as significative as we expected, as it is possible to see in Figure 2, where we can see that most of the contributing features are those from PLSI. However, as previously shown in Figure 1 and Table 5, SVM learning over PLSI has the advantage of being able to capture the co-relation of all arguments, as opposed than the pairwise PLSI model.

As future work, we plan evaluating with larger wordsets, as well as applying our model to other tasks such as anaphora resolution, or sentence coherence detection.

References

Agirre, E., Martinez, D.: Learning class-to-class selectional preferences. In: Workshop on Computational Natural Language Learning, ACL (2001)

Baroni, M., Lenci, A.: One distributional memory, many semantic spaces. In: Proceedings of the EACL 2009 Geometrical Models for Natural Language Semantics (GEMS) Workshop, pp. 1–8. ACL, East Stroudsburg (2009)

Bergsma, S., Lin, D., Goebel, R.: Discriminative Learning of Selectional Preference for Unlabeled Text. In: Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, pp. 59–68 (2008)

Bolshakov, I.A.: An Experiment in Detection and Correction of Malapropisms Through the Web. In: Gelbukh, A. (ed.) CICLing 2005. LNCS, vol. 3406, pp. 803–815. Springer, Heidelberg (2005)

Calvo, H., Inui, K., Matsumoto, Y.: Interpolated PLSI for Learning Plausible Verb Arguments. In: Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation, pp. 622–629 (2009)

Deerwester, S., Dumais, S.T., Furnas, G.W., Thomas, K.L., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society for Information Science, 391–407 (1990)

Deschacht, K., Moens, M.: Semi-supervised Semantic Role Labeling using the Latent Words Language Model. In: Procs. 2009 Conf. on Empirical Methods in Natural Language Processing (EMNLP 2009), pp. 21–29 (2009)

Ferraresi, A., Zanchetta, E., Baroni, M., Bernardini, S.: Introducing and evaluating ukWaC, a very large web-derived corpus of English. In: Procs. of the WAC4 Workshop at LREC, Marrakech, pp. 45–54 (2008)

Hoffmann, T.: Probabilistic Latent Semantic Analysis. In: Procs. Uncertainity in Artificial Intelligence 1999, UAI, pp. 289–296 (1999)

Jiang, J., Conrath, D.: Semantic similarity based on corpus statistics and lexical taxonomy. In: Proc. of the International Conference on Research in Computational Linguistics, ROCLING X (1997)

Kawahara, D., Kurohashi, S.: Japanese Case Frame Construction by Coupling the Verb and its Closest Case Component. In: 1st Intl. Conf. on Human Language Technology Research, ACL (2001)

Korhonen, A.: Using Semantically Motivated Estimates to Help Subcategorization Acquisition. In: Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, Hong Kong, pp. 216–223 (2000)

Lin, D.: Dependency-based Evaluation of MINIPAR. In: Proc. Workshop on the Evaluation of Parsing Systems (1998)

McCarthy, D., Carroll, J.: Disambiguating Nouns, Verbs, and Adjectives Using Automatically Acquired Selectional Preferences. Computational Linguistics 29(4), 639–654 (2006)

Merlo, P., Van Der Plas, L.: Abstraction and Generalisation in Semantic Role Labels: PropBank, VerbNet or both? In: Procs. 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pp. 288–296 (2009)

Ó Séaghdha, D.: Latent variable models of selectional preference. In: Proceedings of the 48th Annual Meeting of the Association of Computational Linguistics, pp. 435–444 (2010)

Reisinger, J., Paşca, M.: Latent Variable Models of Concept-Attribute Attachment. In: Procs. 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pp. 620–628 (2009)

Resnik, P.: Selectional Constraints: An Information-Theoretic Model and its Computational Realization. Cognition 61, 127–159 (1996)

Ritter, A., Mausam, Etzioni, O.: A Latent Dirichlet Allocation method for Selectional Preferences. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pp. 424–434 (2010)

Weeds, J., Weir, D.: A General Framework for Distributional Similarity. In: Procs. Conf. on EMNLP, vol. 10, pp. 81–88 (2003)

Yamada, I., Torisawa, K., Kazama, J., Kuroda, K., Murata, M., de Saeger, S., Bond, F., Sumida, A.: Hypernym Discovery Based on Distributional Similarity and Hierarchical Structures. In: Procs. 2009 Conf. on Empirical Methods in Natural Language Processing, pp. 929–937 (2009)

Combining Diverse Word-Alignment Symmetrizations Improves Dependency Tree Projection

David Mareček

Charles University in Prague,
Institute of Formal and Applied Linguistics
marecek@ufal.mff.cuni.cz

Abstract. For many languages, we are not able to train any supervised parser, because there are no manually annotated data available. This problem can be solved by using a parallel corpus with English, parsing the English side, projecting the dependencies through word-alignment connections, and training a parser on the projected trees. In this paper, we introduce a simple algorithm using a combination of various word-alignment symmetrizations. We prove that our method outperforms previous work, even though it uses McDonald's maximum-spanning-tree parser as it is, without any "unsupervised" modifications.

1 Introduction

Syntactic parsing is one of the basic tasks in natural language processing. The best parsers learn grammar from manually annotated treebanks and for all there holds the rule that increasing amount of training data improves their performance. However, there are many languages for which a very few linguistic resources exist. Developing a new treebank is quite expensive and for some rarer languages it is even impossible to find linguists for the annotations.

In recent years, numerous works have been devoted to developing parsers that would not need much annotated data. One way is the totally unsupervised parsing (e.g. the Klein and Manning's inside-outside method [1]), which infers the dependencies from raw texts only. But the performance of such parsers is quite low so far. This changes when we append a few annotated sentences to the raw texts. Koo proved in [2] that this causes a great improvement.

Hwa came up with an idea [3] to use a parallel corpus. For many languages there exist some form of parallel texts, very often with English or other resource-rich languages being the coupled. The idea is to make a word-alignment, parse the English side of the corpus, then project the dependencies from English to the other language using the alignment, and, finally, train a parser on the resulting trees or tree fragments. Several similar works ([4], [5], [6], [7]) came after and made many improvements on this process. Ganchev [6] and Smith and Eisner [5] combine this method with unsupervised training.

In all our experiments, we project dependency trees from English to other languages, denoted by the attribute *foreign* or letter X throughout the text. However, our approach should be effortlessly applicable also for language pairs with other source language.

When implementing the word alignment task in a parallel corpus, one can hardly expect only perfect 1:1 alignment links (e.g. due to typological language differences). If M:N links are allowed, a wide scale of alignment link types arises. Some of them are necessary or reasonable from one viewpoint, but spurious from the other (e.g. when aligning functional words). Hence, the fact that it is difficult to get one single ultimate word alignment, is not implied only by technical imperfectness of current implementations, but rather by the nature of languages. Simply said, different alignment schemes must be for different tasks. However, this paper shows that we can profit from the diversity.

The novel contribution of this paper lies in exploiting several types of word-alignment links; the previously published works expected a single word alignment on the input, but we show that combining more asymmetric sentence alignments leads to better results. Different types of alignment links can imply different reliability of projected edges, which provides the projection procedure with additional information. Furthermore, we also establish a method for filtering out the noise before training the parser. We use so called *alignment sparseness* and *non-projectivity* metrics for filtering sentences.

In Section 2, we comment previous works related to dependency projection. Section 3 describes word-alignment symmetrizations and discuss their suitability. Our projection algorithm is described in Section 4 and the process of data filtering is in Section 5. In Section 6, we present parsing accuracies on various languages and compare them to previous works. We conclude in Section 7.

2 Related Work

Our projection method was inspired by Hwa's work [4] from 2005, in which word alignment was used for projecting dependencies from English into Spanish and Chinese. They solved the "more counterpart" problem (what to do if an English word has more than one corresponding word) by choosing the left-most corresponding word as a main node; each other corresponding word then becomes dependent on the previous one (left-to-right dependencies). Unlike us, they used only one type of alignment link and they did not specify which method of symmetrization they used. They also introduced the data pruning criteria for filtering out the sentences where the alignment is bad (e.g. too many not aligned words or too many counterparts for one word) and added some hand-written rules to handle heterogeneity of different annotation schemas.

Smith and Eisner introduced in [5] quasi-synchronous grammar features for dependency projection and adaptation of annotation.

Ganchev et al. use in [6] so called "conserved dependencies", in which counterparts of governing and dependent word form a dependency edge (with the same orientation) in English. They ran an unsupervised parser with posterior regularization, in which inferred dependencies should correspond to projected ones.

Jiang and Liu use in [7] a matrix with alignment probabilities instead of the alignment itself in their projection algorithm. They compute a score for each possible link between the two parallel sentences and then, using a threshold, train a parser on the projected dependencies. They made the evaluation on the same Chinese data as in [4] and obtained better accuracy.

3 Alignment Symmetrization Methods

We use GIZA++ tool [8] to make word-alignments in parallel corpora. The alignment created by GIZA++ is asymmetric. For each word in one language just one counterpart in the other language is found, as it is depicted in Figure 1. Standard practice in machine translation tools is to run this alignment twice in both the directions (source-to-target and target-to-source) and use one of symmetrization methods described in [8]. For example, if we make an intersection of the two asymmetric alignments in Figures 1 and 2, we get the symmetric alignment, which is in Figure 3.

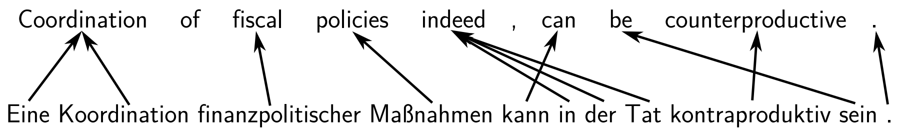

Coordination of fiscal policies indeed , can be counterproductive .
Eine Koordination finanzpolitischer Maßnahmen kann in der Tat kontraproduktiv sein .

Fig. 1. German-to-English alignment example. From each word in the German sentence "Eine*(A)* Koordination*(coordination)* finanzpolitischer*(fiscal)* Maßnahmen*(policies)* kann*(can)* in*(in)* der*(the)* Tat*(fact)* kontraproduktiv*(contraproductive)* sein*(be)*." a link is made to just one English word.

Our task is different from machine translation. We do not need to make any symmetrization, because the task itself is asymmetric. We have a parse tree in English and we want to project it to the other (foreign) language. We would like to know a counterpart for each foreign word, because each foreign word must depend on some other foreign word in the new tree. On the other hand, we do not need to know a counterpart for each English word. From this point of view, the asymmetric alignment *X-to-English* (Figure 2) seems to be more useful for the projection of dependencies than the opposite alignment *English-to-X*.

Of course, symmetrized alignment is useful too. If a connection between two words appears in both *X-to-English* (*XtoEN*) and *English-to-X* (*ENtoX*), it should be more preferred. Beside the *intersection* (*INT*) alignment, which is

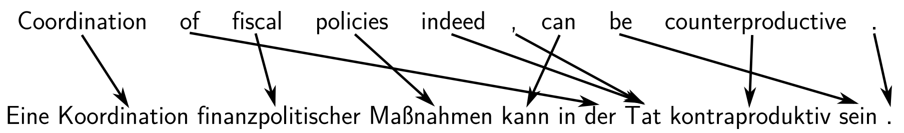

Coordination of fiscal policies indeed , can be counterproductive .
Eine Koordination finanzpolitischer Maßnahmen kann in der Tat kontraproduktiv sein .

Fig. 2. English-to-German alignment example. A link is made from each English word.

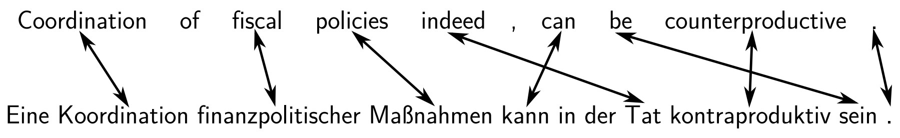

Coordination of fiscal policies indeed , can be counterproductive .
Eine Koordination finanzpolitischer Maßnahmen kann in der Tat kontraproduktiv sein .

Fig. 3. Example of intersection alignment

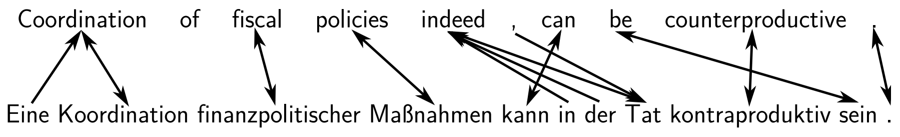

Coordination of fiscal policies indeed , can be counterproductive .
Eine Koordination finanzpolitischer Maßnahmen kann in der Tat kontraproduktiv sein .

Fig. 4. Example of grow-diag-final-and alignment

depicted in Figure 3, we will use in this work also so called *grow-diag-final-and* (*GDFA*) alignment (Figure 4), in which there are all links from the intersection alignment and some other links adjacent to already added links. All the symmetrization methods are described in [8].

4 Algorithm for Projecting Dependency Trees

In this section, we describe our projection algorithm in detail. We present the setting which led to the best projection results across languages we tested. However, it is possible that a slightly different setting would be more useful for some other languages.

4.1 Assignment of Corresponding Words

First of all, we go through the English sentence and for each English word e_i we find a set of corresponding foreign words $C(e_i) = [f_{c_1}, f_{c_2}, \ldots]$. The set can be empty as well as it can contain more than one foreign word. But every foreign word can belong at most to one English word.

The foreign words f_{c_j} in the sets $C(e_i)$ are ordered according to the type of alignment connection between f_{c_j} and e_i. In the first position, there is a word connected by an *intersection* link (if it exists). This links have the highest weight since they are confirmed by both GIZA++ runs. They are followed by words connected by alignment links *X-to-English* that are also in *grow-diag-final-and* alignment. Words connected only by *X-to-English* links are at the end. The whole procedure is described in detail in Figure 5. The first three loops add words into sets sequentially; the last loop searches for English words that have no correspondent so far and if such English word e_i is linked to some word f_j and f_j is not the only word in its set, f_j is transfered to $C(e_i)$.

This assignment method ensures that each foreign word now belongs to just one English word. In this point we differ from previous works. This helps in searching dependencies for words that do not have its own counterpart in English (mostly the function words). Such word then becomes dependent on a word with

```
**Input**: $INT$, $GDFA$, $XtoEN$, $ENtoX$ ... various alignments of a sentence
**Output**: $C$ ... sets of correspondent words for each English word

foreach $[e_i, f_j] \in INT$ do
    Push $f_j$ to $C(e_i)$;
end
foreach $[e_i, f_j] \in (GDFA \cap XtoEN) \setminus INT$ do
    Push $f_j$ to $C(e_i)$;
end
foreach $[e_i, f_j] \in XtoEN \setminus GDFA$ do
    Push $f_j$ to $C(e_i)$;
end
foreach $[e_i, f_j] \in (ENtoX \cup GDFA) \setminus INT$ do
    $e_k \leftarrow$ such English node for which $f_j \in C(e_k)$;
    if $C(e_k)[0] \neq f_j$ then
        Delete $f_j$ from $C(e_k)$;
        Push $f_j$ to $C(e_i)$;
    end
end
```

Fig. 5. Algorithm for assignment of corresponding words

which it shares its counterpart. Since the shared counterpart is only one (it is determined by the *X-to-English* alignment), we do not need to use any heuristic for choosing one as it is in [4].

To conclude, the acquired sets of corresponding words are very close to the *X-to-English* alignment, only some connections are substituted by *English-to-X* links so that more English words would be covered. Of course, there are many other possibilities how to deal with different alignment symmetrizations, but this method seems to be the best for our testing languages.

4.2 Building the Dependency Tree

The algorithm for building the dependency tree of a foreign sentence consists of one recursive function `project_subtree()`. It goes through the English tree in a depth-first manner and builds the foreign tree at the same time. The process is described in pseudo-code in Figure 6. The example of an English-to-German projection is depicted in Figure 7.

When an English node e_i is processed, we choose from the ordered set of corresponding words $C(e_i)$ the first one and declare it as the *main counterpart*. The other corresponding words then become its children.

We demonstrate the algorithm on an example in Figure 7. The English node *indeed* has three corresponding words in the German sentence: *in*, *der*, and *Tat*. While the word *Tat* is connected to *indeed* by an *intersection* link, the other two words are connected only with *grow-diag-final-and* links. This means that $C(\textit{indeed}) = [Tat, in, der]$ and *Tat* becomes the main counterpart of *indeed* and the words *in* and *der* become its children.

```
$e\_root$ = technical root of English parse tree;
$f\_root$ = technical root of foreign parse tree;
build_subtree($e\_root$, $f\_root$);

**function** build_subtree($e\_node$, $f\_node$);
**begin**
    **foreach** $e\_child \in children(e\_node)$ **do**
        **if** $|C(e\_child)| = 0$ **then**
            build_subtree($e\_child$, $f\_node$);
        **else**
            $main\_f\_child \leftarrow C(e\_child)[0]$;
            parent($main\_f\_child$) $\leftarrow f\_node$;
            **foreach** $f\_child \in C(e\_child)$ **do**
                **if** $f\_child \neq main\_f\_child$ **then**
                    parent($f\_child$) $\leftarrow main\_f\_child$;
                **end**
            **end**
            build_subtree($e\_child$, $main\_f\_child$);
        **end**
    **end**
**end**
```

Fig. 6. Algorithm for projection of dependency trees

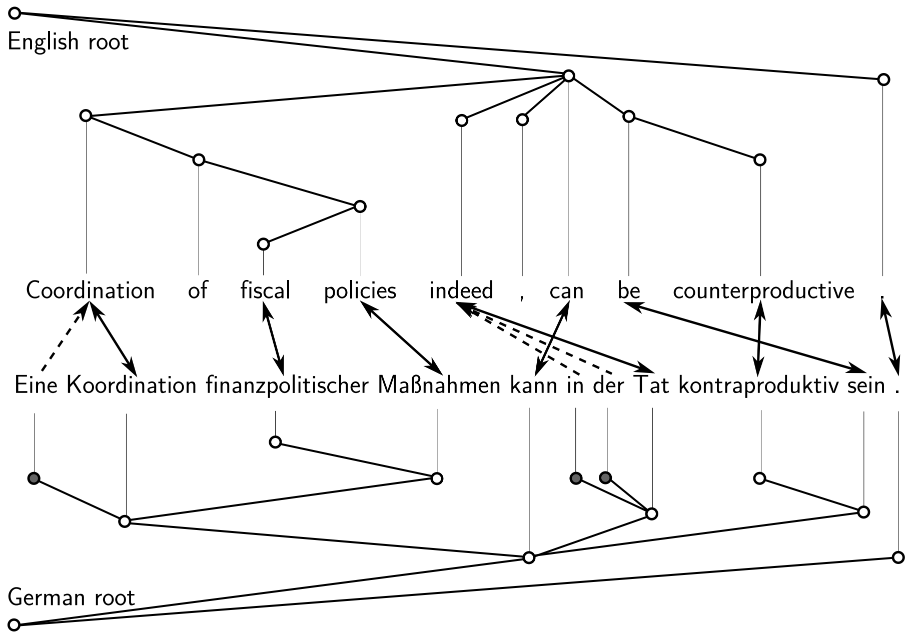

English root
Coordination of fiscal policies indeed , can be counterproductive .
Eine Koordination finanzpolitischer Maßnahmen kann in der Tat kontraproduktiv sein .
German root

Fig. 7. Projection of an English dependency tree into German. Intersection connections are depicted by solid arrows, other are dashed.

Another problem which must be solved is dealing with English nodes that have an empty set of corresponding words, for example the English preposition *of.* In these cases, the algorithm goes directly to its children and counterparts of its children (the words *finanzpolitischer* and *Maßnahmen*) become children of the counterpart of its parent (the word *Koordination*). It means that the node *Maßnahmen* is a child of the node *Koordination*, even though there is one more node (*of*) between *policies* and *coordination.*

5 Data Filtering

When we have the dependency trees projected into the foreign language, we can simply train a parser on them and measure the parsing quality on some manually annotated treebank. The problem is that the quality of some trees on which we are training is very poor. This can be caused by various errors, mainly in preprocessing:

- *non-parallel sentences* – two parallel sentences have completely different meaning. This is caused by an error in sentence-level alignment. Sometimes it happens that only a part of a sentence is translated.
- *completely different structure* – the sentences have the same meaning but their syntactic structures are completely different.
- *wrong word alignment* – in case there are more words with very low frequency in the sentence.

We would like to filter out these bad sentences before training. For this purpose, we established two metrics of the sentence quality: *alignment sparseness* and *non-projectivity.*

5.1 Alignment Sparseness Limit

We define *alignment sparseness* as a relative number of words that have no counterpart in an *intersection* alignment. It is computed as a number of links divided by the average length of the pair of sentences. Its values are between 0 and 1. Value 0 means that the parallel sentences have the same length and there is a perfect 1-to-1 alignment mapping. Value 1 means that there are no intersection links at all.

$$S = 1 - \frac{\#links}{\frac{1}{2}(length(e) + length(f))}$$

All sentences that have higher *alignment sparseness* than a given threshold are filtered out. There is a trade-off between quality and quantity of the training data. Figure 8 shows the experiment in searching for optimal sparseness limit S_{max}. We can see that the higher the limit S_{max}, the higher the number of training sentences. If we train a parser on them and test it on a treebank, accuracy of such parser increases at first, but then it begins to decrease slightly, because the number of wrongly aligned sentences in the training data grows.

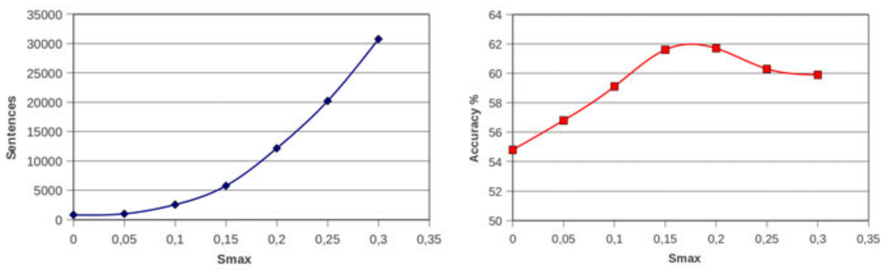


Fig. 8. Relations between alignment sparseness limit, number of sentences after filtering, and unlabeled parsing accuracy. This was measured on English-Czech News commentaries parallel corpus with approximately 100,000 sentence pairs. You can see that the optimal limit S_{max} here is 0.2. For this limit, the parser was trained only on 12,500 sentences, which means that we filter out more than 87% of sentences.

5.2 Non-projectivity Limit

The next criterion for recognition of trees that are not suitable for training the parser is the number of non-projective edges in them. An edge $[d, g]$ in a tree is non-projective, if the parent p of the governing node g lays between d and g. Of course, some non-projective edges can be correct, but after the review of the projected trees, we found out that a majority of non-projective edges are errors caused mainly by the wrong word alignment.

The experiment in which we filter out also such sentences where there were more non-projectivities than a given limit is shown in Figure 9. We can see that the best parsing accuracy was achieved by filtering out all sentences containing at least one non-projective edge (the limit $NP_{max} = 0$).

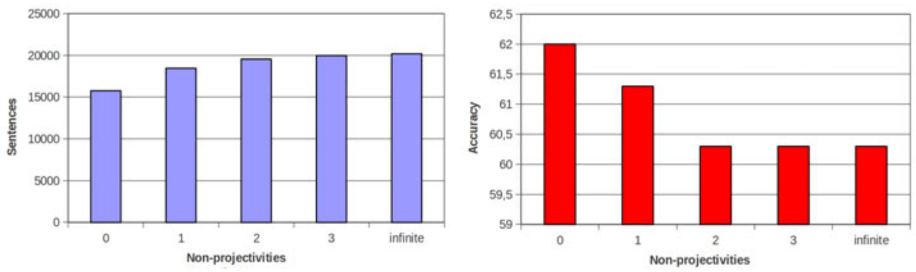


Fig. 9. Experiment with filtering out the sentences containing more non-projectivities than a given limit NP_{max}. It was done on Czech-English parallel corpus already filtered by alignment sparseness limit $S_{max} = 0.25$, which is more than 20,000 sentence pairs.

6 Experiments

We ran our experiments on four languages: Bulgarian, Czech, Dutch, and German. For Czech and German, we used the News commentaries parallel corpus as it was prepared for the WMT10 translation task.[1] For Bulgarian and Dutch, we used

[1] `http://www.statmt.org/wmt10/`

the Acquis Communautaire parallel corpus[2]. The English side of the corpus was tagged by Morce tagger [9] and parsed by McDonald's maximum spanning tree parser [10] which was trained on English CoNLL X[3] data. The foreign (target) sides were tagged by Tree-tagger [11] with appropriate models downloaded from TreeTagger websites.[4]

For filtering the projected trees, we tried several values for *alignment sparseness* limit and *non-projectivity* limit. Once we had the filtered trees, we trained the MST parser[5] on them. The parser was then tested on development-test data from the CoNLL X shared task [12]. The attachment accuracies are in Table 1.

Table 1. Unlabeled parsing accuracies for Bulgarian, Czech, Dutch, and German tested on CoNLL X testing data. S_{max} and NP_{max} are the thresholds used for filtering data before training. The "EN parser" column describes the post-processing steps used for English parsing.

Language	Parallel Corpus	EN parser	S_{max}	NP_{max}	Accuracy
Bulgarian	Acquis Communautaire	*CoNLL+CoordTr*	0.2	0	52.7 %
Czech	News commentaries	*CoNLL+AuxVTr*	0.15	0	62.0 %
Dutch	Acquis Communautaire	*CoNLL+CoordTr*	0.2	0	52.4 %
German	News commentaries	*CoNLL+CoordTr*	0.2	0	55.7 %

Ideally, the testing treebanks should follow the same annotation guidelines as the treebank on which we train English. However, that is not true in CoNLL X. The treebanks differ for example in capturing coordination structures or dealing with auxiliary verbs. For this reasons we implemented two post-processing steps; one for transforming coordination structures (*CoordTr*)[6] and the other for re-hanging auxiliary verbs (*AuxVTr*).[7] The basic parsing (using the CoNLL data) is marked in Table 1 as *CoNLL*.

In order to compare our projection method with previous works, we ran the whole process on Bulgarian with exactly same setting as in [6]. We used the English-Bulgarian OpenSubtitles parallel corpus [13], the English side was parsed by McDonald's MST parser trained on sections 2-21 of the Penn treebank with dependencies extracted using the head rules of Yamada and Matsumoto [14]. The parser was tested on the Bultreebank corpus as it was released for CoNLL X [12]

[2] Only the one-to-one sentence pairs were extracted from the parallel corpus and due to the computability reasons, we used only the first 100,000 parallel sentences which length was higher than two and lower than 30 words.

[3] `http://nextens.uvt.nl/~conll/`

[4] `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/`

[5] We used the McDonald's parser `mstparser-0.4.3b` with these settings: `order:1`, `iters:10`, `decode-type:proj`, `training-k:1`.

[6] In English CoNLL data, the head of coordination structure is the conjunction. However, in Bulgarian and German, the first coordinating member is the head and the other members and conjunctions become its children.

[7] Auxiliary verbs in Czech CoNLL data (*byl*, *bude*, *by*) depend on the main verb, while English auxiliary verbs (*do*, *will*, *be*, *have*) do not.

Table 2. Comparison of unlabeled parsing accuracies on Bulgarian CoNLL X training set. Our method is by 0.3% better than the method presented by Ganchev et al. in their work [6].

Method	Parser	Accuracy
Ganchev et al.	Discriminative model	66.9 %
Ganchev et al.	Generative Model	67.8 %
Our method	MST parser	68.1 %

shared task on the training sentences of up to 10 words. Punctuation was stripped at training time. The results compared in Table 2 show that our method outperforms the previous work [6] in unlabeled accuracy of the parser.

The projection algorithm written in Perl, example data, and the instructions how to run the whole process including syntactic analysis and alignment can be downloaded from `http://www.cicling.org/2011/software/49/`.

7 Conclusions and Future Plans

In this paper, we describe a novel method for projecting dependency trees across parallel texts, in which diverse word-alignment symmetrizations are used. Even though we do not combine the projected dependencies with automatically inferred dependencies, and we train MST parser directly on the projected trees without any modifications, we show that the parsing accuracies reached by our simple projection method are comparable to the previous more complex works. We have made the comparison on Bulgarian data and we outperform the previous state-of-the-art result by 0.3%.

We see two main advantages of the presented algorithm. First, it uses different types of alignment links and therefore some of them may be more preferred in the projection than others. Second, the *X-to-English* alignment ensures that all words on the target side are linked somewhere. This fact helps in attachment of function words which do not have their own counterpart in English.

The biggest problem of this task is differences in annotation guidelines of particular treebanks. This fact makes the evaluation problematic. We would like to solve it in the future by creating more rules to make the treebanks more similar or even create a small multilingual treebank with the same annotation rules for several languages, which would be very useful for evaluating such experiments.

We are also aware of great amounts of trees that are filtered out before training. In the future we plan to incorporate into training data all well-aligned subtrees, not only the whole sentences.

Acknowledgments

This work has been supported by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 247762 (FAUST, FP7-ICT-2009-4-247762), by the doctoral grant "Res Informatica" of the Grant Agency of the Czech Republic (GA201/09/H057), by grant GAUK 116310 of the Grant Agency of Charles University, and by the grants MSM 0021620838 and SVV Projekt 261314/2010. We are grateful to the anonymous reviewers for further research suggestions.

References

1. Klein, D., Manning, C.D.: Corpus-based induction of syntactic structure: Models of dependency and constituency. In: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, ACL 2004. Association for Computational Linguistics, Morristown (2004)
2. Koo, T., Carreras, X., Collins, M.: Simple semi-supervised dependency parsing. In: Proceedings of ACL/HLT (2008)
3. Hwa, R., Resnik, P., Weinberg, A., Kolak, O.: Evaluating Translational Correspondence using Annotation Projection. In: Proceedings of the 40th Annual Meeting of the ACL, pp. 392–399 (2002)
4. Hwa, R., Resnik, P., Weinberg, A., Cabezas, C., Kolak, O.: Bootstrapping Parsers via Syntactic Projection across Parallel Texts. Natural Language Engineering 11, 11–311 (2005)
5. Smith, D.A., Eisner, J.: Parser adaptation and projection with quasi-synchronous grammar features. In: EMNLP 2009: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pp. 822–831. Association for Computational Linguistics, Morristown (2009)
6. Ganchev, K., Gillenwater, J., Taskar, B.: Dependency grammar induction via bitext projection constraints. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing, ACL-IJCNLP 2009, pp. 369–377. Association for Computational Linguistics, Morristown (2009)
7. Jiang, W., Liu, Q.: Dependency parsing and projection based on word-pair classification. In: ACL 2010: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pp. 12–20. Association for Computational Linguistics, Morristown (2010)
8. Och, F.J., Ney, H.: A Systematic Comparison of Various Statistical Alignment Models. Computational Linguistics 29(1), 19–51 (2003)
9. Spoustová, D., Hajič, J., Votrubec, J., Krbec, P., Květoň, P.: The Best of Two Worlds: Cooperation of Statistical and Rule-Based Taggers for Czech. In: ACL 2007: Proceedings of the Workshop on Balto-Slavonic Natural Language Processing, pp. 67–74. Association for Computational Linguistics, Morristown (2007)
10. McDonald, R., Pereira, F., Ribarov, K., Hajič, J.: Non-Projective Dependency Parsing using Spanning Tree Algorithms. In: Proceedings of Human Langauge Technology Conference and Conference on Empirical Methods in Natural Language Processing (HTL/EMNLP), Vancouver, BC, Canada, pp. 523–530 (2005)
11. Schmid, H.: Probabilistic Part-of-Speech Tagging Using Decision Trees. In: Proceedings of the International Conference on New Methods in Language Processing, Manchester, UK, vol. 12, pp. 44–49 (1994)
12. Buchholz, S., Marsi, E.: CoNLL-X shared task on multilingual dependency parsing. In: Proceedings of The Tenth Conference on Natural Language Learning (CoNLL-X), New York City, USA, pp. 149–164 (2006)
13. Tiedemann, J.: Building a Multilingual Parallel Subtitle Corpus. In: Proceedings of CLIN (2007)
14. Yamada, H., Matsumoto, Y.: Statistical Dependency Analysis with Support Vector Machines. In: Proceedings of IWPT, pp. 195–206 (2003)

An Analysis of Tree Topological Features in Classifier-Based Unlexicalized Parsing

Samuel W.K. Chan[1], Mickey W.C. Chong[1], and Lawrence Y.L. Cheung[2]

[1] Dept. of Decision Sciences
[2] Dept. of Linguistics & Modern Languages,
Chinese University of Hong Kong,
Shatin, Hong Kong SAR
{swkchan,mickey_chong,yllcheung}@cuhk.edu.hk

Abstract. A novel set of "tree topological features" (TTFs) is investigated for improving a classifier-based unlexicalized parser. The features capture the location and shape of subtrees in the treebank. Four main categories of TTFs are proposed and compared. Experimental results showed that each of the four categories independently improved the parsing accuracy significantly over the baseline model. When combined using the ensemble technique, the best unlexicalized parser achieves 84% accuracy without any extra language resources, and matches the performance of early lexicalized parsers. Linguistically, TTFs approximate linguistic notions such as grammatical weight, branching property and structural parallelism. This is illustrated by studying how the features capture structural parallelism in processing coordinate structures.

Keywords: parsing, unlexicalized model, topological features, machine learning.

1 Introduction

Advances in parsing have been made on two major fronts, namely, learning models and algorithms, and parsing features. In addition to improving probabilistic modelling and classifier-based methods, new parsing features have been developed in the last two decades, for example, the propagation of lexical head feature (Magerman, 1995; Collins, 2003) and semantic features. This paper explores a novel set of features, collectively called "tree topological features" (TTFs). TTFs can be easily computed with no extra language resources and be integrated into parsing models. They also deliver significant improvement over the baseline model.

This study is motivated by the fact that mainstream parsers seldom consider the shape of subtrees dominated by these nodes and rely primarily on matching POS/syntactic tags. As a result, an NP with a complicated structure is treated the same as an NP that dominates only one word. However, linguists working in syntactic processing have long observed that the size and shape of subtree also affect word ordering and parse tree building. For example, (i) branching property, (ii) heaviness of phrases in construction such as dative alternation and heavy NP-shift, and (iii) structural correlation of conjuncts in coordinate structures. We have proposed features to quantify subtree configuration and how they can be integrated into our parser.

A classifier-based unlexicalized parser was built to evaluate the contribution of the TTFs to the parser performance. While much research has centred on lexicalized parsers (Magerman, 1995; Ratnaparkhi, 1997; Collins, 2003) which use word token information, there are two reasons why we have highlighted unlexicalized parsing as our evaluation platform. First, TTFs can be more fairly assessed by minimizing interference from other features. The presence of other features may mask the contribution of features under scrutiny. In all our experiments, the only features used in addition to TTFs are tag features, i.e. POS and syntactic tags in the input. Second, due to the unavailability of lexical tokens in the input, there is a practical demand for richer feature set to better inform various tasks in unlexicalized parsers. Although it was believed that the lexicalized parsers should outperform unlexicalized parsers (Klein & Manning, 2003), recent studies (Matsuzaki *et al.*, 2005; Petrov & Klein, 2007) actually show that unlexicalized parsers can match lexicalized parsers in performance, for example, using the grammar rule splitting technique. TTFs will be shown to be a profitable direction in parsing research.

The organization of the paper is as follows. Section 2 reviews the range of features commonly used in parsing. The four subsets of TTFs will be introduced in Section 3. Section 4 discusses the architecture of the unlexicalized parser. The experimental evaluation is presented in Section 5. In Section 6, we will discuss the effectiveness and advantages of TTFs in parsing and possible enhancement. This is followed by a conclusion in Section 7.

2 Related Work

2.1 Parsing Features

This section reviews major types of information in parsing.

Tag Features: The dominant types of information that drive parsing and chunking algorithms are POS/phrase tags, context-free grammar (CFG) rules, and their statistical properties. Matching tags against CFG rules to form phrases is central to all early parsing algorithms such as CKY algorithm (Kasami, 1965), and the Earley algorithm (Earley, 1970), and the chart parsing approach (Kay, 1986). Even in the latest state-of-the-art parsers, tag features still contribute a lot to the performance.

Word Token-based Features: Machine learning and statistical modelling emerged as an ideal computational approach to feature-rich parsing. Classifiers can typically capitalize on a large set of features in decision making. Magerman (1995), Ratnaparkh (1999), Charniak (2000) among others used classifiers to model dependencies between word pairs. They popularized the use of head word tokens and their corresponding POS as attributes in lexicalized parsing. Collins (1999, 2003) also integrated information like head word token and distance from head into the statistical model to enhance probabilistic chart parsing. Since then, word tokens, head words and their statistical derivatives have become standard features in many state-of-the-art parsers. Word token information is also fundamental to dependency parsing (Kübler *et al.*, 2009) because dependency grammar is rooted in the idea that the head and the dependent word are related by different dependency relations.

Semantic-based Features: Some efforts have also been made to consider semantic features such as sense tags in parsing. Words are first tagged with semantic classes, often using WordNet-based resources. The lexical semantic class can be instructive to the selection of the correct parse from a set of candidate structures. It has been reported that the lexical semantics of words is effective in resolving structural ambiguity, especially PP-attachment (Black *et al.*, 1992; Stetina & Nagao, 1997; Xiong *et al.*, 2005; Agirre *et al.*, 2008). Nevertheless, the use of semantic features has still been relatively rare. They incur overheads in acquiring semantic language resources, such as sense-tagged corpora and WordNet databases. Semantic-based parsing also requires accurate sense-tagging. Since substantial gain from POS-based features is unlikely in the near future and deriving semantic features is often a tremendous task, there is a pressing need to seek for new features.

2.2 Linguistically-Motivated Features

In this section, a review of the linguistic motivation behind the TTFs is provided.

Grammatical Weight: Apart from syntactic categories, linguists have long observed that the number of words (often referred to as "weight" or "heaviness") in a phrase can affect syntactic processing of sentences (Quirk *et al.*, 1972: 943; Wasow, 1997; Rosenbach, 2005). It corresponds roughly to the span feature as described in later section. The effect of grammatical weight often manifests in word order variation. Heavy NP shift, dative alternation, particle movement and extraposition in English are canonical examples where "heavy" chunks get dislocated to the end of a sentence. Charniak & Johnson (2005) utilized grammatical weight as a criterion for re-ranking the N-best candidates.

Tree Topology: CFG-based parsing approach hides the structural properties of the dominated subtree from the associated syntactic tag. Structural topology, or tree shape, however, can be useful in guiding the parser to group tags into phrases. Structures significantly deviating from left/right branching, e.g. center embedding, are much more difficult to process and rare in production (Chomsky & Miller, 1963; Gibson, 1998). Another example is the resolution of scope ambiguity in coordinate structures. Coordinate structures are common but notoriously difficult to parse due to massive scope ambiguity when the conjuncts are complex (Collins, 1999; Kübler *et al.*, 2009). One good cue to the problem is that humans prefer coordinate structures with parallel internal syntactic structures (Frazier *et al.*, 2000; Dubey *et al.* 2008). The implication to syntactic parsing is that preference should be given to bracketing in which the conjuncts are structurally similar.

3 Tree Topological Features

Most parsers depend largely on matching syntactic tags to build parse trees. However, it is well-known that the shape of subtrees (including grammatical weight, skewness, etc.) can affect the well- formedness of phrase formation. For example, although the sequence `VP → VBD PP NP` (*gave [$_{PP}$ to Bill] [$_{NP}$ the books]*) is normally not accepted, the sentence becomes easily accepted if the NP is "heavy", e.g. *gave [$_{PP}$ to Bill] [$_{NP}$ the books which my friend bought yesterday]*. Consequently, it is helpful to add weight information to the rule, as in (1).

$$\texttt{VP} \rightarrow \texttt{VBD} \quad \texttt{PP} \quad \texttt{NP}_{\texttt{heavy}} \qquad (1)$$

Previous empirical studies (Frazier *et al.*, 2000; Dubey *et al.*, 2008) have shown that coordinate structures tend to have a balanced structure across conjuncts. The chunking of coordinate structures could be enhanced if the rule includes indicators describing the tree shape of the conjuncts, as in (2).

$$\texttt{VP} \rightarrow \texttt{VP}_{\texttt{shape}} \quad \texttt{CC} \quad \texttt{VP}_{\texttt{shape}} \qquad (2)$$

Tree topological features (TTFs) are thus formulated to capture the shape or topology of subtree quantitatively. Our approach involves examining four sets of features, without any assumption of the word tokens, namely, (i) Node Coordinates (*NC*s), (ii) Span Ratio (*SR*), (iii) Aspect Ratio (*AR*), and (iv) Skewness Measure (*SM*).

*Node Coordinates (NCs): NC*s record the position of the root node of the subtree from the most embedded terminal node and the beginning of the sentence. They include the level of focus (*LF*) and the relative position (*RP*) of the target subtree. The *LF* is defined as the total number of levels under the target node, with the terminal level inclusive. In other words, it is the number of nodes in the path from the subtree root node to the terminal node farthest away from the root in the subtree. The *RP* indicates the linear position of the target node in that level. In Fig. 1, the *LF* for subtree *A* and *B* are the same, i.e. *LF* = 4. The *RP*'s for subtree *A* and *B* are 1/2 and 2/2 respectively.

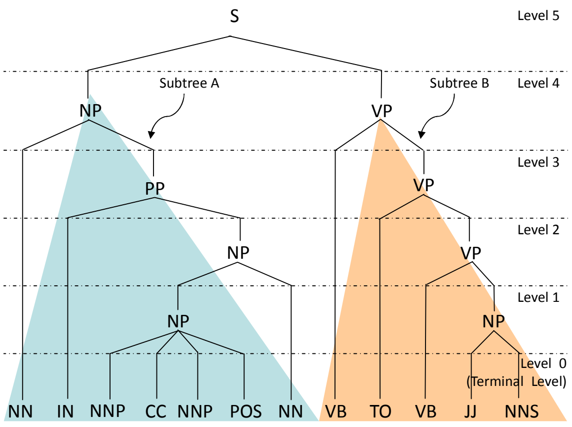


Fig. 1. Two different subtrees in the sentence `S`

Span Ratio (SR): The *SR* is defined as the total number of terminal nodes spanned under the target node and is divided by the length of the sentence. In Figure 1, the span ratio for the target node `VP` at subtree *B* is 5/12. This ratio illustrates not only how many terminal nodes are covered by the target node, but also how far the target node is from the root S. When the *SR* is close to 1, the node is closer to the root S.

Aspect Ratio (AR): The *AR* of a target node in a subtree is defined as the ratio of the total number of non-terminal nodes involved to the total number of terminal nodes spanned. The *AR* is indicative of the average branching factor of the subtree. It also measures the flatness of the subtree.

Skewness Measure (SM): The *SM* estimates the degree to which the subtree leans towards either left or right. In this research, the *SM* of a subtree is evaluated by the distribution of the length of the paths connecting the target node and each terminal node it dominates. The length of a path from a target node *V* to a terminal node *T* is the number of edges between *V* and *T*. For a tree with *n* terminal nodes, there are *n* paths. A pivot is defined as the [*n*/2]th terminal node when *n* is odd and between [*n*/2]th and [(*n*+1)/2]th terminal nodes if *n* is even, where [ ] is a ceiling function. The *SM* is defined as

$$SM = \frac{1}{\sum_{\rho_i > 0} \rho_i} \left(\frac{\sum_{i=1}^{n} \rho_i (x_i - \bar{x})^3}{\sigma^3} \right) \qquad \text{Eqn (1)}$$

where x_i is the length of the *i*-th path pointing to the *i*-th terminal node, $\bar{x}$ and σ are the average and standard deviation of the length of all paths at that level of focus (*LF*). ρ_i is the distance measured from the *i*-th terminal node to the pivot. The distance is positive if the terminal node is to the left of the pivot, zero if right at the pivot, and negative if the terminal node is to the right of the pivot. Obviously, if the lengths of all paths are the same in the tree, the numerator of Eqn (1) will be crossed out and the *SM* returns to zero. The pivot also provides an axis of vertical flipping where the *SM* still holds. The farther the terminal node from the pivot, the longer the distance. The distances ρ provide the moment factors to quantify the skewness of trees. For illustration, let us consider subtree *B* with the target node VP at level of focus (*LF*) = 4 in Figure 1. Since there are five terminal nodes, the pivot is at the third node VB. The lengths of the paths x_i from left to right in the subtree are 1, 2, 3, 4, 4 and the moment factors ρ_i for the paths are 2, 1, 0, -1, -2. Assuming that $\bar{x}$ and σ for all the trees in the Treebank at level 4 are, say, 2.9 and 1.2 respectively, then *SM* = -3.55. It implies that subtree *B* under the target node VP has a strong right branching tendency, even though it has a very uniform branching factor which is usually defined as the number of children at each node. In our parser, to determine whether the two target nodes at level 4, i.e., `NP` and `VP`, should be merged to form a S at level 5 or not, an attribute vector with TTFs for both `NP` and `VP` are devised as a training case while the corresponding target attribute is a binary value, i.e., chunking vs. merging.

4 A Classifier-Based Parser

4.1 Basic Architecture

Our parser is based on classical chunk-based parsing which is a bottom-up derivation strategy (Abney, 1991; Magerman, 1995; Ramshaw & Marcus, 1995; Sang, 2001; Tsuruoka & Tsujii, 2005; Sagae & Lavie, 2005). It is divided into three major modules, namely, (i) chunker, (ii) phrase recognizer, and (iii) learning module. The input is a string of POS and/or syntactic tags without word tokens. The input is subject to two passes. In the first pass, the chunker locates the boundaries of chunks (or phrases) of the lowest level in the target parse tree. In the second pass, the phrase recognizer assigns non-terminal syntactic tags (e.g. `NP`, `VP`, etc.) to the identified phrases. The updated tag sequence is fed back to the chunker for processing at the next level. The iteration continues until a complete parse is formed. The learning module acquires the knowledge encoded in the Penn Treebank to support the classification tasks in the two passes. The details of the machine learning algorithm will be provided in Section 4.2. Table 1 shows the parsing algorithm.

Chunk/Merge Approach
To identify phrases, we develop a "Chunk/ Merge Approach", which is a binary classification that determines whether the point between two adjacent tags should be classified as a phrase boundary. Let the input of the chunker be a tag sequence $<x_0 \ldots x_n \ldots x_m>$ where $0 \leq n \leq m$. Define the focus point y_{n} as the point between two consecutive tags x_n and x_{n+1}. The chunker classifies a focus point as either a chunking point or merging point at the relevant level. A focus point y_n is a merging point if x_n and x_{n+1} share the same parent node in the target parse tree. Otherwise, y_n is a chunking point. In machine learning, a feature vector is set up for each focus point in training. The target attribute is classified as either chunking point or merging point.

Table 1. Parsing algorithm

```
▪ Prepare training data from the Treebank based on topological features
▪ Train the chunker and phrase recognizer using the ensemble technique
▪ For any input POS tag sequence l,
  WHILE l contains more than one element DO
      IDENTIFY the status, + or %, of each focus point in l
      RECOGNIZE the syntactic tag (ST) of each identified phrase
      UPDATE l with the new ST sequence
  ENDWHILE
▪ Display the parse tree
```

Consider the Penn Treebank POS sequence in (3) and the expected classification of points. Chunking points are marked with "%" and merging points with "+".

```
L0: PRP % VBZ % DT % RB + JJ % NN        (3)
    He    is    a    very nice  guy
```

The point between RB and JJ is a merging point because they are siblings of the parent node ADJP in the target parse tree. The point between DT and RB is a chunking point. DT and RB are not siblings and do not share the same parent node. Chunks are defined as the consecutive tag sequences in the chunker output that are not separated by %. By the same procedure, the output strings of Level 1—3 chunking are represented as (4)—(6) respectively.

```
L1: NP % VBZ % DT + ADJP + NN            (4)
L2: NP % VBZ  + NP                       (5)
L3: NP + VP                              (6)
```

Phrase Recognition

The phrase recognizer assigns a syntactic tag to each chunk identified by the chunker in each iteration. Recall (3), where `RB JJ` constitutes a phrase. The function of the phrase recognizer is to label the phrase as `ADJP`. The recognizer is a classifier that predicts the phrase label based primarily on the tag sequence supplied by the chunker. The use of a classifier instead of a rule table is preferred for several reasons. The classifier approach makes it possible to assign a label to rules not seen in the training data. Also, some ambiguous rules can be better resolved by considering contextual features. For example, `IN S` could have two different tag assignments depending on the context as shown in Table 2.

Table 2. Ambiguity of `IN+S`

Rule	***Example***
`IN+S`→`PP`	"by + [$_S$ buying big blocks of stock]"
`IN+S`→`SBAR`	"as + [$_S$ UAL stopped trading]"

In Section 2—21 of the Penn Treebank, 13,554 distinct rules are found. Table 3 shows that 94.3% of the tag sequences correspond to only one possible syntactic tag, i.e. unambiguous, and the rest correspond to multiple possible tags.

Table 3. Ambiguity of rules

No. of Tags	***Rule Count***	***%***	***Frequency***	***%***
N = 1	12,786	94.3	138,554	18.7
N = 2	530	3.9	128,892	17.4
N = 3	142	1.0	101,531	13.7
N = 4+	96	0.7	371,344	50.2
Total:	13,554	100.0	740,321	100.0

However, unambiguous rules only account for 18.7% of rules in terms of occurrence frequency. 81.3% of the rules encountered in the treebank have to be disambiguated by the recognizer. To enhance the classification, the following feature vector, as shown in Table 4, is used in training and prediction.

Table 4. Feature vector for the phrase recognizer

Attribute	***Meaning***
PhraseType	Syntactic tag of the phrase (e.g. NP, VP, etc.) [=Target value]
TotalTag	No. of tags in the phrase
RelPosition	Position of the first tag of the chunk relative to the sentence (i.e. Position of 1st tag / Total no. of tags in the sentence)
Tag1…Tag6	Tags for the first six tags of the phrase
PTag	Tag right before the first tag of the phrase
FTag	Tag right after the last tag of the phrase
LastTag	Last tag of the phrase (for head-final phrases)

4.2 Machine Learning

The classification tasks in chunking and phrase recognition are supported by machine learning. Classifier-based parsing has the advantage of flexibly incorporating a large amount of features. This is different from chart parsing models which require special statistical modelling. The critical issue is the selection of features that are instructive to the classification tasks. POS and syntactic tags form the base set. The 6 tags on each side of the focus point are defined as the context of the focus point. Suppose the focus point y_n between x_n and x_{n+1} is considered. The attributes include the 6 tags preceding the focus point ($x_{n-5}, \ldots, x_n$) and 6 tags following the focus point ($x_{n+1}, \ldots, x_{n+6}$). The four sets of features as discussed in Section 3 will be examined and compared in the experiments.

Table 5. Adaboost algorithm

```
Given: $(x_1, y_1),..,(x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$
For $t = 1, \ldots, T$
• Train a weak learner using distribution $D_t$
• Get a weak hypothesis $h_t : X \rightarrow \{-1, +1\}$ with error
      $\varepsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$
• Choose      $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$
• Update:
      $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases}$
              $= \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
      where $Z_t$ is a normalization factor
• Output:
      $H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$
```

The ensemble technique is adopted to yield a more accurate predictive power (Dietterich, 2000). Ensemble learning creates a finite set of classifiers from random sets of training instances and then uses them together for the classification. Empirically, ensembles tend to yield better results and enhance their predictive power when there is a significant diversity among the data. Boosting, a widely used ensemble technique, is an effective method that produces a very accurate prediction rule by combining rough and moderately inaccurate rules of thumb (Schapire & Singer, 2000). In boosting, an initial base classifier using a set of training instances having equal weight is constructed. When the prediction of the base classifier differs from the expected outcome, the weight of this poorly predicted instance increases. A new training data set is then selected randomly from the weighted instances. As a result, the learning of the next classifier pays more attention to the poorly predicted instances. This process continues until a specified number of iterations is reached or a predefined termination condition is met. In brief, the main idea of boosting is to combine many simple and moderately inaccurate categorization rules into a single, highly accurate categorization rule. The simple rules are trained sequentially; conceptually, each rule is trained on the examples that were the most difficult to classify by the preceding rules. The first practical boosting algorithm, AdaBoost, which was introduced by Freund & Schapire (1997), solved many of the practical difficulties of the earlier boosting algorithms. Table 5 illustrates the main idea of the algorithm. Interested readers can refer to the literature for more detailed discussion (Freund & Schapire, 1997; Hastie *et al.*, 2001).

5 Experimental Results

5.1 Parsing Performance

Our parsing models were trained and tested using the Penn Treebank (Marcus *et al.*, 1993). Following the convention of previous studies, we pre-processed the trees by removing `NULL` elements and functional tags and collapsing `ADVP` and `PRT` into `ADVP`. Sections 2—21 are used for training and Section 23 for testing. To evaluate the contribution of the features, seven different experiments were set up, as in Table 6. E1 is the baseline experiment with tags only. The ±6 tags around the focus point are included in the feature vector. In addition to the tag features in E1, E2—E5 also include the *NC*s, *SR*, *AR* and *SM* respectively. Chan *et al.* (2010) reported that the overall performance after mixing the four types of TTFs. In this paper, the separation of the features presented in this paper allows us to estimate the relative contribution of different features towards the chunker. E6 was created by combining all the rules from E2—E5. All TTFs are therefore considered.

Table 6. Parsing features in seven experiments

Experiment	*Features*
E1	POS only (=baseline)
E2	POS+*NC*s
E3	POS+*SR*
E4	POS+*AR*
E5	POS+*SM*
E6	POS+*NC*s+*SR*+*AR*+*SM*

We first study the impact of feature sets on chunking. CH1—CH6 are evaluated.

Table 7 shows the training and test errors in the chunkers. If one compares CH2—CH5, it is clear that all TTFs enhance sentence chunking. The gain from the *AR* and *SM* (i.e. CH4 and CH5) is more significant. The best chunker, CH6, reduces the test error rate from the baseline 4.36% to 3.59%.

Table 7. Performance of chunkers

Chunker	Training error %	Test error %
CH1	1.66	4.36
CH2	1.02	4.23
CH3	1.04	4.09
CH4	1.03	3.82
CH5	1.01	4.00
CH6	--	3.59

The error rates in training and testing of phrase recognition are 0.09% and 0.68% respectively. The same phrase recognizer was used in all parser evaluation tests.

Table 8. Performance of seven parsers P1—P6

	R	P	F	CBs	0 CBs	≤2 CBs
P1	78.94	77.63	78.28	1.59	48.72	76.43
P2	81.89	82.39	82.14	1.38	53.77	79.89
P3	82.46	82.05	82.25	1.44	52.91	77.78
P4	82.18	82.33	82.26	1.40	52.58	78.56
P5	82.49	82.02	82.26	1.43	53.64	78.48
P6	84.74	83.33	84.03	1.34	54.61	80.53

The chunker and the phrase recognizer were assembled to form a parser. CH1—CH6 were used in P1—P6 respectively. We use the PARSEVAL measures to compare the performance as shown in Table 8. Our baseline parser (P1) performs fairly well. The tag-based parser produces an F-score of 78.28%. The addition of any of the four kinds of TTFs significantly raises the F-score of the baseline model to 82.14—82.26% (P2—P5). They are very close in performance. The *SR* is marginally worse than the other three features. When the rule sets of P2—P5 are combined, P6 leverages the cooperative effect of TTFs (ensemble learning), producing the best F-score of 84.03%.

5.2 Analysis of the Aspect Ratio and Skewness Measure

To better understand why TTFs are instructive to parsing, we present in Tables 9 and 10 the preliminary statistical analysis of the *AR* and *SM* of some major phrase types, including `VP`, `NP`, `S`, and `PP`, based on Sections 2—21 of the Penn Treebank. The separation of TTFs in the experiments enables us to examine the contribution of individual TTFs more clearly than the contribution of the TTFs as an aggregate, as in our previous study (Chan *et al.* 2010). The tables reveal that most of the *AR* and *SM* values are significantly different across levels, suggesting that the features can record reliably the TTF differences of subtrees. For example, in Table 9, the *AR* mean and standard deviation of `L2-VP` subtrees are 0.398 and 0.120 respectively.

We performed *t*-tests for difference in means between various levels, even under the same phrase type. The *t*-score for the difference in mean between `L2-VP` and `L3-VP` is -103.280, which indicates a strong difference in their *AR* values between these two levels. The means progressively increases as the level goes up. The ratio is potentially useful in deciding whether a focus point is merging. In a `VP`, the verb is usually far less complex than the object `NP`, but in a coordinate structure, the conjunct phrases tend to be equally complex. The *AR* potentially provides a useful measure of the tendency.

Similarly, *t*-tests for difference in means between various levels and phrase types were conducted in *SM* as shown in Table 10. Again, they indicate a strong difference in their *SM* values between these two levels. The means of all phrases beyond Level 2 are negative, consistent with the fact that English is generally a right branching language. When we compare the *SM* values across phrase types, it is easy to notice that `VPs` and `PPs` have larger negative values, meaning that the skewness to the right is more prominent. Even within the same phrase type, the *SM* values may differ significantly as one moves from its current level to parent level. The *SM* offers an indicator that differentiates different phrase types with different syntactic levels.

Table 9. *AR* values for various phrases (* = the mean in the column is statistically significantly different from the mean in the immediately following column, with *d.f.* >120)

VP	L2-VP	L3-VP	L4-VP	L5-VP
N	18,406	22,052	18,035	15,911
Mean	0.398	0.530	0.625	0.698
S.D.	0.120	0.137	0.152	0.157
t_{score}	-103.280*	-65.060*	-43.392*	
NP	**L2-NP**	**L3-NP**	**L4-NP**	**L5-NP**
N	23,270	28,172	10,827	8,375
Mean	0.312	0.568	0.599	0.706
S.D.	0.139	0.156	0.161	0.189
t_{score}	-196.680*	-17.175*	-41.464*	
S	**L2-S**	**L3-S**	**L4-S**	**L5-S**
N	2,233	5,020	7,049	7,572
Mean	0.782	0.679	0.687	0.705
S.D.	0.223	0.192	0.171	0.165
t_{score}	18.9275*	-2.3599	-6.4684*	
PP	**L2-PP**	**L3-PP**	**L4-PP**	**L5-PP**
N	53,589	11,329	11,537	5,057
Mean	0.350	0.471	0.639	0.670
S.D.	0.109	0.139	0.150	0.154
t_{score}	-87.162*	-87.867*	-12.030*	

Table 10. *SM* values for various phrases (* = the mean in the column is statistically significantly different from the mean in the immediately following column, with *d.f.* >120)

VP	L2-VP	L3-VP	L4-VP	L5-VP
N	18,406	22,052	18,035	15,911
Mean	-1.022	-4.454	-4.004	-3.738
S.D.	1.018	1.406	1.438	1.405
t_{score}	284.085*	-31.483*	-17.216*	
NP	**L2-NP**	**L3-NP**	**L4-NP**	**L5-NP**
N	23,270	28,172	10,827	8,375
Mean	1.013	-1.313	-1.432	-2.171
S.D.	1.284	2.013	1.821	1.628
t_{score}	158.748*	5.609*	29.614*	
S	**L2-S**	**L3-S**	**L4-S**	**L5-S**
N	2,233	5,020	7,049	7,572
Mean	0.688	-1.825	-1.459	-1.517
S.D.	1.229	2.732	2.451	2.128
t_{score}	54.031*	-7.568*	1.523	
PP	**L2-PP**	**L3-PP**	**L4-PP**	**L5-PP**
N	53,589	11,329	11,537	5,057
Mean	-1.337	-3.322	-3.951	-3.301
S.D.	0.935	1.148	1.112	1.183
t_{score}	172.352*	42.073*	-33.173*	

6 Discussion and Further Work

6.1 Effectiveness of Tree Topological Features to Our Parser

The findings reported in Section 5 indicate that the best *unlexicalized* parser, P6, performs on a par with first generation *lexicalized* parsers, as shown in Table 11. The experiments have two implications. First, the integration of TTFs produces substantial gain over the baseline model, P1. To the best of our knowledge, TTFs have not been systematically investigated in parsing before. The effectiveness of these new features suggests that in addition to improving parsing algorithms, practitioners should not overlook efforts in devising new features. Second, the implementation of TTFs is very easy and computationally inexpensive. In fact, no extra resources or complicated algorithms are needed to compute TTFs. Most importantly, they are highly suitable to the stringent requirements of unlexicalized parsing in which no word token information is allowed. The features can be added to mainstream parsers relatively easily without substantial changes.

Table 11. Parsers comparison (Length ≤ 40 words)

	R	P	F
Our parser (=P6) (unlexicalized)	84.7	83.3	84.0
Magerman 1995 (lexicalized)	84.6	84.9	84.7
Klein & Manning 2003 (unlexicalized)	85.7	86.9	86.3

6.2 Tree Topological Features and Coordinate Structure Parsing

Our study has provided a way to quantitatively capture linguists' various insights that tree topology is helpful in syntactic structure building (e.g. grammatical weight, subtree shape, etc.). Apart from the marco performance presented in Section 5, let us examine more closely how parsing coordinate structures (*CS*s) can be benefitted from a TTF-aware parser. *CS*s are chosen because they are notoriously difficult to parse due to scope ambiguity (Collins, 1999, 2003). Corpus studies show that conjuncts tend to be similar structurally. A TTF-aware parser can exploit the cue of the *AR* and *SM* to produce *CS*s with structurally similar conjuncts. We extracted all rules that have the form "XP → XP 'and' XP" from the training data, and compared the *AR* and *SM* of phrases with and without *CS*s. *t*-tests for difference in mean between them were performed. The *t*-score is based on unequal sample sizes and unequal variances.

As shown in Tables 12 and 13, *CS*s and non-*CS*s are statistically significantly different from each other in terms of the *AR* and *SM*, when we keep the phrase type and level constant. For example, the means of *AR* for *CS*s and non-*CS*s in VP (Level 3) are 0.529 and 0.647 respectively. They are statistically different. The *SM* is an even better indicator. *CS* phrases are much more balanced with a smaller *SM* value ranging from -0.4 to -1.2. The *SM* values in non-*CS* columns are generally several times larger. The *SM* offers information for the chunkers to avoid over- or under-chunking conjuncts in phrases with a coordination marker (e.g. 'and') because over- or under-chunking may lead to a large *SM* value. In essence, the *SM* captures the syntactic branching property.

Table 12. *AR* values of coordinate structures (+*CS* = node that immediately dominates a *CS*; -*CS* otherwise; * = the mean in the column is statistically significantly different from the mean in the immediately following column)

NP	*L3 (-CS)*	*L3-(+CS)*	*L4 (-CS)*	*L4-(+CS)*
N	27,950	222	10,222	605
Mean	0.569	0.433	0.598	0.623
S.D.	0.156	0.145	0.161	0.149
t_{score}	13.911*		-3.991*	
PP	*L3 (-CS)*	*L3-(+CS)*	*L4 (-CS)*	*L4-(+CS)*
N	11,288	41	11,522	15
Mean	0.471	0.544	0.640	0.532
S.D.	0.139	0.124	0.150	0.088
t_{score}	-3.761*		4.744*	
VP	*L3 (-CS)*	*L3-(+CS)*	*L4 (-CS)*	*L4-(+CS)*
N	21,855	197	17,711	324
Mean	0.529	0.647	0.624	0.666
S.D.	0.137	0.121	0.152	0.133
t_{score}	-13.609*		-5.617*	

We also want to briefly compare our parser with parser in Charniak & Johnson (2005). Their parser also integrated some global features such as the conjunct parallelism, right-branching, and the "heaviness" of the phrases. They re-rank the output of a k-best parser with a final F-score of 91%. The re-ranking procedure, which is a means of post-processing, goes beyond the history-based models and captures any possible features numerically. However, this global optimization approach suffers from the drawback of reparsing the training Treebank repeatedly. A serious problem in the approach is that the truly correct parse may not even be included in the restricted small subset of all possible analyses. As shown in Collin (2000), 41% of the correct parses were not included in the set of 30-best parses which are supposed to be shortlisted by the subsequent re-ranking mechanism.

Table 13. *SM* values of coordinate structures (+*CS* = node that immediately dominates a *CS*; -*CS* otherwise; * = the mean in the column is statistically significantly different from the mean in the immediately following column).

NP	*L3 (-CS)*	*L3-(+CS)*	*L4 (-CS)*	*L4-(+CS)*
N	27,950	222	10,222	605
Mean	-1.321	-0.397	-1.448	-1.162
S.D.	2.010	2.190	1.806	2.047
t_{score}	-6.266*		-3.360*	
PP	*L3 (-CS)*	*L3-(+CS)*	*L4 (-CS)*	*L4-(+CS)*
N	11,288	41	11,522	15
Mean	-3.332	-0.580	-3.955	-0.353
S.D.	1.136	1.068	1.104	1.743
t_{score}	-16.465*		-8.002*	
VP	*L3 (-CS)*	*L3-(+CS)*	*L4 (-CS)*	*L4-(+CS)*
N	21,855	197	17,711	324
Mean	-4.488	-0.628	-4.063	-0.793
S.D.	1.350	2.136	1.364	1.676
t_{score}	-25.319*		-34.908*	

6.3 Further Work

The reported parser still has some room for improvement. First, our current bottom-up chunking strategy considers only the best chunking and merging sequence at each time. It is better to introduce *N*-best chunking sequence and propagated more than one possible structure up the tree. Second, it would be interesting to integrate TTFs in combination with other design features, such as rule splitting, into the parser. Third, an insight from Collins (2003) is that head POS (in addition to head words) in lexicalized parsing can capture head-modifier relationships and subcategorization frames. In unlexicalized models, we can attach the feature of the head POS tag of the phrase to enrich the information about the phrases.

7 Conclusion

We propose to capture and quantify tree topological information using the four sets of TTFs. On our unlexicalized parser, all four sets of features have been demonstrated to produce the performance gain over the baseline model. Taking advantage of the ensemble learning technique, our best unlexicalized parsing F-score stands at 84.0%, similar to that of the first generation lexicalized parser. TTFs can be inexpensively computed and flexibly incorporated into different types of parsers. TTFs are effective in capturing basic linguistic properties, such as grammatical weight and branching direction, which are overlooked in previous studies of parsing. Our treebank statistical analysis shows that the proposed features are good indicators of tree topology of various phrase types. The parser can integrate the information in the decision of phrase formation. The analysis of TTFs in *CS*s and non-*CS*s highlights how the *SM* can help resolve the scope ambiguity in *CS*s. Unlike some approaches which take advantages of tree topology in their post-processing phase, our approach will put to good use of the features in predicting phrase boundaries and their syntactic tags, as a critical ambiguity resolution issue, during the early stage of parsing.

Acknowledgments

The work described in this paper was partially supported by the grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project Nos. CUHK440607 and CUHK440609). We also thank Henry Lee, our computer officer, for his network support during the experiments.

References

Abney, S.: Parsing by Chunks. In: Berwick, R., Abney, S., Tenny, C. (eds.) Principle-Based Parsing. Kluwer Academic, Dordrecht (1991)

Agirre, E., Baldwin, T., Martinez, D.: Improving Parsing and PP Attachment Performance with Sense Information. In: Proceedings of the 46th Annual Meeting of the Human Language Technology Conference (HLT 2008), pp. 317–325 (2008)

Black, E., Jelinek, F., Lafferty, J., Magerman, D., Mercer, R., Roukos, S.: Towards History-based Grammars: Using Richer Models for Probabilistic Parsing. In: Proceedings of the 5th DARPA Speech and Natural Language Workshop, pp. 31–37 (1992)

Chan, S., Cheung, L., Chong, M.: Tree Topological Features for Unlexicalized Parsing. In: Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010): Poster Volume, pp. 117–125 (2010)

Charniak, E.: A Maximum-Entropy-Inspired Parser. In: Proceedings of NAACL 2000, pp. 132–139 (2000)

Charniak, E., Johnson, M.: Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005), pp. 173–180 (2005)

Chomsky, N., Miller, G.: Introduction to the Formal Analysis of Natural Languages. In: Luce, R., Bush, R., Galanter, E. (eds.) Handbook of Mathematical Psychology, vol. 2, pp. 269–321. Wiley, New York (1963)

Collins, M.: Head-driven Statistical Models for Natural Language Parsing. Ph.D. thesis, University of Pennsylvania, Philadelphia (1999)

Collins, M.: Discriminative Reranking for Natural Language Parsing. In: Machine Learning: Proceedings of the Seventeenth International Conference (ICML 2000), Stanford, California, pp. 175–182 (2000)

Collins, M.: Head-Driven Statistical Models for Natural Language Parsing. Computational Linguistics 29(4), 589–637 (2003)

Dietterich, T.G.: Ensemble Methods in Machine Learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 1–15. Springer, Heidelberg (2000)

Dubey, A., Keller, F., Sturt, P.: A Probabilistic Corpus-based Model of Syntactic Parallelism. Cognition 109(3), 326–344 (2008)

Earley, J.: An Efficient Context-Free Parsing Algorithm. Comm. ACM 6(8), 94–102 (1970)

Frazier, L., Munn, A., Clifton, C.: Processing Coordinate Structures. Journal of Psycholinguistic Research 29(4), 343–370 (2000)

Freund, Y., Schapire, R.E.: A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)

Gibson, E.: Linguistic Complexity: Locality of Syntactic Dependencies. Cognition 68, 1–76 (1998)

Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer, Heidelberg (2001)

Kasami, T.: An Efficient Recognition and Syntax-analysis Algorithm for Context-free Languages. Scientific Report AFCRL-65-758, Air Force Cambridge Research Lab, MA (1965)

Kay, M.: Algorithm Schemata and Data Structures in Syntactic Processing. In: Readings in Natural Language Processing, pp. 35–70. Morgans Kaufmann Publishers Inc., San Francisco (1986)

Klein, D., Manning, C.: Accurate Unlexicalized Parsing. In: Proceedings of the 41st Meeting of the Association for Computational Linguistics, pp. 423–430 (2003)

Kübler, S., McDonald, R., Nivre, J.: Dependency Parsing. Morgan & Claypool Publishers (2009)

Magerman, D.: Statistical Decision-tree Models for Parsing. In: Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics, pp. 276–283 (1995)

Marcus, M., Santorini, B., Marcinkiewicz, M.: Building a Large Annotated Corpus of English: the Penn Treebank. Computational Linguistics 19(2), 313–330 (1993)

Matsuzaki, T., Miyao, Y., Tsujii, J.: Probabilistic CFG with Latent Annotations. In: Proceedings of the 43rd Annual Meeting of the ACL, pp. 75–82 (2005)

Petrov, S., Klein, D.: Learning and Inference for Hierarchically Split PCFGs. In: Proceedings of the 22nd Conference on Artificial Intelligence, Nectar Track, Vancouver (2007)

Quirk, R., Greenbaum, S., Leech, G., Svartvik, J.: A Grammar of Contemporary English. Longman, London (1972)

Ramshaw, L.A., Marcus, M.P.: Text Chunking Using Transformation-based Learning. In: Proceedings of the Third Workshop on Very Large Corpora, pp. 82–94 (1995)

Ratnaparkhi, A.: Learning to Parse Natural Language with Maximum Entropy Models. Machine Learning 34, 151–175 (1999)

Rosenbach, A.: Animacy versus Weight as Determinants of Grammatical Variation in English. Language 81(3), 613–644 (2005)

Sagae, K., Lavie, A.: A Classifier-Based Parser with Linear Run-Time Complexity. In: Proceedings of the Ninth International Workshop on Parsing Technologies (IWPT), pp. 125–132 (2005)

Sang, E.: Transforming a Chunker to a Parser. In: Veenstra, J., Daelemans, W., Sima‘an, K., Zavrel, J. (eds.) Computational Linguistics in the Netherlands 2000, pp. 177–188 (2001)

Schapire, R.E., Singer, Y.: BoosTexter: A Boosting-based System for Text Categorization. Machine Learning 39, 135–168 (2000)

Stetina, J., Nagao, M.: Corpus-based PP Attachment Ambiguity Resolution with a Semantic Dictionary. In: Proceedings of the 5th Workshop on Very Large Corpora, Beijing, China, pp. 66–80 (1997)

Tsuruoka, Y., Tsujii, J.: Chunk Parsing Revisited. In: Proceedings of the 9th International Workshop on Parsing Technologies, pp. 133–140 (2005)

Wasow, T.: Remarks on Grammatical Weight. Language Variation and Change 9, 81–105 (1997)

Xiong, D., Li, S., Liu, Q., Lin, S., Qian, Y.: Parsing the Penn Chinese Treebank with Semantic Knowledge. In: Dale, R., Wong, K.-F., Su, J., Kwong, O.Y. (eds.) IJCNLP 2005. LNCS (LNAI), vol. 3651, pp. 70–81. Springer, Heidelberg (2005)

Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics?

Christopher D. Manning

Departments of Linguistics and Computer Science,
Stanford University,
353 Serra Mall, Stanford CA 94305-9010
manning@stanford.edu

Abstract. I examine what would be necessary to move part-of-speech tagging performance from its current level of about 97.3% token accuracy (56% sentence accuracy) to close to 100% accuracy. I suggest that it must still be possible to greatly increase tagging performance and examine some useful improvements that have recently been made to the Stanford Part-of-Speech Tagger. However, an error analysis of some of the remaining errors suggests that there is limited further mileage to be had either from better machine learning or better features in a discriminative sequence classifier. The prospects for further gains from semi-supervised learning also seem quite limited. Rather, I suggest and begin to demonstrate that the largest opportunity for further progress comes from improving the taxonomic basis of the linguistic resources from which taggers are trained. That is, from improved descriptive linguistics. However, I conclude by suggesting that there are also limits to this process. The status of some words may not be able to be adequately captured by assigning them to one of a small number of categories. While conventions can be used in such cases to improve tagging consistency, they lack a strong linguistic basis.

1 Isn't Part-of-Speech Tagging a Solved Task?

At first glance, current part-of-speech taggers work rapidly and reliably, with per-token accuracies of slightly over 97% [1–4]. Looked at more carefully, the story is not quite so rosy. This evaluation measure is easy both because it is measured per-token and because you get points for every punctuation mark and other tokens that are not ambiguous. It is perhaps more realistic to look at the rate of getting whole sentences right, since a single bad mistake in a sentence can greatly throw off the usefulness of a tagger to downstream tasks such as dependency parsing. Current good taggers have sentence accuracies around 55–57%, which is a much more modest score. Accuracies also drop markedly when there are differences in topic, epoch, or writing style between the training and operational data.

Still, the perception has been that same-epoch-and-domain part-of-speech tagging is a solved problem, and its accuracy cannot really be pushed higher. I think it is a common shared meme in at least the U.S. computational linguistics community that interannotator agreement or the limit of human consistency on part-of-speech tagging is 97%. As various authors have noted, e.g., [5], the second wave of machine learning part-of-speech taggers, which began with the work of Collins [6] and includes the other taggers cited above, routinely deliver accuracies a little above this level of 97%, when tagging material from the same source and epoch on which they were trained. This has been achieved by good modern discriminative machine learning methods, coupled with careful tuning of the feature set and sometimes classifier combination or semi-supervised learning methods. Viewed by this standard, these taggers now *clearly exceed* human performance on the task. Justifiably, considerable attention has moved to other concerns, such as getting part-of speech (POS) taggers to work well in more informal domains, in adaptation scenarios, and within reasonable speed and memory limits.

What is the source of the belief that 97% is the limit of human consistency for part-of-speech tagging? It is easy to test for human tagging reliability: one just makes multiple measurements and sees how consistent the results are. I believe the value comes from the `README.pos` file in the `tagged` directory of early releases of the Penn Treebank. It suggests that the "estimated error rate for the POS tags is about 3%".[1] If one delves deeper, it seems like this 97% agreement number could actually be on the high side. In the journal article on the Penn Treebank [7], there is considerable detail about annotation, and in particular there is description of an early experiment on human POS tag annotation of parts of the Brown Corpus. Here it was found that if two annotators tagged for POS, the interannotator disagreement rate was actually 7.2%. If this was changed to a task of correcting the output of an automatic tagger (as was done for the actual Penn Treebank), then the disagreement rate dropped to 4.1%, and to 3.5% once one difficult text is excluded. Some of the agreement is then presumably both humans adopting the conventions of the automatic POS tagger rather than true human agreement, a topic to which I return later.

If this is the best that humans can give us, the performance of taggers is clearly at or above its limit. But this seems surprising – anyone who has looked for a while at tagger output knows that while taggers are quite good, they regularly make egregious errors. Similarly, examining portions of the Penn Treebank by hand, it is just very obvious that there are lots of errors that are just mistakes rather than representing uncertainties or difficulties in the task. Table 1 shows a few tagging errors from the beginning of section 02 of the training data.[2] These are all cases where I think there is no doubt about what the correct tag should be, but that nevertheless the annotator failed to assign it. It seems

[1] This text appears up through LDC95T7 Treebank release 2; the statement no longer appears in the much shorter README included in the current LDC99T42 Treebank release 3.). This error rate is also mentioned in [7, pp. 327–8].

[2] My informal impression is that the accuracy of sections 00 and 01 is considerably worse, perhaps reflecting a "burn in" process on the part of the annotators. I think it is in part for this reason that parsers have been conventionally trained on sections 02–21 of the Penn Treebank. But for POS tagging, most work has adopted the splits introduced by [6], which include sections 00 and 01 in the training data.

clear that the inter-annotator agreement of humans depends on many factors, including their aptitude for the task, how much they are paying attention, how much guidance they are given and how much of the guidance they are able to remember. Indeed, Marcus et al. [7, p. 328] express the hope that the POS error rate can be reduced to 1% by getting corrections from multiple annotators, adjudicating disagreements, and using a specially retrained tagger. However, unfortunately, this work never took place. But using the tools developed over the last two decades given the existence of the Penn Treebank, we are now in a much better position to do this, using semi-automated methods, as I discuss below.

Table 1. Examples of errors in Penn Treebank assigned parts-of-speech, from section 02 of the *WSJ*

Time , the/DT largest/JJS ***newsweekly/RB*** *, had average circulation of*
  Correct: **newsweekly/NN**
below the $ 2.29 billion value United Illuminating ***places/NNS*** *on its bid*
  Correct: **places/VBZ**
Rowe also noted that political concerns also ***worried/VBN*** *New England Electric .*
  Correct: **worried/VBD**
Commonwealth Edison now faces an additional court-ordered refund on its summer/winter rate differential collections ***that/IN*** *the Illinois Appellate Court has estimated at $ 140 million .*
  Correct: **that/WDT**
Joseph/NNP M./NNP Blanchard/NNP , 37 , vice president , engineering ; Malcolm/NNP ***A./NN*** *Hammerton/NNP*
  Correct: **A./NNP**

2 Approaching the Asymptote: Continuing to Push Up POS Tagging Numbers

Since the time of our last POS tagger paper [1], I've added a few features that have slightly pushed up the performance of the Stanford POS tagger. I give the details of those models here. But it is noticeable that they do not improve overall performance by very much. Other people seem to be hitting the same wall, and while there are fractionally better results from others, none are much better. Suppose somehow that more machine learning magic can get numbers up from 97.3% per-token accuracy to 97.5% per-token accuracy. That would still mean that the last decade will only have solved about 1/6 of the errors remaining in part of speech taggers.

2.1 Incremental Improvements

The experiments I present here describe incremental work: There are no big changes to the architecture, but some improvements in the features, parameters, and learning methods give small incremental gains in POS tagging performance, bringing it close to parity with the best published POS tagging numbers in 2010. These numbers are on the now fairly standard splits of the *Wall Street Journal* portion of the Penn Treebank for POS tagging, following [6].[3] The details of the corpus appear in Table 2 and comparative results appear in Table 3.

Table 2. *WSJ* corpus for POS tagging experiments

Set	Sections	Sentences	Tokens	Unknown
Training	0-18	38,219	912,344	0
Development	19-21	5,527	131,768	4,467
Test	22-24	5,462	129,654	3,649

Table 3. Tagging accuracies with different feature templates and other changes on the *WSJ* 19-21 development set

Model	Feature Templates	# Feats	Sent. Acc.	Token Acc.	Unk. Acc.
3GRAMMEMM	See text	248,798	52.07%	96.92%	88.99%
NAACL 2003	See text and [1]	460,552	55.31%	97.15%	88.61%
Replication	See text and [1]	460,551	55.62%	97.18%	88.92%
Replication′	+rareFeatureThresh = 5	482,364	55.67%	97.19%	88.96%
5W	$+\langle t_0, w_{-2}\rangle, \langle t_0, w_2\rangle$	730,178	56.23%	97.20%	89.03%
5WSHAPES	$+\langle t_0, s_{-1}\rangle, \langle t_0, s_0\rangle, \langle t_0, s_{+1}\rangle$	731,661	56.52%	97.25%	89.81%
5WSHAPESDS	+ distributional similarity	737,955	56.79%	97.28%	90.46%

3GRAMMEMM shows the performance of a straightforward, fast, discriminative sequence model tagger. It uses the templates $\langle t_0, w_{-1}\rangle$, $\langle t_0, w_0\rangle$, $\langle t_0, w_{+1}\rangle$, $\langle t_0, t_{-1}\rangle$, $\langle t_0, t_{-2}, t_{-1}\rangle$ and the unknown word features from [1]. The higher performance NAACL 2003 tagger numbers come from use of a bidirectional cyclic dependency network tagger, which adds the feature templates $\langle t_0, t_{+1}\rangle$, $\langle t_0, t_{+1}, t_{+2}\rangle$, $\langle t_0, t_{-1}, t_{+1}\rangle$, $\langle t_0, t_{-1}, w_0\rangle$, $\langle t_0, t_{+1}, w_0\rangle$, $\langle t_0, w_{-1}, w_0\rangle$, $\langle t_0, w_0, w_{+1}\rangle$ The next line shows results from an attempt to replicate those numbers in 2010. The results are similar but a fraction better.[4] The line after that shows that the numbers are pushed up a little by lowering the support threshold for including rare word features to 5. Thereafter, performance is improved a little by adding features. 5W adds the words two to the left and right as features, and 5WSHAPES also adds word shape features that we have described for named entity recognition elsewhere [8].[5] These features map words to equivalence classes

[3] In this paper, when I refer to "the Penn Treebank", I am actually referring to just the *WSJ* portion of the treebank, and am using the LDC99T42 Treebank release 3 version.

[4] I think the improvements are due to a few bug fixes by Michel Galley. Thanks!

[5] As far as I am aware, features of this sort were first introduced by Collins [9].

based on character type, such as *Mexico* to *Xxxxx* and *IA-64* to *XX-dd*. Some of the other recent taggers cited earlier have made use of even more higher order features and conjunctive features [2, 10], but in our tagger they seem to provide marginal to negative gains. Something that does really help is adding features for words based on induced distributional similarity classes, as shown on the last line.[6] Here, we use the method and code of [11], though other methods for introducing distributional similarity classes would probably work roughly as well. While the overall gains on the *WSJ* are modest, taken together, these features and the word shape features give a significant gain in performance on unknown words: errors on unknown words are reduced by 13% (relative). And one would expect these features to be even more useful when the tagger is subsequently used on text from other domains or epochs. Note, however, that the last line is for a model where the distributional similarity classes were trained separately on about 300 million words of data in an unsupervised fashion, whereas all the other models are trained only on the *WSJ* training set.

I present these numbers to show that while small amounts of progress remain possible, we clearly seem to be entering an era of diminishing returns. It seems like about 2.4% of the remaining 2.6% error rate might need to be approached from a different angle.[7]

2.2 Splitting Tags

I have shown in other work that parsing performance on the Penn Treebank can be improved enormously by splitting certain of the categories, both part-of-speech and phrasal categories, and parsing with the resulting split-category treebank grammar [12]. One might reasonably think that the same strategy could be applied successfully to the POS tagging problem, especially as a number of the most useful state splits for parsing are splits of part-of-speech categories. But, unfortunately, splitting tags seems to be largely a waste of time for the goal of improving POS tagging numbers. A thorough exploration of the possibilities can be found in [13]. My own more limited experimentation point in the same direction.

3 Error Analysis

How, then, can we solve the other 5/6 of the errors of POS taggers? An examination of the things that taggers get wrong on same-domain test text makes it clear that little of the remaining error is going to be solved by better local features of the kind used by current state-of-the-art sequence model taggers. How

[6] This line corresponds to the released version 3.0 of the Stanford POS Tagger, available at `http://nlp.stanford.edu/software/tagger.shtml`

[7] Since this investigation was part of a series of experiments on different models, they were all evaluated only on the development set (section 19–21). It is now clear from several studies, including the numbers below, that the final test set is a bit easier, and it could be expected that final test set numbers would be almost 0.1% higher. See table 6 for results on the final test set.

are we going to get the rest? To answer that, we need to understand what kinds of errors there are. In Table 4, I give a rough breakdown of where we need to look.[8]

I did a small error analysis, taking a sample of 100 errors from section 19 of the treebank. I divided errors into seven classes, as shown in Table 4. Many errors are hard to classify. When things were unclear, I allowed an error to be assigned to two classes, giving it 1/2 a point under each. I exemplify the seven classes below.

Table 4. Frequency of different POS tagging error types

Class	Frequency
1. Lexicon gap	4.5%
2. Unknown word	4.5%
3. Could plausibly get right	16.0%
4. Difficult linguistics	19.5%
5. Underspecified/unclear	12.0%
6. Inconsistent/no standard	28.0%
7. Gold standard wrong	15.5%

1. **Lexicon gap:** Here, the word occurred a number of times in the training data, but never with the tag which it has in this context. Given the nature of discriminative POS taggers, it is always going to be very difficult for context to override lexical features in this situation. For example, below, *slash* is clearly a noun, but in the training set, it occurs only but several times as a verb.
   a/DT 60/CD %/NN slash/NN in/IN the common stock dividend
2. **Unknown word:** Here the tagger has to rely only on context features, and contexts are often ambiguous. For example, below, *substandard* is a word which does not appear in the training data and it is also very reasonable for a POS tagger to guess that it might be a noun (as it did).
   blaming the disaster on/IN substandard/JJ construction/NN
3. **Could plausibly get right:** Here, you could imagine a sequence model tagger with a context of a few words or tags on either side getting the right answer, though it may be quite difficult in practice. For example, below, it seems like a sequence tagger should be able to work out that *overnight* is here functioning as an adverb rather than an adjective (the tag it chose), since it is here a verb modifier not pre-modifying a noun.
   market/NN players/NNS overnight/RB in/IN Tokyo/NNP began bidding up oil prices
4. **Difficult linguistics: Needs much syntax/semantics/discourse:** Here, it seems very clear that determining the right tag requires broad contextual knowledge that must be beyond a sequence tagger with local features. For

[8] An early, but somewhat imprecise, discussion of the different sources of tagging disagreement can be found in [14].

example, below, a tagger just cannot correctly choose between the present (VBP) and past (VBD) tag for *set* without an understanding of a multi-sentence discourse context, and happens to choose wrongly.

They/PRP set/VBP up/RP absurd/JJ situations/NNS , detached from reality

5. **Underspecified/unclear:** The tag is underspecified, ambiguous, or unclear in the context. There are several common cases of this, such as whether to choose a verbal or adjectival tag for words which have a participial inflectional form and modify a head, and whether to choose a verbal or noun tag for gerunds. While there are linguistic tests that can be used to distinguish the two categories in both these cases, often in particular contexts the correct analysis is just underspecified. For example, below, it is unclear whether *discontinued* should be regarded as an adjective or verbal participle.

   it will take a $ 10 million fourth-quarter charge against/IN discontinued/JJ operations/NNS

6. **Gold standard inconsistent or lacks guidance:** Here, there should be a right answer, but the tagging manual does not define what to do and in practice the annotators have been inconsistent, so it is not surprising that the tagger gets such things right only half the time by chance. For example, for expressions like *the '30s* below, or indeed corresponding ones like *the 1930s*, the treebank is inconsistent in sometimes tagging them as CD and at other times as NNS. There should be a clear answer here which should be consistently used, but none was defined, and human annotators were inconsistent. (If the tag CD is construed fairly strictly as cardinal numbers – for example, ordinals are definitely excluded and tagged as adjectives (JJ), then it seems to me like these expressions shouldn't be tagged CD, and that NNS is correct, and below we retag in this fashion, but in the Treebank, the two taggings are almost exactly equally common, with a couple of tokens also tagged as NN, to add variety.)

   Orson Welles 's Mercury Theater in/IN the/DT '30s/NNS ./.

7. **Gold standard wrong:** The tag given in the gold standard is clearly wrong. For example, below, the tag of VB for *hit* is just wrong. It should be a VBN, as the passive participle complement to *got*. Other examples of this sort appeared in Table 1.

   Our market got/VBD hit/VB a/DT lot/NN harder/RBR on Monday than the listed market

What conclusions can we draw? While semi-supervised methods like the distributional similarity classes above are very useful for handling unknown words, their ability to improve overall tagger performance numbers appear quite limited. At most they can address errors in classes 1 and 2, which account for less than 10% of the errors, and in practice they are likely to address only errors in class 2, which are about 5% of the errors, since in discriminative sequence models, lexical features are very strong and it is difficult for context to override them.[9] The progress that has been made in the last decade in POS tagging has

[9] But, again, this is for same-epoch-and-domain testing; their impact is much greater when tagging data from disparate domains.

presumably come mainly from handling some of the cases in class 3, and there is presumably still a fraction of space for improvement here. But many of the cases in class 3 shade off into cases of class 4, where it is hard to imagine a sequence model POS tagger getting them right, except sometimes by a lucky guess. At any rate, classes 3 and 4 together comprise less than one third of the errors. The cases in class 5 are inherently difficult; we return to them at the end. The easiest path for continuing to improve POS tagging seems to be to look at the cases in classes 6 and 7, where the gold standard data is just wrong or is inconsistent because of the lack of clear tagging guidelines. These classes comprise over 40% of the data, and, indeed, if some of the cases that I regard as unspecified or unclear (class 5) could be made clear by tightening up the guidelines, then we might be dealing here with over half the remaining errors. The road on this side of the fence is much less traveled, but I believe it now provides the easiest opportunities for tagging performance gains.

4 Correcting the Treebank

From the earliest days of the resurgence of statistical NLP, there has been a very strong current against fixing data. I think the attitude originated at IBM. For example, one can find a discussion of the issue in David Magerman's thesis [15, p. 101]. I think the idea is that the world is noisy, and you should just take the data as is, in contrast with old-style NLP, which dealt with constructed and massaged data. In addition there are also clear concerns about the overfitting of models, and of model builders being influenced to assign the labels that their models predict. At any rate, one of the big advantages of this perspective is that everyone is using exactly the same training and test sets, and so results are exactly comparable and should be reproducible (after pinning down a few more things about evaluation metrics, etc.).

While it is of course important for everyone to be aware of changes that particular experimenters have made to data sets, and there is certainly value in constant training and test sets for the sake of comparable experiments, it seems that a desire for constancy can be and has been carried much too far. We are now 15 years from the distribution of Penn Treebank release 2, which was the final version of the *WSJ* data, and many researchers have variously noticed mistakes and deficiencies in the annotation, but virtually no attempt has been made to correct them.[10] For example, Ratnaparkhi [16] notes that a

[10] This is not fully true: Work on the PropBank did lead to revisions to and corrections of the Treebank as part of a PropBank-Treebank merge activity, and some other ideas for improving treebank structure (for noun phrase structure and hyphenation) have been incorporated into OntoNotes (LDC2009T24), a new, unified corpus which includes large sections (but not all) of the classic *WSJ* Treebank. However, there hasn't been correction of a lot of the miscellaneous small-scale errors, and transitioning the community to OntoNotes is still very much a work in progress, with the vast majority of current work on English treebanks and POS tagging and parsing still using the original Penn Treebank Wall Street Journal data.

large source of errors in his tagger is that the tags for certain common words like *about* are inconsistent across the corpus, and, indeed, that the name of the annotator of an example is one of the best predictors of tag assignment. Similarly, Abney et al. [17] examine the most anomalous word tokens that get the highest weights when applying boosting to POS tagging and show that many of these tokens have erroneous tags.

At some point the desire for corpus constancy becomes dysfunctional. The de facto situation with the *WSJ* treebank contrasts with what you see in other fields such as taxonomic biology. It is just not the case that because the first person who collected a certain specimen said it was an Acacia species that for all time it continues to be called an Acacia species, even when further evidence and testing makes it perfectly clear that it is not. At both the individual and species level, the taxonomic biology world has been willing to tolerate quite large scale renamings and disruptions so as to improve the ontological basis of the field. Such is scientific progress in a taxonomic field. The same thing should happen with the content of treebanks.[11]

In computational linguistics, the main work that has been done on improving the taxonomy of tags to allow clearer automatic tagging and improving the conventions by which tags are assigned has been done within the English Constraint Grammar tradition [18, 19]. Contrary to the results above, this work has achieved quite outstanding interannotator agreement (up to 99.3% *prior* to adjudication), in part by the exhaustiveness of the conventions for tagging but also in part by simplifying decisions for tagging (e.g., all *-ing* participles that premodify a noun are tagged as adjectives, regardless). It is surprising the extent to which this work has been ignored by the mainstream of computational linguistics. In some ways, the present work tries to apply some of the same approach to generating consistent taggings, but without performing revisions to the tag set used.

While one way to achieve the goal of correcting the treebank would for humans to carefully check tag assignments, further linguistic annotation work and the developments in language technology provide other methods. A very good way to find errors and inconsistencies in tag assignments is to see where tools like taggers go wrong, as in the examples cited above [16, 17]. Inconsistencies can also be detected by methods aimed just at this task, an idea notably explored by Dickinson [20]. But for the Penn Treebank there is also another profitable approach to pursue. An examination of the corpus makes clear that the *treebanking* was done much more carefully and consistently than the POS tagging. Since, following the ideas of X$'$ theory, the POS tag of words can often be predicted from phrasal categories, we can often use the tree structure and phrasal categories to tell us what the POS tags should have been. This is the main strategy used here to reduce the error and inconsistency rate in the Penn Treebank. While Dickinson's methods are interesting, they do not provide a sufficient level of precision for fully automatic treebank correction, whereas using the treebank

[11] One venue where this may increasingly happen in the future is in the more open data Manually Annotated Sub-Corpus (MASC) of the American National Corpus: `http://www.anc.org/MASC/`

syntactic structure commonly does. So, I use testing on the training data to identify inconsistencies, and then information from the treebank structure to guide correction.

5 Fixing Some of the Errors

Many of the errors and inconsistencies in the Penn Treebank are quite systematic and are well-suited to fixing by deterministic rules. Here, we use Tsurgeon scripts [21], which work by matching a tree pattern using Tregex (a tgrep-like language; cf. [22]) and then performing operations on matched parts of the tree. Here are a couple of cases of the kinds of errors we can straightforwardly fix.[12]

5.1 Past Tense versus Past Participles

There are quite frequent tagging errors as to when verb forms are marked as past tense (VBD) versus past participles (VBN). In general, if a past participle is not adjacent to a passive or perfective auxiliary indicating a VBN, then it is quite frequently wrongly tagged as VBD.[13] But such cases can usually be detected and fixed by rules over tree patterns. For instance, we can use rules such as this one:

```
@VP < VBD=bad [ > (@VP < (/^VB/ < be_have_get )) | > (@VP <
CONJP|CC > (@VP < (/^VB/ < be_have_get ))) | > (@NP < @NP) ]
relabel bad VBN
```

That is, a verb in a VP that is under a VP containing a passive or perfective auxiliary verb (perhaps inside a conjunction structure) or modifying a noun phrase should really be a participle VBN and not a past finite VBD.

5.2 Plurals as Singulars

The (reasonable) convention for practical part of speech tagging is that words receive a single word class. This flies in the face of commonly accepted ideas of linguistic morphology where there can be zero derivation of an X^0 category from another X^0 category, with a change in category. There are many such cases in the Penn Treebank, some clearer and some less clear. One case is with plural nouns that become incorporated into named entities which are then treated as singular. Examples include *(the) United States*, *(the) Parks Council*, and *Kawasaki Heavy*

[12] I will not dwell on the problems caused by hyphenation in the original Penn Treebank, since they are widely recognized and have been reformed in more recent LDC treebanking projects, including the OntoNotes corpus, which contains many of the trees of the Penn Treebank in an updated form which improves the representation of hyphenated terms and complex NPs with left-branching structure.

[13] One could suspect that these errors in many cases reflect a bias resulting from the use of an automatic tagger in the construction of the Penn Treebank, since these were probably cases that the tagger got wrong, and the human annotator then failed to correct.

Industries. The noun of interest is clearly morphologically plural. But despite the fact that it would normally be regarded as the head of the noun phrase in the first and third examples, the whole noun phrase takes singular verb agreement (*the United states is changing ...*). Given the stated preference for the Penn Treebank to tag on the basis of syntactic function, it seems like the verbal agreement is a good reason to tag such words as singular NNP, but in practice they are usually – though not consistently – tagged as NNPS. The right answer isn't entirely clear here, involving both how fossilized the formation is and whether to choose the original or final category in cases of X^0 zero derivation. However, in this case, most of the compounds are fairly transparent, and annotators prefer to go with the morphology around three-quarters of the time. Since it is in fashion at the moment to go with the wisdom of crowds, I will adopt this convention.

Another similar problem is when non-nouns become involved in proper nouns, such as *United* in either *United Airlines* or *(the) United States.* It is unclear whether to stick with the adjectival tag for *United* or to call it a proper noun because the whole phrase is clearly a proper noun. In this case, human annotators overwhelmingly went with the NNP choice, and, again, I will follow the wisdom of the crowd. I will summarize these two decisions as the United States Principles.[14] In practice, we can handle cases of these principles simply with rules like:

```
NNP=bad < Industries|Airlines
relabel bad NNPS
```

5.3 *That*

That is a hard word for taggers, since it can function as all of a determiner, complementizer, relative pronoun, and an adverb (*he isn't that sick*), and has separate tags for each function (DT, IN, WDT, and RB). Some cases of *that* are clearly ones that need syntactic structure to get right and are beyond the reach of a POS tagger. But there are also lots of errors in the tagging of *that* in the training data, and we may as well at least correct those, so that the POS tagger

[14] There are further issues here. On proper nouns getting a noun tagging, the Penn Treebank part-of speech tagging guidelines [23, sec 5.3] make a stronger claim, saying that any capitalized word that is part of a name should be tagged NNP or NNPS. This seems too strong, since sometimes verbs and function words are capitalized as parts of names, such as in the titles of books like *Gone With The Wind.* I feel that this is just wrong and very confusing to an POS tagger. Such titles are larger-level syntactic units. While the annotators sometimes followed this dictate, the majority of the time they ignored it. Again, we will follow the wisdom of crowds. This rule will only be applied to content words of a base NP that are part of a name.

Secondly, there are also proper adjectives such as *Australian* or *North Korean.* These are also tagged inconsistently as adjectives or proper nouns in the Penn treebank. Arguably, it is a bad defect of the the Penn Treebank tag set that it lacks a proper adjective category that would cover these cases. But, given the current tag set, when these expressions occur as adjectival modifiers (rather than as a noun referring to a person) then it seems clear that they should be tagged as JJ. These are not proper nouns.

gets the best chance it can to learn the distinctions. Again, we use the parse structure to guide the correction:

```
@NP < (IN|WDT=bad < /^(?:a|that|That)$/)
relabel bad DT

@SBAR < (DT|WDT|NN|NNP|RB=bad < that|because|while|Though)
relabel bad IN

@ADJP < JJ < (IN=bad < that)
relabel bad RB
```

The first rule matches 173 times in the Penn Treebank, while the second rule matches 285 times. These aren't really rare errors we are talking about.

5.4 Miscellaneous Errors and Inconsistencies

Many of the details of inconsistencies are particular to individual lexical items and quite mundane. To take one example that turns up a bunch of times, for *K mart*, annotators were inconsistent on whether to tag *mart* as a proper noun or not, presumably because it is not capitalized. It seems to me that it should be treated as still a proper noun, and at any rate, this should just be consistent. This rule make it consistent:

```
@NP < (NNP < K $+ (NN=bad < mart))
relabel bad NNP
```

5.5 Tagging Results

Overall, I defined a couple of hundred such rules, based on examination of the training data, some of which changed several hundred tags, others of which changed only a single tag. Some were aimed at outright errors and others at inconsistencies. The rules certainly don't exhaust all the errors and inconsistencies found in the training data, but there are enough covering a number of the most common problem that we can get some idea as to whether such taxonomic improvements might noticeably lift tagging accuracy.

The one complication is how to assess this with respect to test sets. As I show below, if you only correct the training data, then no gains are achieved. This is because the test data has all the same errors and inconsistencies as before. Indeed, the uncorrected tagger may pick up some of any patterning that exists in "inconsistent" tagging, and do better on the test set. Therefore, the strategy adopted here is as follows: Change rules are developed looking at the training data. These rules are then tested by examining their effect on the development data. It is checked that they do not apply wrongly in any situations (if they do, they are refined to make their application more limited and precise (and the process repeated), or just discarded following Hippocratic reasoning of first doing no harm. Then, the final rules are applied to the final test data without examining their effect. That is, the changes are assumed to all be correct for the test data. Of course, there is a small risk here that a rule could misapply, but the sanctity of the final test data is preserved. Moreover, based on the precise nature of the change rules and examination of their effects on development test data, I feel highly confident that at least 98% of the changes will be good corrections of consistentizations of the test data.

In Table 5 we show the effects of this process on the development data. Note that the number of features in the tagger goes down a bit with data correction because there is less entropy in tag assignments. The error reduction between the first and last lines is already quite substantial (by the standards of these things), and would presumably increase further with further extension and refinement of the correction rule set. Finally, Table 6 show the scores of several models on the final test data.

Table 5. Effect of correction on tagging accuracy on the *WSJ* 19–21 development set

Model	**Corrected Train**	**Corrected Test**	**# Feats**	**Sent. Acc.**	**Token Acc.**	**Unk. Acc.**
5wShapesDS	no	no	737,955	56.79%	97.28%	90.46%
	no	yes		57.95%	97.38%	90.60%
	yes	no	735,679	55.87%	97.21%	90.58%
	yes	yes		62.66%	97.75%	90.75%

Table 6. Accuracy of taggers on the final test set *WSJ* 22–24

Model	**Corrected Data**	**Sentence Accuracy**	**Token Accuracy**	**Unknown Accuracy**
naacl 2003	no	55.75%	97.21%	88.50%
Replication	no	56.44%	97.26%	89.31%
5wShapes	no	56.65%	97.29%	89.70%
5wShapesDS	no	56.92%	97.32%	90.79%
5wShapesDS	yes	61.81%	97.67%	90.49%

6 Foundational Issues

Notwithstanding the significant progress that can be made by removing errors and improving the consistency of the treebank, there are interesting foundational linguistic issues as to which decisions are linguistically well-justified, and which turn into arbitrary conventions of treebank annotation. The latter can still give consistency, but cannot really be linguistically justified.[15]

What I want to look at is the *validity* of what is in the Penn Treebank:

> "Measurement requires three things: An *object* to be measured, a well-defined *property* of the object to measure, and a *measuring instrument* that actually does the job" [24, 135].

[15] For instance, consistency could be trivially guaranteed by always giving the same tag to each token of a word type.

The objects at hand here are clear: words and sentences of English newswire. My concerns touch the other two issues: Are part-of-speech labels well-defined discrete properties enabling us to assign each word a single symbolic label? Secondly, is the measuring instrument up to the task? Answering questions like this is one clear place where linguists should have something useful to offer to the modern world of Statistical NLP.

The first question is in many ways the more interesting. If the properties of part of speech and syntactic category are not well-defined, then the variables assigned by the coder lack a coherent basis. Is it possible to assign to each word in a context a single symbol that represents the word's syntactic category? Or do we need something like squishy categories [25]? While the use of discrete categories underlies most of modern generative linguistics, fuzziness is readily accepted by a descriptive grammar such as [26], which regularly refers to the "fuzzy borders between word classes".[16] A thorough recent examination of the issues is found in [27]. Given that the behavior of some words has gradually changed from one part of speech to another over time,[17] some gradable notion of category is presumably necessary. On the other hand, one needs to account for the fact that it seems reasonable and feasible to assign such a category as *noun* or *verb* to the vast majority of the words in the lexicon. This could perhaps be connected up with work on categorical perception [30] which attempts to explore how phenomena which are grounded in continuous physical quantities are perceived by human beings as belonging to discrete categories, with only a little fuzz around the edges.

What is it that treebankers are actually assigning as categories? [29] showed that many of the criteria that people often use for part of speech are actually sensitive to semantic sorts. Are treebankers mainly influenced by semantic function or are they really picking out structural categories? According to generative wisdom, notional (semantic) criteria for part of speech are "extremely unreliable" [31, 57], but given that they are what is taught in school, if anything ("a noun is a person, place or thing"), there is a high probability that treebankers often use these rather than true syntactic distributional categories. Here I present one example of this phenomenon. I discussed a couple of others in [32].

6.1 Transitive Adjectives

Maling [29] discusses the three words *near*, *like*, and *worth*, arguing that these words were historically clearly adjectives, but that with the loss of case marking

[16] Where a treebanker was uncertain concerning the proper part-of-speech tag, they could give words disjunctive tags, and the journal paper [7] describes this as part of a policy of not having annotators make arbitrary decisions. However, in practice, this option was little used, with only 0.01% of tokens (147 tokens) receiving an ambiguous tag. In the vast majority of cases of indeterminacy, it is clear that the annotator either did just make an arbitrary decision or else accepted the decision of the automatic tagger that preceded them. Hence the inconsistency noted in the previous section.

[17] For examples and discussion, see [28], [29], and the discussion below.

in English, *like* and *worth* shifted syntactic category to become prepositions (the more appropriate category for uninflected words that take an NP complement), while *near* is perhaps the only surviving case of a transitive adjective in English. In various footnotes, two other candidate surviving transitive adjectives are suggested: *opposite* and *due*. Searching the treebank reveals another possible transitive adjective: *outside*.[18] Table 7 shows a summary of the occurrence of these words in the Penn Treebank Wall Street Journal corpus.

Table 7. Parts of speech assigned to putative transitive adjectives in the Penn Treebank

	Total	IN	JJ	NN	NNS	RB	VB(P)
due	371		344	2	1	24	
like	580	461	26				93
near	126	97	24			5	
outside	145	80	52	8		5	
opposite	19	1	12	6			
worth	114	10	65	39			

The case of *worth* is well-studied. It is a recognized problem word and the treebank manuals have specific, if inconsistent, instructions for it. The initial guide to part of speech tagging said [23, p. 31]:

> *worth* is a preposition (IN) when it precedes a measure phrase, as in *worth ten dollars*.

The subsequent Treebanking manual provides an odd mixture of descriptive and prescriptive advice, but seems to reverse this earlier judgment [33, pp. 308–309]:

> *worth*:
> 1. with complement: ADJP
>   Note that some instances of this use of *worth* are labeled PP-PRD, as in (b); however the use of ADJP-PRD, as in (a), predominates.
>   (a) [S [NP-SBJ [NP the results], [ADJP however general],] [VP are [ADJP-PRD worth [NP the search]]]]
>   (b) [S [NP-SBJ [NP the results], [ADJP however general],] [VP are [PP-PRD worth [NP the search]]]]
> 2. *dollars worth*: NP
>   There is considerable variation, but here is a common way of analyzing expressions like *five dollars worth*:
>   [VP issue [NP [NP [ADJP [QP some $ 3 million to $ 4 million] *u*] worth] [PP of [NP Rural Roads Authority bonds]]]]

Commented out in the file is: "Sorry, there ain't no 'right' way for these. –R.". This is the essence of the problem. It is generally accepted that *worth* appears in

[18] The only other word tagged as an adjective and followed by an NP complement is one instance of *such*, but this is because of a clear typo in the newswire source: **Akzo has high hopes for some emerging fiber businesses, such carbon fibers and aramid.*

certain contexts as a noun, but, in the remaining cases, is *worth* a preposition, as Maling and Santorini propose, an adjective as the new Treebanking manual proposes (and also, both the Oxford English Dictionary and Huddleston and Pullum [34]), or should we un-ask this question?

6.2 Treebank Evidence

There are 114 instances of *worth*, selectively shown in Table 8. 10 examples are tagged as a preposition, 8 in phrases that treebankers later tagged as ADJPs (1–2) and two that were later tagged as PPs (3–4). In one of the former, the complement is incorrectly tagged as an adverbial (5). 65 examples were tagged as JJ, 48 placed in ADJPs (6–7), 13 placed in PPs (8–9) and 4 which occur inside noun phrases and should have been tagged as NN (10–11). 39 examples were tagged as NN: 2 of these were incorrect and should have been given a non-noun tag (12–13). The rest are noun uses including after a quantifier phrase (14–15) and in other noun uses including compounds (16–17). In 4 cases involving quantifiers (all cases involving a following PP), an extra erroneous level of ADJP structure has been added (18).

There are various questions and concerns here. The OED lists *worth* as a noun, and as an adjective (and as an obsolete verb). [26] appears to regard *worth* as both a preposition and an adjective. On p. 1064 they argue that:

> The prepositional status of *worth* ... is confirmed by the fact that it can govern a noun phrase, a nominal *-ing* clause with a genitive subject, and a nominal relative clause (but not a *that*-clause or a *to*-infinitive

but later (p. 1230) it is listed as a canonical example in the section on "Adjective complementation by an *-ing* participle clause" with an additional note on it being unclear whether to regard *worthwhile* as an adjective or as *worth* followed by a noun (which is reflected in inconsistent spelling). At any rate, they seem to beg the question of the existence of transitive adjectives, by declaring anything with NP complements to be a preposition. Huddleston and Pullum [34] reject the criterion of taking an NP complement as being decisive and come out strongly in favor of *worth* as a transitive adjective, unlike similar words like *like*, *unlike* and *due* which they suggest belong to both the adjective and preposition categories.

Contra Maling, there is some evidence that *worth* is still more like an adjective than a preposition, but it seems fairly clear that it has mixed properties that make it partly like adjectives and partly like prepositions, but not like a canonical member of either category. Even Huddleston and Pullum admit that *worth* "differs markedly from central members of the adjective category". That is, it is a case of syntactic gradience resulting from historical changes [27]. In such cases, it is artificial to demand a categorical classification, whatever its convenience for current part-of-speech tagging technology.

One pragmatic solution in such cases might just be to accept that certain high frequency words may have odd properties and we should just give them tags by convention, however imperfect their assignment to a category. There are probably few applications of NLP which will be much affected by the choice of an

Table 8. Selected citations of *worth* in the Penn Treebank WSJ corpus

1	Northeast says its bid is (ADJP-PRD	(IN worth)	(NP *e*)).
2	Each share point is (ADJP-PRD	(IN worth)	(NP about $60 million) in sales)
3	grain elevators are (PP-PRD	(IN worth)	(S-NOM *e* preserving for aesthetic . . .))
4	should be (PP-PRD	(IN worth)	(NP 30 a share))
5	assets are (ADJP-PRD	(IN worth)	(NP-ADV more to private buyers than . . .)
6	a good number decide it's not (ADJP-PRD	(JJ worth)	(NP it))
7	and decide it's (ADJP-PRD	(JJ worth)	(NP the astronomical price) to add it
8	It was (PP-PRD	(JJ worth)	it), just for the look on . . .
9	the company . . . is (PP-PRD	(JJ worth)	(NP $70 a share)) if broken up
10	are in need of (NP billions of dollars	(JJ worth)	of repair)
11	is one of the (JJS earliest) (NN high-net)	(JJ worth)	(NNS banks) (PP in the U.S.)
12	Not even . . . makes this trip (ADJP-PRD	(NN worth)	(S taking))
13	What is UAL stock (ADJP-PRD	(NN worth)	(NP *e*))
14	an additional $200 to 300 million	(NN worth)	per month
15	could pile up $150	(NN worth)	of quarters on a slanted coin
16	The company's net	(NN worth)	cannot fall below $185 million
17	thus dilute the	(NN worth)	and voting power of ASKO
18	will sell (NP (ADJP (QP $25 million))	(NN worth)	(PP of his clothes))

adjective or preposition tag for *worth*. If anything, applications are mainly likely to gain from the treatment being consistent. But in such cases, we must accept that we are assigning parts of speech by convention for engineering convenience rather than achieving taxonomic truth, and there are still very interesting issues for linguistics to continue to investigate, along the lines of [27].

Acknowledgments

Thanks to all the people who have worked on the Stanford Part-of-Speech Tagger over the years: foremost, Kristina Toutanova, but also Dan Klein, Yoram Singer, William Morgan, Anna Rafferty, Michel Galley, and John Bauer. And thanks to Alexander Gelbukh for the CICLing 2011 conference invitation, which prompted me to write this material up.

References

1. Toutanova, K., Klein, D., Manning, C.D., Singer, Y.: Feature-rich part-of-speech tagging with a cyclic dependency network. In: NAACL 3, pp. 252–259 (2003)
2. Shen, L., Satta, G., Joshi, A.: Guided learning for bidirectional sequence classification. In: ACL 2007 (2007)
3. Spoustová, D.j., Hajič, J., Raab, J., Spousta, M.: Semi-supervised training for the averaged perceptron POS tagger. In: Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009), pp. 763–771 (2009)
4. Søgaard, A.: Simple semi-supervised training of part-of-speech taggers. In: Proceedings of the ACL 2010 Conference Short Papers, pp. 205–208 (2010)
5. Subramanya, A., Petrov, S., Pereira, F.: Efficient graph-based semi-supervised learning of structured tagging models. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pp. 167–176 (2010)
6. Collins, M.: Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In: EMNLP 2002 (2002)
7. Marcus, M.P., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of English: The Penn treebank. Computational Linguistics 19, 313–330 (1993)

8. Finkel, J., Dingare, S., Manning, C., Nissim, M., Alex, B., Grover, C.: Exploring the boundaries: Gene and protein identification in biomedical text. BMC Bioinformatics 6 (suppl. 1) (2005)
9. Collins, M.: Ranking algorithms for named entity extraction: Boosting and the voted perceptron. In: ACL 40, pp. 489–496 (2002)
10. Tsuruoka, Y., Tsujii, J.: Bidirectional inference with the easiest-first strategy for tagging sequence data. In: Proceedings of HLT/EMNLP 2005, pp. 467–474 (2005)
11. Clark, A.: Combining distributional and morphological information for part of speech induction. In: EACL 2003, pp. 59–66 (2003)
12. Klein, D., Manning, C.D.: Accurate unlexicalized parsing. In: ACL 41, pp. 423–430 (2003)
13. MacKinlay, A.: The effects of part-of-speech tagsets on tagger performance. Honours thesis, Department of Computer Science and Software Engineering, University of Melbourne (2005)
14. Church, K.W.: Current practice in part of speech tagging and suggestions for the future. In: Mackie, A.W., McAuley, T.K., Simmons, C. (eds.) For Henry Kučera: Studies in Slavic Philology and Computational Linguistics. Papers in Slavic philology, vol. 6, pp. 13–48. Michigan Slavic Studies, Ann Arbor (1992)
15. Magerman, D.M.: Natural language parsing as statistical pattern recognition. PhD thesis, Stanford University (1994)
16. Ratnaparkhi, A.: A maximum entropy model for part-of-speech tagging. In: EMNLP 1, pp. 133–142 (1996)
17. Abney, S., Schapire, R.E., Singer, Y.: Boosting applied to tagging and PP attachment. In: Fung, P., Zhou, J. (eds.) Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, pp. 38–45 (1999)
18. Voutilainen, A., Järvinen, T.: Specifying a shallow grammatical representation for parsing purposes. In: 7th Conference of the European Chapter of the Association for Computational Linguistics, pp. 210–214 (1995)
19. Samuelsson, C., Voutilainen, A.: Comparing a linguistic and a stochastic tagger. In: Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics, pp. 246–253 (1997)
20. Dickinson, M., Meurers, W.D.: Detecting errors in part-of-speech annotation. In: Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2003 (2003)
21. Levy, R., Andrew, G.: Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In: 5th International Conference on Language Resources and Evaluation, LREC 2006 (2006)
22. Rohde, D.L.T.: Tgrep2 user manual. MS. MIT, Cambridge (2005)
23. Santorini, B.: Part-of-speech tagging guidelines for the Penn treebank project. 3rd Revision, 2nd printing, February 1995. University of Pennsylvania (1990)
24. Moore, D.S.: Statistics: Concepts and Controversies, 3rd edn. W. H. Freeman, New York (1991)
25. Ross, J.R.: A fake NP squish. In: Bailey, C.J.N., Shuy, R.W. (eds.) New Ways of Analyzing Variation in English, pp. 96–140. Georgetown University Press, Washington (1973)
26. Quirk, R., Greenbaum, S., Leech, G., Svartvik, J.: A Comprehensive Grammar of the English Language. Longman, London (1985)
27. Aarts, B.: Syntactic gradience: the nature of grammatical indeterminacy. Oxford University Press, Oxford (2007)

28. Abney, S.: Statistical methods and linguistics. In: Klavans, J., Resnik, P. (eds.) The Balancing Act. MIT Press, Cambridge (1996)
29. Maling, J.: Transitive adjectives: A case of categorial reanalysis. In: Heny, F., Richards, B. (eds.) Linguistic Categories: Auxiliaries and Related Puzzles, vol. 1, pp. 253–289. D. Reidel, Dordrecht (1983)
30. Harnad, S. (ed.): Categorical perception: the groundwork of cognition. Cambridge University Press, Cambridge (1987)
31. Radford, A.: Transformational Grammar. Cambridge University Press, Cambridge (1988)
32. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press, Boston (1999)
33. Bies, A., Ferguson, M., Katz, K., MacIntyre, R. (colleagues): Bracketing guidelines for Treebank II style: Penn treebank project. MS, University of Pennsylvania (1995)
34. Huddleston, R.D., Pullum, G.K.: The Cambridge Grammar of the English Language. Cambridge University Press, Cambridge (2002)

Ripple Down Rules for Part-of-Speech Tagging

Dat Quoc Nguyen[1], Dai Quoc Nguyen[1],
Son Bao Pham[1,2], and Dang Duc Pham[1]

[1] Human Machine Interaction Laboratory,
Faculty of Information Technology,
University of Engineering and Technology,
Vietnam National University, Hanoi
{datnq,dainq,sonpb,dangpd}@vnu.edu.vn
[2] Information Technology Institute,
Vietnam National University, Hanoi

Abstract. This paper presents a new approach to learn a rule based system for the task of part of speech tagging. Our approach is based on an incremental knowledge acquisition methodology where rules are stored in an exception-structure and new rules are only added to correct errors of existing rules; thus allowing systematic control of interaction between rules. Experimental results of our approach on English show that we achieve in the best accuracy published to date: 97.095% on the Penn Treebank corpus. We also obtain the best performance for Vietnamese VietTreeBank corpus.

1 Introduction

Part-of-speech (POS) tagging is one of the most important tasks in Natural Language Processing, which assigns a tag representing its lexical category to each word in a text. After the text is tagged or annotated, it can be used in many applications such as: machine translation, information retrieval etc. A number of approaches for this task have been proposed that achieved state-of-the-art results including: Hidden Markov Model-based approaches [1], Maximum Entropy Model-based approaches [2] [3] [4], Support Vector Machine algorithm-based approaches [5], Perceptron learning algorithms [2][6]. All of these approaches are complex statistics-based approaches while the obtained results are progressing to the limit. The combination utilizing the advantages of simple rule-based systems [7] can surpass the limit. However, it is difficult to control the interaction among a large number of rules.

Brill [7] proposed a method to automatically learn transformation rules for the POS tagging problem. In Brill's learning, the selected rule with the highest score is learned on the context that is generated by all preceding rules. In additions, there are interactions between rules with only front-back order, which means an applied back rule will change the results of all the front rules in the whole text. Hepple [8] presented an approach with two assumptions for disabling interactions between rules to reduce the training time while sacrificing a small fall of accuracy.

Ngai and Florian [9] presented a method to impressively reduce the training time by recalculating the score of transformation rules while keeping the accuracy.

In this paper, we propose a failure-driven approach to automatically restructure transformation rules in the form of a Single Classification Ripple Down Rules (SCRDR) tree [10][11][12]. Our approach allows interactions between rules but a rule only changes the results of selected previous rules in a controlled context. All rules are structured in a SCRDR tree, which allows a new exception rule to be added when the system returns an incorrect classification. Moreover, our system can be easily combined with existing part of speech tagger to obtain an even better result. For Vietnamese, we obtained the highest accuracy at present time on VietTreebank corpus [13]. In addition, our approach obtains promising results in term of the training time in comparison with Brill's learning.

The rest of paper is organized as follows: in section 2, we provide some related works including Brill's learning, SCRDR tree, among others and describe our approach in section 3. We describe our experiments in section 4 and discussion in section 5. The conclusion and future works will be presented in section 6.

2 Related Works

2.1 Transformation-Based Learning

The well-known transformation-based error-driven learning method had been introduced by Brill [7] for POS tagging problem and this method has been used in many natural language processing tasks, for example: text chunking, parsing, named entity recognition. The key idea of the method is to compare the golden-corpus that was correctly tagged and the current-corpus created through an initial tagger, and then automatically generate rules to correct errors based on predefined templates. For example, corresponding with a template *"transfer tag of current word from **A** to **B** if the next word is **W**"* is some rules like as: *"transfer tag of current word from **JJ** to **NN** if the next word is **of**"* or *"transfer tag of current word from **VBD** to **VBN** if the next word is **by**"...*

Transformation-based learning algorithm runs in multiple iterations as follows:

- **Input:** Raw-corpus that contains the entire raw text without tags extracted from the golden-corpus that contains manually tagged *word/tag* pairs.
- **Step 1:** Annotated-corpus is generated using an initial tagger where its input is the raw-corpus.
- **Step 2:** Comparing the annotated-corpus and the golden-corpus to determine tag errors in the annotated-corpus. From these errors, all templates are used for creating potential rules.
- **Step 3:** Each rule will be applied to a copy of annotated-corpus. The score of a rule is computed by subtracting of number of additional errors from number of correctly changed tags. The rule with the best score is selected.
- **Step 4:** Update the annotated-corpus by applying selected rule.

– **Step 5:** Stop if the best score is smaller than a predefined threshold T, else repeat step 2.
– **Output:** Front-back ordered list of transformation rules.

The training process of Brill's tagger includes two phases:

– The first-phase is used to assign the most likely tag for unknown words. Initially, the most likely tag for unknown words starting with a capital letter is **NNP** and otherwise it is **NN**. In this phase, the lexical transformation rules are used to predict the most likely tag for unknown words. The transformation templates in this phase depend on character(s), prefix, suffix of a word and only the preceding/following word. For example, *"change the most likely tag of an unknown-word to **Y** if the word has suffix **x**, $|x| <= 4$"*, *"change the most likely tag of an unknown-word to **Y** if the last (1, 2, 3, 4) characters of the word are **x**"* or *"change the most likely tag of an unknown-word to **Y** if the word **x** ever appears immediately to the left/right of the word"*
– The second phase uses transformation-based error-driven learning for producing contextual transformation rules. Each word is assigned a tag by the initial tagger: known-words were annotated with the highest frequency tag using the lexicon extracted from corpus that was used for learning lexical transformation rules, and unknown-words were assigned with default tags **NNP** or **NN** and subsequently the ordered lexical transformation rules were applied.

To tag raw texts, the known-words are assigned by the highest frequency tag using lexicon extracted from the training corpus and unknown-words are assigned with default tags **NNP** or **NN** and then the ordered lexical transformation rules are applied to these unknown-words. Finally, the ordered contextual transformation rules will be applied to all words. In the tagging process, a word can be tagged multiple times. At each the iteration during the training phase, all possible rules will be generated and each rule's score is computed based on the entire corpus. Therefore, training phase in Brill's learning takes a significant amount of time.

The transformation-based learning of Brill allows interactions between learnt rules. A new rule can change the result of any previous rules.

Hepple [8] presented a method to impressively improve about 950 times [9] at the training time while there was a small fall in the precision by using two assumptions: independence and commitment, which disables any interaction between learned rules. The commitment assumption assumes that a tag was changed at most once by a rule in the whole training period. And the independence assumption imposes that if a rule changes a tag, it will not change the context relevant to the firing of a future rule. Ngai and Florian [9] proposed an approach called as Fast TBL to significantly reduce about 340 times at the training time on corpus of about 1 million words while achieved the same accuracy as a standard transformation-based tagger. The central idea of this approach is to save the number of corrected-tags and the number of additional errors for each rule for recalculation when applying a newly selected rule to the current corpus.

Another drawback of Brill's learning is that it is not able to estimate probabilities of class memberships. An approach presented in [14] shows how to convert transformation-based rules list to decision trees for resolving this problem.

2.2 Single Classification Ripple Down Rules

Ripple Down Rules (RDR) [10][15][12] were developed to allow users incrementally add rules to an existing rule-based system whiles systematically controlling interactions between rules and ensuring consistency among existing rules.

Suppose the system's classification produced by some rule ***R*** is deemed incorrect by the expert. As the justification for the decision that the classification is incorrect, the expert creates a new rule ***Re*** which acts as an *exception* to the rule ***R***. The justification would refer to attributes of the case, such as patient data in the medical domain, or a linguistic pattern matching the case in the natural language domain[15].

The new rule ***Re*** will only be applied to cases for which the provided conditions in ***Re*** are true and for which rule ***R*** would produce the classification, if rule ***Re*** had not been entered. In other words, in order for ***Re*** to be applied to a case as an exception rule to ***R***, rule ***R*** has to be satisfied as well. A sequence of nested exception rules, of any depth, may occur. Whenever a new exception rule is added, a difference to the previous rule has to be identified by the expert. This is a natural activity for the expert when justifying his/her decision to colleagues or apprentices. The case which triggered the addition of an exception rule is stored along with the new rule. This case, called the *cornerstone case* of the rule ***R***, is retrieved when an exception to ***R*** needs to be entered. The cornerstone case is intended to assist the expert in coming up with a justification, since a valid justification must point at differences between the cornerstone case and the case at hand for which ***R*** does not perform satisfactorily. A number of RDR-based systems also store with every rule all cases for which the rule has given a correct conclusion. These systems effectively store all *seen cases.* This enables the consistency test to be checked against not only the cornerstone cases but all previously seen cases.

A SCRDR tree [10][12] is a finite binary tree with two distinct types of edges. These edges are typically called *except* and *if not* (or *false*) edges as shown in figure 1. Associated with each node in a tree is a rule. A rule has the form: *if* α *then* β where α is called the *condition* and β the *conclusion.*

An SCRDR tree is evaluated for a case by passing the case to the root of the tree. At any node in the tree, if the condition of a node N's rule is satisfied by the case, the case is passed on to the *except child* of N if it exists. Otherwise, the case is passed on to N's *if not child* if it exists. The conclusion given by this process is the conclusion from the last node in the SCRDR tree which *fired.* To ensure that a conclusion is always given, the root node typically contains a trivial condition which is always satisfied. This node is called the *default node.*

A new rule is added to an SCRDR tree when the evaluation process returns a wrong conclusion using the *fired rule* ***R***. A new node containing the new rule is attached to the last node evaluated in the tree provided the new rule is consistent

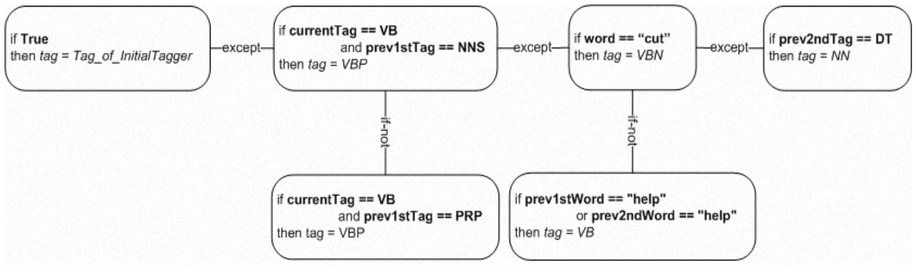


Fig. 1. A part of SCRDR tree for POS tagging

with the existing knowledge base. This is done by making sure cases that have been previously classified correctly by the rule ***R*** do not match the new rule. If the node has no exception link, the new node is attached using an *exception link*, otherwise an *if not link* is used.

2.3 Vietnamese POS Tagging Problems

Dinh and Hoang [16] proposed an approach for Vietnamese POS tagging problem that gave accuracy of 87% by building an English-Vietnamese bilingual corpus which contains approximately 5 million words. They tagged the English corpus using transformation-based learning of Brill [7] and convert POS-annotation tags from English side to Vietnamese using existing word-alignment tools.

Three available tools using machine learning methods: Conditional Random Fields [17], Maximum Entropy Model, Support Vector Machine were used to combine with morpheme-based approach in [18] for Vietnamese POS tagging, that achieved the highest averaged accuracy using the 5-fold cross-validation of 91.64% on Vietnamese Treebank corpus [13].

3 Our Approach

In this section, we describe a transformation-based failure-driven approach to automatically build a single classification Ripple Down Rule (SCRDR) tree for POS tagging problem. Figure 2 describes the learning model used in our approach.

The *Raw corpus* is annotated by using an *Initial tagger* to create the *Annotated corpus*. By comparing the annotated corpus with the *Golden corpus*, an *Object-driven dictionary* is generated based on the *Object Template* which captures the context containing the current word and its tag, (1^{st}, 2^{nd}, 3^{rd}) previous and next words and (1^{st}, 2^{nd}, 3^{rd}) previous and next tags in following format (*previous* 3^{rd} *word, previous* 3^{rd} *tag, previous* 2^{nd} *word, previous* 2^{nd} *tag, previous* 1^{st} *word, previous* 1^{st} *tag, word, currentTag, next* 1^{st} *word, next* 1^{st} *tag, next* 2^{nd} *word, next* 2^{nd} *tag, next* 3^{rd} *word, next* 3^{rd} *tag*) in the annotated corpus.

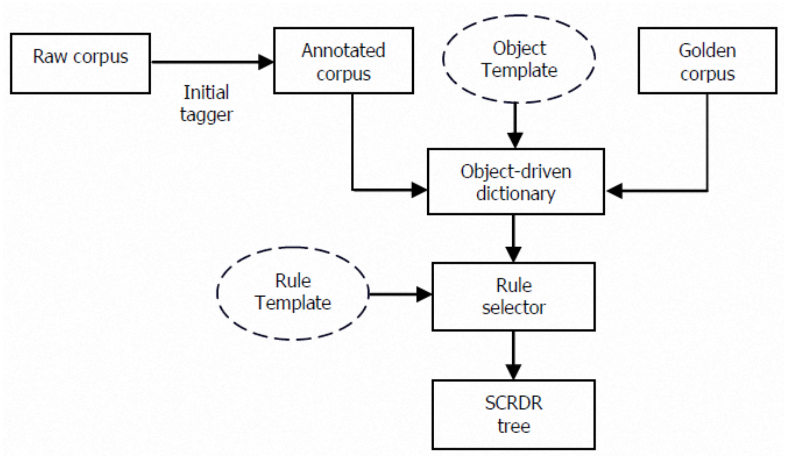


Fig. 2. The diagram describing our approach

An object-driven dictionary is a set of the format (***Object***, *correct Tag*) in which *Object* captures the context of the current word in the annotated corpus and *correct Tag* is the corresponding tag in the golden corpus.

Rule1: *if* {word == **"object.word"**} *then* tag = **"correctTag"**
Rule2: *if* {next1stTag == **"object.next1stTag"**} *then* tag = **"correctTag"**
Rule3: *if* {prev1stTag == **"object.prev1stTag"**} *then* tag = **"correctTag"**

Fig. 3. Some rule examples

From the object template, rule templates are created based on the templates of Brill's tagger for *Rule selector*. Examples of rule templates are shown in figure 3 where elements in bold will be replaced by concrete values for creating concrete rules.

Training algorithm:

- **Step 1:** Load raw corpus and assign initial tags using an initial tagger.
- **Step 2:** Create Object-driven dictionary by comparing output of the initial tagger and the golden corpus.
- **Step 3:** Build the default node representing the initial tagger.
- **Step 4:** At a node-FR in SCRDR tree, let SE be the set of elements from the object-driven dictionary that fired at the node-FR but theirs tags are incorrect i.e. node-FR gives wrong conclusions for elements in SE.

  To select a new exception rule, a list of all concrete rules is generated based on rule-templates from all elements in SE and unsatisfied cornerstone case of node-FR. The rule with the highest value by subtracting B from A would be selected where A is the number of elements in SE that is correctly modified by the rule and B is the number of elements in SE that is incorrectly changed by the rule.

The newly selected rule is added to the SCRDR tree where the cornerstone case is the case in SE that is correctly modified by the selected rule.

This step process is repeated until the score for the selected rule is under a given threshold. At each iteration, a new exception rule is added to correct an error made by the existing rule-based system.

To illustrate how the new exception rules are added, lets consider the following rule (a node in the SCRDR tree)

if currentTag == **"vb"** and prev1stTag == **"nns"** *then* tag = **"vbp"**

cc: *('the', 'dt', 'latest', 'jjs', 'results',* ***'nns'****, 'appear',* ***'vb'****, 'in', 'in', 'today', 'nn', ''s', 'pos')*

Suppose we have a case that this rule fires but returns a wrong conclusion i.e. incorrect tag. The following rule can be added as an exception rule of the rule in the SCRDR tree with the cornerstone case (cc) being the case that was misclassified originally:

if word == **"cut"** *then* tag = **"vbn"**

cc: *('keeping', 'vbg', 'their', 'prp$', 'people',* ***'nns'****,* ***'cut'****,* ***'vb'****, 'off', 'rp', 'from', 'in', 'the', 'dt')*

To take a further example, suppose we have a new case that the above newly added rule fires but the conclusion is incorrect. The following exception is added to correct the mistake:

if prev2ndTag == **"dt"** *then* tag = **"nn"**

cc: *('to', 'to', 'the',* ***'dt'****, 'capital-gains',* ***'nns'****,* ***'cut'****,* ***'vb'****, ',', ',', 'which', 'wdt', 'has', 'vbz')*

Tagging process:

- Raw texts are tagged by initial tagger to create the annotated texts.
- Make objects to capture context surrounding current *word/tag* in annotated texts.
- Each object is classified by SCRDR tree for generating output tag.

In our method, we use two thresholds: one for finding rules for nodes at the depth of 1 and the other is used for nodes at higher levels. One reason is that the default node has no cornerstone case.

4 Experiment

We apply our approach to both English and Vietnamese part of speech tagging tasks.

4.1 Results for English

Following [2] [3] [4] [5] [6], we split the Penn Wall Street Journal Treebank [19] into training, development and test sets as shown in table 1 for our experiments for English. We retrained Brill's tagger on training data at default threshold of 2 resulting in 1595 rules with the time cost for learning contextual transformation rules of 2700 minutes.

Table 1. Data Set

DataSet	Sections	Sentences	Tokens
Training	0-18	38,219	912,344
Develop	19-21	5,527	131,768
Test	22-24	5,462	129,654

For our method, RDR tree was built on whole training data using Brill's retrained initial tagger as the initial tagger that achieved the baseline accuracies of 93.67% and 93.58% on development data and test data respectively.

Brill's retrained tagger achieved an accuracy of 96.57% on development data while the result of our taggers on development data is shown in table 2. The accuracy is comparable while our method improves up to 33 times in training time.

Table 2. Pos tagging in accuracy of development data of our approach

Threshold	Number of rules	Accuracy (%)	Training time (minutes)
(50, 20)	133	95.76	14
(10, 10)	393	96.21	30
(5, 5)	830	96.42	48
(3, 2)	**2517**	**96.55**	**82**
(1, 1)	18310	96.35	512

Table 3 shows the performance for our method using the best threshold and Brill's tagger on test data.

Table 3. Pos tagging accuracy on test data

Method	Accuracy (%)
Brill	96.530
Our approach	96.548

Table 4 shows the accuracy of our method depending on the depth of the RDR tree.

Table 4. Accuracy and speed tagging in our method on test data on PenIV 2.66GHZ of CPU, 1G of RAM

Depth	Number of rules	Accuracy (%)	Speed tagging (number of words second)
<= 1	1433	96.372	161
<= 2	2467	96.540	160
<= 3	2517	96.548	160

Table 5. Accuracy of our method with different initial taggers on test data

Initial Tagger (IT)	Accuracy of IT (%)	Accuracy of IT and RDR tree(%)	Number of rules in RDR tree
Brill's tagger	96.53	96.68	322
Tagger of Tsuruoka and Tsujii	96.987	97.095	130

Table 5 shows the returned results when we used Brill's retrained tagger and tagger of Tsuruoka and Tsujii [4] that was trained on same WSJ 0-18 training data at default parameters as initial taggers for building an RDR tree in our approach. It can be seen that our approach can be used to improve performance of existing approaches by adding more exception rules.

4.2 Results for Vietnamese

We ran experiments for Vietnamese on the same corpus as in [18] on Vietnamese Treebank corpus [13]. This corpus contains approximately 10000 sentences with a tag set of 17 labels. We randomly divide the corpus into five folds; giving one fold size of around 44±1K words. Each time, four folds were merged as the training set and the remaining fold is selected as the test set. Final result is the averaged results of five runs using the best threshold found for the English experiment.

Table 6 shows the result of our approach, which we use an open dictionary assigning the most frequent tag in whole training set for a word as the initial tagger. For this open dictionary assumption, when a word (in test set) not in the dictionary, it would be tagged as Np if the first character is an upper letter or N if otherwise by default. With the open dictionary assumption, the accuracy of the initial tagger is 90.38%.

Table 6. Five-fold cross validation accuracy of our method in open dictionary assumption

Method	Final accuracy (%)
Open dict	92.24

From table 6, it can be seen that our method achieves a higher accuracy than accuracy of 91.64% of the method described in [18].

We also trained our method using the closed dictionary assumption where the initial tagger assigns a word with the most frequent tag that extracted from whole training and test sets. In addition, we retrained Brill's tagger for Vietnamese using the closed dictionary assumption without the process of learning lexical transformation rules. Both of methods obtained an accuracy of 92.81% at the initial state.

Table 7 shows the results for our approach and Brill's tagger in closed dictionary assumption.

Table 7. Five-fold cross validation accuracy in closed dictionary assumption

Method	Final accuracy (%)
Brill's tagger	94.72
Our method	94.61

It can be seen that our approach is comparable to that of Brill's in this corpus. Due to the small size of this corpus, our approach can utilize experts to add rules instead of learning rule from the corpus.

5 Discussion

For Brill's approach and Hepple's approach, a new rule is selected based on the accuracy of context, so the output of initial tagger is always changed when the new rule applies. In our approach, objects always are static so that new rules are selected based on the original state of the output of the initial tagger. Therefore our approach can be easily combined with existing state of the art taggers to improve their performance as demonstrated when we improve the existing result of Tsuruoka and Tsujii [4].

Another important point is that our approach is very suitable to use experts to add new exception rules given a concrete case at hand that is misclassified by the system. This is especially important for under-resourced languages where obtaining a large annotated corpus is difficult.

6 Conclusion

In this paper, we propose a failure-driven approach to automatically restructure transformation rules in the form of a Single Classification Ripple Down Rules tree. Our approach allows controlled interactions between rules where a rule only changes the results of a limited number of other rules. On the Penn Treebank, our approach achieves the best performance published to date of 97.095%. For Vietnamese, our approach achieves an accuracy of 92.24% for open-dictionary assumption and an accuracy of 94.61% for close-dictionary assumption. This is also the best result to date to the best of our knowledge.

In the future we will involve experts to manually add more exception rules to the current rule based system to even improve the performance of the system further.

Another avenue is to use Ngai and Florian's method [9] to improve the training time and extend the lexical transformation rule learning of Brill for Vietnamese.

Acknowledgment

This work is partly supported by the research project No. QG.10.39 granted by Vietnam National University, Hanoi and IBM Faculty Award 2009 for the third author.

References

1. Brants, T.: Tnt – a statistical part-of-speech tagger. In: Proc. ANLP, pp. 224–231 (2000)
2. Collins, M.: Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In: Proc. EMNLP, pp. 1–8 (2002)
3. Toutanova, K., Klein, D., Manning, C.D., Singer, Y.: Feature-rich part-of-speech tagging with a cyclic dependency network. In: Proc. NAACL-HLT, pp. 173–180 (2003)
4. Tsuruoka, Y., Tsujii, J.: Bidirectional inference with the easiest-first strategy for tagging sequence data. In: Proc. HLT-EMNLP, pp. 467–474 (2005)
5. Giménez, J., Màrquez, L.: Svmtool: A general pos tagger generator based on support vector machines. In: Proc. LREC, pp. 43–46 (2004)
6. Shen, L., Satta, G., Joshi, A.: Guided learning for bidirectional sequence classification. In: Proc. ACL, pp. 760–767 (2007)
7. Brill, E.: Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. Computational Linguistics 21(4), 543–565 (1995)
8. Hepple, M.: Independence and commitment: assumptions for rapid training and execution of rule-based pos taggers. In: Proc. ACL, pp. 278–277 (2000)
9. Ngai, G., Florian, R.: Transformation-based learning in the fast lane. In: Proc. NAACL, pp. 1–8 (2001)
10. Compton, P., Jansen, R.: Knowledge in context: a strategy for expert system maintenance. In: Proc. AI 1988, pp. 292–306 (1988)
11. Compton, P., Jansen, R.: A philosophical basis for knowledge acquisition. Knowl. Acquis. 2(3), 241–257 (1990)
12. Richards, D.: Two decades of ripple down rules research. Knowl. Eng. Rev. 24(2), 159–184 (2009)
13. Nguyen, P.T., Vu, X.L., Nguyen, T.M.H., Nguyen, V.H., Le, H.P.: Building a large syntactically-annotated corpus of vietnamese. In: Proc. LAW, pp. 182–185 (2009)
14. Florian, R., Henderson, J.C., Ngai, G.: Coaxing confidences from an old friend: probabilistic classifications from transformation rule lists. In: Proc. EMNLP, pp. 26–34 (2000)
15. Pham, S.B., Hoffmann, A.: Efficient knowledge acquisition for extracting temporal relations. In: Proc. ECAI, pp. 521–525 (2006)

16. Dien, D., Kiem, H.: Pos-tagger for english-vietnamese bilingual corpus. In: Proc. HLT-NAACL WBT, pp. 88–95 (2003)
17. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proc. ICML, pp. 282–289 (2001)
18. Tran, O.T., Le, C.A., Ha, T.Q., Le, Q.H.: An experimental study on vietnamese pos tagging. In: Proc. IALP, pp. 23–27 (2009)
19. Marcus, M.P., Marcinkiewicz, M.A., Santorini, B.: Building a large annotated corpus of english: the penn treebank. Computational Linguistics 19(2), 313–330 (1993)

An Efficient Part-of-Speech Tagger for Arabic[⋆]

Selçuk Köprü

Teknoloji Yazılımevi, Ltd.
METU Technopolis
06531, Ankara, TR
selcuk.kopru@tyazilimevi.com

Abstract. In this paper, we present an efficient part-of-speech (POS) tagger for Arabic which is based on a Hidden Markow Model. We explore different enhancements to improve the baseline system. Despite the morphological complexity of Arabic our approach is a data driven approach and does not utilize any morphological analyzer or a lexicon as many other Arabic POS taggers. This makes our approach simple, very efficient and valuable to be used in real-life applications and the obtained accuracy results are still comparable to other Arabic POS taggers. In the experiments, we also thoroughly investigate different aspects of Arabic POS tagging including tag sets, prefix and suffix analyses which were not examined in detail before. Our part-of-speech tagger achieves an accuracy of 95.57% on a standard tagset for Arabic. A detailed error analysis is provided for a better evaluation of the system. We also applied the same approach on different languages like Farsi and German to show the language independent aspect of the approach. Accuracy rates on these languages are also provided.

1 Introduction

The continous increase in the demand for processing Arabic faces many challenges. One important challenge at this point is the requirement to tag part-of-speech (POS) information in a given Arabic text with high accuracy and efficiently. Part-of-speech tagging is the task of classifying the words in a sentence into a set of classes. Output of POS taggers has a widespread usage in many Natural Language Processing (NLP) applications, such as Machine Translation (MT) or Information Retrieval (IR). The efficiency and accuracy of the POS tagger has usually a reasonable impact on the overall quality of the entire system. There are many studies on Statistical Machine Translation (SMT) systems that report gain in evaluation scores when a POS tagger is used [14,18,17,6].

POS tagging is the task of labeling words in an input sentence with part-of-speech and additional linguistic information. The words in the input sentence must be tokenized before the tagging process. There are two main approaches to POS tagging: rule-based tagging and stochastic tagging. Rule based taggers

[⋆] This work was supported by Applications Technology, Inc. (Apptek).

process a set of rules that resolve the POS of a word. Stochastic taggers disambiguate the POS by calculating probabilities based on a large corpus [11]. There are also some combined approaches like Transformation Based Learning (TBL) which include characteristics from both rule-based and stochastic approaches.

In this paper, we investigate the different aspects of Arabic POS tagging and present a Hidden Markov Model (HMM) based method. The proposed POS tagging system achieves an accuracy of 95.6% using a restricted tagset of 17 tags. The tagger is language independent and has been used successfully for different languages such as English and Persian.

The rest of this paper is organized as follows. In the next section we give a brief overview of related work. In Section 3, the approach and the system that is implemented in this study is explained. Next, we talk about the training procedure and data in particular. In Section 5 the system is evaluated with detailed experiments and error analysis. Finally, we conclude the paper with Section 7.

2 Related Work

The huge demand for Arabic POS taggers in different NLP systems causes researchers to focus on this problem. In the recent years, different studies have emerged but they are not enough and the problem is not studied as sufficiently as in other languages such as English.

[8] present an approach to use a morphological analyzer for tokenizing and morphological tagging. First, they obtain all possible morphological analyses from the morphological analyzer. Next, they use a Support Vector Matrix (SVM) based classifier to for the POS and other morphological features. The work by [5] and [4] also use an SVM based approach to automatically tokenize, POS tag and annotate base phrases in Arabic text.

[16] proposes a tagger of Semitic languages that treats both Arabic and Hewbrew. They also use a morphological analyzer in their HMM based POS tagger. The work by [9] is similar in the sense that it combines a morphological analyzer in a HMM model. In [12], a hybrid appraoch to Arabic POS tagging is presented which makes use of a tag set including 131 tags. In [1], the application of TBL to Arabic POS tagging is inspected.

Our approach is different from all the above studies in the sense that it is purely stochastic, corpus-driven, HMM based and without utilizing a morphological analyzer during the tagging process.

3 The Approach

POS tagging is considered as a sequence classification task and Hidden Markov Model can be applied to different language related classification problems successfully. Our work is inspired by systems which employ the HMM approach to POS tagging in different languages such as [2]. Given an observation of a sequence of words, the aim is to find the best sequence of POS tags that correspond to the observed words. The Bayesian explanation for this problem is formulated in the following well-known equation:

$$\hat{t}_1^n = \arg\max_{t_1^n} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1}) \tag{1}$$

In Equation (1) the sequence of the words in the observed sentence are represented as $w_1 \dots w_n$. In order to calculate the estimate of the tag sequence $\hat{t}_1^n$, tag transition probabilities and word likelihoods are used. The tag transition probability $P(t_i|t_{i-1})$ is estimated with a bigram Language Model (LM) that is constructed from a tag transition corpus. The word likelihood $P(w_i|t_i)$ is computed from counts in the training data. The HMM is represented as a weighted finite state machine (FSM) with hidden states. In POS tagging, the HMM states correspond to the tags and the output symbols represent the words. Viterbi algorithm is used to determine the most likely tag sequence.

We use the IRSTLM toolkit [7] to estimate, store and access the language models required in the tagger. The toolkit supports the Witten-Bell smoothing [19] and the Kneser-Ney smoothing [13] methods. In order to overcome the sparse data problem in high order n-grams, we use interpolation in combination with the smoothing methods.

$$\begin{aligned}\hat{P}(t_n|t_{n-1}, t_{n-2}) = {} & \lambda_1 P(t_n|t_{n-1}, t_{n-2}) + \\ & \lambda_2 P(t_n|t_{n-1}) + \\ & \lambda_3 P(t_n)\end{aligned} \tag{2}$$

Equation (2) shows how different n-grams are joined to interpolate second order probability values. The weights in the interpolation formula, i.e. the λ values, sum to 1 and they are calculated by deleted interpolation according to the frequencies in the training data as described in [2]. Normalization of the λ values ensures that the result is still a probability distribution.

The word likelihood probability $P(w_i|t_i)$ is computed according to Equation (3) which makes use of the frequencies in the training corpus. $C(t_i, w_i)$ is the count of the times the word w_i is tagged as t_i in the training corpus.

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)} \tag{3}$$

Data sparsity problem is again a big problem while computing the word likelihood as it was in the tag transition probabilities. Whenever a word in the observation was not seen in the training corpus, the above formula will evaluate to zero. If the tag is an open-class tag, then the likelihood is estimated using prefix and suffix probabilities as in Equation (7) where p_i^j denotes the w_i prefix including the first j characters and s_i^k denotes the w_i suffix including the last k

characters. Moreover, $\hat{P}_p(p_i^j|t_i)$ is the estimation of the probability that the prefix p_i^j can occur if the tag t_i is observed. Similarly, $\hat{P}_s(s_i^k|t_i)$ is the estimation of the probability that the suffix s_i^k can occur if the tag t_i is observed. The terms prefix and suffix do not have any linguistic meaning, they are only character strings. While calculating the prefix and suffix probabilities, the longest prefix and suffix is used that is matched in the training set. Once a match is found no other shorter strings are searched further.

$$\hat{P}_p(p_i^j|t_i) = \frac{C(t_i, p_i^j)}{C(p_i^j)^2} \tag{4}$$

$$\hat{P}_s(s_i^k|t_i) = \frac{C(t_i, s_i^k)}{C(s_i^k)^2} \tag{5}$$

$$\hat{P}(w_i|t_i) = \alpha \hat{P}_p(p_i^j|t_i) + (1-\alpha)\hat{P}_s(s_i^k|t_i) \tag{6}$$

$$\hat{P}(w_i|t_i) = \alpha \frac{C(t_i, p_i^j)}{C(p_i^j)^2} + (1-\alpha)\frac{C(t_i, s_i^k)}{C(s_i^k)^2} \tag{7}$$

Empirically, the α weight is found to be 3/5 indicating that the prefix probility has slightly more predictive power over the suffix probability in Arabic. Setting the weight according to the length of the matched prefix and suffix did not yield better results. In English, prefix probability does not have any predictive power because of its morphological structure and the α value is set to zero.

We use the algorithm listed in Figure 1 to estimate the word likelihood. If the observed word is matched in the training corpus then the `word-likelihood` function simply returns the result in Equation (3). If the word is not matched and if the tag is a closed-class tag, then the function returns zero. Otherwise, prefix and suffix analysis is performed on the word. The algorithm ensures that a non-zero probability value is returned for open-class tags. The functions `prefix-prob` and `suffix-prob` always return a non-zero value because at least one character of the word matches as prefix and suffix.

The Viterbi algorithm is used to decode the most desirable sequence of tags. The algorithm takes as input the language models and the sequence of observed words and outputs the optimal tag sequence. Both first-order and second-order HMMs are constructed. The first-order model utilizes bigram language models and the second-order model utilizes trigram language models.

4 Training

4.1 Data

The Penn Arabic Treebank [15] parts 1, 2 and 3 (ATB) is used for training and testing purposes. The entire corpus contains 629,866 words in 1,835 news stories from four distinct sources. In ATB, Buckwalter's morphological analyzer [3] is used to generate a candidate list of POS values for each word/token and the

```
word-likelihood(word , tag)
  if count(tag, word) > 0 then
    return count(tag, word) / count(tag)
  else
    if tag is open-class then
      return $\alpha \times$ prefix_prob(word, tag)
        $+ (1-\alpha) \times$ suffix_prob(word, tag)
    else
      return 0
    end if
  end if

suffix-prob(word , tag)
  for i=2 to length(word)
    suffix = substr(word, 2, i)
    if count(suffix, word) > 0 then
      return count(tag,suffix)/count(tag)$^2$
    end if
  end for

prefix-prob(word , tag)
  for i=length(word)-1 down to 1
    prefix = substr(word, 1, i)
    if count(prefix, word) > 0 then
      return count(tag,prefix)/count(tag)$^2$
    end if
  end for
```

Fig. 1. Algorithm to calculate the word likelihood for a word

appropriate POS value is selected by human annotators. Detailed information about the size of the corpus parts is given in Table 1. We have used 95% of the ATB corpus for training and the remaining 5% for testing.

4.2 Arabic Tag Sets

The annotation level in ATB can be described as morphological analysis tags rather than simple POS tags. The annotation labels contain more than the linguistic category of the word, e.g. determiner, gender, person, number, definitness information and more. Morphological richness in the languages increases the nuumber of tags in the tagset. There are in total 1731 different labels which are formed by combining various morphological information. [10] gives similar numbers for Slovene and Czech. Some frequent labels from the ATB corpus are listed below:

DET+ADJ+NSUFF_FEM_SG+CASE_DEF_GEN
CONJ+PV+PVSUFF_SUBJ:3MS
ADJ+CASE_INDEF_GEN

Table 1. Size of the ATB corpus

Part	Genre	Stories	Words
1	News	734	145,386
2	News	501	144,199
3	News	600	340,281
Total		1,835	629,866

Different tasks require different tag sets and the selection of the tags in the tag set affects the performance rate of the tagger. In order to compare with previously published results, similar tag sets should be used. In [8], a reduced tag set consisting of 15 tags is used: V (verb), N (noun), PN (proper noun), AJ (adjective), AV (adverb), PRO (pronoun), P (preposition/participle), D (determiner), C (conjunction), NEG (negative particle), NUM (number), AB (abbreviation), IJ (interjection), PX (punctuation) and X (unknown). The frequency counts of the tags in our training corpus is given in Table 2.

Table 2. Frequency counts of the tags in the reduced tag set

N	209366	PN	51526	AB	2904
P	74402	PRO	27009	D	261
PX	71414	C	17352	IJ	196
AJ	67268	NUM	13415	NEG	40
V	55635	AV	4111	X	31

In [5], a tag set similar to the English Penn Treebank tag set is used for Arabic and it contains 24 different tags: CC, CD, CONJ+NEG, PART, DT, FW, IN, JJ, NN, NNP, NNPS, NNS, NO_FUNC, NUMERIC, COMMA, PRP, PRP$, PUNC, RB, UH, VBD, VBN, VBP, WP, WRB. In [4], an extended tag set comprising 75 tags is used which aim to cover the morphological richness of Arabic including marking for case, number, gender, definiteness, mood, person, voice, tense and others.

In our tagger, the tag set is not hard coded and the tags are defined externally which makes the tagger independent of the tag set in use. The tag set definition includes a list of the tags in use and other information like whether they are open class tags or closed class tags. The words and tokens that belong to a closed class tag are fixed and out-of-vocabulary (OOV) words are assigned only to open class tags during tagging.

4.3 Language Model Estimation

First step in LM estimation is the collection of frequency counts and n-grams. The estimation for the tag transition proability in Equation (1) is calculated with a tag transition model. A tag sequence file is extracted from the ATB corpus in the following format for each sentence:

$tag_1\ tag_2\ tag_3\ \cdots$

This tag sequence file is used to create a tag frequency file and a tag transition n-gram using the IRSTLM toolkit. Discounting and interpolation methods are used in combination in order to smooth the probability values as described in Section 3. Next, a "word to tag" mapping file is extracted from the ATB corpus in the following format:

$word_1/tag_1 \; word_2/tag_2 \; word_3/tag_3 \; \cdots$

This word to tag mapping file is used to create a frequency file which contains the counts of how many times a word was tagged with a specific tag. This frequency file is used to estimate the word likelihood probability in Equation (1). For OOV words, prefix frequency files, suffix frequency files, prefix to tag frequency files and suffix to tag frequency files are created for differet prefix and suffix lengths. These frequency files are created using the following sequence files that are extracted from the ATB corpus.

$prefix_1 \; prefix_2 \; prefix_3 \; \cdots$
$suffix_1 \; suffix_2 \; suffix_3 \; \cdots$
$prefix_1/tag_1 \; prefix_2/tag_2 \; prefix_3/tag_3 \; \cdots$
$suffix_1/tag_1 \; suffix_2/tag_2 \; suffix_3/tag_3 \; \cdots$

5 Evaluation

5.1 Experiments

We have two different goals in the experiments. First goal is to analyze the effect of different aspects in Arabic POS tagging. The second goal in the experiments is to evaluate and compare our POS tagger against other published results. The test file, which is 2%of the ATB corpus, contains 984 sentences and over 12K tokens of which 5,597 are unique. The OOV rate in the test file is 5.4%. The accuracy results are given in percent correct rates.

The unigram baseline system produces 84.3% accuracy and with bigram language model the accuracy jumps to 93.69%. In IRSTLM, tag transition probabilities are smoothed by default with Witten-Bell discounting. If the discounting method is changed to improved Kneser-Ney discounting, then an accuracy of 93.67% is achieved. Applying deleted interpolation in combination with the discounting approaches in the tag transition probabilities increases the accuracy to 94.13%. Suffix analysis without prefix analysis adds 1.22% accuracy and prefix analysis without suffix analysis adds 1.17% accuracy. If suffix analysis and prefix analysis are applied together then the accuracy increase becomes 1.44%. We did not achieve any gain by switching from a bigram language model to a trigram language model. The accuracy results are summarized in Table 3.

Table 3. Accuracy results of the model with different features

Features	Accuracy
Unigram LM	84.3%
Bigram LM + Witten-Bell Smoothing	93.69%
Bigram LM + Kneser-Ney Smoothing	93.67%
+ Deleted Interpolation	94.13%
+ Suffix Analysis	95.35%
+ Prefix Analysis	95.57%
Trigram LM	95.50%

In the next setup, we have experimented on the prefix and suffix length. Table 4 lists the accuracy results for different lengths from 1 to 10. It came out that prefix/suffix length longer than 9 did not have any effect on the results.

Table 4. Accuracy results of the model with different prefix and suffix lengths

Prefix/Suffix Length	Accuracy	Gain
1	94.25%	0.12%
2	94.55%	0.30%
3	94.99%	0.44%
4	95.22%	0.23%
5	95.31%	0.09%
6	95.46%	0.15%
7	95.49%	0.03%
8	95.54%	0.05%
9	95.57%	0.03%
10	95.57%	0%

[8] report an accuracy score of 98.1% on the tag set with 15 tags compared to our 95.57%. We believe that the difference results from the utilization of the morphological analyzer.

5.2 Error Analysis

In this section we present the details of the types of errors that occured in the experiments. Table 5 lists the number of times a POS category is predicted erronously. Nouns (N) are tagged 175 times incorrectly which corresponds to 31.47% of the errors. However, because of the high frequency of N in the test file, the percent error of the class is only 3.97%. Thus, we can conclude that the tagger performs well on nouns. The highest percent error rate belongs to proper nouns (PN) and it is 9.43% (if we exclude interjection POS - IJ because of its very low frequency). The high error rate in proper nouns is explainable because of the high OOV rate. The second highest error rate is on adjectives (AJ) with 8.49% and the third highest error rate is on verbs (V) with 7.97%.

Table 5. Error counts and error rates based on word classes

POS	Error		Reference	
	Count	Percent	Count	Percent
N	175	31.47%	4,403	3.97%
AJ	122	21.94%	1,437	8.49%
PN	95	17.09%	1,007	9.43%
V	94	16.91%	1,180	7.97%
P	39	7.01%	1,599	2.44%
PRO	15	2.70%	557	2.69%
NUM	7	1.26%	335	2.09%
AV	5	0.90%	77	6.49%
C	3	0.54%	372	0.81%
IJ	1	0.18%	3	33.33%
Total	556	100%	12,501	NA

Table 6 lists the number of times a specific tagging error has occurred. The most common error is the AJ - N mismatch. In 108 instances, an adjective in the reference file is tagged as a noun which corresponds to 19.42% of all errors. The second most common error is the V - N mismatch with 84 instances and 15.11%.

Table 6. Error types and their frequencies

Reference	Hypothesis	Count	Percent
AJ	N	108	19.42%
V	N	84	15.11%
N	AJ	78	14.03%
PN	N	71	12.77%
N	PN	38	6.83%
N	V	25	4.50%
N	P	23	4.14%
P	N	18	3.24%
PN	AJ	13	2.34%
P	PRO	9	1.62%
P	C	9	1.62%
AJ	PN	9	1.62%
PRO	P	7	1.26%
PRO	N	6	1.08%
other low freq. errors		58	10.43%
Total		556	100%

Table 7 lists the wrong hypotheses and their frequencies.

Table 7. Wrong hypotheses and their frequencies

Wrong Hypotheses	Count	Percent
N	292	52.52%
AJ	96	17.27%
PN	52	9.35%
P	42	7.55%
V	32	5.76%
PRO	19	3.42%
C	11	1.98%
NUM	7	1.26%
D	2	0.36%
AB	2	0.36%
AV	1	0.18%
Total	556	100%

6 Experiments on Farsi and German

In order to see the language independent aspect of the approach we have conducted experiments on Farsi and German. The results are summarized in Table 8. The Farsi tagger is trained on data of 2.6 M words and it achieved an accuracy rate of 95.54%. The German tagger is trained on a data of 1 M words and it achieved an accuracy rate of 95.80%. It is very interesting that the accuracy for all languages are very close to each other.

Table 8. POS accuracy rates for Farsi and German

	Train Data		Test Data		Accuracy
	Words	Sentences	Words	Sentences	
Farsi	2.6 M	90 K	26 K	900	95.54 %
German	1 M	50 K	9.3 K	510	95.80 %

7 Conclusions

In this study we present an efficient HMM based POS tagger for Arabic that is comparable to other state-of-the-art Arabic taggers. The tagger is efficient so that it can be used in real-life applications. For example, it is integrated into an SMT system and it is utilized for reordering purposes.

Different aspects of Arabic POS tagging including tag sets, prefix and suffix analyses are investigated and verified with the experiments. We have obtained comparable results to other state-of-the-art Arabic POS tagging systems. We also present a detailed error analysis of the system.

The system is also trained on languages like Farsi and German which show different morphological and syntactic properties compared to Arabic. Results obtained for the other languages are similar as in Arabic.

As a future work, we aim to improve the accuracy in the pos tagger more. We also want to lay out the effect of POS tagging in the performance of Arabic SMT.

References

1. AlGahtani, S., Black, W., McNaught, J.: Arabic part-of-speech tagging using transformation-based learning. In: Proceedings of the Second International Conference on Arabic Language Resources and Tools. The MEDAR Consortium, Cairo (2009)
2. Brants, T.: Tnt – a statistical part-of-speech tagger. In: Proceedings of the Sixth Conference on Applied Natural Language Processing, pp. 224–231. Association for Computational Linguistics, Seattle (2000)
3. Buckwalter, T.: Buckwalter arabic morphological analyzer, version 2.0 (2004)
4. Diab, M.: Improved arabic base phrase chunking with a new enriched pos tag set. In: Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources, pp. 89–96. Association for Computational Linguistics, Prague (2007)
5. Diab, M., Hacioglu, K., Jurafsky, D.: Automatic tagging of arabic text: From raw text to base phrase chunks. In: Susan Dumais, D.M., Roukos, S. (eds.) HLT-NAACL 2004: Short Papers, pp. 149–152. Association for Computational Linguistics, Boston (2004)
6. Elming, J., Habash, N.: Syntactic reordering for English-Arabic phrase-based machine translation. In: Proceedings of the EACL 2009 Workshop on Computational Approaches to Semitic Languages, pp. 69–77. Association for Computational Linguistics, Athens (2009), `http://www.aclweb.org/anthology/W09-0809`
7. Federico, M., Bertoldi, N., Cettolo, M.: IRSTLM: an open source toolkit for handling large scale language models. In: Proceedings of Interspeech 2008, Brisbane, Australia (September 2008)
8. Habash, N., Rambow, O.: Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005), pp. 573–580. Association for Computational Linguistics, Ann Arbor (2005)
9. Hadj, Y.O.M.E., Al-Sughayeir, I.A., Al-Ansari, A.M.: Arabic part-of-speech tagging using the sentence structure. In: Proceedings of the Second International Conference on Arabic Language Resources and Tools. The MEDAR Consortium, Cairo (2009)
10. Hajic, J.: Morphological tagging: Data vs. dictionaries. In: Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics, pp. 94–101. Association for Computational Linguistics, Seattle (2000)
11. Jurafsky, D., Martin, J.H.: Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics. Prentice-Hall, Englewood Cliffs (2009)
12. Khoja, S.: Apt: Arabic part-of-speech tagger. In: Proceedings of the Student Workshop at the Second Meeting of the North American Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, Pittsburgh (2001)

13. Kneser, R., Ney, H.: Improved backing-off for m-gram language modeling. In: ICASSP 1995: International Conference on Acoustics, Speech, and Signal Processing, vol. 1, pp. 181–184 (May 1995)
14. Koehn, P., Hoang, H.: Factored translation models. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pp. 868–876. Association for Computational Linguistics, Prague (2007)
15. Maamouri, M., Bies, A., Buckwalter, T., Mekki, W.: The penn arabic treebank: Building a large-scale annotated arabic corpus. In: Proceedings of the NEMLAR Conference on Arabic Language Resources and Tools, Cairo, Egypt, pp. 102–109 (2004)
16. Mansour, S., Sima'an, K., Winter, Y.: Smoothing a lexicon-based pos tagger for arabic and hebrew. In: Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources, pp. 97–103. Association for Computational Linguistics, Prague (2007)
17. Niehues, J., Kolss, M.: A POS-based model for long-range reorderings in SMT. In: Proceedings of the Fourth Workshop on Statistical Machine Translation, pp. 206–214. Association for Computational Linguistics, Athens (2009), `http://www.aclweb.org/anthology/W09-0435`
18. Rottman, K., Vogel, S.: Word reordering in statistical machine translation with a POS-based distortion model. In: Proceedings of the 11th International Conference on Theoretical and Methodological Issues in Machine Translation, pp. 171–180. University of Skovde, Skovde (2007)
19. Witten, I.H., Bell, T.C.: The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression. IEEE Transactions on Information Theory 37(4), 1085–1094 (1991)

An Evaluation of Part of Speech Tagging on Written Second Language Spanish

M. Pilar Valverde Ibañez

Aichi Prefectural University
Department of Spanish and Latin American Studies
Kumabari, Nagakute-cho, Aichi, 480-1198, Japan
valverde@for.aichi-pu.ac.jp

Abstract. With the increase in the number and size of computer learner corpora in the field of Second Language Acquisition, there is a growing need to automatically analyze the language produced by learners. However, the computational tools developed for natural language processing are generally not considered as appropiate because they are designed to treat native language. This paper analyzes the reliability of two part-of-speech taggers on second language Spanish and investigates the most frequent tagger errors and the impact of learner errors in the performance of the taggers.

1 Introduction

An increasing number of learner corpora are being compiled in the field of Second Language Acquisition. Such corpora, defined as electronic collections of spoken or written texts produced by foreign or second language learners [1], require linguistic annotation that enables the study of learner language in a systematic way. However, most of the work on annotating learner texts has traditionally focused on the (generally manual) annotation of errors [2], and little attention has been given to the purely linguistic annotation, irrespectively of errors. With the increase in the number and size of corpora, such annotation is becoming even more necessary to study not only misuse of words but also other aspects of learner language such as the under- and overuse of words and structures.

For these reasons, it is necessary to study how state-of-the-art natural language processing tools such as part-of-speech (PoS) taggers can be (re)used in the learner domain [3, 4]. The aim of this paper is to evaluate the performance of two PoS taggers on second language Spanish and investigate what are the most frequent tagger errors and how they are related to learner errors. For the evaluation, we have tagged a sample of 5,000 words of learner language with two Spanish PoS taggers. First, we have manually revised the PoS tags assigned by them and second, we have manually annotated the sample with learner error information. Finally, we have extracted quantitative information about the taggers and the learners' performance.

The paper is organized as follows. Section 2 deals with the type of texts and taggers used for the evaluation and makes a distinction between two types of information encoded in the manual revision: tagger errors and learner errors. Section 3 deals with the results of the evaluation: the precision of the taggers, the analysis of the taggers' and learner errors, and the impact of learner errors on PoS precision. Finally, section 4 summarizes the conclusions.

2 Evaluation

2.1 The Sample File

We have extracted the texts for the evaluation from the CORANE corpus [5], which contains 1,091 Spanish essays written by 321 learners with different proficiency levels and different native languages.

Our study deals with texts written by Japanese learners with an intermediate level of Spanish.[1] The CORANE corpus contains essays from 19 Japanese learners with a B1 level and 9 learners with a B2 level, and we randomly selected one essay from each learner, which results in a text sample made up by 28 essays written by 28 different learners, totallying approximately 5,000 words (excluding punctuation).[2]

2.2 The PoS Taggers

The sample was tagged with two Spanish PoS taggers: FreeLing [6] and HISPAL [7].[3] Tested on native language, FreeLing offers a state-of-the-art accuracy of about 97%, and HISPAL, of 99%.

With regard to the strategies used by the taggers, FreeLing provides two algorithms for PoS tagging with a similar performance: an HMM trigram model and a relaxation labelling model (the latter enables the use of hand-written rules together with the statistical models). For the analysis of our sample, the former model was used. FreeLing Spanish dictionary was built by hand, and contains over 81,000 forms corresponding to more than 7,100 different combinations of lemma and PoS.

HISPAL is a rule-based parser with a modular architecture. However, unlike other rule-based systems, the morphological analysis is carried out by means

[1] The levels B1 and B2 of the Common European Framework of Reference for Languages.

[2] The topics of the essays are: My first impression of Spain (1 essay); My holidays (3); Wishes for the year 2001 (3); My life in Spain (2); Write about a person you know (2); Dialogue (1); Recipe (3); What I would do if I were invisible (3); November 1st (1); A Japanese Wedding (1); A Party (3); My opinion about a piece of news (3); Big Brother (1); Free time (1).

[3] We did not test the performance of the Spanish TreeTagger [8] because the information it provides is not easily comparable to the other taggers. TreeTagger's tagset uses broad categories with little morphological information: it lacks gender and number information for nouns and adjectives, and mode, tense and person information for verbs.

of a lexicon-free morphological analyzer. Specifically, HISPAL "uses a full form lexicon only for about 220 closed class tokens, while everything else is treated analytically through affix classes with or without stem conditions. [...] Which of these alternatives are real Spanish words will be decided by lexicon look-up." The lexicon was created with a bootstrapping method, from corpora, and finally a large list of lexeme strings was manually checked.

With regard to the backfall strategies and heuristics they use to treat unknown words or contexts (typical of learner texts), in FreeLing unknown words "are handled via conditional probabilities of PoS tags given word suffixes, so that the most suitable PoS tags are proposed for each word not included in the morphological dictionary." In HISPAL, if one or more of the suggested readings for a word have a lexicon-support (extracted from corpora) for their roots, other readings are discarded.

With regard to the tagset used by the taggers, both assign not only word class but also morphological information like gender, number, person, case, tense and mode. Every tagger uses a slightly different tagset, and we evaluated the accuracy of the taggers with respect to their own tagset.

Table 1. Morphological information for inflecting word classes in FreeLing and HISPAL

Word class	Gender	Number	Person	Case	Tense	Mode
Nouns (Common)	●	●				
Verbs (Finite)			●		●	●
Verbs (Participle)	●	●				
Adjectives	●	●				
Pronouns (Determiners)	●	●				
Pronouns (Personal)	●	●	●	●		
Pronouns (Independent)	●	●				

The size of the tagsets, which may be related to the taggers' accuracy, is difficult to quantify. However, each tagger encodes similar information, and during the revision of the sample we found that the information provided by each tagger is easily comparable. Both provide morphological information for inflecting word classes (nouns, verbs, adjectives, determiners and pronouns). Specifically, both specify the gender (masculine or feminine), number (singular or plural), person (first, second or third), case (nominative, accusative, dative or genitive), tense (present, imperfect, future, past or conditional) and mode (indicative, subjunctive, imperative, infinitive, gerund or participle), as shown in table 1. Adverbs, conjunctions, prepositions, interjections and numerals are non-inflecting word classes and then only receive a word class tag.

The main difference between the taggers' tagsets are the following. On the one hand, FreeLing provides information about the type of adjective, adverb, determiner and pronoun in the PoS tag,[4] while HISPAL encodes this information as secondary lexicon information which is not the target of this study.[5] On the other hand, HISPAL provides information about the number and gender of proper nouns while FreeLing does not, and classifies the traditional word classes Determiner and Pronoun into three classes: Determiner (defined as inflecting pronouns, that can be used as prenominals), Personal (defined as person-inflecting pronouns) and Independent pronouns (defined as non-inflecting pronouns, that can not be used as prenominals).

2.3 Tagger Errors vs. Learner Errors

First, we annotated the fragment of 5,000 words with each tagger, and merged the two tagged files into one file, with one word per row, where the columns contain the form, the lemma and the PoS tag assigned by each tagger. Second, we manually revised the file to look for:

(i) Tagger (PoS) errors : when the PoS tag assigned by the parser is wrong we have added the correct tag (according to each tagger's guidelines), based on the stem, the morphology and the distribution of the word.
(ii) Learner errors: when the word used by the learner is wrong from the native point of view we have added information about the type of error (mispelling, wrong ending, wrong distribution, etc.).[6]

While (i) informs us about the accuracy of the parser, (ii) informs us about the accuracy of the learner. It is important to note the different meaning that the word "wrong" or "error" has in each case. In (i), "wrong" means that the PoS is not appropiate for the word used by the learner, wether the word is "correct" or not from the native point of view. For example, in (1) the tagger assigns a PoS of proper noun to the word *mi* 'my', but in fact this word is a possessive pronoun (the tagger probably assigned the proper noun tag incorrectly because the word appears at the beginning of the sentence). In this case, the human annotator adds the correct PoS tag to the word.

(1) Mi idea ha cambiado mucho.
My idea has changed a lot.

[4] Adjectives can be qualifying or ordinal. Adverbs can be general or negative. Determiners can be demonstrative, possesive, interrogative, exclamative, indefinite or article. Pronouns can be personal, demonstrative, possessive, indefinite, interrogative, relative or exclamative.

[5] The secondary tags in HISPAL contain additional information about the word: not only word class "type" but also valency and semantic prototype information. Due to the amount of information, and to keep the tagsets as comparable as possible, we decided to take into account only the primary PoS tag.

[6] Our error classification is an adaptation of [9]. It should also be kept in mind that error annotation always contains an element of subjectivity, as the very notion of error is far from clear-cut [10].

However, in (2) *muchísimos* (masculine plural determiner, 'a lot of') is correctly analyzed as a masculine plural determiner by the tagger, even though in this context (followed by the feminine noun *habitaciones*, 'rooms'), there is a gender agreement error and the determiner should be feminine (*muchísimas*) from a native point of view.

(2) Este palacio es muy grande y tiene *muchísimos* habitaciones [...].
This palace is very big and has a lot of rooms

On the contrary, in (ii), "wrong" means that, from the native point of view, there is an error in the lexical stem, the morphology or the distribution of the word. For example, in (2), there is a learner error in the word *muchísimos*, which should be feminine instead of masculine, so the human annotator adds a tag that encodes the type of error (in this case, it is a Distribution-Morphology mismatch error).

This two-layered model can be also explained in terms of "form" (i.e. the learner language 'as is') and "function" (an interpretation of the intended meaning of the learner). The goal of the PoS tagger is to analyze the learner language as is. Given this type of PoS annotation, one can build another module that analyzes the intended meaning or construction, that is, learner error annotation (like the gender mismatch in (2)), or other type of linguistic annotation.

While PoS errors and learner errors are related and some times coincide, in approximately half of the tokens in our sample PoS errors do not correspond to learner errors, and learner errors don't result in PoS errors, as one may expect (section 3.4).

3 Results

3.1 Precision of the Taggers

After manually revising the text sample, we calculated the precision of the two taggers, which is considerably high: FreeLing has a precision of 92.6% and HISPAL, of 95.0%.[7]Therefore, the taggers experiment a decrease of between 4.4 and 4 points with respect to their documented accuracy on native texts (which is 97% and 99%, respectively).

However, to establish a more fair comparison between the taggers' precision on learners texts with respect to native texts, we corrected our learner texts changing them into something a native speaker would say, but trying to keep the text as close to the original as possible. We then tagged and corrected the

[7] We calculated the precision of the taggers by diving the number of correct PoS tags by the number of tokens in the sample excluding punctuation (5,101 tokens). Even though during the analysis the taggers merge and split some words differently, the number of tagged tokens is the same in the two samples because we aligned the two files. This means that the merged tokens are counted as correct as many times as the number of words they contain, in the same way as the split tokens. When both taggers merge the words in the same way, they are counted only once.

texts and the precision of the taggers was 96.1% for Freeling and 98.1% for HISPAL (exactly 0.9 points lower than the documented precision), which allows us to say that the taggers show a slightly lower performance when they deal with learner texts: in our sample, their precision decreases between 3.1 and 3.5 points with respect to the near native version of the same texts.

This means that, in our sample, both taggers succeeded to assign a correct PoS tag to more than half of the tokens containing a learner error.[8] As table 2 shows, with FreeLing 64.9% of the tokens with learner errors are correctly assigned a PoS tag, and with HISPAL, 69.4%. These figures are considerably high taking into consideration that both taggers assign not only a word class but also morphological information to every token.

Table 2. PoS precision with respect to learner errors

		PoS error	No PoS error	Total
FreeLing	Learner error	35.1	64.9	100.0
	No learner error	5.4	94.6	100.0
	Total	7.4	92.6	100.0
HISPAL	Learner error	30.6	69.4	100.0
	No learner error	3.1	96.9	100.0
	Total	5.0	95.0	100.0

The slightly different performance of the two taggers can be related to the different strategies employed to deal with faulty input (section 2.2) but also to the different treatment of inherently ambiguous morphological forms.

In Spanish, several verb forms are systematically ambiguous. Specifically, the first and third person singular of several verbal tenses have the same form (the indicative and subjunctive imperfect tense, the conditional tense and the subjunctive present tense). The particle *se* is also a highly ambiguous form (it can be an impersonal mark, part of a pronominal pronoun or a pronoun).

While FreeLing tries to disambiguate those systematically ambiguous forms, HISPAL assigns an underspecified tag to them (additional information is included in secondary tags). This may be one of the reasons why the FreeLing tagger has a slightly lower accuracy in our sample compared with HISPAL. Indeed, the majority of the errors that affect verbal morphology and the particle *se* in FreeLing are not related to learner language, and actually do not co-occur with learner errors. If we substracted those cases from FreeLing errors, the accuracy of the tagger would raise from 92.6% to 94.3%, a figure that is very close to the 95.0% achieved by HISPAL.

[8] However, when the tagger manages to assign a correct PoS tag to an unknown word, the lemma assigned is generally the same word form (generally an inflected form, instead of the dictionary form), which may be problematic for later processing that requires information about the lemma of the words.

3.2 Analysis of the Taggers' Errors

With regard to the type of PoS information that is assigned wrongly in our learner texts, the most frequent error is word class confusion, which gives account of 92.3% of the tagger errors in HISPAL and 73.9% in FreeLing. The errors in the morphological features, that is, in-class errors, concern mainly the verb, since this is the word class with a higher inflecting potential (table 1).

The precision and recall of the different PoS tags for each word class is shown in table 3.[9] The tag with the lowest precision is the Proper Noun tag (58.7%), in the FreeLing sample (apart from the Interjection tag in HISPAL, which appears only once). In addition to that, the Adjective and Independent Pronoun tags have a low precision in both taggers. As for the two most frequent tags (Common Noun and Verb), interestingly, the Verb tag has a low precision in FreeLing, while the Common Noun tag has a low precision in HISPAL. This seems to indicate that the distinction between these two word classes is problematic and taggers are inclined towards one of the two readings in favour of the other, as we will see in table 4. With regard to recall, in FreeLing the Common Noun, Verb and Adverb PoS tags have a significant lower recall. HISPAL follows the same tendency except for the fact that the Adjective, instead of the Adverb, is the tag with the lowest recall.

Table 3. Recall and precision of PoS tags (word class)

	FreeLing					HISPAL				
Word class	TP+FP	TP	FN	Recall	Precision	TP+FP	TP	FN	Recall	Precision
Common Noun	964	921	86	91.5	95.5	1116	957	69	93.3	85.75
Verb	1041	923	118	88.7	88.7	1005	956	61	94.0	95.12
Pronoun-Det.	663	654	12	98.2	98.6	745	729	16	97.9	97.85
Preposition	627	621	12	98.1	99.0	621	619	10	98.4	99.68
Adverb	325	317	43	88.1	97.5	401	392	13	96.8	97.76
Adjective	275	235	14	94.4	85.5	292	246	31	88.8	84.25
Coordinating Conj.	233	233	2	99.1	100.0	234	233	0	100.0	99.57
Pronoun-Personal	263	246	15	94.3	93.5	227	217	12	94.8	95.59
Subordinating Conj.	185	180	20	90.0	97.3	160	151	18	89.3	94.38
Proper Noun	218	128	6	95.5	58.7	135	114	9	92.7	84.44
Pronoun-Indep.	110	91	14	86.7	82.7	82	69	7	90.8	84.15
Numeral	65	64	0	100.0	98.5	82	81	0	100.0	98.78
Interjection	7	5	5	50.0	71.4	1	0	9	0.0	0

[9] We calculated the precision of every tag by diving the number of times a given tag was assigned correctly (TP) by the number of times the tag was assigned (TP+FP). Accuracy was calculated by dividing the number of times a given tag was assigned correctly (TP) by the same number plus the number of times the tag should have been assigned but it was not (FN).

On the other hand, the tags that contribute the most to the overall error rate in FreeLing are verbs (33.0%) and proper nouns (25.1%), while in HISPAL are common nouns (46.4%), verbs (14.3%) and adjectives (13.4%).[10]

Table 4 shows the five most common error pairs in each sample. For FreeLing, the most frequent error concerns the morphological features of the verb. While this pair is also considerably frequent in the HISPAL sample, in FreeLing this is mostly due to the design of the tagger (that tries to disambiguate inherently ambiguous forms) and not to learner errors (that only explain 23.2% of the cases). The same happens with the ambiguous Spanish particle *se*.

Table 4. The five most frequent PoS error pairs (word class) and their co-ocurrence with learner errors

	False PoS	True PoS	Freq	%	L error	% L error
FreeLing	Verb	Verb	82	21.6	19	23.2
	Proper noun	Common noun	40	10.5	9	22.5
	Adjective	Common noun	22	5.8	4	18.2
	Se (impersonal)	Se (verbal)	21	5.5	0	0
	Common noun	Verb	19	5	8	42.1
HISPAL	Verb	Common noun	24	9.2	8	33.3
	Common Noun	Verb	22	8.4	19	86.4
	Adjective	Common noun	19	7.3	11	57.9
	Common noun	Adjective	19	7.3	6	31.6
	Verb	Verb	13	5.0	8	61.5

The second most frequent PoS error in the FreeLing sample consists on the confusion between proper and common nouns. This agrees with the fact that the Proper Noun tag is the one with the lowest precision in FreeLing (table 3). In both samples the confusion between nouns and verbs and nouns and adjectives, in both directions, is very frequent.

With regard to the relationship between tagger and learner errors (section 3.4), we can see in the last column of table 4 that FreeLing's errors can be attributed mainly to the tagger's performance itself, while in HISPAL the tagger's errors and learner errors are more strongly related.

3.3 Analysis of Learners' Errors

As expected, the accuracy of the taggers decreases when analyzing learner language. In our sample, FreeLing error rate is 7.4% and HISPAL error rate is 5%. To measure the effect that learner language has on the accuracy of the taggers, we manually tagged our sample with learner error annotation.

[10] We calculated each tag's contribution to the error of the tagger by dividing the number of times a given tag was assigned incorrectly by the total number of tag errors.

Our sample contains 359 learner errors at the word level, that is, 7% of words are wrong from a native point of view.[11] As shown in table 5, the most frequent error type is distribution-morphology mismatches,[12] followed by misspellings, accents, spare words, errors related to punctuation marks, uppercase or lowercase letters, stem-distribution mismatches[13] and wrong word choice.

Table 5. Frequency of learner errors

Error	Freq.	%
Distribution-Morphology mismatch	113	31.5
Misspelling	79	22.0
Accent	37	10.3
Spare word	37	10.3
Punctuation mark or upper/lowercase	35	9.7
Stem-Distribution mismatch	18	5.0
Word choice	13	3.6

The occurence of learner errors is specially frequent in some word classes, as table 6 summarizes. In our sample, one out of every ten verbs, adjectives and personal pronouns contain some kind of learner error. Verbs and personal pronouns are probably the most complex word classes from the morphological point of view (they are the most inflecting word classes, as seen in table 1), and as a result they are the most difficult and error-prone for learners. In the case of the verb and the adjective, the most common error is the mismatch between distribution and morphology and for the personal pronouns it is the spare use of such pronouns. Proper nouns are also a common source of errors, due to the use of uppercase or lowercase letters and punctuation signs in an unconventional way.

3.4 The Impact of Learner Language on PoS Precision

The relationship between tagger errors and learner errors is shown in table 7, where we can see that 33.2% of the PoS errors in the FreeLing sample co-occur with learner errors, while in HISPAL this percentage raises to 43.0%. Therefore, 2.5 and 2.2 points of the error rate in FreeLing and HISPAL, respectively, can be attributed to learner errors directly. Even though we cannot measure it in our

[11] We did not annotate errors consisting of a missing word, wrong word order or grammatically correct but semantically unnatural constructions. The annotation of these phenomena would give a higher learner error rate.

[12] Defined as the cases where "the lexical word class specification for the stem accords with its distribution and morphology, but the inflectional morphology does not match the distribution, i.e. the grammatical context" [9]. This is the case of agreement errors in gender and number, for example.

[13] Defined as the cases where "a lexeme of a given word class appears in a distributional slot which is not available to instances of that word class".

Table 6. Word classes and frequency of learner errors

Word class	(TP+FN)	Learner error	%
Common Noun	1007	57	5.7
Verb	1041	105	10.1
Pronoun-Determiner	666	48	7.2
Preposition	633	21	3.3
Adverb	360	18	5.0
Adjective	249	27	10.8
Coordinating Conjunction	235	0	0.0
Pronoun-Personal	261	28	10.7
Subordinating Conjunction	200	10	5.0
Proper Noun	134	16	11.9
Pronoun-Independent	105	5	4.8
Numeral	64	1	1.6
Interjection	10	2	20.0

sample, learner errors may also affect the performance of the parser indirectly, when a learner error causes a tagger error that in turn causes another tagger error. If for example a given word in the text has to be disambiguated by means of a contextual rule that includes a neighbouring wrong tag, the rule is likely to assign another wrong tag to the word.

Table 7. Frequency of PoS errors and learner errors

		PoS error		No PoS error		Total	
		Freq.	%	Freq	%	Freq.	%
FreeLing	Learner error	126	33.2	233	4.9	359	7.0
	No learner error	254	66.8	4488	95.1	4742	93.0
	Total	380	100	4721	100	5101	100
HISPAL	Learner error	110	43.0	249	5.1	359	7.0
	No learner error	146	57.0	4596	94.9	4742	93.0
	Total	256	100	4845	100	5101	100

Some learner errors co-occur more frequently than others with taggers' errors. As for the word classes affected, if we go back to table 4, we can see in the last column that some PoS errors co-occur very often with learner errors, while others have a low co-occurency. Interestingly, in both taggers the pairs Common Noun-Verb and Verb-Common Noun (that is, the confusion between verbs and common nouns) are the pairs that co-occurs with learner errors more frequently (in 42.1% and 86.4% of cases in FreeLing and HISPAL, respectively). This may indicate that the distinction between verbs and nouns are specially vulnerable to learner errors. The next pair that co-occurs more often with learner errors is the Verb-Verb pair, that is, the errors in the morphological features of the verb. The different verb forms are also vulnerable to errors. For example, for the tagger the difference between subjunctive present and indicative past is only the absence or presence of an accent in the verb ending. Then, the accentuation of the verb is critical to achieve a correct morphological interpretation.

As for the different type of learner errors considered, as table 8 shows, in our sample, the errors that have a bigger impact on the performance of the tagger are the less "linguistic" ones: those related to accents, the misuse of punctuation marks or uppercase or lowercase letters, and misspellings. For the first two, the taggers have a precision of around 30% (HISPAL has a considerably higher precision (60%) when dealing with the latter type of errors). For misspelling, both taggers have a precision of around 50%.

Table 8. Most frequent learner errors and precision of the taggers

	Tokens	%	FreeLing	%	HISPAL	%
Distribution-Morphology mismatch	113	31.5	94	83.2	98	86.7
Misspelling	79	22.0	40	50.6	41	51.9
Accent	37	10.3	12	32.4	10	27.0
Spare word	37	10.3	37	100.0	37	100.0
Punctuation, upper/lowercase	35	9.7	12	34.3	21	60.0
Stem-Distribution mismatch	18	5.0	17	94.4	16	88.9
Word choice	13	3.6	12	92.3	13	100.0

Interestingly, Distribution-Morphology mismatches, which is the most common type of error in our sample, receive a correct tag in more than 80 of cases, and Stem-Distribution mismatches, in more than 88%. The wrong word choice, as well as spare words, as expected do not interfere with tagger's precision.

The different performance of the taggers when dealing with learner language may be related to the backfall strategies and heuristics they use to treat unknown words or contexts (section 2.2). However, it is clear that their results could be improved with the help of an ortographic correction module designed to deal with learners errors related to accents, punctuation, the use of upper- or lowercase and misspellings.

4 Conclusions

We have evaluated the performance of two Spanish PoS taggers on Spanish learner texts. The taggers were designed to analyze native texts and employ different strategies to deal with faulty input. The results show a considerable high precision of both taggers dealing with texts written by intermediate-level learners of Spanish whose mother tongue is Japanese. The FreeLing tagger has a precision of 92.6%, and the HISPAL tagger, of 95.0%. In addition to that, part of the differences between the taggers can be due to the fact that FreeLing tries to disambiguate systematically ambiguous forms (if we substract those cases from the error rate, its precision raises to 94.3%).

The PoS tag with the lowest precision is the Proper Noun tag, in the FreeLing sample, due to punctuation and upper- or lowercase misuse. Next, the Verb has a low precision in FreeLing, while the Noun tag is the tag with the lowest precision in HISPAL. As for learner errors, in our sample 7% of words are wrong from a native point of view, for a variety of reasons: distribution-morphology mismatches is the most common type of error, followed by misspellings and accents, which together give account of two thirds of learner errors.

With regard to the relationship between PoS errors and learner errors, in HISPAL the former co-occur with learner errors in 43.0% of cases, and in FreeLing, in 33.2%. The confusion between verbs and common nouns is specially affected by learner errors, which indicates that the distinction between these two word classes is specially vulnerable to learner errors. The type of learner errors that have a bigger impact on the taggers' performance are ortographic errors like accents, punctuation, the use of upper- and lowercase letters and misspelling, while the taggers can deal reasonably well with grammatical errors like distribution-morphology or stem-distribution mismatches. Other type of errors, like the use of spare words or the wrong word choice does not interfere with the taggers' performance.

Overall, both taggers succeeded to analyze most of the learner errors. FreeLing correctly analyzed 64.9% of the words with learner errors, and HISPAL, 69.4%. However, given the influence of ortographic errors on their performance, their results could be improved with the help of an ortographic correction module.

To sum up, our evaluation shows that PoS taggers developed for native language can deal reasonably well with learner language and the learner errors frequently contained in it. While there is still room for improvement, we advocate for the use of PoS taggers for the analysis of (intermediate-level) learner language analysis, if possible with an ortographic module that detects errors in accents, punctuation and upper- or lowercase letters, which are the type of learner errors that cause more tagger errors. Such an ortographic module may be useful not only for learner input but also for native input, where ortographic rules are not stable either.

The annotation of learner corpora with PoS information will facilitate the study of learner language [11, 12] as well as the development of new tools for automatic error-tagging or Intelligent Computer Assisted Language systems [13].

Acknowledgments

This work was supported by *kakenhi* (22820047), Grant-in-Aid for Scientific Research (Start-up) from the Japan Society for the Promotion of Science.

References

1. Granger, S.: Computer learner corpus research: current status and future prospects. In: Connor, U., Upton, T. (eds.) Applied Corpus Linguistics: A Multidimensional Perspective, pp. 123–145. Amsterdam & Atlanta, Rodopi (2004)
2. Nicholls, D.: The cambridge learner corpus – error coding and analysis for lexicography and elt. In: Wilson, A., et al. (eds.) Proceedings of the Corpus Linguistics 2003 Conference (CL 2003), Technical papers 16, Lancaster University, Archer et al. (2003)
3. Rooy, B.V., Schäfer, L.: Automatic POS tagging of a learner corpus: The influence of learner error on tagger accuracy. In: Archer, D., Rayson, P., Wilson, A., McEnery, T. (eds.) Proceedings of the Corpus Linguistics 2003 Conference, CL 2003 (2003)
4. Thouesny, S.: Increasing the reliability of a part-of-speech tagging tool for use with learner language. In: Automatic Analysis of Learner Language, AALL 2009 (2009)
5. Mancera, A.M.C., Martínez, I.P., Canales, A.B., Fernández, L.C., Granda, J.F.S.: Corpus para el análisis de errores de aprendices de E/LE (CORANE). In: Sanz, A.G. (ed.) Actas del XII Congreso Internacional de ASELE: tecnologías de la información y de las comunicaciones en la enseñanza de la E/LE, pp. 527–534 (2001)
6. Atserias, J., Casas, B., Comelles, E., González, M., Padró, L., Padró, M.: Freeling 1.3: Syntactic and semantic services in an open-source nlp library. In: Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006), ELRA (2006)
7. Bick, E.: A constraint grammar-based parser for spanish. In: Proceedings of TIL 2006 - 4th Workshop on Information and Human Language Technology, Ribeirão, Preto (2006)
8. Schmidt, H.: Probabilistic part-of-speech tagging using decision trees. In: Proceedings of the International Conference on New Methods in Language Processing (1994)
9. Díaz-Negrillo, A., Meurers, D., Valera, S., Wunsch, H.: Towards interlanguage POS annotation for effective learner corpora in SLA and FLT. Language Forum 36(1–2) (2010)
10. Granger, S.: Error-tagged learner corpora and CALL: A promising synergy. CALICO Journal 20(3), 465–480 (2003)
11. Aarts, J., Granger, S.: Tag sequences in learner corpora: a key to interlanguage grammar and discourse. In: Learner English on Computer, pp. 132–141. Longman, Redwood City (1998)
12. Tono, Y.: A corpus-based analysis of interlanguage development: analysing POS tag sequences of EFL learner corpora. In: Practical Applications in Language Corpora (1999)
13. Heift, T., Schulze, M.: Errors and Intelligence in Computer-Assisted Language Learning: Parsers and Pedagogues. Routledge Studies in Computer Assisted Language Learning (2007)

Onoma: A Linguistically Motivated Conjugation System for Spanish Verbs

Luz Rello[1,⋆] and Eduardo Basterrechea[2]

[1] NLP & Web Research Group
Dept. of Information and Communication Technologies
Universitat Pompeu Fabra
Barcelona, Spain
[2] Molino de Ideas s.a.
Nanclares de Oca, 1F
Madrid, Spain

Abstract. In this paper we introduce a new conjugating tool which generates and analyses both existing verbs and verb neologisms in Spanish. This application of finite state transducers is based on novel linguistically motivated morphological rules describing the verbal paradigm. Given that these transducers are simpler than the ones created in previous developments and are easy to learn and remember, the method can also be employed as a pedagogic tool in itself. A comparative evaluation of the tool against other online conjugators demonstrates its efficacy.

1 Introduction

Although the literature about online Spanish conjugators is scarce, it does reveal that some are fully memory based (DRAE)[1] while others rely on finite state morphological rules [17][2].

To the best of our knowledge, the goal of most of the work related to verbal morphology was not the creation of an end-user tool such as a conjugator. However, both machine learning and rule-based approaches have been taken into consideration when processing inflectional morphology. While instance based-learning algorithms can induce efficient morphological patterns from large training data [2,1,5,13], approaches using finite state transducers [19,8,6] do enable the implementation of robust morphological analyzer-generators which are successful in handling concatenation phenomena [4].

The Onoma conjugator[3] was implemented as a cascade of finite state transducers that implements a decision tree. The use of finite state transducers (FSTs)

⋆ While developing this work the first author's institution was Molino de Ideas s.a.
[1] Conjugator from the Dictionary of the Royal Spanish Academy (DRAE). Available at: `http://buscon.rae.es/draeI/`
[2] The conjugator developed by Grupo de Estructuras de Datos y Lingüística Computacional (GEDLC) at the University of Las Palmas de Gran Canaria, which is available at: `www.gedlc.ulpgc.es/investigacion/scogeme02/flexver.htm`
[3] Developed and funded by Molino de Ideas. `http://conjugador.onoma.es`

provides the possibility of generating verbal paradigms as well as the reverse process: the analysis of inflectional verb forms [9]. Further, the use of a cascade structure facilitates the implementation of ordered alternation rules [10,11].

The remainder of the paper is structured as follows: the data and methodology used in this study is explained in Section 2, while Section 3 describes Spanish verbal morphology. Section 4 discusses the architecture of the system. A comparative evaluation of the system against other online conjugators is presented in Section 5. Finally, in Section 6, conclusions are drawn.

2 Data and Methodology

A database named the MolinoIdeas Verb Conjugation Database (MIVC-DB) was used for the modeling process. It contains 15,367 verbs (plus their corresponding verbal paradigms) including all the verbs registered in the Royal Spanish Academy Dictionary (11,060 verbs) [15], the Spanish Wikipedia, and the verbs found in a collection of 3 million journalistic articles from newspapers written in Spanish from America and Spain[4].

Our conjugator differs from the other Spanish processors in its architecture [17] (the GEDLC conjugator relies on the interaction of a segmentation program, three lists containing prefixes, verbal endings and pronouns, and two modules: one for the verbal endings and another for obtaining required external information) and in the design of the transducers, which are not based on concatenation rules [19] (in this FST model, a specific ending is added to 62 conjugation classes, giving as a result almost 150 verb-stem final states), but on rules which modify a hypothetical regular verb form, providing the possibility to extend such rules for the conjugation and analysis of verb neologisms in Spanish.

When designing the rules and patterns for each FST, the Spanish verbal inflectional paradigm was analyzed in detail from a linguistic point of view. This analysis led to the derivation of a simpler description of the inflectional verb paradigm which can be fully expressed (except for six verbs, see Section 4) using just nine patterns and a set of rules, as opposed to approximately one hundred and twenty conjugation models as in other approaches [7,18]. Given that the FSTs used in this system are easy to learn and remember, the description can be employed as a pedagogic tool in its own right by students of Spanish as a foreign language. It helps in the learning of the Spanish verb paradigm since currently existing methods (e.g. [14,12]) do not provide guidance on the question of whether verbs are regular or irregular. This is due to the fact that the system can identify the nature of any possible verb by reference only to its infinitive form[5] following just seven steps. [16].

For the design of the algorithm, in order to validate the rules and patterns extracted from the analysis of the MIVC-DB, an error-driven approach was taken.

[4] Newspapers with the major representation in our corpus are: *El País, ABC, Marca, Público, El Universal, Clarín, El Mundo* and *El Norte de Castilla.*

[5] In some rare cases, external information which the system also provides is required, see Section 4.

3 Spanish Verb Morphology

In Spanish, inflected verb forms exist for the nineteen tenses/moods as shown in Table 1[6].

Table 1. Inflected forms from the verbal paradigm

Tense/mood	**Examples, verb *ayudar* (to help)**
present tense/indicative	*ayudo*, 1st person singular
present tense/subjunctive	*ayude*, 1st person singular
present tense/imperative	*ayuda*, 2nd person singular
preterite imperfect tense/indicative	*ayudaba*, 1st person singular
preterite imperfect tense/subjunctive 1	*ayudara*, 1st person singular
preterite imperfect tense/subjunctive 2	*ayudase*, 1st person singular
preterite perfect composed tense/indicative	*he ayudado*, 1st person singular
preterite perfect composed tense/subjunctive	*haya ayudado*, 1st person singular
past perfect tense/indicative	*ayudé*, 1st person singular
past perfect composed tense/subjunctive	*hube ayudado*, 1st person singular
preterite pluscuanperfect tense/indicative	*había ayudado*, 1st person singular
preterite pluscuanperfect tense/subjunctive 1	*hubiera ayudado*, 1st person singular
preterite pluscuanperfect tense/subjunctive 2	*hubiese ayudado*, 1st person singular
future tense/indicative	*ayudaré*, 1st person singular
future tense/subjunctive	*ayudare*, 1st person singular
future perfect tense/indicative	*habré ayudado*, 1st person singular
future perfect tense/subjunctive	*hubiere ayudado*, 1st person singular
conditional simple tense/indicative	*ayudaría*, 1st person singular
conditional perfect tense/indicative	*habría ayudado*, 1st person singular

Except for the imperative, each tense possesses seven inflected forms corresponding to grammatical person. Furthermore, there are two infinitives and two gerunds (present and perfect) plus four forms of the participle form, depending on its number/gender variations. The potential therefore exists for up to 140 different forms per verb.

A Spanish verb consists of its stem, tense-mood inflections and person-number inflections. Most of the complexity resides in four factors:

1. Both kinds of inflection (tense-mood and person-number) can sometimes be realized by the same morphological segment;
2. the stem can be realised by different variations, i.e. the same verb can have more than one stem;
3. prefixes and suffixes can be added to the stem; and
4. the verb can be irregular which means that either the stem, the inflections or both are different from the hypothetical regular paradigm of conjugation.

Of 15,367 verbs, 4,225 are irregular (27.5 %). Moreover, 26.8% of the verbal neologisms in Spanish are irregular [16]. This group of irregular neologisms follow

[6] Throughout the paper, the solidus will be used when denoting tense/mood combinations.

the inflectional patterns of established verbs and conflates genuine paradigmatic irregularity and orthographic issues regarding grapheme realization on stem final consonants among others, shown in Section 4.

Most morphological processing systems are based on combining stems with inflections [19,7,12]. By contrast, our verbal paradigm description is based on patterns and transformational rules. Here, the term *rule* is used to denote an alteration that affects the hypothetical regular form of an irregular verb to generate the irregular form that matches with the appropriate irregular conjugation. Such rules are applied to a *pattern* which is the set of inflected forms affected by the irregularity rules (see subsection 4.1) in the verbal conjugation paradigm of the particular verb.

4 System Architecture

The system is composed of two modules, which employ finite state machines. The first one (**Classifier**) is designed to recognize the verb form and extract the information needed for its conjugation or analysis. This information is: (1) the word from which the verb form derives (if there is one) and (2) some formal information on the verb form which is derived via seven finite state automata (regular expressions) which detect wether the verb is regular or irregular based on its ending [16] or, in some cases, from the word that the verb is derived from. This module makes use of two additional purpose-built submodules: one to detect the word from which the verb is derived and another to identify the stress pattern of the verb. These two submodules are used to detect the verb root and to provide information that will later be exploited for its inflection or analysis. When the verb form is irregular, this information will be used to select the irregularity rules and patterns to be applied (see subsection 4.1).

By means of the first module, the verbs are classified into two groups [3]: (a) *regular verbs* and (b) *irregular verbs*. When identified, irregular verbs are further divided into (b.1) the so-called *Magnificent verbs*, *traer* (to bring), *valer* (to be worth), *salir* (to go out), *tener* (to have), *venir* (to come), *poner* (to put), *hacer* (to do), *decir* (to say), *poder* (can), *querer* (to want), *saber* (to know), *caber* (to fit), *andar* (to walk), and their derivations; (b.2) verbs which undergo diphthongization or a vowel replacement in their root; (b.3) verbs which are affected by diacritic rules of irregularity; (b.4) verbs which suffer orthographic changes in their endings; (b.5) verb forms whose root ends in a vowel and will undergo heterogeneous rules of irregularity, and finally; (b.6) the irreducible verbs which are a set of six verbs whose conjugations are stored in memory: the auxiliary verb (*haber*, (to have)), the copulative verbs, *ser* (to be) or *estar* (to be), and the monosyllabic verbs: *ir* (to go) *dar* (to give) and *ver* (to see). Apart from the irreducible verbs, the rest of the verbal paradigm system is based entirely on rules and patterns implemented in Module 2 (**Modeling**).

Module 2 is composed of two conjugation modules. The first module **(2.1 Hypothetical verb form)** conjugates –or analyses– the verb form as if it were regular by concatenating the root with the corresponding inflections depending on its tense, mood, person and number. The second conjugation module **(2.2 Modifying the hypothetical verb form)** – is composed of several finite state machines. For irregular verbs, it first detects the type of patterns and rules of irregularity that should be applied to the hypothetical verb forms generated by Module 2.1. Next, it applies the selected irregularity rules and patterns to generate the correct irregular paradigm. There are a total of 40 rules and seven patterns, plus two additional ones for the Magnificent verbs.

4.1 Module 2.2: Modifying the Hypothetical Verb Form

Patterns: Each pattern is composed of the set of grammatical person, tense, and number forms which are affected by the associated rule. The patterns are correlated with groups of verbs that satisfy a set of formal conditions. The names of the patterns and the characteristics of the inflectional verbs affected by them are stated below:

(1) Pattern **To**: recognizes verbs whose root contains the stressed syllable.
(2) Pattern **Te**: for verbs whose inflection contains the stressed syllable.
(3) Pattern **Dei**: recognizes verbs whose inflections begin with the vowels *e* or *i*.
(4) Pattern **Dao**: recognizes verbs whose inflections begin with the vowels *a* or *o*.
(5) Pattern **Di**: recognizes verbs with a stressed inflection that begins with an unstressed *i*.
(6) Pattern **Dti**: recognizes verbs whose inflections begin with a stressed *i*.
(7) Pattern **Dt-i**: is used to recognize verbs with a stressed inflection that begins with any vowel except *i*.

Depending on the pattern and the formal composition of the verb form, a specific irregularity rule is activated by means of one of the FSTs in Module 2.2: modifying the hypothetical verb form.

To illustrate: pattern **Dei** activates the irregularity modifications (subsection 4.1) which always affect third person singular and first and third person plural forms of the present tense/imperative, all the person forms of the present tense/subjunctive and the first person singular of the preterite perfect simple tense/indicative. For example, the form *escenifique* from *escenificar* (to stage) substitutes the letter *c* by *qu* in the first person singular present tense/subjunctive form.

Similarly, the irregularity rules (see subsection 4.1) activated by the pattern **Di** will only affect the gerund, the third person singular and the first person plural forms of the preterite perfect simple tense/indicative plus all the grammatical person forms of the preterite imperfect tense/subjunctive and the future tense/subjunctive. To illustrate, the verb form *cayere* from verb *caer* (to fall) adds a *y* between its root and the inflections in all person forms of the preterite imperfect/subjunctive and the future/subjunctive tenses/moods.

The Magnificent verbs are recognized using two specific patterns:

(8) Pattern **Fc**: for all the grammatical person forms of future and conditional tenses/indicative moods.

(9) Pattern **í4**: allows recognition of verbs for all person forms belonging to preterite perfect simple tense/indicative mood and preterite imperfect/subjunctive and the future/subjunctive tenses/moods.

Irregularity Transformational Rules: Finally, Module 2.2 applies the pertinent irregularity modifications over the hypothetical regular forms generated by Module 2.1. in order to generate the corresponding irregular verb form.

The rules perform one of the following three types of alteration:

– **substitution**, (e.g. *z* is substituted by *c* in pattern **Dei**, to derive the first person singular present tense/subjunctive inflected form *trace* from *trazar* (to trace));
– **addition**, (e.g. *z* is added in the root for verb forms recognized using the pattern **Dao**, as illustrated when the first person singular present tense/ indicative form *conozco* is derived from *conocer* (to know));
– **deletion**, (e.g. the vowel *i* is removed from the inflections of verbs recognized by means of the **Di** pattern, as illustrated when the first person singular present tense/indicative form *taño* is derived from *tañer* (to strum)).

Overall, 40 irregularity rules have been implemented. They are divided into five groups:

(1) **Consonantal orthographic transduction rules:** These comprise 9 FSTs which modify the verb in order to ensure that the derived form obeys Spanish orthographic conventions. These rules are activated for verbs recognized using the patterns **Dei**, **Dao** and **Di** (e.g. one rule of this type enables the first person singular, present tense/indicative form *sigo* to be derived from *seguir* (to follow), when it is activated by the pattern **Dao**).
(2) **Diacritic transduction rules:** Comprised by 2 FSTs activated by the pattern **To** (The processing that they perform is illustrated by the derivation of the first person singular, present tense/indicative form *vacío* from *vaciar* (to empty)).
(3) **Root vowel transduction rules:** Comprised by 8 FSTs that operate on the root vowel, which can be either diphthongized or replaced by another vowel. These irregularity rules are activated by patterns **To** and **Dti** (e.g. when the first person singular, present tense/indicative form *sirvo* is derived from *servir* (to serve), having been activated by the pattern **Dti**).
(4) **Vowel root ending transduction rules:** Comprised of 8 FSTs which apply heterogeneous transduction rules affecting those verbs whose infinitive form root ends in a vowel. The use of these rules is illustrated by the derivation of *oyes* from *oír* (to hear) by addition of the letter *y* after the root, having been activated by the pattern **Te**.
(5) **Specific Magnificent verbs transduction rules:** Comprising 13 FSTs activated by the patterns **Fc**, **í4**, **Dao** and **To**. To illustrate, the root of the verb *tener* (to have) is changed (when the rule is activated by the pattern **í4**). *(Tuve)*, is modified by adding the letter *g* after its root (when the rule is activated by the pattern **Dao**) and *(tengo)* is modified by addition of *d* after the root in the verb forms recognized by the pattern **Fc** *(tendré).*

The FSTs exploited in Module 2.2 are arranged in a cascade as their order of application is important, given that most of the irregular verbs activate several rules. For instance, *dormir* (to sleep) undergoes substitution of its root vowel *o* by *u* when recognized by pattern **Dti** (firstly, the second person plural of present tense/subjunctive form *durmáis*, is derived. This is followed by diphthongization of the root vowel when it is recognized by the pattern **To** to derive the first person singular of present tense/indicative form *duermo*.

5 Comparative Evaluation

A comparative evaluation of the system was carried out against seven Spanish conjugators that are available online. They are:

1. Royal Spanish Academy Conjugator: `http://buscon.rae.es/draeI/`.
2. Reverso conjugator:
   `http://conjugador.reverso.net/conjugacion-espanol.html`.
3. WordReference Spanish Verb Conjugator:
   `http://www.wordreference.com/conj/EsVerbs.asp`.
4. University of Oviedo conjugator:
   `http://www6.uniovi.es/dic/conjuga.html`.
5. The conjugator developed by Grupo de Estructuras de Datos y Lingüística Computational from University of Las Palmas de Gran Canaria:
   `http://www.gedlc.ulpgc.es/investigacion/scogeme02/flexver.htm`.
6. SpanishDict Verb Conjugator: `http://www.spanishdict.com/conjugate/`.
7. Verbix Spanish Verb Conjugator v.2.0:
   `http://www.verbix.com/languages/spanish.shtml`.

Please notice the comparison between these results should be done with caution since there is no reason to assume that the other conjugators are aiming to address the same task, specifically, the conjugation of verbal neologisms.

A list containing 40 heterogeneous verb forms (inflectional forms as well as infinitives) was tested against each conjugator. The verb forms used in the evaluation were carefully selected on account of their difficulty. They can be classified into five *ad hoc* categories: (1) regular and irregular verb neologisms[7], formed by concatenating a prefix to an existing verb: *autodestruir* (to self destroy); or (2) verb neologisms formed from words which are not verbs: *googlear* (to google); (3) verbs with multiple conjugation: *roer* (to gnaw) which, for example, can be conjugated as *roo*, *roigo* or *royo* in its first person singular form in present tense/indicative mood); (4) verbs with double meanings whose paradigm of conjugation differs depending on the meaning: *acostar* (*acuesto*, to put in bed; *acosto*, to reach the coast) (Table 2); and (5) ambiguous inflected forms (Table 3). Of the 40 verb forms, 10 belong to class 1, 10 to class 2, 6 instances belong to classes 3 and 6 to class 5. 8 ambiguous examples belong to class 5.

Table 2 presents which systems are able to generate the different kinds of verbal paradigms, while Table 3 shows which systems analyze inflected verb forms

[7] The neologisms chosen for the evaluation are not present in MICV-DB.

Table 2. Comparative evaluation: generation of verbal paradigms

System	**Conjugation**	**1 Prefix neologisms**	**2 New word neologisms**	**3 Multiple conjugation**	**4 Double meaning**
Conjugator 1	yes	no	no	yes	yes (see text)
Conjugator 2	yes	no (see text)	no	no	no
Conjugator 3	yes	no	no	yes	no
Conjugator 4	yes	no	no	yes	no
Conjugator 5	yes	yes	no	yes	no
Conjugator 6	yes	no	no	no	no
Conjugator 7	yes	yes	yes	yes	no
Onoma	yes	yes	yes	yes	yes

and ambiguous inflected forms. Table 4 presents the accuracy of the conjugation or the analysis of 40 verb forms in the systems.

Conjugator 1 (in Table 2) does present the two different conjugations of a verb with double meanings, although it does not state which type of verbal paradigm corresponds to which meaning, as our system does. *Onoma* offers the user the opportunity to first choose the appropriate meaning and then displays the verb paradigm depending on the user's choice. Conjugator 2 does conjugate some verb neologisms formed with prefixes although it does not cover all cases (e.g. *cohacer* (to do at the same time)).

Only half of the tested conjugators (including *Onoma*) analyze inflected verb forms (see Table 3) and ambiguous verb forms (5) were used to test this type of analysis. For instance, *sé* can either be the first person singular form indicating the present tense/indicative mood of the verb *saber* (to know), or the second person singular present tense/imperative form of the verb *ser* (to be).

Table 3. Comparative evaluation: analysis of inflected verb forms

System	**Analysis**	**5 Ambiguous verb forms analysis**
Conjugator 1	no	no
Conjugator 2	yes	no
Conjugator 3	yes	yes
Conjugator 4	no	no
Conjugator 5	yes	yes
Conjugator 6	no	no
Conjugator 7	no	no
Onoma	yes	yes

As can be inferred from the evaluation presented, no other existing online conjugation system is as extensive in its functionality and the range of features employed. This is particularly evident in its ability to identify and analyze

Table 4. Comparative evaluation: accuracy of conjugation and analysis

System	**Conjugation accuracy: neologisms and registered verbs**	**Analysis accuracy: registered verbs**
Conjugator 1	37.5%	none
Conjugator 2	25.0%	37.5%
Conjugator 3	31.2%	87.5%
Conjugator 4	31.2%	none
Conjugator 5	50.0%	100%
Conjugator 6	15.6%	none
Conjugator 7	81.2%	none
Onoma	100%	87,5%

ambiguous inflected forms, verb neologisms and to deal with verbs that have double meanings and therefore, double conjugations.

Nevertheless, our system does not conjugate six outdated verbs (e.g. *far* (to do), *caler* (to be necessary), etc.) as, to our knowledge, such paradigms have been largely unexplored, though two conjugators (2 and 7) do present the possible verbal paradigm for this small group of verbs. Given that they are not used in contemporary Spanish, their treatment is considered beyond the scope of the present paper.

On this data set, *Onoma* is able to conjugate verbal paradigms with 100% accuracy, while its accuracy in analysing verb forms is 87,5%. Onoma, as well as displaying the correct verb paradigms in its analysis of all registered verbs sometimes includes paradigms of verbs that may possibly occur in Spanish, but are not registered in existing dictionaries. When paradigms of non-registered verbs are included in its analyses of a verb, this is considered an error, regardless of whether or not the rest of the analysis is correct. While this strict approach to evaluation adversely affects the performance level reported for *Onoma*, it allows a feasible comparison to be made with those systems that do not treat neologisms. Overall, it can be concluded that *Onoma* compares favorably with the other conjugators in terms of the accuracy of analysis and conjugation.

6 Final Remarks

In this paper we have presented *Onoma*, a system which conjugates Spanish verbs, including neologisms. *Onoma's* linguistically motivated model for verb paradigms is novel and has great potential for pedagogic applications in teaching the intricacies of the Spanish verb conjugation system.

As the *Onoma* transducers are implemented on a database management system, they are simple to modify independently of the rest of the software. In future work, we plan to integrate the *Onoma* algorithm into a general Spanish morphological processor to treat the rest of the open-class lexical categories.

Acknowledgements

We would like to express our gratitude to the skillful Molino de Ideas s.a. engineering team who have successfully implemented the method, particularly Daniel Ayuso de Santos and Alejandro de Pablos López. Special thanks to Prof. Leo Wanner who pointed us helpful literature

References

1. Albright, A., Hayes, B.: Modeling English Past Tense Intuitions with Minimal Generalization. In: Proceedings of the 6th Workshop of the ACL Special Interest Group in Computational Phonology (SIGPHON), pp. 58–69 (2002)
2. Anick, P., Artemieff, S.: A high-level morphological description language exploiting inflectional paradigms. In: Proceedings of COLING 1992, pp. 67–73 (1992)
3. Basterrechea, E., Rello, L.: El verbo en español. Construye tu propio verbo. Molino de Ideas, Madrid (2010)
4. Beesley, K.R., Karttunen, L.: Finite-State Non-Concatenative Morphotactics. In: Proceedings of the 5th Workshop of the ACL Special Interest Group in Computational Phonology (SIGPHON), pp. 1–12 (2000)
5. Creutz, M., Lagus, K.: Induction of a Simple Morphology for Highly-Inflecting Languages. In: Proceedings of the 7th Meeting of the ACL Special Interest Group in Computational Phonology: Current Themes in Computational Phonology and Morphology, pp. 43–51 (2004)
6. Gasser, M.: Semitic Morphological Analysis and Generation Using Finite State Transducers with Feature Structures. In: Proceedings of the 12th Conference of the European Chapter of the ACL, pp. 309–317 (2009)
7. Gomis, P., Segura, L.: Vademécum del verbo español. In: SGEL. Sociedad General Española de Librería, Madrid (1998)
8. Görz, G.: A Finite State Approach to German Verb Morphology. In: Proceedings of COLING 1988, pp. 212–215 (1988)
9. Kaplan, R.M., Kay, M.: Regular models of phonological rule systems. Computational Linguistics 20, 331–378 (1994)
10. Karttunen, L.: KIMMO: A general morphological processor. Texas Linguistic Forum 22, 165–185 (1983)
11. Karttunen, L., Kaplan, R.M., Zaenen, A.: Two-level morphology with composition. In: Proceedings of COLING 1992, pp. 141–148 (1992)
12. Mateo, F.: Bescherelle. Les verbes espagnols. Hatier, Paris (2008)
13. Parkes, C.H., Malek, A.M., Marcus, M.P.: Towards Unsupervised Extraction of Verb Paradigms from Large Corpora. In: Proceedings of the 6th Workshop on Very Large Corpora, pp. 110–117 (2007)
14. Puebla, J.: Cómo conjugar todos los verbos del español. Playor, Madrid (1995)
15. Real Academia Española. Diccionario de la lengua española, 22th edn. Espasa, Madrid (2001)
16. Rello, L., Basterrechea, E.: Automatic conjugation and identification of regular and irregular verb neologisms in Spanish. In: Proceedings of the NAACL 2010, Workshop on Computational Approaches to Linguistic Creativity, CALC 2010 (2010)

17. Santana, O., Pérez, J.R., Hernández, Z.J., Carreras, F.J., Rodríguez, G.: FLAVER: Flexionador y lematizador automático de formas verbales. Lingüística Española Actual 19(2), 229–282 (1997)
18. Santana, O., Carreras, F.J., Hernández, Z.J., Pérez, J.R., Rodríguez, G.: Manual de la conjugación del español. 12 790 verbos conjugados. Arco Libros, Madrid (2002)
19. Tzoukermann, E., Liberman, M.Y.: A Finite-State Morphological Processor for Spanish. In: Proceedings of the 13th Conference on Computational Linguistics, vol. 1, pp. 277–282 (1990)

Measuring Similarity of Word Meaning in Context with Lexical Substitutes and Translations

Diana McCarthy

Lexical Computing Ltd.
Brighton, BN1 6WE
diana@dianamccarthy.co.uk

Abstract. Representation of word meaning has been a topic of considerable debate within the field of computational linguistics, and particularly in the subfield of word sense disambiguation. While word senses enumerated in manually produced inventories have been very useful as a start point to research, we know that the inventory should be selected for the purposes of the application. Unfortunately we have no clear understanding of how to determine the appropriateness of an inventory for monolingual applications, or when the target language is unknown in cross-lingual applications. In this paper we examine datasets which have paraphrases or translations as alternative annotations of lexical meaning on the same underlying corpus data. We demonstrate that overlap in lexical paraphrases (substitutes) between two uses of the same lemma correlates with overlap in translations. We compare the degree of overlap with annotations of usage similarity on the same data and show that the overlaps in paraphrases or translations also correlate with the similarity judgements. This bodes well for using any of these methods to evaluate unsupervised representations of lexical semantics. We do however find that the relationship breaks down for some lemmas, but this behaviour on a lemma by lemma basis itself correlates with low inter-tagger agreement and higher proportions of mid-range points on a usage similarity dataset. Lemmas which have many inter-related usages might potentially be predicted from such data.

1 Introduction

Words mean different things in different contexts and if we want systems to interpret and produce language as humans do then we need to build systems that can handle this. Work in computational lexical semantics has been dominated by work involving predefined inventories of senses, particularly in the subfield of word sense disambiguation (WSD). The use of inventories such as WordNet [1] have been a major catalyst for work in lexical semantics and certainly there are good reasons why one might want to use such an inventory, for example to exploit the other information contained therein. However, frequently inventories are used because that is how the gold-standard data has been annotated, rather than with regard to whether the distinctions that are being made are in fact useful. The lack of consensus on the appropriate representation of sense has been a major cause of contention within the WSD community.

While we know that the inventory should be selected for the purposes of the application, we have no clear understanding of how to determine the appropriateness of an inventory, certainly for monolingual applications, but also when the target language is unknown in cross-lingual applications.[1] Additionally, while there has been good deal of research in WSD using predefined inventories, there is a need to examine and evaluate other representations of lexical semantics such as inventories that can be derived automatically from distributional similarity data [2, 3].

In this paper we provide an overview of three different datasets that have been designed so as to provide alternative annotations of word meaning in context. These three datasets are:

- the English Lexical Substitution Task dataset (LEXSUB) [4]
- the Cross-Lingual Lexical Substitution Task dataset (CLLS) [5]
- the Usage Similarity dataset (Usim) [6]

It is quite possible to use these gold-standards to evaluate preexisting inventories, but since the gold-standards were not produced using any particular inventory they also allow comparison between different representations.

The three different datasets have been created using lexical substitutes, i.e. paraphrases, (LEXSUB),[2] translations (CLLS) and ordinal scale usage similarity judgements (Usim). The datasets all used a portion of data in common, allowing us to compare these different annotations. In this paper we examine to what extent these alternative annotations correlate which we would expect them to do as representations of the same underlying phenomena: word meaning in context.

We are of course only able to examine this relationship in the context of the data we have available. Our results will depend on the lemma and occurrences of that lemma. We also examine these relationships on a lemma by lemma basis and show that the lemmas where the relationship tends to break down also tend to be the lemmas with poor inter-tagger agreement in the more fundamental usage similarity task. We show that the proportion of mid range judgements for a lemma in the usage similarity task also correlates with the inter-tagger agreement for that lemma. This mid range measure therefore might be used as a predictor of words where meanings are difficult to distinguish.

The paper is structured as follows. In the next section we give some background to word meaning annotation. We also provide an overview of the LEXSUB, CLLS and Usim datasets and the methodologies used to create them. In section 3

[1] Even in the case of machine translation, it is possible that translations won't form neat clusters, or that a particular usage might warrant a translation that has not occurred in training data for that lexeme.

[2] We use the terms paraphrases and substitutes interchangeably in this paper to mean lexical paraphrase of a lemma in context within the same language. We use translations for the cross-lingual substitutes in the CLLS data.

we motivate our analysis and highlight the subset of the underlying corpus data that is common to all datasets. In subsection 3.1 we describe our methodology for calculating similarity between two usages of the same lemma in terms of the overlap of paraphrases or translations. In subsection 3.2 we examine the correlations of the overlap and Usim scores over the data that is common to all three datasets. In subsection 3.3 we do the same analysis on a lemma by lemma basis. We demonstrate that lemmas where the correlation between the overlap (LEXSUB, CLLS) and similarity measures (Usim) is low or not evident, tend also to be those lemmas with low inter-tagger agreement on Usim and have a higher proportion of mid-range scores indicating less certainty of the annotators and more inter-relationships between usages.

The data and software that have been used for the analysis in this paper will be available at http://www.cicling.org/2011/software/diana, along with pointers to the related datasets from which the data described here was taken.

2 Word Meaning Annotations

2.1 Background

There has been a great deal of work on word meaning annotations in computational linguistics, in particular the SENSEVAL and SemEval series[3] Most of the datasets used have been produced with the same methodology whereby a list of senses is given and each annotator is asked to select the best one. For example, given the WordNet listing of the noun *match* in figure 1, and the sentence below from LEXSUB:

> *Like the Philadelphia match, this event was covered live on the World Wide Web*

It is unlikely that any native speaker would have difficulty in assigning sense 2 for the word *match* in this context. However, it is easy to find problematic cases such as the following (from the same dataset):

> *This is at least 26 weeks by the week in which the approved match with the child is made*

In this case, senses 8 and 9 both seem relevant. Sense 8 is not quite right because this sentence seems to be talking about adoption, not a partnership (married or otherwise) of equals. Meanwhile sense 9 does not seem quite right as the word *resembles* and the example provided in the definition in figure 1 seem to indicate visual appearance.

Problematic cases such as these are commonplace [7]. Despite these issues, word sense tagging may be useful when one specifically needs to apply the inventory to corpus data, for example to exploit semantic relations [8]. WSD is

[3] The SENSEVAL series was renamed SemEval for the 2007 event due to the increasing range of semantic phenomena that the tasks covered, rather than just word senses.

1. *match, lucifer, friction match* – (lighter consisting of a thin piece of wood or cardboard tipped with combustible chemical; ignites with friction; "he always carries matches to light his pipe"; "as long you've a lucifer to light your fag")
2. *match* – (a formal contest in which two or more persons or teams compete)
3. *match* – (a burning piece of wood or cardboard; "if you drop a match in there the whole place will explode")
4. *match, mate* – (an exact duplicate; "when a match is found an entry is made in the notebook")
5. *match* – (the score needed to win a match)
6. *catch, match* – (a person regarded as a good matrimonial prospect)
7. *peer, equal, match, compeer* – (a person who is of equal standing with another in a group)
8. *couple, mates, match* – (a pair of people who live together; "a married couple from Chicago")
9. *match* – (something that resembles or harmonizes with; "that tie makes a good match with your jacket")

Fig. 1. WordNet 3.0 senses for the noun *match*

often performed on the premise that it will ultimately prove useful for natural language processing applications, but without specifically addressing its utility. There is some research as to the utility of WSD, but unfortunately the evidence is equivocal [9–13]. A popular proposal is to make coarse grained senses such that tagging by humans and machine is more reliable [14–16]. While this undoubtedly helps boost inter-tagger reliability and WSD system performance [17], we do not yet know whether this increased performance will help in applications. Sometimes the coarse distinctions are resolved implicitly, for example by virtue of the collocations within a query and it may be the subtler distinctions that will improve the results [18].

While some lemmas may be easy to partition into senses, this is not always the case [19]. Senses often fall between one another making sense grouping decisions difficult. For example, the senses of the noun *child* in WordNet as shown in figure 2 are clearly related to one another, however if one was to group these there are several ways to go. Sense 1 seems to lie somewhere between sense 2 and sense 3. That is, it is related to both 2 and 3 more than sense 2 and sense 3 are to each other. Sense 2 likewise seems to lie somewhere between sense 1 and sense 4. In SENSEVAL-2, sense 1 and 3 formed one group and sense 2 and 4 another. While this is quite a plausible grouping, using the **youth** vs **descendant** distinction, it does not reflect the fact that that sense 1 and 2 are clearly related, and can be translated the same way in some languages (for example, both might be translated as *enfant* in French).

While we do certainly acknowledge that annotating and disambiguating word senses may be necessary for some applications, in this paper our aim is to examine datasets that do not use a predefined inventory and therefore allow for

1. *child, kid, youngster, minor, shaver, nipper, small fry, tiddler, tike, tyke, fry, nestling* – (a young person of either sex; "she writes books for children"; "they're just kids"; "'tiddler' is a British term for youngster")
2. *child, kid* – (a human offspring (son or daughter) of any age; "they had three children"; "they were able to send their kids to college")
3. *child, baby* – (an immature childish person; "he remained a child in practical matters as long as he lived"; "stop being a baby!")
4. *child* – (a member of a clan or tribe; "the children of Israel")

Fig. 2. WordNet 3.0 senses for the noun *child*

comparison of different approaches that represent lexical semantics, including fully unsupervised distributional approaches, without biasing to an approach that uses a specific inventory. The datasets involve tasks (lexical paraphrasing, translation and similarity) that test systems in ways that should bode well for applications such as summarisation, translation and question-answering, both monolingual and cross-lingual.

2.2 Three Datasets Annotated without Recourse to a Specific Inventory

LEXSUB: The English Lexical Substitution Dataset. LEXSUB was devised for SemEval 2007 as a means of evaluating WSD systems without recourse to a predefined inventory. For the gold standard dataset, five human annotators produced lexical paraphrases (substitutes) for target words in the context of a sentence. The 201 target words (nouns, verbs, adjectives and adverbs) each have ten sentences which were extracted from the English Internet Corpus [20]. The annotators were permitted to supply up to three substitutes, provided each were equally valid, and they were also permitted to provide a NULL response in the event that they could not find a suitable substitute. The details are reported in [4, 21]. Systems that perform well on this task have potential in paraphrasing applications. The task merges two subtasks of i) finding the substitutes and ii) selecting the appropriate ones for the context. It is possible to evaluate these two subtasks separately [21] on retrieval of the union of substitutes for the ten sentences (for evaluating inventories) and by providing the union of substitutes as input for matching to the context.

We provide an example of the gold-standard data for the adjective *stiff* in figure 3.[4] The sentence number (1 to 10) is given in the first column and the English substitutes (lexical paraphrases) provided by the five annotators are given in the second column. The frequency of each substitute from all five

[4] In the caption of this figure and elsewhere in this paper we use the suffix {a,r,n,v} for the PoS classes {adjective, adverb, noun and verb} respectively.

Sent	LEXSUB substitutes	CLLS translations
1	rigid 4; inelastic 1; firm 1; inflexible 1	duro 4; tieso 3; rigido 2; agarrotado 1; entumecido 1
2	rigid 3;unyielding 1;firm 1	duro 3; tieso 3; rigido 2; agarrotado 1; fuerte 1; yerto 1; firme 1
3	strong 2; firm 2; good 1; solid 1; hard 1	duro 4; definitivo 1; severo 1; fuerte 1
4	strong 4; tough 2; intense 1	duro 3; dificil 2; consistente 1; fuerte 1; protocolario 1; marcado 1; ceremonioso 1; firme 1
5	aching 3; frozen 1; rheumatic 1; cricked 1; painful 1; sore 1	torticolis 2; entumecido 2; duro 1; anquilosado 1; torcido 1; tieso 1
6	aching 3; sore 2	duro 2; tieso 2; rigido 1; agarrotado 1; yerto 1; firme 1
7	stern 1; formal 1; firm 1; unrelaxed 1; constrained 1; unnatural 1; unbending 1	duro 2; forzado 2; fijo 1; rigido 1; acartonado 1; insipido 1
8	harsh 2; heavy 2; severe 1; strong 1; rigid 1	duro 3; severo 3; contundente 1; rigodo 1; estricto 1
9	firm 3; rigid 2; unyielding 1; hard 1	duro 2; protocolario 1; severo 1; ceremonioso 1; firme 1
10	strong 4; heavy 2; powerful 1	consistente 3; duro 3; contundente 1; fuerte 1; dificil 1

Fig. 3. Substitutes and Translations for the *stiff.a* sentences in LEXSUB and CLLS

annotators is shown after that substitute and before the semi-colon.[5] In this data one can see the different meanings of *stiff*: *rigid, strong (good), aching, stern, harsh.* However, we also see that there are overlaps in the substitutes suggesting a potential overlap in meaning, for example the meanings of *stiff* in sentences 1, 3 and 7 are clearly different yet they share the substitute *firm*:

1) *Even though it may be able to pump a normal amount of blood out of the ventricles, the stiff heart does not allow as much blood to enter its chambers from the veins.*
3) *One stiff punch would do it.*
7) *In 1968 when originally commissioned to do a cigarstore Indian, he rejected the stiff image of the adorned and phony native and carved " Blue Nose, " replica of a Delaware Indian.*

CLLS: The Cross-Lingual Lexical Substitution Dataset. This task was run as part of SemEval 2010. For this dataset four human annotators annotated all sentences for 130 lemmas taken from the LEXSUB dataset. All annotators were native Spanish speakers with a high level of proficiency in English. Their task was to produce Spanish translations for the target word in the context of a

[5] Unlike a traditional WSD task the annotators are not selecting from a fixed inventory and all substitutes with their respective frequencies are used in the gold-standard.

sentence. They were asked to provide as many translations as possible. Unlike a full blown machine translation task [11], annotators and systems are not required to translate the whole context but just the target word. Such a facility might be useful for language learners, as an aid to human translation, or the output might be useful for cross-lingual information retrieval and translation. Previous work has used translations for sense distinctions [22] however in the CLLS task there was no requirement for distinct groupings of translations. The annotators were simply asked to provide as many translations as they felt were appropriate for the target word in the context.

By way of an example, the translations of the adjective *stiff* are given in the third column of figure 3. The relationships between the different usages are apparent from the translations (all have *duro* as a translation, and many have *rigido*), but also we can see differences which correspond to those found in the paraphrases (for example only sentences 5 and 6 have substitutes *aching, sore* and the translation *tieso*).

The Usage Similarity Dataset (Usim). This dataset has been collected in two rounds. The first set of annotations was collected as one of the annotation tasks described in [6]. A further set of such annotations has subsequently been collected using approximately the same method. We refer to the two rounds as Usim-1 and Usim-2. The data for both rounds was taken from LEXSUB. The task involves the users being given every pair of 10 sentences for each lemma, i.e. 45 sentence pairs per lemma, and the annotators were asked to give a judgement between 1 and 5 of how similar the usage of that lemma was in the two sentences. Rather than use a word sense, substitute or translation as an annotation, this task aims to represent lexical meaning in context as similarity between two occurrences. This has been used previously to look at the correlation between the overlap of substitutes and usage similarity judgements [6], and we extend that analysis to compare overlap of translations with the overlap of substitutes, as well as both with usage similarity. The Usim dataset also has potential for evaluating distributional methods which represent lexical meaning without necessarily inducing clusters (word senses).

Both rounds use data taken from the LEXSUB dataset. The first round used data for 34 lemmas whereas the second round used data for 26 lemmas. There were four lemmas that were common to both rounds. The annotation proceeded in more or less the same way except that in the first round only one sentence of context was given as in LEXSUB, whereas in the second round the annotators could also see the previous and following sentence. Also, in the first round there were three annotators whereas in the second round we used eight annotators to reduce the impact of individual variation.

3 Analysis

In this paper we examine the extent that lexical substitutes (from LEXSUB) and translations (from CLLS) correlate with each other and with Usim judgements using the overlap of paraphrases or translations as a measure of similarity between two usages of the same lemma.

In the analysis reported here, we examine only the LEXSUB data that is common to both CLLS and Usim. There are 32 lemmas from Usim-1 also in CLLS and 24 lemmas from Usim-2 also in CLLS. There are four lemmas that overlap these two sets. Each lemma has 10 sentences other than:

– one sentence (for *bar.n*) which had one sentence discounted because of a mistagging in the trial dataset from LEXSUB,
– one sentence (for *lead.n*) which was discounted in LEXSUB as only one substitute from the five annotators was provided
– two sentences (one for *flat.a* and one for *shade.n*) which were discounted in Usim (rounds 1 and 2 respectively) because one of the annotators provided a 'don't know' response [6]

Thus the results here are based on 516 sentences for these 52 lemmas. We report the overall correlations between datasets separately for the two rounds of Usim. Since the second round of Usim had more annotators and consequently has less individual variation, when we give results by lemma we use round 2 for the four lemmas that are in both Usim-1 and -2.

3.1 Overlap Calculation

In order to compare tasks, we follow previous work [6] in calculating the overlap of substitutes (and for CLLS, these are the translations) at each sentence pair for a lemma. The overlap, INTER, is the multiset intersection of the substitutes in two sentences ($s1$ and $s2$). INTER is calculated as follows. Let $subs_{s1}$ and $subs_{s2}$ be the sets of substitute types for $s1$ and $s2$ respectively. Let $freq_{w1}$ be the frequency of a substitute (w) for $s1$ from the LEXSUB (or CLLS) gold standard and $freq_{w2}$ be the equivalent for $s2$. Let $maxfreq(freq_{s1}, freq_{s2})$ be the larger of $freq_{s1}$ and $freq_{s2}$ where these are the total frequency of all substitutes for the respective sentence. Then INTER is measured as:

$$\text{INTER}(s1, s2) = \frac{\sum_{w \in subs_{s1} \cap subs_{s2}} min(freq_{w1}, freq_{w2})}{maxfreq(freq_{s1}, freq_{s2})} \tag{1}$$

3.2 Correlations between LEXSUB, CLLS and Usim

As in [6], and following [23], we calculate spearman's rank correlation coefficient (ρ) between the overlap measures and the average judgement from Usim. We normalise all Usim judgements to z-scores before taking the average to allow for the fact that the annotators may have used the 1-5 scale differently. In table 1 we see the correlations between the LEXSUB, CLLS and Usim datasets. The correlations are calculated over every sentence pairing for every lemma within that dataset. These correlations are all highly significant as can be seen from the probability (p-value) of this occuring by chance. For the second round of Usim annotation there were more annotators and this results in less noise from individuals and stronger correlation (larger values of ρ). We also note that the Usim tasks both have highercorrelations with the paraphrases in LEXSUB compared

Table 1. Correlations between datasets

datasets	ρ	p-value
LEXSUB-CLLS	0.519	2.04e-169
LEXSUB-Usim-1	0.570	9.37e-122
LEXSUB-Usim-2	0.724	9.24e-172
CLLS-Usim-1	0.523	1.10e-99
CLLS-Usim-2	0.625	5.74e-115

to the translations in CLLS (i.e. comparing LEXSUB-Usim-1 to CLLS-Usim-1 and also LEXSUB-Usim-2 to CLLS-Usim-2). It seems that while both overlap of paraphrases and translations give us a useful gauge of the similarity of two usages, the paraphrases gave a slightly stronger relationship. This was surprising because it was thought that finding translations might be easier for annotators (as well as machines), although the pairwise inter-annotator agreement for CLLS was only slightly higher than for LEXSUB [5]. The total number of lexical substitute and translations types for this set of lemmas was 1250 and 1217 respectively. Since these are similar we do not believe this should be a major factor in the differences in correlation.

3.3 Correlations by Lemma

We investigated whether the correlation between translation and paraphrase overlap, and paraphrase and translation overlap with Usim, holds for all lemmas. To do this we did the same calculation on all sentence pairs as in table 1, but this time for the data for one specific lemma at a time. We also provide the Usim inter-annotator agreement (IAA) for each lemma using spearman's correlation following [6] so that we can see whether the lemmas where there is a mismatch in correlation between the various annotations are also those where humans disagree on similarity between usages. Such disagreements are probably due to the inter-relationships between meanings of the different usages. In addition to this, we provide MID which is the proportion of judgements from the annotators in the intermediate points of the Usim 1-5 scale, that is 2, 3 or 4, for each lemma. This is calculated in the following way. Let $a \in A$ be an annotator from the set A of all annotators, and $j_a \in P_l$ be the judgement of annotator a for a sentence pair for a lemma from all possible such pairings for that lemma (P_l). Then the MID score is calculated as:

$$MID(lemma) = \frac{\sum_{a \in A} \sum_{j_a \in P_l} 1 \ if \ j_a \in \{2,3,4\}}{|A| \cdot |P_l|} \qquad (2)$$

These scores by lemma are shown in table 2 which is ordered alphabetically by lemma. To make it easier to see the patterns we provide table 3 which uses the scores from table 2 to sort the lemmas into rank order for each column respectively. We use the ρ coefficients for all columns and sort in ascending order except rev MID which orders the lemmas in descending order of the MID score. The reason for this is that we anticipate that lemmas with more judgements in the

Table 2. Scores by lemma. The lemmas are ordered alphabetically.

lemma	Usim round	LEXSUB-CLLS	LEXSUB-Usim	CLLS-Usim	MID	IAA
account.n	2	0.322	0.517	0.516	0.389	0.66
bar.n	1	0.583	0.615	0.620	0.296	0.35
bright.a	2	0.402	0.555	0.125	0.553	0.53
call.v	2	0.708	0.849	0.699	0.178	0.65
charge.v	1	0.640	0.618	0.320	0.244	0.68
check.v	1	0.396	0.389	0.765	0.519	0.35
clear.v	1	0.339	0.538	0.685	0.452	0.63
coach.n	2	0.875	0.695	0.836	0.269	0.74
dismiss.v	2	0.829	0.783	0.656	0.606	0.52
draw.v	1	0.352	0.498	0.266	0.526	0.50
dry.a	1	0.448	0.580	0.372	0.378	0.59
execution.n	1	0.763	0.885	0.753	0.459	0.78
field.n	1	0.712	0.449	0.358	0.474	0.25
figure.n	1	0.757	0.757	0.567	0.393	0.50
fire.v	2	0.921	0.853	0.905	0.169	0.93
fix.v	2	0.750	0.529	0.637	0.339	0.57
flat.a	1	0.719	0.559	0.775	0.435	0.85
fresh.a	1	0.506	0.390	0.409	0.756	0.17
function.n	1,2	0.106	0.781	0.063	0.533	0.14
hard.r	1	0.743	0.119	-0.065	0.637	0.34
heavy.a	1	0.357	0.385	0.555	0.600	0.57
hold.v	2	0.202	0.475	0.106	0.478	0.47
investigator.n	1,2	0.653	0.471	0.322	0.272	0.23
lead.n	2	0.058	0.027	0.666	0.493	0.47
light.a	1	0.526	0.287	0.192	0.363	0.49
match.n	1	0.677	0.743	0.755	0.326	0.59
neat.a	2	-0.110	0.622	0.077	0.581	0.31
new.a	2	0.282	0.141	-0.310	0.225	0.01
order.v	1,2	0.613	0.720	0.708	0.344	0.65
paper.n	1	0.764	0.738	0.812	0.437	0.63
poor.a	1	0.309	0.779	0.237	0.341	0.43
post.n	1	0.556	0.745	0.618	0.222	0.69
put.v	1	0.720	0.146	0.270	0.622	0.34
range.n	2	0.690	0.782	0.654	0.344	0.74
raw.a	1	0.451	0.117	0.243	0.733	0.29
rich.a	2	0.677	0.857	0.696	0.406	0.73
right.r	1	0.441	0.618	0.616	0.481	0.65
ring.n	2	0.319	0.759	0.459	0.325	0.53
rough.a	2	0.568	0.635	0.827	0.350	0.63
rude.a	1	0.224	0.299	0.846	0.533	0.61
severely.r	2	0.472	0.212	0.527	0.325	0.78
shade.n	2	0.704	0.740	0.608	0.302	0.42
shed.v	2	0.188	0.594	0.166	0.494	0.53
skip.v	2	0.417	0.666	0.504	0.381	0.70
soft.a	2	0.397	0.554	0.590	0.436	0.51
solid.a	1	0.237	0.618	0.329	0.630	0.49
special.a	1	0.107	0.202	0.368	0.704	0.37
stiff.a	1,2	0.444	0.493	0.305	0.497	0.40
strong.a	1	0.040	0.170	0.507	0.733	0.31
tap.v	1	0.578	0.725	0.786	0.452	0.70
throw.v	1	-0.183	0.290	-0.061	0.696	0.32
work.v	1	0.033	0.497	-0.054	0.637	0.27

Table 3. Lemmas ordered by the score (from table 2) in each respective column. All columns except rev MID are in ascending order, rev MID is in descending order of the MID score.

LEXSUB-CLLS	LEXSUB-Usim	CLLS-Usim	rev MID	IAA
throw.v	lead.n	new.a	fresh.a	new.a
neat.a	raw.a	hard.r	raw.a	function.n
work.v	hard.r	throw.v	strong.a	fresh.a
strong.a	new.a	work.v	special.a	investigator.n
lead.n	put.v	function.n	throw.v	field.n
function.n	strong.a	neat.a	hard.r	work.v
special.a	special.a	hold.v	work.v	raw.a
shed.v	severely.r	bright.a	solid.a	neat.a
hold.v	light.a	shed.v	put.v	strong.a
rude.a	throw.v	light.a	dismiss.v	throw.v
solid.a	rude.a	poor.a	heavy.a	hard.r
new.a	heavy.a	raw.a	neat.a	put.v
poor.a	check.v	draw.v	bright.a	bar.n
ring.n	fresh.a	put.v	function.n	check.v
account.n	field.n	stiff.a	rude.a	special.a
clear.v	investigator.n	charge.v	draw.v	stiff.a
draw.v	hold.v	investigator.n	check.v	shade.n
heavy.a	stiff.a	solid.a	stiff.a	poor.a
check.v	work.v	field.n	shed.v	hold.v
soft.a	draw.v	special.a	lead.n	lead.n
bright.a	account.n	dry.a	right.r	light.a
skip.v	fix.v	fresh.a	hold.v	solid.a
right.r	clear.v	ring.n	field.n	draw.v
stiff.a	soft.a	skip.v	execution.n	figure.n
dry.a	bright.a	strong.a	clear.v	soft.a
raw.a	flat.a	account.n	tap.v	dismiss.v
severely.r	dry.a	severely.r	paper.n	bright.a
fresh.a	shed.v	heavy.a	soft.a	ring.n
light.a	bar.n	figure.n	flat.a	shed.v
post.n	charge.v	soft.a	rich.a	fix.v
rough.a	right.r	shade.n	figure.n	heavy.a
tap.v	solid.a	right.r	account.n	dry.a
bar.n	neat.a	post.n	skip.v	match.n
order.v	rough.a	bar.n	dry.a	rude.a
charge.v	skip.v	fix.v	light.a	clear.v
investigator.n	coach.n	range.n	rough.a	paper.n
match.n	order.v	dismiss.v	order.v	rough.a
rich.a	tap.v	lead.n	range.n	call.v
range.n	paper.n	clear.v	poor.a	order.v
shade.n	shade.n	rich.a	fix.v	right.r
call.v	match.n	call.v	match.n	account.n
field.n	post.n	order.v	ring.n	charge.v
flat.a	figure.n	execution.n	severely.r	post.n
put.v	ring.n	match.n	shade.n	skip.v
hard.r	poor.a	check.v	bar.n	tap.v
fix.v	function.n	flat.a	investigator.n	rich.a
figure.n	range.n	tap.v	coach.n	coach.n
execution.n	dismiss.v	paper.n	charge.v	range.n
paper.n	call.v	rough.a	new.a	execution.n
dismiss.v	fire.v	coach.n	post.n	severely.r
coach.n	rich.a	rude.a	call.v	flat.a
fire.v	execution.n	fire.v	fire.v	fire.v

middle range (2, 3 and 4) are more likely to be those with lower correspondence between paraphrases, translations and usage similarity.

From table 3, we observe a tendency for lemmas where the relationship between paraphrases and translations (LEXSUB-CLLS) is weak or absent to also have lower IAA and higher average MID scores. The same tendency to low IAA and higher MID scores occurs where the relationship between paraphrases or translations and usage similarity (LEXSUB-Usim and CLLS-Usim) breaks down. These trends are highlighted by the position of the lemmas in the columns in table 3. These are of course only tendencies as factors other than the meaning of the usage affect the translations and paraphrases available for a lemma. Nevertheless, there is a striking tendency that words where the different meanings are interrelated (*work.v* and *special.a*) tend to occur in the early rows in all (or most) columns whereas words where the meanings are more easily distinguished tend to occur in the final section (e.g. *coach.n* and *fire.v*). To measure this relationship, we calculated spearman's ρ between i) the MID scores or ρ values for LEXSUB-CLLS, LEXSUB-Usim and CLLS-Usim by lemma and ii) the IAA. The results are shown in table 4. The IAA–MID correlation is negative because lower values of MID tend to occur for lemmas with higher IAA. The ρ values support the finding we observe from looking at the positions of the lemmas in the columns of table 3. All columns show a correlation with IAA, but the extent that a mismatch between translation overlap and Usim judgements provides the strongest indicator of poor IAA for that lemma.

Table 4. Correlation of score by lemma with IAA for that lemma

score	ρ	p-value
LEXSUB-CLLS	0.424	0.0017
LEXSUB-Usim	0.526	6.177e-05
CLLS-Usim	0.670	5.558e-08
MID	-0.486	0.00026

3.4 Discussion

Our results support the view that lexical translations and substitutes, where they exist, can be used as a representation of word meaning. Though of course the correspondence in substitute or translation overlap between sentences may be coincidental, on the whole an overlap in meaning typically produces an overlap in lexical paraphrase or translation. Although the correspondence is not evident for all lemmas, it seems from our analysis that when translations and paraphrase overlap does not concur with usage similarity, this is often because the meanings of the word is inter-related which is reflected by lower IAA on Usim. The MID score is also a useful indicator of IAA as it is simple to obtain. One could apply this MID score in cases where the annotators had been asked to annotate different lemmas to reduce annotation effort. One could then put more effort into sense induction and disambiguation of lemmas with lower MID values on the grounds that their senses are distinguished more easily.

4 Conclusion

We have examined the CLLS and LEXSUB datasets and found a highly significant correlation between the overlap of translations and the overlap of substitutes of the same lemma for the same sentence pairings. We have also found a highly significant correlation between both of these measures and usage similarity (Usim) judgements. This relationship is not evident across the board however, with lemmas such as *fire.v* displaying an extremely strong correlation while lemmas with more relationships between different usages, such as *special.a*, with very low correlation. These correlations scores on a lemma-by-lemma basis are themselves correlated with IAA of the lemma. The lower the IAA, the lower the correlation between LEXSUB and CLLS overlap, and LEXSUB or CLLS overlap with Usim judgements. There is also a negative correlation with the number of usage similarity judgements in the mid range (2, 3 or 4) of a 1-5 scale and the IAA for a lemma. This suggests that scores in the MID range may be useful for predicting low IAA. This may help in identifying lemmas with distinct usages and those with inter-related usages which may prove problematic for word sense induction and disambiguation.

There are many directions we anticipate for future work, in particular the use of these datasets for comparing fully unsupervised representations of lexical meaning. The standard use of LEXSUB and CLLS is for evaluating lexical paraphrases and translations, but since the overlap in paraphrases and translations from these sets correlate with usage similarity judgements (in Usim) one could use overlap of substitutes and translations in full LEXSUB and CLLS datasets to evaluate a system that estimates the similarity of two usages. Another direction we hope to investigate is the extent that the paraphrases, and translations can be clustered (see [5] and [24]).

Acknowledgments

I thank the following collaborators for their work in creating the datasets used in this analysis, without whose help and ideas none of this work would have been possible. These are Roberto Navigli, for his collaboration on LEXSUB, Katrin Erk and Nick Gaylord for their collaboration on Usim and related datasets which have inspired this analysis and Rada Mihalcea and Ravi Sinha on CLLS. I'd also like to thank Huw McCarthy for the interface for the second round of Usim.

References

1. Fellbaum, C. (ed.): WordNet, An Electronic Lexical Database. The MIT Press, Cambridge (1998)
2. Schütze, H.: Automatic word sense discrimination. Computational Linguistics 24, 97–123 (1998)
3. Pantel, P., Lin, D.: Discovering word senses from text. In: Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Edmonton, Canada, pp. 613–619 (2002)

4. McCarthy, D., Navigli, R.: SemEval-2007 task 10: English lexical substitution task. In: Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval 2007), Prague, Czech Republic, pp. 48–53 (2007)
5. Mihalcea, R., Sinha, R., McCarthy, D.: Semeval-2010 task 2: Cross-lingual lexical substitution. In: Proceedings of the 5th International Workshop on Semantic Evaluation, pp. 9–14. Association for Computational Linguistics, Uppsala (2010)
6. Erk, K., McCarthy, D., Gaylord, N.: Investigations on word senses and word usages. In: Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing. Association for Computational Linguistics, Suntec (2009)
7. Kilgarriff, A.: Word senses. In: Agirre, E., Edmonds, P. (eds.) Word Sense Disambiguation, Algorithms and Applications, pp. 29–46. Springer, Heidelberg (2006)
8. Resnik, P.: Selection and Information: A Class-Based Approach to Lexical Relationships. PhD thesis, University of Pennsylvania (1993)
9. Sanderson, M.: Word sense disambiguation and information retrieval. In: 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 142–151. ACM Press, New York (1994)
10. Carpuat, M., Wu, D.: Word sense disambiguation vs. statistical machine translation. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005). Association for Computational Linguistics, Ann Arbor (2005)
11. Carpuat, M., Wu, D.: Improving statistical machine translation using word sense disambiguation. In: Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007), pp. 61–72. Association for Computational Linguistics, Prague (2007)
12. Resnik, P.: WSD in NLP applications. In: Agirre, E., Edmonds, P. (eds.) Word Sense Disambiguation, Algorithms and Applications, pp. 299–337. Springer, Heidelberg (2006)
13. Clough, P., Stevenson, M.: Evaluating the contribution of eurowordnet and word sense disambiguation to cross-language retrieval. In: Second International Global WordNet Conference (GWC 2004), pp. 97–105 (2004)
14. Ide, N., Wilks, Y.: Making sense about sense. In: Agirre, E., Edmonds, P. (eds.) Word Sense Disambiguation, Algorithms and Applications, pp. 47–73. Springer, Heidelberg (2006)
15. Hovy, E., Marcus, M., Palmer, M., Ramshaw, L., Weischedel, R.: Ontonotes: The 90% solution. In: Proceedings of the HLT-NAACL 2006 Workshop on Learning Word Meaning from Non-linguistic Data. Association for Computational Linguistics, New York City (2006)
16. Navigli, R., Litkowski, K.C., Hargraves, O.: SemEval-2007 task 7: Coarse-grained english all-words task. In: Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval 2007), Prague, Czech Republic, pp. 30–35 (2007)
17. Navigli, R.: Meaningful clustering of senses helps boost word sense disambiguation performance. In: Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics Joint with the 21st International Conference on Computational Linguistics (COLING-ACL 2006), Sydney, Australia, pp. 105–112 (2006)

18. Stokoe, C.: Differentiating homonymy and polysemy in information retrieval. In: Proceedings of the Joint Conference on Human Language Technology and Empirical methods in Natural Language Processing, Vancouver, B.C., Canada, pp. 403–410 (2005)
19. McCarthy, D.: Relating wordnet senses for word sense disambiguation. In: Proceedings of the EACL 2006 Workshop: Making Sense of Sense: Bringing Psycholinguistics and Computational Linguistics Together, Trento, Italy, pp. 17–24 (2006)
20. Sharoff, S.: Open-source corpora: Using the net to fish for linguistic data. International Journal of Corpus Linguistics 11, 435–462 (2006)
21. McCarthy, D., Navigli, R.: The English lexical substitution task. In: Language Resources and Evaluation Special Issue on Computational Semantic Analysis of Language: SemEval-2007 and Beyond, vol. 43(2), pp. 139–159 (2009)
22. Ng, H.T., Chan, Y.S.: SemEval-2007 task 11: English lexical sample task via english-chinese parallel text. In: Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval 2007), Prague, Czech Republic, pp. 54–58 (2007)
23. Mitchell, J., Lapata, M.: Vector-based models of semantic composition. In: Proceedings of ACL 2008: HLT, pp. 236–244. Association for Computational Linguistics, Columbus (2008)
24. Lefever, E., Hoste, V.: SemEval-2007 task 3: Cross-lingual word sense disambiguation. In: Proceedings of the 5th International Workshop on Semantic Evaluations (SemEval 2010), Uppsala, Sweden (2010)

A Quantitative Evaluation of Global Word Sense Induction

Marianna Apidianaki and Tim Van de Cruys

Alpage, INRIA & University Paris 7
Domaine de Voluceau
Rocquencourt, B.P. 105
F-78153 Le Chesnay cedex
{Marianna.Apidianaki,Tim.Van_de_Cruys}@inria.fr

Abstract. Word sense induction (WSI) is the task aimed at automatically identifying the senses of words in texts, without the need for hand-crafted resources or annotated data. Up till now, most WSI algorithms extract the different senses of a word 'locally' on a per-word basis, i.e. the different senses for each word are determined separately. In this paper, we compare the performance of such algorithms to an algorithm that uses a 'global' approach, i.e. the different senses of a particular word are determined by comparing them to, and demarcating them from, the senses of other words in a full-blown word space model. We adopt the evaluation framework proposed in the SemEval-2010 Word Sense Induction & Disambiguation task. All systems that participated in this task use a local scheme for determining the different senses of a word. We compare their results to the ones obtained by the global approach, and discuss the advantages and weaknesses of both approaches.

1 Introduction

Word sense induction (WSI) methods automatically identify the senses of words in texts, without the need for predefined resources or annotated data. These methods offer an alternative to the use of expensive hand-crafted resources developed according to the 'fixed list of senses' paradigm, which present several drawbacks for efficient semantic processing [1]. The assumption underlying unsupervised WSI methods is the distributional hypothesis of meaning [2], according to which words that occur in similar contexts tend to be similar. In distributional semantic analysis, the co-occurrences of words in texts constitute the features that serve to calculate their similarity. Following this approach, data-driven WSI algorithms calculate the similarity of the contexts of polysemous target words and group them into clusters. The resulting clusters describe the target word senses.

The unsupervised algorithms used for WSI can be distinguished into *local* and *global*. Local algorithms work on a per-word basis, determining the senses for each word separately. Algorithms that use a global approach determine the different senses of a particular word by comparing them to, and demarcating them from, the senses of other words in a full-blown word space model.

In this paper, we compare the performance of these two types of algorithms for sense induction. The comparison is carried out using the evaluation framework proposed in the SemEval-2010 Word Sense Induction & Disambiguation (WSI&D) task [3,4]. The SemEval WSI tasks [4,5] provide a common ground for comparison and evaluation of different sense induction and discrimination systems. All the systems that participated in the SemEval-2010 WSI&D task use a local scheme for determining the different senses of a word. We compare their results to the ones obtained by the global approach, and discuss the advantages and weaknesses of both approaches.

The paper is organized as follows. We first explain how word senses are identified in the local and the global approaches to sense induction, and we present the global algorithm used in our research. Section 3 describes the evaluation setting that we adopt and the metrics that will be used in order to evaluate the performance of the algorithms. In Section 4, we present the evaluation results of the global approach, and compare them to the results obtained by the local systems that participated in the SemEval-2010 WSI&D task. Our last section draws conclusions, and lays out some avenues for future work.

2 WSI Algorithms

2.1 Inducing Word Senses on a Per-word Basis

Local methods to word sense induction discover the senses of a target word (w) by clustering its instances in texts according to their semantic similarity. Following the distributional hypothesis of meaning, words that are used in similar contexts carry similar meanings [2,6]. So, the instances of w that appear in similar contexts are considered as semantically similar and its senses can be discovered by clustering its contexts [7].

The features used for calculating the similarity of the instances of w are their co-occurrences in a fixed-sized window of text. So, the different instances of w in a corpus can be represented by feature vectors created from their contexts [8,9]. The grouping of the context vectors according to their similarity generates a number of clusters that describe the different senses of w. The context of w may be taken into account in different ways : it may be modeled as a first-order context vector, representing the direct context of the instances of w in the corpus [10,11], or by using higher-order vectors, i.e. by considering the context vectors of the words occurring in the target context [8].

Other methods use the words found in the context of target words in order to construct co-occurrence graphs. In a graph of this type, the vertices correspond to the words appearing in the contexts of the target words and the edges represent their relations. These relations may be grammatical [12] or they may be co-occurrences of the words in fixed contexts [13,14]. The senses of the target words are discovered by partitioning the co-occurrence graph using clustering techniques, or by using a PageRank algorithm.

2.2 Global Approach to Sense Induction

In contrast to the local approach to sense induction, where senses are discovered by clustering contexts for each word individually, the global approach discovers senses by clustering semantically similar senses of words in a global manner, comparing them and demarcating them from the senses of other words in a full-blown word space model. The similarity between the senses is calculated on the basis of their common features, e.g. the syntactic dependencies a particular sense occurs with [15].

In Pantel and Lin's [16] method, the similarity of word senses is calculated on the basis of the dependency relations in which the senses take part (extracted from a syntactically annotated corpus). Each word is represented by a feature vector, where each feature corresponds to a syntactic context (dependency triple) in which the word occurs. Each feature is weighted and its value corresponds to the pointwise mutual information between the feature and the word. The algorithm first discovers a set of tight clusters (called 'committees') in the similarity space. Each word is then assigned to the closest committee by comparing the word's feature vector to the centroid of a committee (i.e. the mean of the feature vectors of the committee members). After a word is assigned to a particular committee, the overlapping features are deleted from the word's vector, which allows for the discovery of less dominant senses. Each cluster that a word belongs to describes one of its senses.

2.3 Non-negative Matrix Factorization for Sense Induction

Sense induction. The global algorithm implemented here is based on the one proposed by Van de Cruys [17]. This algorithm creates semantic word models by using an extension of non-negative matrix factorization (NMF) [18], that combines both the bag of words approach and the syntax-based approach to sense induction. The intuition in this is that the syntactic features of the syntax-based approach can be disambiguated by the semantic dimensions found by the bag of words approach. The algorithm finds a small number of latent semantic dimensions, according to which nouns, contexts and syntactic relations are classified.

Nouns are classified according to both bag-of-words context and syntactic context, so three matrices are constructed that capture the co-occurrence frequency information for each mode. The first matrix contains co-occurrence frequencies of nouns cross-classified by dependency relations, the second matrix contains co-occurrence frequencies of nouns cross-classified by words that appear in the noun's context window, and the third matrix contains co-occurrence frequencies of dependency relations cross-classified by co-occurring context words. NMF is then applied to the three matrices and the separate factorizations are interleaved (i.e. the results of the former factorization are used to initialize the factorization of the next matrix). A graphical representation of the interleaved factorization algorithm is given in figure 1.

When the factorization is finished, the three different modes (nouns, bag-of-words context words and syntactic relations) are all represented as a limited number of semantic dimensions.

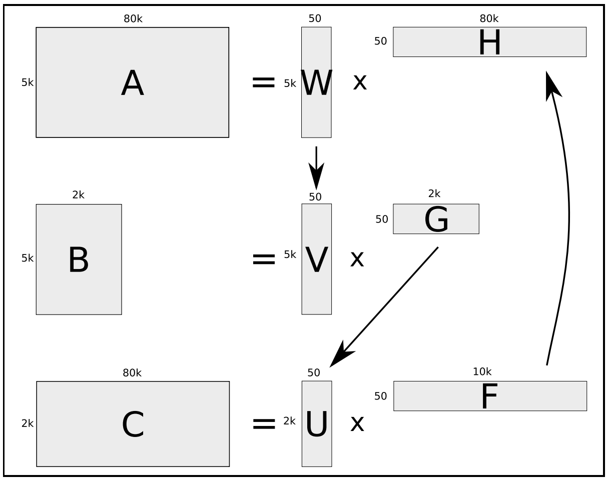


Fig. 1. A graphical representation of the extended NMF

Next, the factorization that is thus created is used for word sense induction. The intuition is that a particular dimension of an ambiguous word is ‘switched off’, to reveal possible other senses of the word. Matrix H indicates the importance of each syntactic relation given a semantic dimension. With this knowledge, the syntactic relations that are responsible for a certain dimension can be subtracted from the original noun vector. This is done by scaling down each feature of the original vector according to the load of the feature on the subtracted dimension.

The last step is to determine which dimension(s) are responsible for a certain sense of the word. In order to do so, the method is embedded in a clustering approach. First, a specific word is assigned to its predominant sense (i.e. the most similar cluster). Next, the dominant semantic dimension(s) for this cluster are subtracted from the word vector, and the resulting vector is fed to the clustering algorithm again, to see if other word senses emerge. The dominant semantic dimension(s) can be identified by ‘folding in’ the cluster centroid into the factorization.

A simple k-means algorithm is used to compute the initial clustering. k-means yields a hard clustering, in which each noun is assigned to exactly one (dominant) cluster. In the second step, it is determined for each noun whether it can be assigned to other, less dominant clusters. First, the salient dimension(s) of the centroid to which the noun is assigned are determined. The centroid of the cluster is computed by averaging the frequencies of all cluster elements except for the target word we want to reassign, and weighting the resulting vector with pointwise mutual information [19]. After subtracting the salient dimensions from the noun vector, it is checked whether the vector is reassigned to another cluster centroid. If this is the case, (another instance of) the noun is assigned to the cluster, and the second step is repeated. If there is no reassignment, we continue with the next word. The target element is removed from the centroid to make sure that only the dimensions associated with the sense of the cluster are subtracted. When the algorithm is finished, each noun is assigned to a number of clusters, representing its different senses.

We use two different methods for selecting the final number of candidate senses. The first method, NMF_{con}, takes a conservative approach, and only selects candidate senses if – after the subtraction of salient dimensions – another sense is found that is more similar to the adapted noun vector. The second method, NMF_{lib}, is more liberal, and also selects the next best cluster centroid as candidate sense until a certain similarity threshold ϕ is reached. Experimentally (examining the cluster output), we set $\phi = 0.2$.

Sense disambiguation. The sense inventory that results from the induction step can now be used for the disambiguation of individual instances as follows. For each instance of the target noun, we extract its context words, i.e. the words that co-occur in the same paragraph, and represent them as a frequency vector. Using matrix G from our factorization model (which represents context words by semantic dimensions), this co-occurrence vector can be 'fold in' into the semantic space, thus representing the probability of each semantic dimension for the particular instance of the target noun. Likewise, the candidate senses of the noun (represented as centroids) can be folded into our semantic space using matrix H, which represents the dependency relations by semantic dimensions. This yields a probability distribution over the semantic dimensions for each centroid. As a last step, we compute the Kullback-Leibler divergence between the context vector and the candidate centroids, and select the candidate centroid that yields the lowest divergence as the correct sense.

Example. Let us clarify the process with an example for the noun *chip*. The sense induction algorithm finds the following candidate senses:

1. *cache*, *CPU*, *memory*, *microprocessor*, *processor*, *RAM*, *register*
2. *bread*, *cake*, *chocolate*, *cookie*, *recipe*, *sandwich*
3. *accessory*, *equipment*, *goods*, *item*, *machinery*, *material*, *product*, *supplies*

Each candidate sense is associated with a centroid (the average frequency vector of its members), that is fold into the semantic space, which yields a 'semantic fingerprint', i.e. a distribution over the semantic dimensions. For the first sense, the 'computer' dimension will be the most important. Likewise, for the second and the third sense the 'food' dimension and the 'manufacturing' dimension will be the most important.[1]

Let us now take a particular instance of the noun *chip*, such as the one in (1).

(1) An N.V. Philips **unit** has **created** a **computer system** that **processes video images** 3,000 times faster than conventional **systems**. Using **reduced instruction - set computing**, or RISC, chips made by Intergraph of Huntsville, Ala., the **system** splits the **image** it '**sees**' into 20 **digital representations**, each **processed** by one *chip*.

[1] In the majority of cases, the induced dimensions indeed contain such clear-cut semantics, so that the dimensions can be rightfully labeled as above.

Looking at the context of the particular instance of *chip*, a context vector is created which represents the semantic content words that appear in the same paragraph (the extracted content words are printed in boldface). This context vector is again folded into the semantic space, yielding a distribution over the semantic dimensions. By selecting the lowest Kullback-Leibler divergence between the semantic probability distribution of the target instance and the semantic probability distributions of the candidate senses, the algorithm is able to induce the 'computer' sense of the target noun *chip*.

Implementational details. The SemEval training set has been part of speech tagged and lemmatized with Stanford Part-Of-Speech Tagger [20,21], and parsed with MaltParser [22] trained on sections 2-21 of the Wall Street Journal section of the Penn Treebank extended with about 4000 questions from the QuestionBank[2] in order to extract dependency triples. The SemEval test set has only been tagged and lemmatized, as our disambiguation model did not use dependency triples as features (contrary to our induction model).

The three matrices needed for our factorization model were constructed using the 5K nouns, 80K dependency relations, and 2K context words (excluding stop words) with highest frequency in the training set, which yields matrices of 5K nouns $\times$ 80K dependency relations, 5K nouns $\times$ 2K context words, and 80K dependency relations $\times$ 2K context words. For our initial k-means clustering, we cluster the 5K nouns into 600 clusters.

The sense induction and disambiguation algorithms were implemented in Python. The interleaved NMF factorization model itself was implemented in Matlab, using 50 iterations, and factorizing the model to 50 dimensions.

3 Word Sense Induction Evaluation in SemEval 2010

3.1 Training and Evaluation Datasets

Our WSI algorithm is trained and tested on the dataset of the SemEval-2010 WSI&D task [4]. The main difference of this task from the SemEval-2007 WSI task [5] is that the training and testing data are treated separately, which allows for a more realistic evaluation of the clustering models. Word senses are induced from the training data while testing data are used for tagging new instances of the words with the previously discovered senses.

The SemEval-2010 WSI&D task is based on a dataset of 100 target words, 50 nouns and 50 verbs. For each target word, a *training* set is provided from which the senses of the word have to be induced without using any other resources. The training set for a target word consists of a set of target word instances in context (sentences or paragraphs). In this paper, we will focus on the set of nouns, that consists of 716,945 instances.

The senses induced during training are used for disambiguation in the *testing* phase. In this phase, the systems are provided with a testing dataset that consists

[2] `http://maltparser.org/mco/english_parser/engmalt.html`

of unseen instances of the target words. The testset comprises 5,285 noun instances. The instances in the testset are tagged with OntoNotes senses [23]. The systems need to disambiguate these instances using the senses acquired during training.

3.2 Supervised and Unsupervised Evaluation

The results of the systems participating in the SemEval-2010 WSI&D task are evaluated both in a supervised and in an unsupervised manner. In the *supervised* evaluation, one part of the testing dataset is used as a *mapping* corpus, which serves to map the automatically induced clusters to gold standard (GS) senses, and the other part as an *evaluation* corpus, used to evaluate the methods in a standard WSD task. The mapping between clusters and GS senses serves to tag the evaluation corpus with GS tags.

In the *unsupervised* evaluation, the induced senses are evaluated as clusters of examples (*tw* contexts) which are compared to the sets of examples tagged with the GS senses (corresponding to classes). So, if the testing dataset of a *tw* comprises a number of instances, these are divided into two partitions : a set of automatically generated clusters and a set of GS classes. A number of these instances will be members of both one GS class and one cluster. Consequently, the quality of the proposed clustering solution is evaluated by comparing the two groupings and measuring their similarity.

3.3 Evaluation Measures

The supervised evaluation in the SemEval-2010 WSI&D task follows the scheme employed in the SemEval-2007 WSI task [5], with some modifications. The induced senses (clusters) are mapped to GS senses using a mapping corpus, which is a part of the testing sense-tagged dataset. Then, the evaluation corpus, which corresponds to the rest of the testing dataset, is used to evaluate WSI methods in a standard WSD task. The evaluation is performed according to the precision and recall measures employed for the evaluation of supervised WSD systems.

Two evaluation metrics are employed during the unsupervised evaluation in order to estimate the quality of the clustering solutions, the *V-measure* [24] and the *paired F-Score* [25]. *V-Measure* assesses the quality of a clustering by measuring its *homogeneity* (h) and its *completeness* (c). Homogeneity refers to the degree that each cluster consists of data points primarily belonging to a single GS class, while completeness refers to the degree that each GS class consists of data points primarily assigned to a single cluster. V-Measure is the harmonic mean of h and c.

$$VM = \frac{2 \cdot h \cdot c}{h + c} \tag{1}$$

In the *paired F-Score* [25] evaluation, the clustering problem is transformed into a classification problem [4]. A set of instance pairs is generated from the automatically induced clusters (*F(K)*), which comprises pairs of the instances found in each cluster. Similarly, a set of instance pairs is created from the GS classes

(*F(S)*), containing pairs of the instances found in each class. *Precision* is then defined as the number of common instance pairs between the two sets to the total number of pairs in the clustering solution (cf. formula 2). *Recall* is defined as the number of common instance pairs between the two sets to the total number of pairs in the GS (cf. formula 3). Precision and recall are finally combined to produce the harmonic mean (cf. formula 4).

$$P = \frac{|F(K) \cap F(S)|}{|F(K)|} \tag{2}$$

$$R = \frac{|F(K) \cap F(S)|}{|F(S)|} \tag{3}$$

$$FS = \frac{2 \cdot P \cdot R}{P + R} \tag{4}$$

The obtained results are also compared to two baselines. The Most Frequent Sense (*MFS*) baseline groups all testing instances of a *tw* into one cluster. The *Random* baseline randomly assigns an instance to one of the clusters.[3]. This baseline is executed five times and the results are averaged.

4 Evaluation Results

4.1 Unsupervised Evaluation

In table 1, we present the performance of a number of algorithms on the V-measure. We compare our V-measure scores with the scores of the best-ranked systems in the SemEval 2010 WSI&D task. The second column shows the number of clusters induced in the test set by each algorithm. The *MFS* baseline has a V-Measure equal to 0, since by definition its completeness is 1 and homogeneity is 0.

Table 1. V-measure for SemEval noun testset

	VM (%)	**#Cl**
UoY	20.6	11.54
Hermit	16.7	10.78
KSU KDD	18.0	17.5
NMF$_{lib}$	13.5	5.42
Duluth-WSI	11.4	4.15
Random	4.2	4.00
NMF$_{con}$	3.9	1.58
MFS	0.0	1.00

[3] The number of clusters of *Random* was chosen to be roughly equal to the average number of senses in the GS.

NMF$_{con}$ – our model that takes a conservative approach in the induction of candidate senses – does not beat the random baseline. NMF$_{lib}$ – our model that is more liberal in inducing senses – reaches better results. With 13.5%, it scores similar to other algorithms that induce a similar average number of clusters, such as Duluth-WSI [26].

Pedersen [26] has shown that the V-Measure tends to favour systems producing a higher number of clusters than the number of GS senses. This is reflected in the scores of our models as well.

In table 2, the paired F-Score of a number of algorithms is given. The paired F-Score penalizes systems when they produce a higher number of clusters (low recall) or a lower number of clusters (low precision) than the GS number of senses. We again compare our results with the scores of the best-ranked systems in the SemEval 2010 WSI&D task.

Table 2. Paired F-score for SemEval noun testset

	FS (%)	**#Cl**
MFS	57.0	1.00
Duluth-WSI-SVD-Gap	57.0	1.02
NMF$_{con}$	54.6	1.58
NMF$_{lib}$	42.2	5.42
Duluth-WSI	37.1	4.15
Random	30.4	4.00

NMF$_{con}$ reaches a score of 54.6%, which is again similar to other algorithms that induce the same average number clusters. NMF$_{lib}$ scores 42.2%, indicating that the algorithm is able to retain a reasonable F-Score while at the same time inducing a significant number of clusters. This especially becomes clear when comparing its score to the other algorithms.

4.2 Supervised Evaluation

Table 3 shows the recall of our algorithms in the supervised evaluation, again compared to other algorithms evaluated in the SemEval 2010 WSI&D task.

Table 3. Supervised recall for SemEval noun testset, 80% mapping, 20% evaluation

	SR (%)	**#S**
UoY	59.4	1.51
NMF$_{lib}$	57.3	1.93
Duluth-WSI	54.7	1.66
NMF$_{con}$	54.5	1.21
MFS	53.2	1.00
Random	51.5	1.53

NMF$_{lib}$ gets 57.3% and NMF$_{con}$ reaches 54.5%, which again indicates that our algorithm is in the same ballpark as other algorithms that induce a similar average number of senses.

5 Conclusion and Future Work

In this paper, we presented a quantitative evaluation of a global approach to word sense induction, and compared it to more prevailing local approaches to word sense induction, that induce senses on a per-word basis. The results indicate that the global approach performs equally well, reaching similar results to the state-of-the-art performance of local approaches. Moreover, the global approach is able to reach similar performance on an evaluation set that is tuned to fit the needs of local approaches. The evaluation set contains an enormous amount of contexts for only a small number of target words, favouring methods that induce senses on a per-word basis. The global approach is likely to induce a more balanced sense inventory using a more balanced, unbiased corpus, and is likely to outperform local methods when such an unbiased corpus is used as input. We therefore think that a global approach to word sense induction, such as the one presented here, provides a genuine and powerful solution to the problem at hand, and deserves further attention.

We conclude with some issues for future work. First of all, we would like to evaluate the approach presented here using a more balanced an unbiased corpus, and compare its performance on such a corpus to local approaches. Secondly, we would also like to include grammatical dependency information in the disambiguation step of the algorithm. For now, the disambiguation step only uses a word's context words; enriching the feature set with dependency information is likely to improve the performance of the disambiguation.

Acknowledgments

This work is supported by the Scribo project, funded by the French "pôle de compétitivité" System@tic, and by the French national grant EDyLex (ANR-09-CORD-008).

References

1. Ide, N., Wilks, Y.: Making sense about sense. In: Agirre, E., Edmonds, P. (eds.) Word Sense Disambiguation, Algorithms and Applications, pp. 47–73. Springer, Heidelberg (2007)
2. Harris, Z.: Distributional structure. Word, 146–162 (1954)
3. Manandhar, S., Klapaftis, I.P.: Semeval-2010 task 14: Evaluation setting for word sense induction & disambiguation systems. In: Proceedings of the NAACL HLT Workshop on Semantic Evaluations: Recent Achievements and Future Directions, Boulder, Colorado, pp. 117–122 (2009)

4. Manandhar, S., Klapaftis, I.P., Dligach, D., Pradhan, S.: Semeval-2010 task 14: Word sense induction &disambiguation. In: Proceedings of the 5th International Workshop on Semantic Evaluation, ACL 2010, Uppsala, Sweden, pp. 63–68 (2010)
5. Agirre, E., Soroa, A.: Semeval-2007 task 02: Evaluating word sense induction and discrimination systems. In: Proceedings of the 4th International Workshop on Semantic Evaluations, pp. 7–12. ACL, Prague (2007)
6. Miller, G., Charles, W.: Contextual correlates of semantic similarity. Language and Cognitive Processes 6, 1–28 (1991)
7. Navigli, R.: Word sense disambiguation: a survey. ACM Computing Surveys 41, 1–69 (2009)
8. Schütze, H.: Automatic word sense discrimination. Computational Linguistics 24, 97–123 (1998)
9. Purandare, A., Pedersen, T.: Word sense discrimination by clustering contexts in vector and similarity spaces. In: Proceedings of the Conference on Computational Natural Language Learning (CONLL), Boston, MA, pp. 41–48 (2004)
10. Pedersen, T., Bruce, R.: Distinguishing word senses in untagged text. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Providence, RI, pp. 197–207 (1997)
11. Bordag, S.: Word sense induction: Triplet-based clustering and automatic evaluation. In: Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL), Trento, Italy, pp. 137–144 (2006)
12. Widdows, D., Dorow, B.: A graph model for unsupervised lexical acquisition. In: Proceedings of the 19th International Conference on Computational Linguistics (COLING), Taipei, Taiwan, pp. 1093–1099 (2002)
13. Véronis, J.: Hyperlex: lexical cartography for information retrieval. Computer Speech & Language 18, 223–252 (2004)
14. Agirre, E., Martínez, D., de Lacalle, O.L., Soroa, A.: Two graph-based algorithms for state-of-the-art wsd. In: Proceedings of the Empirical Methods in Natural Language Processing (EMNLP) Conference, Sydney, Australia, pp. 585–593 (2006)
15. Lin, D.: Automatic retrieval and clustering of similar words. In: Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL 1998), Montreal, Quebec, Canada, vol. 2, pp. 768–774 (1998)
16. Pantel, P., Lin, D.: Discovering word senses from text. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Edmonton, Alberta, Canada, pp. 613–619 (2002)
17. Van de Cruys, T.: Using three way data for word sense discrimination. In: Proceedings of the 22nd International Conference on Computational Linguistics (COLING), Manchester, pp. 929–936 (2008)
18. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: Advances in Neural Information Processing Systems, pp. 556–562 (2000)
19. Church, K.W., Hanks, P.: Word association norms, mutual information & lexicography. Computational Linguistics 16, 22–29 (1990)
20. Toutanova, K., Manning, C.D.: Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In: Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC), pp. 63–70 (2000)
21. Toutanova, K., Klein, D., Manning, C., Singer, Y.: Feature-rich part-of-speech tagging with a cyclic dependency network. In: Proceedings of the Human Language Technology / North American Association for Computational Linguistics conference (HLT-NAACL), pp. 252–259 (2003)

22. Nivre, J., Hall, J., Nilsson, J.: Maltparser: A data-driven parser-generator for dependency parsing. In: Proceedings of the Language Resources and Evaluation Conference (LREC), Genoa, Italy, pp. 2216–2219 (2006)
23. Hovy, E., Marcus, M., Palmer, M., Ramshaw, L., Weischedel, R.: Ontonotes: the 90% solution. In: Proceedings of the Human Language Technology / North American Association for Computational Linguistics conference (HLT-NAACL), Companion Volume: Short Papers on XX, New York, NY, pp. 57–60 (2006)
24. Rosenberg, A., Hirschberg, J.: V-measure: A conditional entropy-based external cluster evaluation measure. In: Proceedings of the Joint 2007 Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), Prague, Czech Republic, pp. 410–420 (2007)
25. Artiles, J., Amigó, E., Gonzalo, J.: The role of named entities in web people search. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 534–542 (2009)
26. Pedersen, T.: Duluth-wsi: Senseclusters applied to the sense induction task of semeval-2. In: Proceedings of the 5th International Workshop on Semantic Evaluation, pp. 363–366. Association for Computational Linguistics, Uppsala (2010)

Incorporating Coreference Resolution into Word Sense Disambiguation

Shangfeng Hu and Chengfei Liu

Faculty of Information and Communication Technologies
Swinburne University of Techonolgy,
Hawthorn, VIC 3122, Australia
{shu,cliu}@groupwise.swin.edu.au

Abstract. Word sense disambiguation (WSD) and coreference resolution are two fundamental tasks for natural language processing. Unfortunately, they are seldom studied together. In this paper, we propose to incorporate the coreference resolution technique into a word sense disambiguation system for improving disambiguation precision. Our work is based on the existing instance knowledge network (IKN) based approach for WSD. With the help of coreference resolution, we are able to connect related candidate dependency graphs at the candidate level and similarly the related instance graph patterns at the instance level in IKN together. Consequently, the contexts which can be considered for WSD are expanded and precision for WSD is improved. Based on Senseval-3 all-words task, we run extensive experiments by following the same experimental approach as the IKN based WSD. It turns out that each combined algorithm between the extended IKN WSD algorithm and one of the best five existing algorithms consistently outperforms the corresponding combined algorithm between the IKN WSD algorithm and the existing algorithm.

Keywords: Word sense disambiguation, coreference resolution, natural language processing.

1 Introduction

Word sense disambiguation (WSD) is one of the core research topics of natural language processing for identifying which sense of a word is used in a sentence, when the word has multiple meanings. It remains as an open problem in natural language processing and has important applications in areas such as machine translation, knowledge acquisition, and information retrieval [1].

Supervised WSD approaches provide the state of the art performances in benchmark evaluation [2]. Decadt et al. [3] proposed a memory-based approach which provides the best performance in senseval-3 all word tasks. Unsupervised WSD approaches were also proposed because manual supervision is a cost heavy task. Some WSD systems are built on lexical knowledge base [4-9]. Navigli [20] also proposed an integration of a knowledge-based system to improve supervised systems. They explore and calculate the semantic relationships between concepts in semantic networks [4, 5]. Some of them are graph based approaches [6-9]. The simi-supervised approach [10] shows the potential of getting better WSD results by a relative small manual training set. In this paper, we focus on the discussion of the supervised WSD approaches.

Most WSD algorithms use fixed size windows for word collections. Many approaches [10, 11] are based on collocations [12]. The collocations are identified by a sliding window. The relations between words are always simplified as whether appearing in the collection window or whether appearing in some particular position in the window. The size of windows is always fixed. Since these approaches ignore the details of many relations between the words, enlarging the size of the windows may not improve the performance obviously. As such, other natural language processing techniques such as coreference resolution are seldom considered to improve the performance of WSD because they cannot affect the semantic context which is decided by the window. Some other works use flexible size windows. Personalizing PageRank approach [9] builds the context of at least 20 content words for each sentence to be disambiguated, taking the sentences immediately before and after it when the original sentence is too short. Navigli and Velardi [6] used the sentence as the border of the semantic context of a word in their Structural Semantic Interconnection approach. Coreference resolution may help these works to enlarge the contexts. However, the related words in their contexts are order-free. They consider all the words in a context equally without considering the semantic or syntactic relations between the words in the context. A larger context may not be helpful for them to improve the precision.

Recently, Hu et al. [13] proposed a new WSD approach based on an instance knowledge network (IKN). It keeps all the information of the training set at the instance level of the IKN. When attempting to disambiguate word senses for a candidate sentence, it discovers the knowledge by a graph matching algorithm from the IKN. Because they used Stanford dependency parser [14, 15] to parse the text into dependency graphs, and Stanford parser can only work on separate sentences properly. The size of the dependency graph limited the performance of the IKN approach. Actually, the instance network structure with syntactic relations between instance nodes provides the potential to enlarge the contexts. Based on this observation, we found that coreference resolution techniques can be used to extend the structure of IKN and consequently help to improve the performance of WSD.

Up to now, WSD and coreference resolution have been considered as two separate tasks in natural language processing due to some reasons discussed above. We reckon that natural language understanding is an integrated process. If possible, different techniques should be integrated to help each other to improve the performance of natural language text.

In this paper, we aim to use coreference resolution technique to improve performance of WSD. To the best of our knowledge, this is the first attempt in this topic. We propose to employ the results of coreference resolution in WSD based on the IKN approach [13]. We enlarge the contexts for WSD by connecting separate dependency graphs of IKN with the help of coreference resolution.

The rest of the paper is organized as follows. In Section 2, we briefly introduce the IKN WSD approach, which sets the basis of this work. We present the extended IKN structure, its graph matching algorithm and algorithms for training and WSD in

Section 3. The experimental evaluations are given in Section 4. In Section 5, we discuss some issues for incorporating coreference resolution into WSD. Section 6 concludes the paper with an indication of the future work.

2 Instance Knowledge Network and Its WSD Approach

We first give a brief introduction of the IKN WSD [13] approach as the work presented in this paper is based on it. The IKN is a knowledge representation model, which has three levels – the word level, the type synset level and the instance level. The word level and the type synset level are from WordNet [16]. To build the instance level of the IKN, they parse the sentences in a sense tagged corpus into dependency graphs. Each word node in a dependency graph is set a unique identifier. Each word node becomes an instance node. Then by the sense tags for the word, they connect the instance node to the corresponding tagged sense synset of WordNet. By this way, they convert each dependency graph as an instance graph patterns (IGP). So the instance level of the IKN is composed of all the IGPs which are created from the corpus.

There are four types of relations in the IKN: sense relations between each word at the word level and its sense synsets at the synset level, semantic relations between the synsets at the synset level, instance relations between each synset at the synset level and its instance nodes at the instance level, and dependency relations between instance nodes at the instance level.

In IKN, each word may have multiple senses, each sense in turn may be tagged in multiple positions in the corpus, and an instance node is created for each tagged word. Therefore, a word may be associated with multiple instance nodes.

To discover the knowledge in the IKN, a graph matching algorithm was proposed. The algorithm attempts to find matching sub graphs in the IGPs at the instance level of the IKN for a particular candidate dependency graph which is parsed from a candidate sentence. They named a matching sub graph as an instance matching sub-graph (IMSG).

Figure 1 shows a simplified structure of the IKN and a general picture on how graph matching works. Given a candidate dependency graph *G*, first the algorithm finds, for each candidate word *w* in *G*, the semantic related synsets (SRSs) at the type synset level and then the semantic related instance nodes (SRINs) at the instance level. Then, for each edge $e(w_1, w_2)$ in *G*, we find its matching edges in all IGPs at the instance level. An edge $e'(iw_1, iw_2)$ in an IGP *G'* is called a matching edge of *e* if iw_1 and iw_2 are an SRIN of w_1 and w_2, respectively, and they have the same dependency relation. Finally, the matching edges by shared candidate words and instance nodes are connected to get the IMSGs.

For each returned SRIN *n* of a word *w*, their semantically related instance relationship SRIR(*w*, *n*) is denoted as (*w*, *s*, *t*, *n*) where *s* is the sense synset of *w* on the path and *t* is the semantic relation between *s* and the SRS *s'* on the path which directly connects to *n*.

Because the IKN approach is based on Stanford dependency parser which can only parse sentences, the candidate dependency graphs and the IGPs can only represent separate sentences. Consequently, each IMSG is also limited in a single sentence.

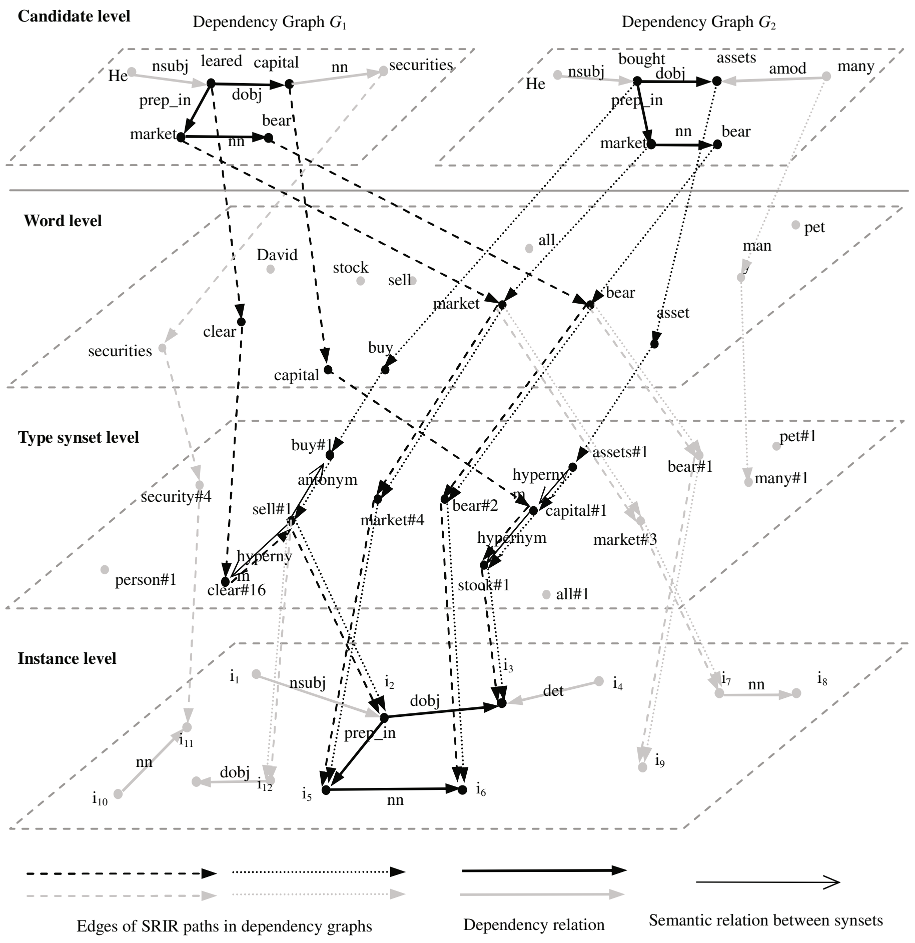


Fig. 1. Graph Matching Algorithm on Instance Knowledge Network

Base on the graph matching algorithm, a probabilistic training algorithm and a WSD algorithm are developed. The probabilistic training algorithm attempts to calculate conditional probabilities for each pair <i_1, i_2> of instance nodes in each IGP. The instance node pairs are not limited to edges only. Since multiple candidate word pairs {< w_{1i}, w_{2j} >} may match <i_1, i_2>, and matched word pairs can be classified by their SRCs t_1 and t_2 of their SRIRs, respectively. Here SRIR(w_{1i}, i_1) = (w_{1i}, s_{1i}, t_1, i_1), SRIR(w_{2j}, i_2) = (w_{2j}, s_{2j}, t_2, i_2), multiple conditional probabilities are defined for <i_1, i_2> and they are directional. For instance, P(i_2, t_2 | i_1, t_1) is defined as the probability of s_{2j} being the proper sense of w_{2j} when s_{1i} is the proper sense of w_{1i}. P(i_1, t_1 | i_2, t_2) can be defined similarly.

After the training process, conditional probabilities from one sense synset to others are obtained. Since a pair of words <w_1, w_2> may appear in different candidate dependency graphs with different syntactic relations between them and each candidate dependency graph may match different IGPs, each pair <s_{1i}, s_{2j}> of sense synsets of <w_1, w_2> may have multiple matched instance node pairs {<i_{1i}, i_{2j}>}. Consequently, for each pair <s_{1i}, s_{2j}>, it is associated with multiple pairs of conditional probabilities, each pair coming from a matched instance node pair <i_{1i}, i_{2j}>.

The WSD algorithm attempts to find IMSGs in the IKN given a candidate dependency graph. Different word pairs of a particular sense synset pair <s_1, s_2> may have different syntactic relations, and their matched sets of instance node pairs may also be different. This shows different contexts of <s_1, s_2>. For each context, the pair of conditional probabilities for <s_1, s_2> are calculated as the maximal conditional probabilities from the matched set of instance node pairs.

Based on these context sensitive conditional probabilities between synsets, an iterative process is deployed for calculating the probability for each sense of a candidate word until it gets stable. The final probability for each word sense is considered as the probability of the sense being the proper sense of the candidate word. The sense with maximal probability is considered as the WSD result of the word and its probability is defined as the confidence of the disambiguation.

3 Extending IKN with Coreference Resolution

The IKN approach [13] opens the door to use coreference resolution to improve the performance of WSD. To incorporate the results of coreference resolution, we first extend the structure of the IKN. Based on the extended IKN structure, we then present the corresponding extended graph matching algorithm. We finally discuss the probabilistic training and WSD algorithms for the extended IKN.

3.1 Extending the Structure of IKN

Stanford dependency parser works on sentences and parses each sentence into a dependency graph. A dependency graph contains words as nodes and dependency relations between the words as edges. There is no edge between different dependency graphs of different sentences.

Coreference resolution is to discover the words or phrases in a sentence or different sentences which refer to the same entity. When there are two words or phrases refer to the same entity, we connect them by a coreference relation. If these two words or phrases belong to different sentences, then their corresponding dependency graphs are connected, and consequently, the context for WSD is enlarged.

To use coreference resolution to improve the performance of WSD, we employ BART coreference system [17, 18]. In BART, the coreference resolution results are represented as tags for phrases or single words and the phrases or words with the same tag are deemed coreferenced.

To extend the IKN with the results of BART, we first select the base word in each coreferenced tagged phrase or word. For pronoun, the base word is the pronoun itself. For noun phrase which contains adjective or multiple nouns, we select the last noun as the base word. Then in each group of coreferenced phrases and words with the same tag, we select the base word of the first phrase or word in the text as the prime base word. Finally, we connect all the nodes of the base words in the group to the node of the prime base word by adding new edges representing the coreference relation. As a result, some previously separate dependency graphs are connected to form a so-called joint dependency graph (JDG). A JDG may represent part of an article or even the whole article.

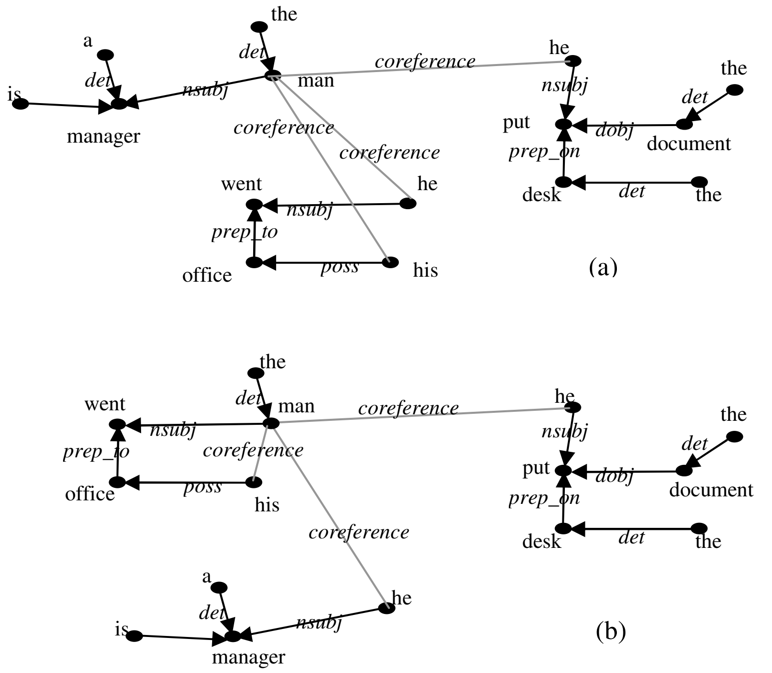


Fig. 2. Joint dependency graph

Definition (Joint Dependency Graph): A joint dependency graph JG is connected graph represented as $JG(T, CG)$ where T = $\{t_1, \ldots, t_n\}$ and $CG = \{CG_1, \ldots, CG_n\}$. $t_i \in T$ is a tag. $CG_i = \{G_{i0}, G_{i1}, \ldots, G_{in_i}\}$ is a group of dependency graphs connected by the coreference relation represented by t_i, and each G_{ij} ($1 \le j \le n_i$) is connected to G_{i0} which owns the prime base word tagged by t_i.

Figure 2(a) shows an example of a JDG. In the example, we have three sentences "*The man is a manager. He went to his office. He put the document on the desk.*" These three sentences are parsed as three dependency graphs.

The coreference resolution results for these three sentences are four coreferenced phrases: *the man* in the first sentence, *he* and *his* in the second sentence, and *he* in the last sentence. We select the base words for each phrase, where *man* is the base word for the first phrase, thus the prime base word for these three sentences, *he*, *his* and *he* are base words for the other three phrases. So we connect the base words *he*, *his* and *he* to the prime base word *man* with the coreference relation, and it results in a star-shaped connected JDG shown in Figure 2(a).

The star-shaped connection method may lead to different graph structures when similar sentences are presented in different order. For example, we may present the above three sentences as "*A man went to his office. He put the document on the desk. He is a manager.*" The JDG for these three sentences is shown Figure 2(b). We will show that this JDG will be treated as the same as the JDG shown in Figure 2(a) by out extending graph matching algorithm.

In the IKN approach, we can extend the dependency graphs to JDGs at both the candidate level and the instance level (shown in Figure 1). That is, by coreference resolution, we can connect coreferenced candidate graphs to a candidate JDG at the candidate level, and connect coreferenced IGPs to a joint IGP (JIGP) at the instance level.

3.2 Graph Matching Algorithm between Candidate JDG and JIGP

The graph matching algorithm of the IKN WSD approach [13] is limited to match some IGPs at the instance level of the IKN with the given candidate dependency graph. Now, we present the extended graph matching algorithm between candidate JDGs and JIGPs. Due to page limitation, we only highlight the main difference between the extended algorithm and the original algorithm which is briefly introduced in Section 2.

Given a candidate JDG *JG*, we find all matching sub-graphs of JIGPs in the extended IKN (EIKN for short) for *JG*.

In the IKN, the matching algorithm basically first finds each pair of matching edges between a candidate dependency graph and an IGP, and then tries to maximally connect to those matched edges in both the candidate dependency graph and the IGP. After that, one or many IMSGs in the IGP can be identified for their corresponding subgraphs in the candidate dependency graph.

In the EIKN, a graph matching is between a candidate JDG and a JIGP. As such, a subgraph of the candidate JDG may cover multiple candidate dependency graphs. Similarly, an IMSG may involve multiple IGPs in a JIGP. This is because that two matching edges belonging to different dependency graphs or different IGPs can be connected by coreference relations.

Figure 3 shows the difference of connecting matching edges between the graph matching algorithms of existing IKN system and our extended IKN system. Figure 3(a) shows a normal matching between a candidate dependency graph and an IGP in the IKN. The connection happens because the candidate word w_2 and its SRIN i_2 are shared by two pairs of matching edges.

Figure 3(b), Figure 3(c) and Figure 3(d) show the differences of the extended graph matching algorithm on how to connect matching edges between a JDG and a JIGP. Coreference relation holds between w_2 and w'_2 at the candidate level and between i_2 and i'_2 at the instance level. A coreference relation is transitive, which means that if coreference relation holds between a and b as well as between b and c, then coreference relation also holds between a and c.

In Figure 3(b), instance node i_2 is a SRIN for both candidate words w_2 and w'_2 and there is a coreference relation between w_2 and w'_2. We can join the two pairs of matching edges $\{(w_1, w_2), (i_1, i_2)\}$ and $\{(w'_2, w_3), (i_2, i_3)\}$ together. So instance dependency graph with node set $\{i_1, i_2, i_3\}$ is an IMSG of the JDG with node set $\{w_1, w_2, w'_2, w_3\}$.

In Figure 3(c), both instance nodes i_2 and i'_2 are an SRIN of candidate word w_2 and there is a coreference relation between i_2 and i'_2. So the joint instance dependency graph with node set $\{i_1, i_2, i'_2, i_3\}$ is considered as an IMSG of the dependency graph with node set $\{w_1, w_2, w_3\}$.

In Figure 3(d), there is a coreference relation between w_2 and w'_2 and there is also a coreference relation between i_2 and i'_2. In this case, the joint instance dependency graph with node set $\{i_1, i_2, i'_2, i_3\}$ is an IMSG of the JDG with node set $\{w_1, w_2, w'_2, w_3\}$.

Through these methods, we extend the graph matching algorithm for the EIKN. We can find all (joint) IMSGs in all JIGPs of the EIKN for a given candidate JDG.

3.3 Extending Training and WSD Approaches

Based on the graph matching algorithm for the EIKN, the probabilistic training algorithm and the WSD algorithm for the EIKN can be developed similar to the IKN. In the IKN system, both a candidate dependency graph and an IGP are limited to a single sentence. In the EIKN system, we train the EIKN by candidate JDGs and the matched IMSGs may span over multiple IGPs (i.e., in a JIGP). As a result, conditional probability needs to be calculated or adjusted for each instance node pair in the matched IMSGs. This helps enlarge the size of IMSGs, and in turn helps WSD for matching text with larger contexts.

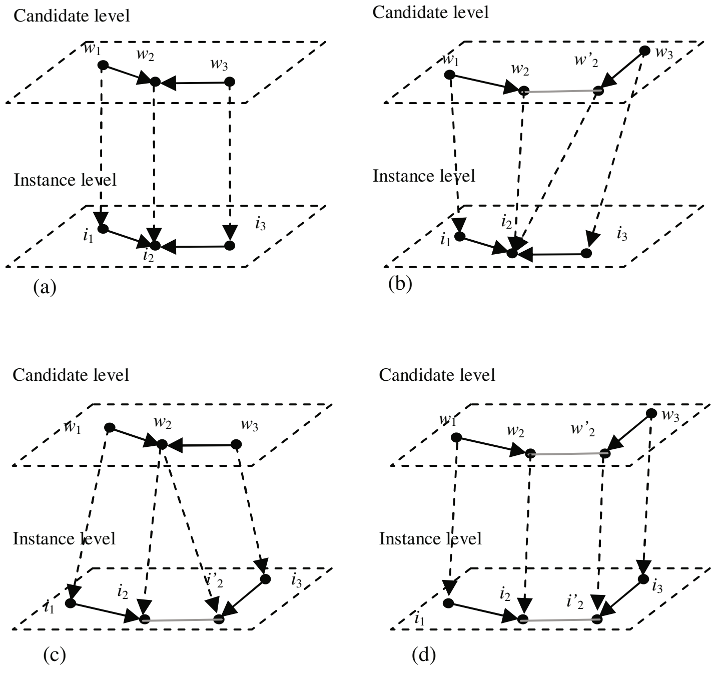


Fig. 3. Comparison between graph matching algorithms of IKN and EIKN

Compared the WSD algorithm for the EIKN with the WSD algorithm for the IKN, there are two main differences. Firstly, instead of a dependency graph for a single sentence, the EIKN WSD algorithm disambiguates a JDG for multiple sentences with a larger context. Secondly, the EIKN WSD algorithm is able to match against a JIGP instead of an IGP. Because of these differences, in the probabilistic reasoning process of the EIKN WSD algorithm, the finding of the maximum conditional probability of a given sense synset from another sense synset is not limited from the dependency graph of a single sentence. For the sense synset s of a candidate word w, we find a maximal conditional probability $P(s \mid s_{ij})$ of s from each sense synset s_{ij} of surrounding candidate word w_i in the JDG which w belongs to. By this approach, we enlarge the size of context which is considered for WSD. Based on the maximal conditional probabilities between synsets in a larger context, we employ a similar iterative process to the IKN WSD for calculating the final probability for each sense of a candidate word. The sense with maximal final probability is considered as the WSD result of the word and its probability is defined as the confidence of the disambiguation.

4 Experiments and Evaluation

Similar to the settings for the IKN experiments, we use Stanford dependency parser to parse the text into dependency graph. The additional feature is that we employ BART coreference resolution system as a basis to connect the dependency graphs and IGPs together in our approach. We build the EIKN by sense tagged corpus SemCor [19] and we train the EIKN by SemCor again. The experiment results of WSD are for Senseval-3 all word tasks.

The trained EIKN has about 500k instance nodes, 2.2M relations and 930k conditional probabilities. The EIKN building process and training process cost about 110 hours on a windows XP professional system with 2.2GHz CPU and 3 GB RAM. The disambiguation for the texts of senseval-3 all word tasks takes about one hour and a half on the same computer.

To compare with the IKN approach, the EIKN high-precision WSD results are combined with those of each given existing WSD algorithm A through the following method which is the same to the IKN approach. For each test word w_i, we define $s_{\mathrm{EIKN}}(w_i)$ as the result sense, $c_{\mathrm{EIKN}}(w_i)$ as the confidence of disambiguation, and θ as the confidence threshold for the EIKN WSD approach. We also define $s_{\mathrm{A}}(w_i)$ as the result sense for the existing WSD algorithm. Then we can get the result sense $s_{\mathrm{A+EIKN}}(w_i)$ of w_i by the combined algorithm as $s_{\mathrm{A+EIKN}}(w_i) = s_{\mathrm{EIKN}}(w_i)$ if $c_{\mathrm{EIKN}}(w_i)$ is greater than or equal to the threshold θ, and $s_{\mathrm{A+EIKN}}(w_i) = s_{\mathrm{A}}(w_i)$ otherwise.

Through this method, we use the EIKN WSD results with high confidence values (greater than the threshold θ) to replace the corresponding WSD results of the existing algorithm. If these selected EIKN WSD results have better precision than the replaced existing WSD results, the combined WSD results will get better recall than the existing WSD algorithm.

Table 1. The comparision between the recalls of combined algorithms of Extended IKN and existing algorithms and the combined algorithms of IKN and the corresponding existing algorithms

	Single	Combined		
IKN/EIKN Threshold θ	N/A	0.7	0.8	0.9
GAMBL+IKN	65.2%	65.2%	65.4%	65.4%
GAMBL+EIKN	65.2%	65.1%	65.6%	65.8%
SenseLearner+IKN	64.6%	65.2%	65.1%	65.0%
SenseLearner+EIKN	64.6%	65.8%	65.9%	65.7%
Koc+IKN	64.1%	64.7%	64.7%	64.5%
Koc+EIKN	64.1%	64.8%	65.2%	65.2%
R2D2+IKN	62.6%	63.8%	63.4%	63.2%
R2D2+EIKN	62.6%	63.8%	64.0%	63.8%
Meaning-allwords+IKN	62.4%	63.6%	63.6%	63.5%
Meaning-allwords+EIKN	62.4%	63.3%	63.9%	63.7%

We combine the EIKN algorithm with the five best algorithms of Senseval-3 all word tasks. For each given existing algorithm A, we attempt to get the combined results of A+EIKN and the corresponding combined results of A+IKN for a range of thresholds θ. We compare the results of A+EIKN and A+IKN for each particular θ.

In Table 1, we compare the recalls of each combined algorithm between the EIKN algorithm and one of the five existing algorithms with the corresponding combined algorithm of the IKN algorithm and the existing algorithm. When the threshold θ equals to or is greater than 0.8, for each existing WSD algorithm A and particular θ the combined results of A+EIKN achieve better recall than the corresponding combined algorithm A+IKN. When the threshold θ *=0.8, 0.9,* the improvements are from 0.2% to 0.8%. Compared to the corresponding existing WSD algorithms, the improvements are from 0.4% to 1.5%. Except the combined GAMBL+EIKN algorithm, all the other four combined algorithms improve more than 1% the recall than the corresponding algorithm. The best recall of A+EIKN is also better than A+IKN.

We observed that the effects of combined algorithms may be different. A worse recall of an existing algorithm may lead to better recall of the combined result. For example, the recall 64.6% of SenseLearner algorithm is worse than the recall 65.2% of GAMBL algorithm. However, when θ = 0.8, the recall of SenseLearner+EIKN algorithm is 65.9% which is better than 65.6% the recall of GAMBL+EIKN algorithm.

From the experiments, in the word set with high disambiguation confidence, the EIKN algorithm which uses coreference resolution results can achieve better results than the IKN algorithm and the top five algorithms in Senseval-3 all word tasks.

5 Discussion

Up to now, word sense disambiguation and coreference resolution are still two separate topics in natural language processing area. Nonetheless, the understanding of natural language is an integrated process. Unfortunately, to the best of our knowledge, there is no existing model that is able to consider these two understanding techniques together before the IKN model was proposed.

We found that the IKN model has the potential to incorporate these two different techniques at the instance level, so we propose the EIKN model to realize the incorporation. The EIKN model uses a coreference approach to enlarge the effective context size for WSD. This approach is the first attempt to show how these two different techniques work together for natural language understanding.

Some issues remain in incorporating these two techniques. The main issue is that existing coreference resolution systems may not be able to provide high-precision results. As a WSD result depends on the results of coreference resolution, the quality of results obviously affects the quality of the WSD result. BART, the coreference resolution system used in our EIKN system provides the precision about only 68% in MUC-6 benchmark system. This inevitably affects the performance improvement. Nonetheless, the relative low precision of coreference resolution does not impact much on the accuracy of high confidence WSD results in the EIKN. In the EIKN, the probabilistic training process works on JDGs which incorporate the results of coreference resolution. Therefore the conditional probabilities across multiple dependency graphs that we acquired in the training process already reflect the precision of the coreference resolution results. This phenomenon shows that the IKN probabilistic model is not only effective to WSD. It may be also effective to coreference resolution.

6 Conclusion and Future Work

In this paper, we introduced a new word sense disambiguation approach based on the existing IKN WSD approach by incorporating coreference resolution results. With the help of the results of a coreference resolution system, we build our EIKN system by connecting candidate dependency graphs at the candidate level and IGPs of the IKN at the instance level. This allows us to enlarge the size of contexts which can be considered in both the training process and the disambiguation process.

We run extensive experiments in our EIKN system based on Senseval-3 all-words task. Following the similar evaluation approach of the IKN work, we combined our EIKN WSD algorithm with the best five WSD algorithms. The performance of each combined algorithm of our EIKN algorithm and one existing algorithm is better than the corresponding combined algorithm of the IKN algorithm and the existing algorithm in the word sets with high confidence.

In the future, we will attempt to use high precision WSD results to provide high precision results of coreference resolution. We believe that high precision WSD and high precision coreference resolution can help each other in an iterative process.

Acknowledgement

The research work reported in this paper is supported by Australian Research Council under Linkage Grant LP0669660.

References

1. Navigli, R.: Word sense disambiguation: A survey. ACM Computing Surveys 41(2) (2009)
2. Snyder, B., Palmer, M.: The English all-words task. In: ACL Senseval-3 Workshop 2004 (2004)
3. Decadt, B., Hoste, V., Daelemans, W.: GAMBL, Genetic Algorithm Optimization of Memory-Based WSD. In: ACL Senseval-3 Workshop 2004 (2004)
4. Yarowsky, D., Florian, R.: Evaluating sense disambiguation across diverse parameter spaces. J. Nat. Lang. Eng. 9(4) (2002)
5. Cuadros, M., Rigau, G.: Quality assessment of large scale knowledge resources. In: EMNLP 2006 (2006)
6. Navigli, R., Velardi, P.: Structural Semantic Interconnections: A Knowledge-Based Approach to Word Sense Disambiguation. IEEE Trans. Pattern Anal. Mach. Intell. 27(7) (2005)
7. Sinha, R., Mihalcea, R.: Unsupervised graphbased word sense disambiguation using measures of word semantic similarity. In: ICSC 2007 (2007)
8. Navigli, R., Lapata, M.: Graph connectivity measures for unsupervised word sense disambiguation. In: IJCAI 2007 (2007)
9. Agirre, E., Soroa, A.: Personalizing PageRank for Word Sense Disambiguation. In: EACL 2009 (2009)
10. Pham, T.P., Ng, H.T., Lee, W.S.: Word Sense Disambiguation with Semi-Supervised Learning ACL 2005 (2005)
11. Mihalcea, R.: Using Wikipedia for AutomaticWord Sense Disambiguation. In: NAACL 2007 (2007)
12. Lee, Y.K., Ng, H.T.: An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In: EMNLP 2002 (2002)
13. Hu, S., Liu, C., Zhao, X., Kowalkiewicz, M.: Building Instance Knowledge Network for Word Sense Disambiguation. To appear in Proceedings of the 33rd Australasian Computer Science Conference, ACSC 2011 (2011), `http://www.ict.swin.edu.au/personal/cliu/`
14. Klein, D., Manning, C.: Fast Exact Inference with a Factored Model for Natural Language Parsing. In: NIPS 2002 (2002)
15. Stanford_Parser, `http://nlp.stanford.edu/software/lex-parser.shtml`
16. Fellbaum, C.: WordNet – an electronic lexical database. MIT Press, Cambridge (1998)
17. Versley, Y., Moschitti, A., Poesio, M., Yang, X.: Coreference Systems based on Kernel Methods. In: Coling 2008 (2008)
18. Versley, Y., Ponzetto, S.P., Poesio, M., Eidelman, V., Jern, A., Smith, J., Yang, X., Moschitti, A.: BART: A Modular Toolkit for Coreference Resolution. In: Companion Volume ACL 2008 s(2008)
19. Miller, H., Leacock, C., Tengi, R., Bunker, R.: A semantic concordance. In: ARPA Workshop on HLT (1993)
20. Navigli, R.: A structural approach to the automatic adjudication of word sense disagreements on NLE 2008 (2008)

Deep Semantics for Dependency Structures

Paul Bédaride[1] and Claire Gardent[2]

[1] Universität Suttgart
paul.bedaride@loria.fr
[2] CNRS/LORIA
claire.gardent@loria.fr

Abstract. Although dependency parsers have become increasingly popular, little work has been done on how to associate dependency structures with deep semantic representations. In this paper, we propose a semantic calculus for dependency structures which can be used to construct deep semantic representations from joint syntactic and semantic dependency structures similar to those used in the ConLL 2008 Shared Task.

Keywords: Dependency graphs, Deep Semantics, Graph Rewriting.

1 Introduction

Deep semantics have been developed for stochastic categorial parsers [1] and for parsers based on phrase structure grammars [2, 3, 4]. Much less work has been done, however, on combining dependency parsers with a a deep semantics calculus. Although [5] sketches a syntax-semantics interface for dependency grammar, the proposed approach requires a constraint-based, tightly interleaved construction of dependency, predicate/argument and scoping structure which is not easily adaptable to the output of contemporary dependency parsers. Similarly, [6] presents a formalism for semantic construction from dependency structures. However, the approach incorrectly assumes that semantic dependencies match syntactic dependencies and so fails to generalise (cf. Section 2).

In this paper, we present an approach for rewriting dependency graphs into deep semantic representations that can be applied to joint syntactic and semantic dependency structures similar to those used in the ConLL 2008 Shared Task. We start by discussing a number of issues raised by dependency structures in relation to semantic construction and by motivating the choices underlying our approach (Section 2). We then present our proposal (Section 3).

2 Motivations

In essence, a dependency structure consists of nodes labelled with lexical items (and optionally, parts-or-speech) and linked by binary asymetric relations called dependencies. Figure 5 illustrates this with the plain (non bold) nodes and edges forming a possible dependency graph for the sentence "John seems to love Mary".

Importantly, dependency structures differ from phrase structure trees or categorial derivations in that they describe relations between words and eschew the notions of syntactic constituents and non terminal syntactic categories. Starting with [7] however, a key assumption is that the computation of deep semantics strongly relies on syntax. Thus in phrase structure grammars, each syntactic rule is coupled with a semantic rule specifying how the semantics of its daughters combines to yield the semantics of the constituent being derived. Similarly, in categorial grammars, each word is simultaneously assigned a syntactic and a semantic category describing both how it syntactically combines with other word/category pairs and how its semantics combine with the semantics of the items it combines with. In essence, syntax guides semantic construction in that it constrains the semantic type[1] of word occurrences and specifies how the semantics of constituents combine to yield the semantics of derived constituents. Given this, dependency graphs raise two main issues with respect to semantic construction.

First, the impoverished syntactic categories they include make it difficult to determine the semantic type of a given word occurrence. For instance, given the two sentences in (1), there is no obvious way to determine from their dependency graphs (shown in Figures 5 and 6) that the semantic functor licensed by "seems" combines with a VP semantics in (1a) but with a sentence semantics in (1b).

(1) a. John seems to love Mary
    b. It seems that John loves Mary

The problem is that, in both cases, the syntactic category associated with "seems" is a simple part-of-speech category which fails to indicate the syntactic and hence the semantic type of the verb arguments. To put it another way, there is no indication in a dependency graph of which syntactic type of "seem" is used to build each of the two sentences. To determine that "seems" combines with an infinitival VP in (a) but with a sentential argument in (b), dependencies that are non local to "seems" would need to be checked e.g., Does "love" in the dependency graph dominate a "to" or a "that" node ?

A second issue regarding semantic construction from dependency graphs is that syntactic and semantic dependencies do not necessarily match [3]. In particular, there is sometimes a mismatch between predicate/argument and scope relations. For instance, in questions such as (2), "which man" scopes over the rest of the sentence to yield the meaning *Which is the x such that x is a man and John thinks that Mary likes x ?* Standard dependency structures fail to capture the scope of the wh-element because "man" is related by an object relation to "likes" but not to the main verb "thinks".

[1] Here and in what follows, the term "semantic type" refers to the logical type of the denotation of natural language expressions e.g., in an extensional typed lambda calculus the type t of sentences or the type *((e,t),(e,t),e)* of quantifiers with e the type of individuals and t the type of truth values.

(2) Which man does John think that Mary likes?

In sum, the combined lack in a dependency graph, of a fully fledged syntactic categorial system and of a syntactic structure makes it difficult both to determine the semantic type of a word occurrence (Does "seems" combine with a VP or an S semantics?) and to appropriately describe how meanings should combine (How can both scope and predicate/argument relationships be appropriately captured?). To address these issues, we propose an approach to semantic construction which does not rely on a strict syntax/semantic parallelism but constructs semantic representations based on a small set of general principles describing the syntax-semantic interface. These principles are encoded in graph rewriting rules which determine the semantic type of each word occurrence based on the graph configuration in which it occurs. We show how this approach handles the cases above and a range of various other semantic phenomena.

3 Proposal

We start (Section 3.1) by briefly introducing graph rewriting and discussing termination, confluence and well-formedness. We then describe our approach to semantic construction (Section 3.2) and illustrate its working by describing the derivation of "Every man loves a woman" (Section 3.3). We then go on to sketch how to handle control, raising, modifiers and relative clauses (Section 3.4).

3.1 Graph Rewriting

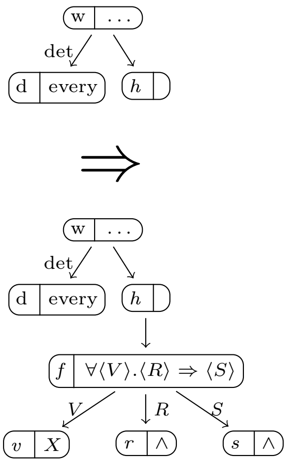
Used in e.g., formal calculus, combinatoric algebra and operational semantics, rewriting is a technique for modelling reduction and simplification. For instance, the rewriting rule $r_1 : x{*}y{+}x{*}z \rightarrow x{*}(y{+}z)$ permits factorising $5{*}6{+}5{*}7{+}5{*}8$ to $5 * ((6 + 7) + 8)$. More generally, a rewriting system consists of a set of rewriting rules of the form $l \rightarrow r$ where l and r are filtering and rewriting patterns respectively. Given a graph g, such a rule will apply to g if g matches the filtering pattern l. The result of applying a rule to a graph g is g where the sub-part of g matched by l is rewritten according to the rewriting pattern r. Matching consists in looking for a homomorphism between the pattern graph l and the host graph g while the allowed rewriting operations include information duplication, deletion and addition[2].

GrGen, a standard graph rewriting system. To define our rewrite rules, we use an existing rewriting system called GrGen [9]. In GrGen, the objects handled by rewriting are directed graphs with typed nodes and edges. Each node and each edge has a type. Additionally, nodes can be associated with a set of attribute value pairs constrained by the node type. In the filtering pattern, attribute value pairs are interpreted as constraints while in the rewriting pattern, they are interpreted as assignments. Finally, nodes names can be used to constrain the mapping between filtering and rewriting pattern in that two nodes with the same names must be identical.

[2] For a precise definition of matching, we refer the reader to [8].

```
rule forall {
 pattern{
  w:word; d:word; h:sem;
  w -det-> d;
  w --> h;
  if {d.word=="every"}
 }
 replace{
  w -det-> d;
  w --> h;
  f:sem; v:sem; r:sem; s:sem;
  eval{
    v.var = "X";
    f.formula = "";
    r.formula = "∧";
    s.formula = "∧";
  }
  h --> f;
  f -V-> v;
  f -R-> r;
  f -S-> s;
 }
}
```

(a) GrGen Rule



(b) Graphical representation

Fig. 1. A rewrite rule which expands the seed node licensed by an "Every N" subgraph with a semantic subgraph encoding the corresponding semantic type namely, the type of a universal quantifier. Here and in what follows, we use the following graphical conventions. Node names (e.g., w, d, *h, f, q, r*) appear to the left of the vertical bar splitting a node description while attribute values (every, $X, \wedge$) appear to its right. Attribute names are omitted. Node types are indicated using different fonts whereby italics indicate a node in the semantic representation structure, a plain font a node in the dependency structure and a bold font a node in the SRL structure. Edge type is not represented but is deducible from the types of the in and out vertices.

Expressive and efficient, GrGen[3] is well suited to specify our semantic construction rules. For instance, the rewrite rule graphically depicted in Figure 1b can be specified as shown in Figure 1a. In essence, the rewrite rule expands the seed node h[4] licensed by an "Every N" dependency subgraph, with a semantic subgraph capturing the corresponding semantic type namely, the type of a universal quantifier.

[3] There are other rewriting systems available such as in particular, the Tsurgen system used in the Stanford Parser to map parse trees into dependency graphs. We opted for GrGen instead because GrGen is efficient, notationally expressive (for specifying graphs but also rules and rule application strategies) and comes with a sophisticated debugging environment.

[4] See Section 3.2 for an explanation of how seed nodes are introduced.

Confluence, termination and well-formedness. The standard approach to semantic construction typically relies on the typed lambda calculus to define and combine meaning representations. This ensures termination (through the typing system), confluence (all ways of combining the same sequence of lambda terms yield the same result) and well-formedness of the resulting formulae (beta-reduction will fail in case of a type clash).

These properties are not garanteed by rewriting. Indeed it is easy to define a non confluent rewriting system that yields ill formed semantic representations. We avoid those pitfalls as follows. We ensure termination and confluence by imposing a total order on rule application. As we shall see in the following section, each rule captures a general semantic construction principle. The order imposed on their application captures the way in which these principles interact (e.g., scope can only be defined after quantifiers and their semantic arguments have been introduced). Further, well formedness is ensured by the translation from semantic graphs to FOL formulae which will fail in case a FOL formula cannot be reconstructed from a given constructed graph.

3.2 Basic Semantic Construction Procedure

Our semantic calculus is semantic rather than syntax driven. Drawing on the global dependency analysis of a sentence, it incrementally constructs a semantic representation by building, linking and labelling the various substructures composing this representation.

To simplify semantic construction, we additionally assume that the dependency graphs we take as input are enriched with semantic role labelling (SRL) information. This permits abstracting over syntactic idiosyncrasies such as active-passive alternations or dative shifts, and making certain semantic dependencies e. g. in control constructions explicit. The dependency graphs are produced by the Stanford parser [10] and augmented with Propbank style semantic role labelling information as described in [11]. Given such joint structures, we use rewrite rules to further extend them with a semantic representation. Here, we illustrate the approach by showing how to build first order logical formulae but nothing hinges on this and other types of semantic representations could be built such as e.g., Discourse Representation Structures (DRSs) or Minimal Recursion Semantics structures (MRSs).

Semantic construction is modelled by a rewriting system consisting of six general syntax-semantic principles implemented as a set of cascaded rules applying in a fixed order, each rule taking as input the output of the previous step. The underlying intuition is as follows. First, "seed nodes" are created by adding as many nodes to the semantic representation as there are words pointed to by semantic role labelling edges. That is, a node is created for each predicate and each argument in the SRL structure. Second, each seed node is expanded with the subformula skeleton representing its meaning. Nominal arguments are expanded with a generalised quantifier tree shape while predicates (verbs or deverbal nominals) are expanded with an existential quantification over eventuality variables.

Third, scope is determined by linking the resulting trees together. Fourth, nodes are labelled with the appropriate predications whereby variables are bound by the appropriate operator.

In sum, semantic construction initialises a structure (seed nodes creation), expands it with structures representing the semantic type of each node (node expansion), determines scope by linking these substructures together (scoping) and finalises the resulting structure by labelling nodes with the appropriate literals and bound variable (node labelling, variable binding). Additional principles are implemented for modelling connectives such as "when" or "if-then", which are not discussed here because of space restrictions.

3.3 Run through Example

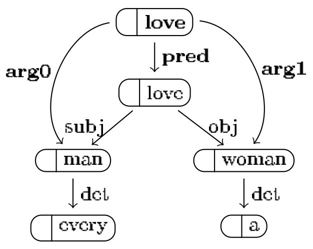
We start by giving a bird eye view of the semantic construction process for the sentence "Every man loves a woman". In the next section, we will show in more detail how the rewrite rules permit appropriately mapping syntax to semantics and more particularly, how they ensure that variables are appropriately bound.

The joint dependency+SRL graph input to semantic construction is shown in Figure 2a. The first step (2b) creates three seed nodes each of which is licensed by an SRL node: the n_0 node is licensed by the predicate node associated with the verb "loves" and the n_1, n_2 nodes are licensed by the two verb argument heads "man" and "woman" respectively.

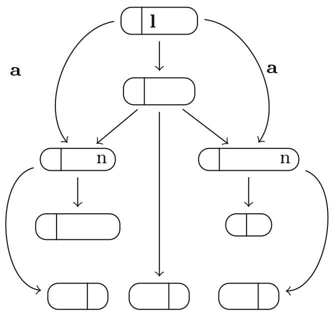
The second step expands these seed nodes to build substructures describing their semantic type. Seed nodes that are licensed by a dependency node with nominal category dominating a determiner node licence the construction of a subtree representing a generalised quantifier i.e., a tripartite structure consisting of a quantifier, a restriction and a scope where the quantifier will be determined by the specific determiner dominated by the noun (e.g., a universal for "every" or "all" and an existential for "a"[5]). In contrast, seed nodes licenced by a predicate (e.g., a verb or a deverbal nominal) trigger the construction of a structure representing an existentially bound event variable. The node expansions licensed by "Every man", "loves" and "A woman" respectively are shown in Figure 2c.

The third step (Figure 2d) connects the substructures built so far thereby determining scope. Scope is specified by adding an edge between the scoping node of each scope bearing operator and the head of the semantic substructure licensed by the next syntactic argument in the sentence (e.g., by adding a link from the scoping node of "every man" to the root node of the head of the subformula licenced by "a woman"). The verb structure is linked to the restriction of its right most argument (the tree for "loves" is linked to the scoping node of "a woman"). Here we make the simplifying assumption that scope is unambiguously determined by the linear order of words in the sentence. Scope ambiguity could be accounted for either by having a rule strategy that supports alternative rules and rule application order thereby inducing several possible solutions or by

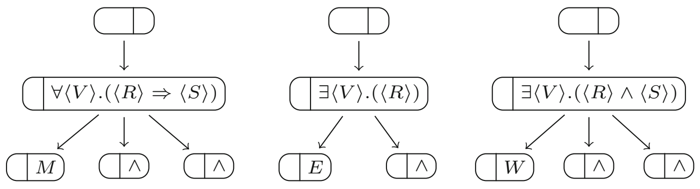
[5] For donkey sentences such as "If a farmer owns a donkey, he beats it" where the indefinite "a farmer" licenses a universal quantifier, the approach should be modified so as to build Discourse Representation Structures rather than FOL formulae.



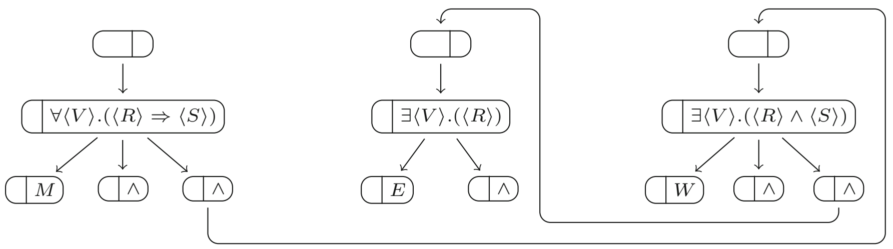
(a) Input dependency+SRL graph



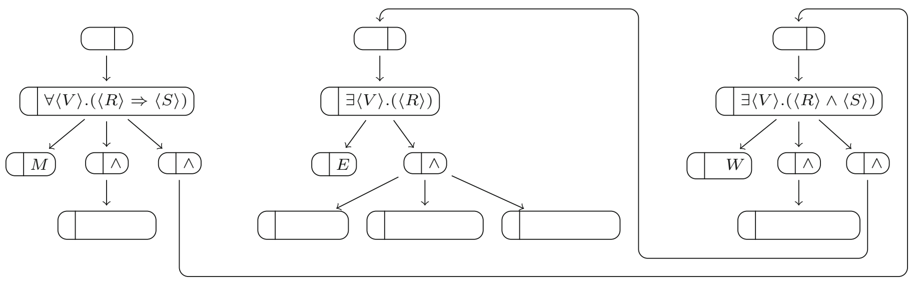
(b) Creation of seed nodes



(c) Specifying types



(d) Specifying scope



(e) Specifying predications

Fig. 2. Derivation example "Every man loves a woman". The notation $\forall\langle V\rangle.(\langle R\rangle \Rightarrow \langle S\rangle)$ is syntactic sugar indicating that the node is labelled with the attribute value pair QUANT:FORALL and that a FOL formula of that shape can be reconstructed from the subgraph rooted in that node. Indeed, from the final representation, the following FOL formula can be derived: $\forall M.(man(M) \wedge \exists W.(woman(W) \wedge \exists E.(love(E) \wedge arg0(E,M) \wedge arg1(E,W))))$

mapping the dependency graphs to underspecified semantic representations such as MRSs. In this case, the scoping links would need to underspecify, rather than specify scope and all argument structures should be linked directly to the verb structure.

Fourth (Figure 2e), predications are handled and existing substructures are expanded with the appropriate literals. For quantifiers, the restriction node is labelled with a literal whose predicate is the lemma of the nominal heading the quantifier and whose variable is the quantifier variable. Similarly the semantic structure licensed by verb and noun predicates are expanded as shown in Figure 2e so as to contain as predicate, the lemma of the licensing verb or deverbal noun and as variable, the variable bound by the existential quantifier licensed by this verb/deverbal. Additionally, literals are added for each of the verb arguments where each literal relates the verb event variable to the argument variable via the thematic role relation given in the SRL structure.

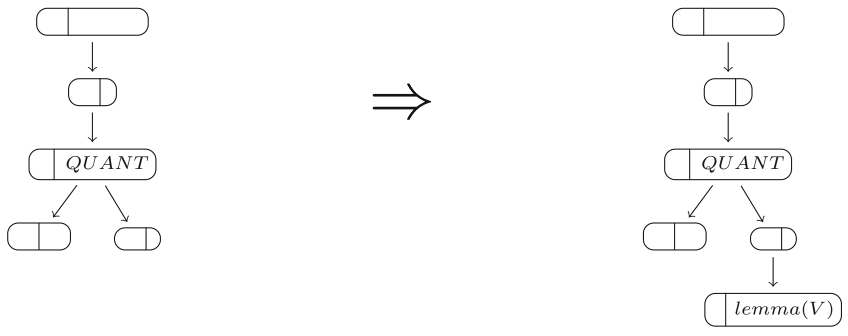
3.4 Rules, Variable Binding and Semantic Phenomena

We now show in more detail how variable binding occurs and sketch the treatment of relative clauses, raising, control and questions.

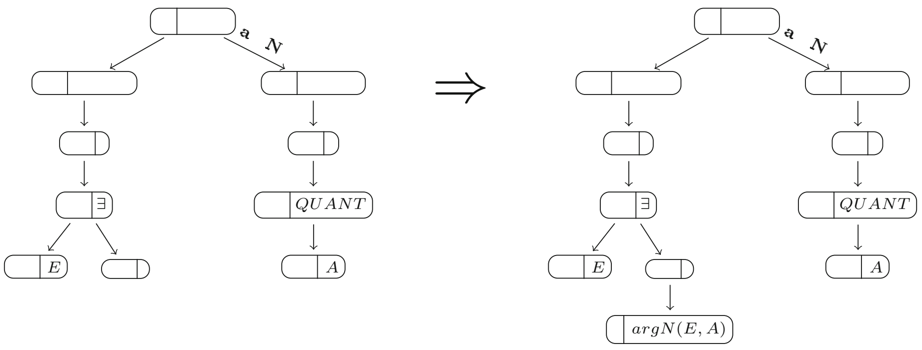
Variable binding (Quantifier restriction and verb semantics). Semantic construction must ensure that the variable bound by a quantifier correctly occurs in its restriction and in its scope. Here, this is ensured by equating the relevant variable in the restriction and in the scope with the quantifier variable. Figure 3 illustrates this graphically. The top rule shows how the quantifier restriction is labelled with a literal *lemma(V)* where *lemma* is the nominal head of the quantifier and V, the variable bound by the quantifier. Similarly (Figure 3b), each **argN** edge in the input graph licenses the introduction in the verb semantics of a literal *ArgN(E,A)* where E is the event variable licensed by the verb and A the variable licensed by the argN argument. Note that the binding of argument variables is mediated not by syntactic functions but by thematic roles thereby simplifying the syntax/semantic interface (because distinct syntactic realisations are abstracted over).

Relative clauses. Relativised arguments are processed in the same way as arguments of a main clause verb because thematic roles relate the verb of a relative clause, not to the relative pronoun, but to its antecedent (cf. Figure 4) and, as just mentioned, the binding of predicate argument variables is mediated by thematic roles. The scoping rules additionnally ensure that the semantics associated with a relative clause is included in the restriction of the relative antecedent.

Control. As for relativised arguments, control verbs do not necessitate any additional rules because the semantic role labeller already provides the information required for appropriately binding the subject (or the object) of the control verb to the subject of its infinitival complement. Thus, in “John promised Mary to shave", “John” is labelled as ARG0 of both “promised” and “shave”, thereby supporting the appropriate variable bindings.



(a)



(b)

Fig. 3. Rules adding the literal licensed by the nominal head of the quantifier (*lemma(V)*) to its restriction (top) and the argument literals (*argN(E,A)*) to its scope (bottom). $QUANT$ and ∃ abreviate the attribute value pairs QUANT:EXISTS OR FORALL and QUANT:EXISTS respectively while V, A and E are variables.The scope branch of the quantifier is not represented in the rule as the rule filter needs only specify the minimal pattern that should be present in the host graph for the rule to be applicable. Additional material is copied over. The top rule states that given a graph containing a dependency node w, labelled with the word "lemma" and linked to the skeleton quantifier subgraph rooted in h, the literal *lemma(V)* should be added to the quantifier restriction. Similarly, the bottom rule rewrites subgraphs relating a verb semantic skeleton (rooted in wh) and any of its argument semantic skeleton (rooted in ah) by adding the literal *argN(E,A)* to the verb semantics.

Adjectival and Adverbial Modifiers. Adjectives and adverbs licence the introduction of a predication over an individual and an event variable respectively. This variable is equated with the variable predicated of by the denotation of the modifiee (i.e., the noun or the verb) using a rule which can be summarised as follows: if the dependency node A is in a modification relation to the dependency node W and W is related to a semantic structure with bound variable X, then the literal $A(X)$ should be included in the restriction of this semantic structure.

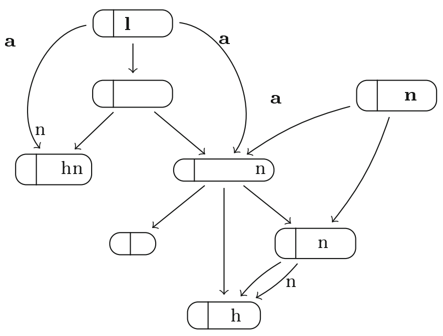


Fig. 4. "John loves a woman who sings". Semantic role labelling relate the predicate licensed by the verb of the relative clause not to the relative pronoun but to its antecedent thereby supporting a uniform semantic treatment of relativised and non relativised argumentsXS

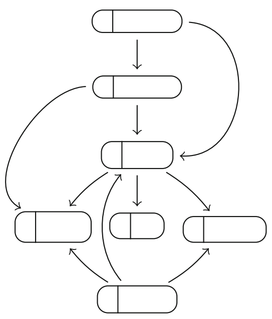


Fig. 5. "John seems to love Mary". The symbolic SRLer we use produces a modification relation between "seems" and its sentential argument thereby supporting a modifier treatment of "seem".

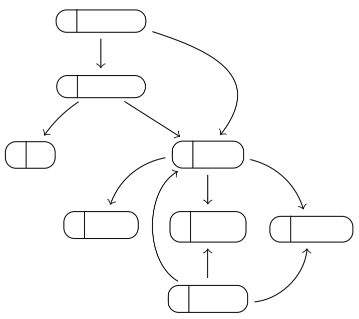


Fig. 6. "It seems that John loves Mary". The symbolic SRLer correctly labels "John" as the **arg0** of "loves".

Raising. Raising verbs such as "seems" in "John seems to love Mary" and "It seems that John loves Mary" are handled as modifiers in that they modify the event variable introduced by the sentential or infinitival object. Semantic role labelling ensures that "John" is the ARG0 of "loves" in both cases (cf. Figure 5) and therefore that the appropriate semantics is constructed.

Questions. As mentioned in Section 2, the dependency graph of questions such as "Which woman does Mary think John likes?" fails to support a strictly compositional semantics because local information is not sufficient to simultaneously determine that "Which woman" is the object of "likes" and takes scope over the whole sentence. In our approach, such sentences are unproblematic: "Which woman" licences the introduction of a quantifier which binds the object variable of "likes" (through the normal predicate/argument binding mechanism); further, its scope is determined to respect the linear order of the words in the input sentence.

Coverage and evaluation. We tested [12] the coverage and the correction of our approach by applying it to a set of 1 000 sentence pairs annotated with an entailment value (true if the first sentence entails the other, false otherwise). For each sentence, the sentences were parsed using the Stanford parser and the semantic role labeller of [11], semantic construction was carried out and the resulting semantic representations translated to FOL. Automated reasoners were then used to check entailment. In all cases, a correct FOL formula was built. Moreover, entailment detection was correct in 71.3% of the cases. Since in many cases, parsing failed to produce a correct analysis, these first results are encouraging. They need to be further tested on real world data though as the testsuite used in this first experiment was artificially constructed and restricted to a limited set of linguistic variations (different verb subcategorisation type and control mainly).

4 Conclusion

By adopting a semantics rather than a syntax driven strategy, the semantic construction approach described in this paper permits bypassing the issues raised by the lack of syntactic information in dependency graphs. More generally, the approach can be seen as defining a set of very general principles governing the construction of semantic representations for predicate/argument structures, quantifiers and modifiers. Contrary to the lambda calculus approach, this allows for a very concise system where a small set of rewrite rules can be used to describe a large number of syntax-semantics interfaces. We are currently extending the approach to cover further semantic phenomena (e.g., comparatives and discourse connectives) and evaluate its coverage and correction using the entailment recognition test.

References

[1] Bos, J., Clark, S., Steedman, M., Curran, J.R., Hockenmaier, J.: Wide-coverage semantic representations from a ccg parser. In: Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004), Geneva, Switzerland, pp. 1240–1246 (2004)

[2] Copestake, A., Flickinger, D., Sag, I., Pollard, C.: Minimal recursion semantics: An introduction. Journal of Research on Language and Computation 3, 281–332 (2005)

[3] van Genabith, J., Frank, A., Crouch, D.: Glue, underspecification and translation (1999)

[4] Gardent, C., Kallmeyer, L.: Semantic construction in ftag. In: Proceedings of the 10th meeting of the European Chapter of the Association for Computational Linguistics, Budapest, Hungary (2003)

[5] Debusmann, R., Duchier, D., Koller, A., Kuhlmann, M., Smolka, G., Thater, S.: A relational syntax-semantics interface based on dependency grammar. In: COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics, p. 176. Association for Computational Linguistics, Morristown (2004)
[6] Cimiano, P.: Flexible semantic composition with dude. In: Proceedings of the 8th International Conference on Computational Semantics, IWCS 2009 (2009)
[7] Montague, R.: The proper treatment of quantification in ordinary English. In: Thomason, R. (ed.) Formal Philosophy. Selected Papers. Yale University Press, New Haven (1974)
[8] Ehrig, H., Heckel, R., Korff, M., Owe, M.L., Ribeiro, L., Wagner, A., Corradini, A.: Algebraic Approaches to Graph Transformation - Part II: Single Pushout A. and Comparison with Double Pushout A. In: Handbook of Graph Grammars and Computing by Graph Transformation, vol. 1, pp. 247–312. World Scientific, Singapore (1999)
[9] Kroll, M., Geiß, R.: Developing graph transformations with grgen.net. Technical report (2007); preliminary version, submitted to AGTIVE 2007
[10] Klein, D., Manning, C.D.: Accurate unlexicalized parsing. In: ACL, Sapporo, Japan, pp. 423–430 (2003)
[11] Bedaride, P., Gardent, C.: Noun/verb inference. In: 4th Language and Technology Conference, Poznan, Poland (2009)
[12] Bedaride, P., Gardent, C.: Benchmarking for syntax-based sentential inference. In: The 23rd International Conference on Computational Linguistics - COLING 2010, Beijing, China (2010)

Combining Heterogeneous Knowledge Resources for Improved Distributional Semantic Models

György Szarvas[⋆], Torsten Zesch, and Iryna Gurevych

Ubiquitous Knowledge Processing Lab, Computer Science Department,
Technische Universität Darmstadt,
Hochschulstr. 10, D-64289 Darmstadt, Germany
`http://www.ukp.tu-darmstadt.de`

Abstract. The Explicit Semantic Analysis (ESA) model based on term cooccurrences in Wikipedia has been regarded as state-of-the-art semantic relatedness measure in the recent years. We provide an analysis of the important parameters of ESA using datasets in five different languages. Additionally, we propose the use of ESA with multiple lexical semantic resources thus exploiting multiple evidence of term cooccurrence to improve over the Wikipedia-based measure. Exploiting the improved robustness and coverage of the proposed combination, we report improved performance over single resources in word semantic relatedness, solving word choice problems, classification of semantic relations between nominals, and text similarity.

1 Introduction

Semantic relatedness (SR) aims at measuring how related the meaning, i.e. the semantic content of two words is. Computing the SR of words finds applications in many classical Natural Language Processing (NLP) problems like Word Sense Disambiguation [24], Information Retrieval [29,22], Cross-Language Information Retrieval [5], Text Categorization [10], Information Extraction [26], Coreference Resolution [27], or Spelling Error Detection [3].

Most of the SR measures proposed in the past have two limitations. First, they only exploit the implicit knowledge encoded in *a single* structured knowledge source like WordNet or Wikipedia, or a large text collection like the World Wide Web, but do not exploit the complementary knowledge in multiple resources through combination. Second, most measures are designed to compute relatedness between words, not between longer text segments. However, SR has important applications both on the word level (Word Sense Disambiguation, Spelling Error Correction), and on the text level (Information Retrieval, Text Categorization). Therefore, in this paper, we address the combination of the knowledge encoded in heterogeneous, independent knowledge resources to obtain better and more robust performance paying attention to direct applicability to word pairs and pairs of texts alike.

[⋆] On leave from Research Group on Artificial Intelligence of the Hungarian Academy of Sciences.

For this purpose, we focus on distributional semantic relatedness, and in particular, following Gabrilovich and Markovitch [9], we employ concept vector based measures to incorporate knowledge from heterogeneous resources (representing encyclopedic knowledge and lexical information in our case) to overcome the weaknesses of single resources. In Section 3, we give a detailed overview on concept vector based measures. We also propose a new formulation of the concept vector based measure that has one less degree of freedom, i.e. it does not require any pruning of word concept vectors. In Section 4, we introduce our combined measure. We implement the combination of independent knowledge sources through combining the relatedness scores provided by concept vector measures based on single resources. As concept vector measures are applicable on the word as well as on the text level, the combined measure preserves the direct applicability to longer texts. In Section 5, we demonstrate the usefulness of the proposed approach through the successful application of the combined measure to three different NLP tasks: i) solving word choice problems, ii) classification of semantic relations between nominals, and iii) text similarity computation. We also show that the combined measure yields stable performance on the word level for different parts of speech, which was not experimentally demonstrated by previous works.

2 Related Work

In the last decade, many different approaches have been proposed to measure the semantic relatedness of natural language units (i.e. words, phrases or texts). **Structural Methods** exploit the structural information through measuring path length [3,27,30], computing PageRank vectors [32], or comparing link vectors [20] in a lexical semantic knowledge resource like WordNet, Wikipedia or ConceptNet. **Distributional Methods** employ distributional relatedness to compare cooccurrence patterns measuring hit counts [4,2], comparing distributional profiles [23,1] or concept vectors [9,34,13] in an underlying collection of representative texts like Wikipedia or the Web.

Structural methods are defined to compute relatedness on the word level[1]. Web-based distributional methods also operate on the word level, calculating hit count based association measures like mutual information between terms. Such methods can be extended to model text-level similarity through measuring the relatedness between all word pairs in the documents to be compared, and then aggregating the word level similarities [21]. This requires $n \cdot m$ calculations to compare two texts of sizes n and m, which can be computationally demanding or even infeasible, e.g. for web-based measures where this entails $n \cdot m$ queries to a search engine.

In contrast to structural and web-based approaches operating solely on the word level, concept vector based methods using a closed collection have the advantage that longer texts can be represented similar to single words. Thus,

[1] Words are the natural unit of representation in the underlying structured resource, e.g. lexical units for WordNet and concepts (i.e. article names) for Wikipedia.

it is straightforward to compute the similarity of longer text segments [6,9]. Concept vector based measures are applicable both on the word and text level using exactly the same formulation, and comparison of longer texts does not require substantial extra computation (i.e. direct comparison of all word pairs).

The performance of concept vector based SR measures heavily relies on how well the underlying knowledge base can be used to assess semantic relatedness based on term cooccurrence. The Explicit Semantic Analysis (ESA) model [9] showed that Wikipedia is seemingly the most appropriate single resource for this purpose, and later work [34] showed that alternative resources can provide comparable performance (or even a noticeable advantage, e.g. for verb pairs). These results in the literature suggest that the combination of multiple resources can lead to improved performance and more robust behavior across different parts of speech at the same time.

The combination of resources has already been shown to be beneficial for word semantic relatedness [1]. Agirre et al. [1] report the best results so far on the English WS-353 [8] and RG-65 [25] datasets, by combining personalized PageRank on the WordNet graph with the contextual and syntactic dependency profiles of words over a large web-based corpus. These approaches are not straightforward to apply to longer texts, at least without significant extra computation (see above). Thus, here we combine resources using concept vector measures and also employ a thorough extrinsic evaluation of combination using three NLP applications.

3 Concept Vector Based Semantic Relatedness

Concept vector based SR methods represent words as a vector of articles in a specific document collection describing world knowledge (each document representing a real world concept). Semantic relatedness is then calculated using a vector similarity function. Formally, for a content word t, the concept vector $\overrightarrow{t}$ is defined as $\overrightarrow{t} = \{w_{c_1 t}, w_{c_2 t}, \ldots, w_{c_n t}\}$, where $w_{c_i t}$ represents the weight of the concept c_i for the word t (e.g. the term frequency of t in c_i) and n is the collection size. The relatedness of terms t_1 and t_2 can thus be calculated using a vector similarity measure, e.g. cosine similarity: $sim_{cosine}(t_1, t_2) = \frac{\sum_i w_{c_i t_1} \cdot w_{c_i t_2}}{\sqrt{\sum_i w_{c_i t_1}^2} \cdot \sqrt{\sum_i w_{c_i t_2}^2}}$.

Following [6,9], longer text segments can be represented using the centroid of the individual term concept vectors. The relatedness of two text segments can then be determined using the same vector similarity function as on the word level. As a result, it is unnecessary to compute the relatedness between all word pairs in the respective documents to get the document-level relatedness score.

3.1 Concept Vector Based SR Parameters

At the core of concept vector based methods is measuring the term cooccurrence statistics over Wikipedia (or a similar resource). The most important technical parameters of such a measure are:

Vector Similarity Function. An arbitrary $f(t_1, t_2) \mapsto \Re$ vector similarity function can be used to compare two concept vectors. Thereby, (t_1, t_2) is considered more similar to each other than (t_3, t_4) if $f(t_1, t_2) > f(t_3, t_4)$.

Gabrilovich and Markovitch [9] used the cosine similarity, and recently Hassan and Mihalcea [13] proposed a Lesk-like [18] vector similarity function, and argued that it is more suitable for cross-language relatedness. We assume that the concept vectors are normalized, and simplify the formulas accordingly:

- $sim_{dotprod}(t_1, t_2) = \sum_i w_{c_i t_1} \cdot w_{c_i t_2}$
- $sim_{Lesk}(t_1, t_2) = \sum_i (w_{c_i t_1} + w_{c_i t_2})$, if both $w_{c_i t_1} > 0$ and $w_{c_i t_2} > 0$,

where t_1 and t_2 denote terms, and $w_{c_i t_j}$ denotes the weight of the concept (document) c_i in the knowledge base, for the term t_j.

Component Weights. Each concept in the underlying knowledge source has to be assigned a weight in a term's concept vector. The weight is usually defined as a function of the term's frequency in the descriptive text of the concept. Thus, terms that are not used in the descriptive text of a concept are naturally assigned a weight of 0 in the corresponding concept vector. Gabrilovich and Markovitch [9] reported to use *TF.IDF* weights, while Hassan and Mihalcea [13] used a normalized *TF* formula:

- *log TF.IDF*: $w_{c_i t} = log(TF_{c_i t} + 1) \cdot IDF_t$ (the logarithm of the number of times term t appears in document c_i, multiplied by the inverted document frequency of the term in the knowledge base),
- *normalized TF*: $w_{c_i t} = TF_{c_i t} * \log(M/|c_i|)$, where M denotes a constant representing the vocabulary size in the entire knowledge base, and $|c_i|$ represents the vocabulary size of document c_i.

Normalization. In concept vector based SR, it is essential to normalize the concept vectors in order to get relatedness values that are comparable to each other. We consider two standard normalization methods: the $L1(\overrightarrow{t}) = \sum_i w_{c_i t}$ and $L2(\overrightarrow{t}) = \sqrt{\sum_i w_{c_i t}^2}$ norms (and divide each vector component by the respective norm value). Even though it is not clearly stated in the literature, we assume that all previous works on concept vector based SR used one of these normalization schemes.

Only the cosine similarity ($dotprod\,L2$) and the ($Lesk\,L1$) satisfy the criterion that for each term $sim(t,t) = 1.0$. Some combinations, like $Lesk\,L2$ might even output values larger than 1.0, but this is not important, as long as the scores of the same measure are meaningfully comparable to each other.

Concept Vector Pruning. Many studies based on reimplementations of ESA [9] mention that the performance of their system improved greatly when they applied a cutoff threshold and kept just the k highest values in each concept vector, setting very small weights to zero. Gabrilovich and Markovitch [9] employed a pruning threshold defined relative to the highest component weight in the vector (they set all weights to zero when the difference of values in a sliding

Table 1. Spearman rank correlations for different concept vector models on the EN, AR, ES and RO WS353 datasets and the DE Gur350 dataset. '?' indicates a parameter that we could not determine with certainty based on the corresponding papers.

	weights	**sim.**	**norm.**	**pruning**	**EN**	**AR**	**ES**	**RO**	**DE**
our measure	log-TF · IDF	avgprod	L2	–	**.73**	**.46**	**.51**	**.50**	**.62**
H&M reimpl.	norm-TF	Lesk	L1	–	.49	.28	.26	.29	.50
G&M reimpl.	log-TF · IDF	cosine	L2	–	.61	.25	.21	.24	.52
H&M reimpl.	norm-TF	Lesk	L1	0.01	.70	.43	.43	.43	.60
G&M reimpl.	log-TF · IDF	cosine	L2	0.001	.69	.37	.34	.36	.58
H&M 2009	norm-TF	Lesk	?	?	.71	**.26**	**.50**	**.28**	–
G&M 2007	log-TF · IDF	cosine	L2	sliding w.	**.75**	–	–	–	–
Z et al. 2008	log-TF · IDF	cosine	L2	?	.31-.62	–	–	–	**.65**

window of size 100 dropped below 5% of the highest weight), and e.g. Yeh et al. [32] reported to keep just the 625 highest values for English.

3.2 Our Concept Vector Measure

In our study we used a slightly different concept vector measure with the similarity function: $sim_{avgprod}(t_1, t_2) = \sum_i (w_{c_i t_1} + w_{c_i t_2}) \cdot w_{c_i t_1} \cdot w_{c_i t_2}$, *log TF.IDF* component weights, and $L2$ normalization. We chose to use the above implementation, because it does not require the application of an ad-hoc cutoff threshold for term vectors to remove small component weights (i.e. concepts with lower TF values for the given term) in order to show competitive performance. We consider this a positive property as pruning would be inevitably tuned on word relatedness datasets which we also use for evaluation.

3.3 Evaluation on Word Semantic Relatedness

Datasets and Evaluation measures. For word semantic relatedness, we use the Spearman rank correlation ρ and the linear Pearson correlation r of SR scores with human judgments as evaluation metrics. Spearman correlation measures how well a monotonic function can describe the relationship between an SR measure and human scores, i.e. how accurately the measure reproduces the relative ordering of word pairs (by humans), while Pearson correlation measures the linear dependence between SR and human scores.

We use publicly available word relatedness datasets for five languages in our experiments. For English, we use the WordSimilarity 353 dataset (EN-WS353) [8]. For German, we use the dataset (DE-Gur350) provided by Gurevych [11]. For Arabic (AR-WS353), Romanian (RO-WS353), and Spanish (ES-WS353), we use the translations of the WS353 dataset provided by Hassan and Mihalcea [13].

Experimental Results. In order to compare the proposed vector measure to those used in previous works, we present results on word relatedness in five languages, with Wikipedia as the underlying knowledge resource in Table 1. We compare our measure to those proposed by Gabrilovich and Markovitch [9]

(G&M 2007)[2] and Hassan and Mihalcea [13] (H&M 2009). In order to cope with potential noise due to different preprocessing steps and Wikipedia versions, we provide the results reported in previous works, together with our reimplementation using the same Wikipedia index and preprocessing. In our reimplementation, we employed a pruning threshold relative to the index size (i.e. the number of concepts in Wikipedia for different languages), and kept the k highest values in a concept vector for $k = threshold \cdot index\_size$. For example, for the reimplementation of the H&M 2009 measure, we kept the highest 1% of the concept vector components and set all other weights to 0). We report results without pruning and with pruning (the threshold was fit to provide the best possible result on the EN-WS353 dataset).

As the results in Table 1 demonstrate, our results are in line with the performance scores reported in previous works and our proposed vector measure gives good performance with one less degree of freedom (i.e. no need of tuning a concept vector pruning threshold on the word level, to set small-weight components to zero). This different behavior of the proposed measure can be attributed to the fact that less weight is given to overlapping low-weight vector components (compared to the other vector measures used here). Our results with this configuration are comparable to (with an advantage for languages with smaller Wikipedias) the Spearman correlation values reported using concept vector based SR with Wikipedia for English (0.75) [9]; for German (0.65) [34]; and for Arabic (0.26), Romanian (0.28) and Spanish (0.50) [13]. However, differences in the Wikipedia versions, preprocessing, etc. make direct comparison to previous works difficult, this is why we replicated the corresponding methods. In our subsequent experiments, we use the parameter set described above, i.e. *avgprod* similarity function, *log TF.IDF* component weights, and *L2* norm.

4 Combination of Multiple Resources

For languages where multiple knowledge resources are available, independent concept vector based models can be constructed [34]. We propose the combination of concept vector based SR values based on different resources to construct a measure that performs well across all parts of speech. This would be crucial for a wide range of applications in NLP, and is achieved through the combination of lexical knowledge with the encyclopedic knowledge in Wikipedia. We perform experiments for German and English, using the knowledge resources *Wikipedia*, *Wiktionary*, and *WordNet/GermaNet* for combination.

4.1 Lexical Semantic Knowledge Resources

A lexical semantic knowledge resource provides textual descriptions of concepts from which the concept vectors can be constructed. We use Wikipedia, Wiktionary, and WordNet for this purpose. Before constructing the vector, we preprocess the textual descriptions using stopword removal and lemmatization (English, German) or stemming (Arabic, Romanian, Spanish).

[2] Zesch et al. [34] reimplemented the ESA model [9].

Wikipedia articles provide detailed textual descriptions for concepts. We used the JWPL Wikipedia API to access the article content. We used the dump from February 6, 2007 (English); September 20, 2009 (Arabic, Spanish); September 19, 2009 (Romanian); February 6, 2007 (German). Following Gabrilovich and Markovitch [9], we discard English Wikipedia articles with less than 100 words and 5 in- or outlinks.

Wiktionary is a multilingual, web-based *dictionary*, *thesaurus*, and *phrase book*, designed as the lexical companion to Wikipedia. In order to get rid of noise from boilerplate text, we used the JWKTL package [34] for fine-grained access to Wiktionary entries. We concatenated the content of all relation types offered by JWKTL for each concept. We used the dump from October 16, 2007 (English) and October 9, 2007 (German).

WordNet [7] and **GermaNet** [16] are lexical databases for English and German. In WordNet, we consider synsets as concept vector components and use the glosses and examples as textual descriptions. As GermaNet contains no glosses, we construct pseudo glosses by concatenating the lemmas of all concepts within a distance of three synsets from the original concept (distance understood as relation path length).

4.2 Combined Concept Vector Measure

A simple and suitable model for combination is to take the individual scores as features, and train regression models to approximate the gold standard scores. For combination, we used the Weka [12] implementations of *Linear Regression (LinReg)* and *Multilayer Perceptron (MLP)* models.

Using regression models, we expect an improved Spearman correlation as the model can learn a nontrivial (and possibly nonlinear) combination of individual values to predict human scores. This setup can also improve the Pearson correlation by seeking an optimal regression model that predicts the human-assigned relatedness values as accurately as possible on the training set.

Datasets and Experimental Setup. For English, we used the EN-WS353 dataset [8], the EN-RG65 dataset [25], and the verb relatedness dataset EN-YP130 [31]. For German, we used the translation of the RG65 dataset (DE-Gur65) and the DE-Gur350 dataset [11]. For machine learning experiments, we always used one complete dataset for evaluation. We then performed the training of regression models using the word pairs in the remaining datasets that did not appear in the actual evaluation dataset. For example, for evaluation on the English EN-YP130 dataset, we used the word pairs in the EN-WS353 and EN-RG65 datasets to train the models. This way, our scores are comparable to previous results on these datasets, as they are trained on a disjoint set of word pairs. We did not perform any parameter tuning in order to avoid using parameters that are tuned to the relatively small training sets. Thus, we used the MLP model with 50 training iterations, and all other parameters set to the default values defined by Weka.

Table 2. Spearman rank (ρ) and Pearson (r) correlations (English)

Method	**EN-WS353**		**EN-YP130**		**EN-RG65**	
	335		*126*		*65*	
	ρ	r	ρ	r	ρ	r
MLP	.781	**.762**	.701	**.722**	**.860**	**.896**
LinReg	**.790**	.661	.642	.654	.858	.820
Wikipedia	.731	.469	.394	.389	.834	.742
Wiktionary	.661	.390	.628	.462	.803	.569
WordNet	.558	.226	**.715**	.509	.811	.297

Table 3. Spearman rank (ρ) and Pearson (r) correlations (German)

Method	**DE-Gur350**		**DE-Gur65**	
	214		*50*	
	ρ	r	ρ	r
MLP	**.774**	**.756**	**.871**	**.891**
LinReg	.769	.679	.870	.809
Wikipedia	.724	.388	.784	.543
Wiktionary	.580	.379	.868	.511
GermaNet	.570	.331	.715	.561

Word Semantic Relatedness Experimental Results. As intrinsic evaluation, we present the results for combining multiple knowledge resources on word relatedness datasets in Tables 2 and 3. We consider this task as intrinsic evaluation as it directly correlates SR scores to human judgments of conceptual similarity. We can assume that the better a measure approximates human scores, the more useful it should be in various NLP applications.

In italics below the dataset name, we show the number of covered word pairs. The first table rows show the results with Multilayer Perceptron (MLP), second rows show Linear Regression (LinReg) for combination. Since all combined measures exploit concept vector based SR on WordNet/GermaNet, Wiktionary, and Wikipedia, we compare them to individual resources (rows 3-5).[3]

As we can see, the use of multiple resources consistently improves over any single resource. Zesch and Gurevych [33] found that increasing the size of the underlying collection (Wikipedia) does not exhibit such remarkable improvements in correlation with human judgments. This confirms our hypothesis that a combination should exploit the advantages of individual resources. The positive effect

[3] The slight differences between values for Wikipedia in Tables 2 and 3 compared to the *'our measure'* row of Table 1 are due to discarding word pairs not covered by Wiktionary or WordNet. This is necessary to ensure a fair comparison of models based on different resources. However, by using only the scores from resources that actually cover the particular word pair, a combined measure with maximal coverage can be constructed. In subsequent extrinsic evaluations, we always employ coverage-maximizing combined measures, assuming 0 values for words not covered by some of the resources.

of the complementarity of the knowledge in different knowledge resources is best demonstrated by the English verb dataset (EN-YP130 column), which was particularly difficult for the otherwise best performing Wikipedia-based measure – the combined measures show a remarkable improvement over the performance of Wikipedia. On the other hand, supervised models perform worse here than WordNet. This is expected, as we used the EN-WS353 and EN-RG65 datasets mostly consisting of noun pairs as the training data, so the models gave more credit to Wikipedia (which performs bad on verbs).

Apart from just one case (EN-YP130 dataset for English) the nonlinear MLP model does not show large improvement over linear regression in terms of Spearman correlation, but it largely improves Pearson correlation. Thus, we suggest the use of linear regression whenever just the ranking of objects is important for an application, as this is the simplest and probably most robust supervised model. The use of a small neural network with sigmoid nodes is a good alternative when one wants to use the measure to retain "similar objects" where "similar" is determined relative to the highest score in a set, as in such settings good Pearson correlation is important.

We consider the application of machine learning to combine the results of concept vector measures built on multiple knowledge resources promising. Our best result on EN-WS353 is competitive to Agirre et al. [1] (0.78), while preserving the favorable aspect of concept vector measures: same formulation for word and text level, and direct applicability to longer texts without the need to compute relatedness scores for all word pairs. As an additional benefit, concept vector based measures – and their combination – return numerical SR scores (not rank positions like the combination proposed by Agirre et al. that learns pairwise preferences and deduces final ranks from the comparison of all pairs). This is required by applications that need to decide whether the confidence in the returned value is sufficient (the top ranked words/documents might still have quite low SR scores).

5 Extrinsic Evaluation and Discussion

To study the beneficial effects of the combined SR measure incorporating heterogeneous knowledge resources, we compare it to single-resource baselines in solving word choice problems, classification of semantic relations, and text similarity computation.

5.1 Word Choice Problems

Word Choice problems [15] consist of a target word and four candidate words or phrases. The objective is to pick the one that is most closely related to the target. The relatedness between the target and each of the candidates is computed by a SR measure, and the candidate with the maximum semantic relatedness value is chosen.

Table 4. Accuracy and coverage in solving word choice problems (English and German)

Method	English			German		
	acc.	cov.	H	acc.	cov.	H
MLP	.742	**.997**	.851	.740	**.848**	.790
LinReg	.746	**.997**	**.853**	.770	**.848**	**.807**
Wikipedia	.600	**.997**	.749	.718	.821	.766
Wiktionary	.835	.602	.700	**.886**	.313	.463
WN / GN	**.855**	.529	.654	.637	.310	.417

Datasets and Evaluation Measures. In our experiments, we used the datasets introduced by Jarmasz and Szpakowicz [15] of 300 Word Choice (WC) problems for English, and by Zesch et al. [34] of 1008 WC problems for German. We lemmatized the target and all candidates. We employed the standard evaluation metrics for this task, i.e. we measured the accuracy (percent of WC problems solved correctly), coverage (percent of WC problems with all alternatives represented in the knowledge base and at least one with nonzero relatedness) and H (harmonic mean of the accuracy and coverage) of SR measures.

Experimental Results. In Table 4, we present results for different measures and their combination in solving word choice problems. The combined measures show very positive characteristics: the coverage is better or equal to the highest coverage of an individual resource, while the accuracy is closer to the most accurate lexical resources than Wikipedia's (which is unpaired in coverage by the other knowledge resources). Overall the combined measure gives an improvement of more than .10 (14% relative increase) in H for English and .04 (5% relative increase) for German. The best results in Table 4 are in the range of the state-of-the-art ($H = .86$ [15] for English and $H = .75$ for German [34]).

5.2 Classification of Semantic Relations between Nominals

The classification of semantic relations between nominals aims at the identification of specific relation types between nouns or base noun phrases appearing in natural language sentences collected from the web. Hendrickx et al. [14] proposed to identify and classify instances of 9 abstract semantic relations between noun phrases, i.e. *Cause-Effect, Instrument-Agency, Product-Producer, Content-Container, Entity-Origin, Entity-Destination, Component-Whole, Member-Collection, Message-Topic.* That is, given two nominals (*e1* and *e2*) in a sentence, systems have to decide whether *relation(e1,e2)* or *relation(e2,e1)* holds for one of the relation types or the nominals' relation is *other* (other relation or unrelated).

Datasets and Evaluation Measures. For the classification of semantic relations between nominals, we use the dataset (8000 train and 2717 test sentences) and standard evaluation measure [14], i.e. the macro averaged F measure for the various relation types. We also provide the classification accuracy scores that serve our goal to compare the effect of SR measures better.

Table 5. Macro average F and classification accuracy in relation classification

Method	**Train (10 fold)**		**Test**	
	macro F	acc.	macro F	acc.
MLP	**.708**	**.668**	**.689**	**.647**
Wikipedia	.694	.654	.680	.635
Baseline	.657	.621	.605	.561
MLP (no lex.)	**.585**	**.550**	–	–
Wikipedia (no lex)	.558	.524	–	–
Baseline (no lex.)	.373	.385	–	–

Experimental Results. For comparison, we employed a baseline system using *standard lexical* (word unigram and lemma uni- and bigrams), *surface* (sentence length, distance of the nouns in tokens), and *contextual* (POS uni-, bi- and trigrams, dependency relations between the nouns) features for classification. To test the added value of semantic relatedness measures, we added SR features to the baseline classifier, describing the relatedness of the nouns to be classified to a set of clue words characteristic for one of the relations (e.g. *goods, cargo, bottle* for *Content-Container*)[4].

In Table 5, we compare the performance of the relation classification system with the baseline features and with extended feature sets using Wikipedia-based SR features and SR features provided by the combined measure. We also compare SR performance without lexical features (i.e. when used in a nonlexicalized classifier). The results show consistent improvements over the Wikipedia-based measure, and huge improvements over the baseline without SR (indicating that SR incorporates useful world knowledge to the classifier model). The best results in Table 5 are in the range of the state-of-the-art performance (0.52-0.82 macro average F measure [14]), with top performance reached using richer representations than the one used here. For details, see Szarvas and Gurevych [28].

5.3 Text Similarity

Two texts are considered similar when their semantic content is closely related to each other. Text similarity computation aims at quantifying the conceptual similarity between two input texts and correlates the calculated similarity scores to the human notion of document similarity through comparison to similarity scores assigned by readers. This task has natural applications in information search and content management.

In our text similarity implementation, we use the document-level aggregation based on centroid vectors [9].

Datasets and Evaluation Measures. For text similarity experiments, we use the 1225 similarity pairs provided by Lee et al. [17], and similar to previous works we use Pearson correlation for evaluation (and also list Spearman correlation).

[4] The list of clue words, feature set and additional material can be found at `http://www.ukp.tu-darmstadt.de/data/sr-combination/`

Table 6. Pearson (r) and Spearman rank (ρ) correlations on the Lee et al. (2005) dataset

Method	full		part 1		part 2	
	r	ρ	r	ρ	r	ρ
MLP	.621	**.576**	**.743**	.616	**.757**	.567
LinReg	**.727**	.571	.702	**.630**	.721	**.570**
Wikipedia	.707	.563	.688	.615	.722	.543
Wiktionary	.500	.376	.411	.350	.563	.407
WordNet	.582	.452	.566	.436	.597	.470
G&M reimpl.	.697	.484	.704	.516	.709	.464
G&M 2007	.72	–	–	–	–	–

Experimental Results. In Table 6, we compare the combined measures to single resources and to our reimplementation of ESA [9]. We used the word similarity datasets for training the combined models. This approach has the weakness that supervised models are trained on word similarity scores which have largely different characteristics from text similarity values. For WordNet and Wiktionary, many word pairs receive 0 similarity (i.e. they do not cooccur at all in the small text definitions in these resources), which is seldom the case for document pairs. This difference in the distribution of feature values is easily noticeable in the Pearson correlation of the nonlinear combination. To mimic a more ideal setting, when combined measures are trained on a set of document pairs (with assigned similarity scores), we cut the Lee et al. dataset into two parts and report results on each part (the combined models here are trained on the other half of the dataset). Besides the unexpected behavior mentioned above, we again see a consistent improvement through combination of different knowledge sources, over single resource measures.

These results suggest that multiple knowledge sources serve as a better basis for comparison of the similarity of text pairs. Or to put that in a wider context, the individual SR measures built on different resources would be good separate features for *learning to rank* (where a similar combination of features is performed to develop improved ranking functions) [19], as their improvements add upon each other. The best results in Table 6 are in the range of the state of the art performance of 0.60 [17] to 0.77 [32] Pearson correlation (Spearman is not used by previous studies).

6 Conclusions and Future Work

This paper demonstrated that better and more robust SR measures (that are applicable to single words and texts alike) can be obtained through the combination of concept vector measures exploiting various independent knowledge resources. First, we provided a detailed overview of concept vector based semantic relatedness measures, identified the most important parameters (term weighting, vector similarity function, vector normalization, and concept vector pruning) that can have major effect on the performance of the concept vector model as an SR measure. Thus, future work should state clearly the parameters of the implementation used, or the results will become difficult to reproduce and compare. Moreover, we proposed a formulation that has one less degree of freedom – it does not require the pruning of the word vectors – and performs well for representative datasets in five languages.

Our second main contribution is the combination of concept vector based SR values computed on different underlying resources by means of machine learning. To combine relatedness measures, we used a regression framework that preserves the direct applicability of concept vector based measures to longer texts. We demonstrated that the combination of resources yields stable performance across parts of speech and consistently improves performance in word relatedness over the standard knowledge base (Wikipedia) for concept vector based measures.

Finally, we performed a thorough extrinsic evaluation using three different NLP tasks: solving word choice problems, classification of semantic relations between nominals, and text similarity. We demonstrated that the improved correlation scores of our combined measure on standard word relatedness datasets actually lead to positive effects in all these applications. Thus, these experimental evaluations prove the feasibility of our approach and the hypothesis that through combining heterogeneous knowledge sources for concept vector based semantic relatedness, more robust and accurate measures can be developed that are also applicable to longer texts. The good results in text similarity calculation also suggest that these vector similarity values based on different knowledge sources are promising candidates for separate features in learning to rank (as their positive characteristics can be combined using machine learning).

In future work, we plan to extend our model to incorporate further resources and similarity measures, and to apply these measures together with traditional Information Retrieval similarity functions, in learning to rank [19].

Acknowledgements

This work was supported by the German Ministry of Education and Research (BMBF) under grant 'Semantics- and Emotion-Based Conversation Management in Customer Support (SIGMUND)', No. 01ISO8042D, and by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program (I/82806).

References

1. Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Pasca, M., Soroa, A.: A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics, pp. 19–27 (2009)
2. Bollegala, D., Matsuo, Y., Ishizuka, M.: Measuring semantic similarity between words using web search engines. In: Proceedings of the World Wide Web Conference, pp. 757–766 (2007)
3. Budanitsky, A., Hirst, G.: Evaluating WordNet-based Measures of Semantic Distance. Computational Linguistics 32(1), 13–47 (2006)

4. Chen, H.-H., Lin, M.-S., Wei, Y.-C.: Novel association measures using web search with double checking. In: Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, pp. 1009–1016 (2006)
5. Cimiano, P., Schultz, A., Sizov, S., Sorg, P., Staab, S.: Explicit versus latent concept models for cross-language information retrieval. In: Proceedings of the 21st International Jont Conference on Artifical Intelligence, pp. 1513–1518 (2009)
6. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society for Information Science 41(6), 391–407 (1990)
7. Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
8. Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G.: Placing Search in Context: The Concept Revisited. ACM Transactions on Information Systems 20(1), 116–131 (2002)
9. Gabrilovich, E., Markovitch, S.: Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence, pp. 1606–1611 (2007)
10. Gabrilovich, E., Markovitch, S.: Wikipedia-based semantic interpretation for natural language processing. Journal of AI Research 34(1), 443–498 (2009)
11. Gurevych, I.: Using the structure of a conceptual network in computing semantic relatedness. In: Dale, R., Wong, K.-F., Su, J., Kwong, O.Y. (eds.) IJCNLP 2005. LNCS (LNAI), vol. 3651, pp. 767–778. Springer, Heidelberg (2005)
12. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update. SIGKDD Explorations 11(1), 10–18 (2009)
13. Hassan, S., Mihalcea, R.: Cross-lingual Semantic Relatedness Using Encyclopedic Knowledge. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1192–1201 (2009)
14. Hendrickx, I., Kim, S.N., Kozareva, Z., Nakov, P., Séaghdha, D.Ó., Padó, S., Pennacchiotti, M., Romano, L., Szpakowicz, S.: Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In: Proceedings of the 5th SIGLEX Workshop on Semantic Evaluation (2010)
15. Jarmasz, M., Szpakowicz, S.: Roget's Thesaurus and Semantic Similarity. In: Proceedings of Recent Advances in NLP, pp. 111–120 (2003)
16. Kunze, C.: Computerlinguistik und Sprachtechnologie. In: Lexikalisch-semantische Wortnetze, pp. 423–431. Spektrum Akad. Verlag (2004)
17. Lee, M.D., Pincombe, B., Welsh, M.: An empirical evaluation of models of text document similarity. In: Proceedings of the 27th Annual Conference of the Cognitive Science Society, pp. 1254–1259 (2005)
18. Lesk, M.: Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to tell a pine cone from an ice cream cone. In: Proceedings of the 5th Annual Int. Conference on Systems Documentation, pp. 24–26 (1986)
19. Liu, T.-Y.: Learning to rank for information retrieval. Foundations and Trends in Information Retrieval 3(3), 225–331 (2009)
20. Milne, D., Witten, I.H.: An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In: Proceedings of WikiAI, pp. 25–30 (2008)
21. Mohler, M., Mihalcea, R.: Text-to-text semantic similarity for automatic short answer grading. In: Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009), pp. 567–575 (2009)

22. Müller, C., Gurevych, I.: A Study on the Semantic Relatedness of Query and Document Terms in Information Retrieval. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1338–1347 (2009)
23. Padó, S., Lapata, M.: Dependency-based construction of semantic space models. Comput. Linguist. 33(2), 161–199 (2007)
24. Patwardhan, S., Banerjee, S., Pedersen, T.: Using Measures of Semantic Relatedness for Word Sense Disambiguation. In: Proceedings of the 4th International Conference on Intelligent Text Processing and Computational Linguistics, pp. 241–257 (2003)
25. Rubenstein, H., Goodenough, J.B.: Contextual Correlates of Synonymy. Communications of the ACM 8(10), 627–633 (1965)
26. Stevenson, M., Greenwood, M.: A Semantic Approach to IE Pattern Induction. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics, pp. 379–386 (2005)
27. Strube, M., Ponzetto, S.P.: WikiRelate! computing semantic relatedness using wikipedia. In: Proceedings of the 21st AAAI National Conference on Artificial Intelligence, pp. 1419–1424. AAAI Press, Menlo Park (2006)
28. Szarvas, G., Gurevych, I.: Tud: Semantic relatedness for relation classification. In: Proceedings of the 5th International Workshop on Semantic Evaluation, pp. 210–213 (2010)
29. Tsatsaronis, G., Panagiotopoulou, V.: A Generalized Vector Space Model for Text Retrieval Based on Semantic Relatedness. In: Proc. of the Student Research Workshop at EACL 2009, pp. 70–78 (2009)
30. Wubben, S., van den Bosch, A.: A semantic relatedness metric based on free link structure. In: Proceedings of the 8th International Conference on Computational Semantics, pp. 355–358 (2009)
31. Yang, D., Powers, D.M.W.: Verb Similarity on the Taxonomy of WordNet. In: Proceedings of the 3rd International WordNet Conference, pp. 121–128 (2006)
32. Yeh, E., Ramage, D., Manning, C.D., Agirre, E., Soroa, A.: WikiWalk: Random walks on Wikipedia for Semantic Relatedness. In: Proceedings of the Workshop on Graph-based Methods for Natural Language Processing, pp. 41–49 (2009)
33. Zesch, T., Gurevych, I.: The more the better? assessing the influence of wikipedias growth on semantic relatedness measures. In: Proceedings of the Seventh International Conference on Language Resources and Evaluation (2010)
34. Zesch, T., Müller, C., Gurevych, I.: Using Wiktionary for Computing Semantic Relatedness. In: Proceedings of the 23rd AAAI Conference on Artificial Intelligence, pp. 861–867 (2008)

Improving Text Segmentation with Non-systematic Semantic Relation

Viet Cuong Nguyen, Le Minh Nguyen, and Akira Shimazu

School of Information Science
Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi, Ishikawa 923-1211, Japan
{cuongnv,nguyenml,shimazu}@jaist.ac.jp

Abstract. Text segmentation is a fundamental problem in natural language processing, which has application in information retrieval, question answering, and text summarization. Almost previous works on unsupervised text segmentation are based on the assumption of lexical cohesion, which is indicated by relations between words in the two units of text. However, they only take into account the reiteration, which is a category of lexical cohesion, such as word repetition, synonym or superordinate. In this research, we investigate the non-systematic semantic relation, which is classified as collocation in lexical cohesion. This relation holds between two words or phrases in a discourse when they pertain to a particular theme or topic. This relation has been recognized via a topic model, which is, in turn, acquired from a large collection of texts. The experimental results on the public dataset show the advantages of our approach in comparison to the available unsupervised approaches.

Keywords: text segmentation, lexical cohesion, topic modeling.

1 Introduction

Text segmentation is one of the fundamental problems in natural language processing with applications in information retrieval, text summarization, information extraction, etc. [15]. It is a process of splitting a document or a continuous stream of text into topically coherent segments. Text segmentation methods can be divided into two categories, by the structure of the output that is linear segmentation [4,7,11,14,16,22,24] and hierarchical segmentation [20], or by the algorithms that are unsupervised segmentation or supervised segmentation. In this research, we focus on the unsupervised-linear text segmentation method. The main advantage of unsupervised approach is that it does not require labeled data and is domain independent.

Almost unsupervised text segmentation methods are based on the assumption of cohesion [10], which is a device for making connections between parts of the text. Cohesion is achieved through the use of reference, substitution, ellipsis, conjunction, and lexical cohesion. The most frequent type is lexical cohesion, which is created by using semantically related words. Halliday and Hasan, in

[10], classified lexical cohesion into two categories: reiteration and collocation. Reiteration includes word repetition, synonym, and superordinate. Collocation includes relations between words that tend to co-occur in the same contexts, which are the systematic and the non-systematic semantic relations.

The current approaches in lexical cohesion-based text segmentation only focus on the first category of lexical cohesion, reiteration. Most of them use reiteration with the assumption that the repetition of words can play as the indicator of the topic coherence in a segment and the topic incoherence between segments. By using reiteration, those approaches can compute the semantic relation between two blocks of texts via some similarity-distance measurement to determine whether they can put a segment boundary between those blocks.

The collocation is the most problematic part in lexical cohesion [10,19,22]. Morris and Hirst in [19] first tried to take into account the collocation in text structuring. However, they could only perform some manual experiments on the text due to the limitation of available electronic resources at that time. In [1], they tried to use WordNet as a device for recognizing synonym and hyponym in text segmentation as an intermediate step to summarize a text. The resource-based approach has some limitations. For instance, WordNet mainly contains relations between nouns and is not available for almost languages. On the other hand, WordNet or thesauri normally contain relation between words, which can be recognized without context, such as {*apple, orange, fruit*}. In other words, those approaches can only take into account the systematic semantic relation.

In this research, we investigate how to recognize the second relation in collocation, the *non-systematic semantic relation*, in order to improve text segmentation performance. This relation holds between two words or phrases in a discourse when they pertain to a particular theme or topic, which is normally hard to classify without context. For instance, {*paper, contribution, review*} in *conference* topic or {*translation, word, meaning*} in *language* topic are examples of classes of non-systematic semantic relation. Due to the nature of that relation, a topic model [3,6,13] estimated based on the co-occurence of words would be appropriate for recognizing it. In the scope of this paper, we attempt to use Latent Dirichlet Allocation (LDA) [3], which has many advantages and is widely adopted in comparison to previous topic model methods, such as Latent Semantic Analysis (LSA) [6] or Probabilistic Latent Semantic Indexing (pLSI) [13]. The LDA model used in this research is estimated from a very large corpus, which contains all the articles of Wikipedia—the free encyclopedia.

In the next section, we provide a concise introduction to lexical cohesion and analyses of the non-systematic semantic relation in the text segmentation task. Section 3 presents some main points of topic modeling based on LDA. Section 4 begins with a presentation of a general framework for lexical cohesion-based text segmentation algorithms. Then, we describe our text segmentation algorithm with non-systematic semantic relation. The evaluation of experiments is discussed in Sect. 5. Section 6 summarizes some related works. We present conclusion in Sect. 7.

2 Non-systematic Semantic Relation

Morris and Hirst [19] were the first to apply lexical cohesion for text segmentation. Based on the reiteration and collocation relationships in [10], they divided lexical cohesion into five types of relationships that are presented in Table 1. The reiteration includes not only identity of reference or word repetition, but also the use of synonym or superordinate. The collocation includes semantic relationships between words that often co-occur. They can be further divided into two categories of relationship: systematic semantic and non-systematic semantic.

Table 1. Five types of lexical cohesion

No.	Type of relation	Example
1	Reiteration with identity of reference	Mary bit into a *peach.* Unfortunately, the *peach* wasn't ripe.
2	Reiteration without identity of reference	Mary ate some *peaches.* She likes *peaches* very much.
3	Reiteration by means of superordinate	Mary ate a *peach.* She likes *fruit.*
4	Systematic semantic relation	Mary likes *green* apples. She does not like *red* ones.
5	Non-systematic semantic relation	Mary spent three hours in the *garden* yesterday. She was *digging* potatoes.

A systematic semantic relation holds between words or group of words that can be classified in a fairly straightforward way. For example, that relation includes antonyms, members of an ordered set such as {*one, two, three*}, members of an unordered set such as {*red, green, blue*}, or part-to-whole relationships like {*eyes, mouth, face*} [19].

A non-systematic semantic relation holds between words that tend to occur in similar lexical environments, in which, they describe things that tend to occur in similar situations or contexts. In other words, they normally belong to a particular theme or topic. On the other hand, words in this relation can have different part-of-speeches. For instance, some classes are {*paper, conference, review, presentation*} or {*language, translate, speak*}. This type of relationship is the most problematic, especially from a knowledge representation point of view. It normally holds between words in a specific context [19].

In this research, to take into account the non-systematic semantic relation, we employ a topic model. A topic model is usually estimated based on the co-occurrence of words in a large collection of documents. In the scope of this research, we use latent Dirichlet allocation (LDA) [3]. A brief description of

LDA will be presented in Sect. 3. In LDA, each word is assigned to a topic with a specific probability, and it can belong to several topics with different probabilities. The inference process of LDA will assign a topic to each word in a document.

In the real world, there are many ways to combine a group of words in a non-systematic semantic relation. If a topic model would like to assign appropriate topics to words in a text, it should be estimated from a collection that contains documents in which those words co-occur. Therefore, to cover almost co-occurrence of words, we need to estimate the topic model on a very large collection of texts, and that collection should be also topical-balanced. In our research, we estimate a LDA model from the whole collection of articles in Wikipedia, which should satisfy our requirements.

In Fig. 1, we show an inferred portion of a text with topics for content words. It is a well-known example in text segmentation entitled "Stargazers" [11]. In that example, the topic number of every word is superscripted.

The Hubble^{680} Space^{680} Telescope^{680}, one of the most important^{375} telescopes^{680} ever built^{272}, will help astronomers^{680} search^{253} for advanced^{365} life^{229} in space^{680} and may find^{664} an answer^{973} to the age-old^{617} question^{973}: are we alone in the universe^{253}.

The information^{973} collected^{827} by the Hubble^{680} will be able to test^{905} the common^{299} assumptions^{617} that we live^{365} on an average^{851} planet^{680} orbiting^{86} an average^{778} star^{86}, that our solar^{680} system^{820} must be typical^{851} of others throughout the galaxy^{86}, and that many advanced^{868} life^{229} forms^{224} have evolved^{375} in the universe^{253}.

Analysis^{827} of data^{827} sent back^{713} over the last 30 years^{713} by unmanned^{680} spacecraft^{680} from distant^{680} regions^{86} of the solar^{86} system^{820} is already seriously questioning^{973} these assumptions^{617}.

The way the earth^{224} evolved^{874} holds^{868} the key^{365} to the question^{973} of life^{229} in space^{680}.

Images^{680} of Mars^{253}, and radar^{680} pictures^{973} of Jupiter^{680}, Saturn^{680}, Uranus^{680} and Neptune^{680} show^{571} that our earth^{253} and its moon^{680} are unique^{664}—at least in the solar^{680} system^{820} .

Fig. 1. The first segment of the article "Stargazers" has been topic-assigned

In the above example, we can see some group of words that are topical-related. They are also assigned the same topic number. For instance, some typical groups are {*Hubble, telescopes, astronomers, planet, spacecraft*} in topic 680, {*orbiting, star, galaxy*} in topic 86, {*search, Mars, universe*}, and so on.

To make the example more illustrative, in Table 2, we show the top 20 most likely words of some topics corresponding to the assigned topics in the example in Fig. 1.

Table 2. Top 20 most likely words of the topic model estimated on Wikipedia

Topic	Top 20 words
86	sun stars star galaxy cluster constellation magnitude ngc galaxies spiral dwarf light orion hd years sky apparent zodiac approximately nebula ...
224	rocks rock volcanic formation formed volcano geology geological lava eruption basin deposits fault plate ago earth surface cone found volcanoes ...
253	earth planet space alien ship universe aliens worlds galaxy race planets science travel mars technology series destroyed ships spaceship fiction ...
365	human humans world race humanity beings life civilization future created nature technology people artificial form living body advanced natural survival ...
680	earth observatory solar mars moon telescope planet sun astronomical astronomy jupiter venus orbit planets comet astronomer observations saturn system planetary ...
827	phase analysis pattern patterns information methods data phases based structure determine identify study identification method specific techniques important activity detection ...

3 Topic Modeling

Latent Dirichlet Allocation (LDA) [3,9] is a probabilistic generative model that can be used to estimate the properties of multinomial observations by unsupervised learning. With respect to text modeling, LDA is a method to perform so-called Latent Semantic Analysis (LSA) [6]. It is shown that the co-occurrence structure of terms in text documents can be used to recover this latent topic structure, notably without any usage of background knowledge. Latent-topic representations of text, in turn, allows modeling of linguistic phenomena like synonymy and polysemy.

The generation process of LDA in Fig. 2 can be interpreted as follows [3]: a document containing N_m words $\boldsymbol{w}_m = \{w_{m,n}\}_{n=1}^{N_m}$ is generated by first picking a distribution over topics $\boldsymbol{\theta}_m$ from a Dirichlet distribution $\mathrm{Dir}(\boldsymbol{\alpha})$, which determines topic assignments for words in that document. Then, the topic assignment for each word placeholder $[m, n]$ is performed by sampling a particular topic $z_{m,n}$ from the multinomial distribution $\mathrm{Mult}(\boldsymbol{\theta}_m)$. Finally, a particular word $w_{m,n}$ is generated for the word placeholder $[m, n]$ by sampling from the multinomial distribution $\mathrm{Mult}(\boldsymbol{\varphi}_{z_{m,n}})$. The topics $\boldsymbol{\varphi}_k$ are sampled once for the entire corpus. K is the number of topics, M is the number of documents in the corpus, and N_m is the number of words in document m.

In this research, we use Gibbs sampling for estimating the LDA model and inferring the topic for every word of the input document. In practice, after doing topic inference on the input document, we compute the topic distribution for every block of text using the topics of words in that block, which is assigned

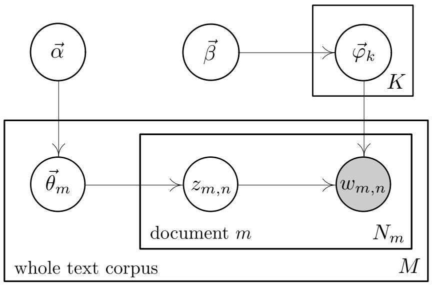


Fig. 2. The generative graphical model of LDA

by a Gibbs sampling process. The topic distribution of the text block will be used to compute the topical-similarity between every pair of text blocks. The probability of topic k of a text block i is computed as follows [12]:

$$\vartheta_{i,k} = \frac{n_i^k + \alpha_k}{\sum_{j=1}^{K} n_i^j + \alpha_k} \tag{1}$$

where n_i^k is the number of words in text block i assigned topic k, α_k is the k-th element of the vector $\boldsymbol{\alpha}$, which is the hyper parameter of the LDA model.

4 Text Segmentation Algorithm

4.1 General Framework

Algorithms for unsupervised text segmentation can be divided into two categories: lexical cohesion-based [4,11,14,16,22] and generative-based [24,7]. The lexical cohesion-based approaches can in turn be divided into lexical chain-based and similarity-based. Indeed, the difference between two sub-categories is minor, because they are also based on the principle of the lexical chain [19]. In the scope of this research, we employ the similarity-based approach. Figure 3 shows the general framework for similarity-based text segmentation.

This process could be interpreted as follows. First, a document has been split into sentences or fixed-size blocks of texts. Then, an occurrence matrix is built based on a vocabulary, in which one dimension is for sentences, and another dimension is for words in the vocabulary. To remove some gaps that are created by short sentences or sentences containing common words, some smoothing technique might be applied to the occurrence matrix. The next step is creating a similarity-distance matrix between all pairs of sentences. This matrix is normally seen as a gray-scale image, which is called DotPlot [22]. Thus, the text segmentation problem can be seen as a special case of the image segmentation problem or the graph partitioning problem. As is common in image processing, some smoothing techniques may be applied to enhance the density of some area and to reduce noise. Last, a segmentation algorithm has been applied to the similarity matrix or DotPlot image to find the boundaries of segments in the given

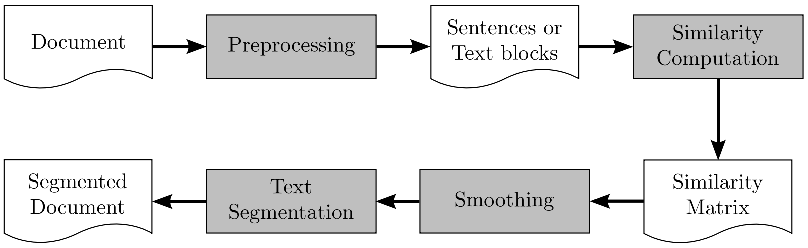


Fig. 3. The general framework of the similarity-based text segmentation algorithm

document. Although the graph partitioning problem is NP-complete, we can easily create a dynamic programming algorithm based on the linear characteristics of the text segmentation problem.

Previous approaches are normally different in the similarity matrix, the smoothing technique, or the segmentation algorithm. For example, Choi et al. [4] built the similarity matrix using the cosine similarity between all pairs of word vectors, which represent sentences. They used a rank filter for smoothing the similarity matrix. Then, similar to [22], a top-down clustering algorithm has been applied to find segment boundaries. A similar strategy was used in [14], but the anisotropic diffusion is used as the smoothing technique.

4.2 Algorithm

In this research, we follow the general framework presented in the previous section. We integrate the topical information into the similarity computation step. In the smoothing step, we apply the anisotropic diffusion technique to the image representation of the similarity matrix to reduce noise in homogeneous regions, make homogeneous regions more homogeneous, and sharpen the boundaries between homogeneous regions [14]. In the segmentation step, we re-implement the minimum cut segmentation algorithm, which has been normally applied to the image segmentation problem [23]. We follow the dynamic programming algorithm in [16] to find an exact solution for the minimum cut problem in text segmentation.

Similarity Matrix. In our model, the similarity between two sentences s_i and s_j is a linear combination of two similarity measures, the lexical-based and the topical-based.

The lexical-based similarity is the cosine of two vectors that represents word frequency in two sentences. To reduce the effect of common words in general English text, we use a vocabulary without stopwords, which does not play any role in semantic relation. On the other hand, this representation model cannot address the issue of words that are not in the stopwords list but occur throughout a text in a particular subject. Those words may play an important role in understanding the meaning of a text, but it has no effect on the text segmentation task. For instance, the word "earth" occurs in almost all sentences in the text "Stargazers" in Fig. 1. Thus, its occurrence does not mark a change in topic. To address this issue, we employ a modified version of the TF-IDF[1] weighting score [4,16], in which we split a text to chunks, which are treated as "documents". At the end, the lexical-based similarity between two sentences is computed as follows:

$$\begin{aligned}\mathrm{sim}_{\mathrm{lex}}(s_i, s_j) &= \cos(\text{tf–idf}_{s_i}, \text{tf–idf}_{s_j}) \\ &= \frac{\sum_{v \in V} \text{tf–idf}_{v,s_i} \times \text{tf–idf}_{v,s_j}}{\sqrt{\sum_{v \in V} \text{tf–idf}^2_{v,s_i}} \times \sqrt{\sum_{v \in V} \text{tf–idf}^2_{v,s_j}}}\end{aligned} \tag{2}$$

where v is a word in the vocabulary V. In practice, we use an exponential version of the above similarity to accentuate differences between low and high lexical similarities $e^{\mathrm{sim}(s_i,s_j)}$.

The topical-based similarity between two sentences is computed based on the topic distribution of those sentences, which is, in turn, computed by (1). Previous studies [3,9] normally use the Kullback–Leibler divergence (KL) for computing the similarity. However, the KL divergence is not a distance measure proper because it is not symetric. Thus, alternatively, we use the information radius (IRad) [17] as the topical-based similarity measure, which is also known as the Jenshen–Shanon divergence (JS)—a variation of the KL divergence.

$$\mathrm{IRad}(p,q) = \mathrm{JS}(p\|q) = \mathrm{KL}\left(p \,\middle\|\, \frac{p+q}{2}\right) + \mathrm{KL}\left(q \,\middle\|\, \frac{q+p}{2}\right) \tag{3}$$

IRad is symmetric ($\mathrm{IRad}(p,q) = \mathrm{IRad}(q,p)$) and there is no problem with infinite values since $\frac{p_i+q_i}{2} \neq 0$ if either $p_i \neq 0$ or $q_i \neq 0$. $\mathrm{IRad}(p,q)$ ranges from 0 for identical distributions to $2\log 2$ for maximally different distributions. The IRad is transformed to the similarity measure as follows [17]:

$$\mathrm{sim}_{\mathrm{topic}}(s_i, s_j) = 10^{-\beta \mathrm{IRad}(p_{s_i}\|p_{s_j})} \tag{4}$$

where β is a parameter that can be tuned for optimal performance. In practice, we normally choose $\beta = 1$.

The final similarity score between two sentences is the linear combination of (2) and (4), which is computed as follows:

$$\mathrm{sim}(s_i, s_j) = \lambda \mathrm{sim}_{\mathrm{lex}}(s_i, s_j) + (1-\lambda)\mathrm{sim}_{\mathrm{topic}}(s_i, s_j) \tag{5}$$

where λ is a model's parameter, which can be tuned based on the development set.

To reduce noise and reduce the number of edges in the graph that represents the similarity matrix, we use a threshold for the similarity score. This threshold is also optimized based on the development set.

[1] TF-IDF: Term Frequency–Inverse Document Frequency.

5 Experiments

In this section, we present experiments on the public dataset with the current available systems on text segmentation. Table 3 shows the list of systems that are involved in our experiments. Those systems are used with their default parameters. Some systems, such as MinCutSeg and JSeg, have been optimized on a separate development set which is extracted from the experimental dataset. Documents in the development set are not used in experiments. Softwares and data are available on `http://www.cicling.org/2011/software/174`.

Table 3. Systems involved in experiments

Name	**Description**	**Ref.**
JTextTile	The Choi's implementation of TextTiling method.	[11]
C99	A widely referred text segmentation system.	[4]
TextSeg	A generative-based text segmentation system.	[24]
MinCutSeg	A minimum cut segmentation system for spoken lectures.	[16]
JSeg	Our system without non-systematic semantic relation.	
JSegT	Our system with non-systematic semantic relation ($\lambda = 0.7$).	

5.1 Dataset

The dataset used in this research is Choi's artificial dataset [4], which has been widely used for benchmarking the performance of text segmentation algorithms [4,14,24,16,18]. This dataset contains 700 documents. Each document is a concatenation of ten text segments. Each segment, in turn, is the first n sentences of a randomly selected document from the Brown corpus[2]. Each document is characterized by the range of n. The corpus was generated automatically according to the description in [4]. Table 4 shows the dataset statistics.

Table 4. Choi's dataset

Range of n	3–11	3–5	6–8	9–11
Number of documents	400	100	100	100

The topic model has been estimated on the Wikipedia dataset, which contains 3,071,253 articles with 6,332,406 distinct words. The vocabulary used for this research contains 233,851 words. The model containing 1,000 topics has been estimated in 200 iterations by GibbsLDA++[3]. The topic inference has been done on all documents in the dataset.

[2] Brown corpus can be freely accessed via NLTK: `http://www.nltk.org`

[3] `http://gibbslda.sourceforge.net/`

5.2 Evaluation Results

Experimental results have been evaluated using two error metrics, P_k [2] and WindowDiff [21]. Low P_k or WindowDiff indicates high accuracy. Table 5 shows the evaluation of experimental results on all the systems on Choi's dataset. The P_k and WindowDiff (WD) scores have been averaged on all the documents in the whole dataset and in sub-datasets.

The parameter λ in (5) of JSegT has been set to 0.7 by tuning on the development set containing five documents.

Table 5. Experimental results

System	3–11		3–5		6–8		9–11		All	
	P_k	WD	P_k	WD	P_k	WD	P_k	WD	P_k	WD
JTextTile	0.524	0.649	0.473	0.541	0.513	0.635	0.533	0.739	0.516	0.644
C99	0.143	0.144	0.115	0.115	0.104	0.104	0.112	0.112	0.129	0.130
TextSeg	0.106	0.107	0.074	0.075	0.052	0.053	<u>0.037</u>	<u>0.037</u>	0.084	0.084
MinCutSeg	0.243	0.251	0.340	0.350	0.241	0.244	0.174	0.175	0.247	0.253
JSeg	0.129	0.130	0.091	0.091	0.107	0.107	0.121	0.126	0.119	0.121
JSegT	<u>0.035</u>	<u>0.036</u>	<u>0.020</u>	<u>0.020</u>	<u>0.030</u>	<u>0.030</u>	0.046	0.046	<u>0.034</u>	<u>0.034</u>

5.3 Discussion

The text segmentation module is normally a part of an application. Therefore, the evaluation of the performance of text segmentation systems is difficult and depends on the application. In this research, we used a widely-used artificial corpus to evaluate our system, and it may be appropriate for comparing the relative performance among text segmentation systems without applications.

The important point to notice in Table 5 is that the JSegT system yields a better result than other systems on the whole dataset. Thus, it confirms the fact that the use of the non-systematic semantic relation can help improve the performance of the text segmentation. On the other hand, the JSeg system, which employs the minimum cut segmentation from the MinCutSeg system, yields a much better result than MinCutSeg in spite of the fact that MinCutSeg was optimized on the same development set as JSeg and JSegT. The reason for this may be that the MinCutSeg was developed for long documents, such as transcripts of spoken lectures in MIT. Our systems, JSeg and JSegT, use an advanced smoothing technique that is effective for short documents.

In the scope of this paper, we also performed experiments with the TextSeg system [24], which is a representative of generative methods in text segmentation. TextSeg yields stable results in experiments, especially in the dataset 9–11. The TextSeg's approach is very different from the similarity-based approach used in this research. Therefore, if the non-systematic semantic relation could be integrated into TextSeg, it may yield potential results.

6 Related Work

Our approach takes into account the non-systematic semantic relation to improve the performance of text segmentation. Such relation has been captured by using a topic model. Choi et al. [5] used LSA in this task but as a method to reduce the number of dimensions of document. Ferret [8] tried to discover topics from the document itself without any priori knowledge. His approach is only appropriate with very long documents, which contain enough information for building the co-occurrence matrix. Eisenstein and Barzilay [7] built a LDA-based model for long documents, which generalizes the generative method introduced by Utiyama and Ishihara [24]. Misra et al. [18] follow the generative approach, in which the topic model is used to compute the topic distribution for every candidate segment to determine whether that segmentation is optimal. It is different from our approach, in which the topic information is used to directly capture the non-systematic semantic relation.

7 Conclusion

We have proposed a method to capture the non-systematic semantic relation in a text using topic modeling to improve the performance of the text segmentation algorithm. This relation has been integrated in the similarity computation process, which will directly affect the quality of the segmentation. The topic model used in this research has been estimated from a large and topic-balanced corpus, Wikipedia, which could help the text segmentation model to apply to a wide range of texts. Experimental results show that our system is more efficient than the availabe systems on Choi's dataset. In future work, we plan to incorporate the systematic semantic relation and to evaluate our method using real corpora.

Acknowledgments

The authors would like to acknowledge the Graduate Research Program of School of Information Science, Japan Advanced Institute of Science and Technology.

References

1. Barzilay, R., Elhadad, M.: Using lexical chains for text summarization. In: Proceedings of ISTS 1997, Madrid, Spain, pp. 10–17 (1997)
2. Beeferman, D., Berger, A., Lafferty, J.: Statistical models for text segmentation. Machine Learning 34(1-3), 177–210 (1999)
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Machine Learning Research 3, 993–1022 (2003)
4. Choi, F.Y.Y.: Advances in domain independent linear text segmentation. In: Proceedings of NAACL 2000, Seattle, USA, pp. 26–33 (2000)
5. Choi, F.Y.Y., Wiemer-Hastings, P., Moore, J.: Latent semantic analysis for text segmentation. In: Lee, L., Harman, D. (eds.) Proceedings of EMNLP 2001, pp. 109–117 (2001)

6. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. The American Society for Information Science 41, 391–407 (1990)
7. Eisenstein, J., Barzilay, R.: Bayesian unsupervised topic segmentation. In: Proceedings of EMNLP 2008, Honolulu, Hawaii, pp. 334–343 (2008)
8. Ferret, O.: Finding document topics for improving topic segmentation. In: Proceedings of ACL 2007, Prague, Czech Republic, pp. 480–487 (2007)
9. Griffiths, T.L., Steyvers, M.: Finding scientific topics. Proceedings of the National Academy of Sciences 101(suppl. 1), 5228–5235 (2004)
10. Halliday, M.A.K., Hasan, R.: Cohesion in English. Longman Pub. Group, Harlow (1976)
11. Hearst, M.A.: Multi-paragraph segmentation of expository text. In: Proceedings of ACL 1994, Las Cruces, New Mexico, USA, pp. 9–16 (1994)
12. Heinrich, G.: Parameter estimation for text analysis. Tech. rep., University of Leipzig, Germany (2005)
13. Hofmann, T.: Probabilistic latent semantic indexing. In: Proceedings of SIGIR 1999, pp. 50–57 (1999)
14. Ji, X., Zha, H.: Domain-independent text segmentation using anisotropic diffusion and dynamic programming. In: Proceedings of SIGIR 2003, Toronto, Canada, pp. 322–329 (2003)
15. Jurafsky, D., Martin, J.H.: Speech and Language Processing, 2nd edn. Prentice Hall, Englewood Cliffs (2008)
16. Malioutov, I., Barzilay, R.: Minimum cut model for spoken lecture segmentation. In: Proceedings of COLING-ACL 2006, Sydney, Australia, pp. 25–32 (2006)
17. Manning, C.D., Schuetze, H.: Foundations of Statistical Natural Language Processing. The MIT Press, Cambridge (1999)
18. Misra, H., Yvon, F., Jose, J.M., Cappe, O.: Text segmentation via topic modeling: an analytical study. In: Proceedings of CIKM 2009, pp. 1553–1556 (2009)
19. Morris, J., Hirst, G.: Lexical cohesion computed by thesaural relations as an indicator of the structure of text. Computational Linguistics 17(1), 21–48 (1991)
20. Otterbacher, J., Radev, D., Kareem, O.: Hierarchical summarization for delivering information to mobile devices. Information Processing and Management 44(2), 931–947 (2008)
21. Pevzner, L., Hearst, M.A.: A critique and improvement of an evaluation metric for text segmentation. Computational Linguistics 28(1), 19–36 (2002)
22. Reynar, J.C.: Topic Segmentation: Algorithms and Applications. Ph.D. thesis, University of Pennsylvania (1998)
23. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(8), 888–905 (2000)
24. Utiyama, M., Isahara, H.: A statistical model for domain-independent text segmentation. In: Proceedings of ACL 2001, Toulouse, France, pp. 499–506 (2001)

Automatic Identification of Cause-Effect Relations in Tamil Using CRFs

Menaka S., Pattabhi R.K. Rao, and Sobha Lalitha Devi

AU-KBC Research Centre, MIT Campus of Anna Univeristy,
Chennai, India
{menakas,pattabhi,sobha}@au-kbc.org

Abstract. We present our work on automatic identification of cause-effect relations in a given Tamil text. Based on the analysis of causal constructions in Tamil, we identified a set of causal markers for Tamil and arrived at certain features used to develop our language model. We manually annotated a Tamil corpus of 8648 sentences for cause-effect relations. With this corpus, we developed the model for identifying causal relations using the machine learning technique, Conditional Random Fields (CRFs). We performed experiments and the results are encouraging. We performed an error analysis of the results and found that the errors can be attributed to some very interesting structural interdependencies between closely occurring causal relations. After comparing these structures in Tamil and English, we claim that at discourse level, the complexity of structural interdependencies between causal relations is more complex in Tamil than in English due to the free word order nature of Tamil.

Keywords: Cause and Effect; CRFs, Tamil; discourse; structural interdependency; machine learning.

1 Introduction

The analysis and modeling of discourse structure has been an important area of linguistic research in recent times and it is indeed crucial for building efficient Natural language processing (NLP) applications. The automatic identification and extraction of discourse relations can improve the performance of NLP applications like Question Answering and Information Extraction. One such discourse relation, the causal relation is the focus of this paper.

Extractions of Causal relations in English [3, 4] and in other languages like Thai [11] have been attempted by researchers from the Data mining or Knowledge Acquisition perspective. Some researchers [9] have focused on recognition of discourse relations using cue phrases or detection of implicit discourse relations, but not extraction of arguments. Others [2, 18] have tried identification of the arguments of all discourse connectives in the PDTB.

Our work aims at extraction of causal relations from a text comprehension perspective i.e., we're interested in what is expressed in text rather than what is a causal relation in the real world. Our objective is to identify causal markers and the text spans of their two arguments. Our work is closer to [2] and [18] in identification of arguments, but closer to [3] and [4] in focusing on only the causal relation, which is not always marked by connectives. We have

a. Identified causal markers in Tamil and their syntactic patterns
b. Identified a set of machine learning features for these markers
c. Annotated a corpus for causal relations- causal markers and its two arguments
d. Trained and tested a corpus using CRFs for language modeling
e. Analyzed errors, found structural interdependencies between causal relations
f. Compared these structures between English and Tamil.

2 The Relation of Cause-Effect in Tamil

The cause-effect relation or the causal relation is a semantic relation between two events: E1, the cause or reason and E2, the effect. The cause is the event, E1 that causes the event, E2. In other words E1 is the reason for E2 to occur. The event, E2 is the effect which is the result or consequence of the cause. Under the taxonomy of discourse relations and definitions proposed by Mann and Thompson [8], we address the relation definitions Volitional Cause, Non-volitional cause, Volitional result and Non-volitional result as cause-effect relations in this work.

Tamil, belonging to the Dravidian family of languages, is an agglutinative language with rich morphology, and has relatively free word order. It is spoken in several countries like India, Srilanka, Myanmar, Malaysia and Singapore. Sobha Lalitha Devi and Menaka S [15] have studied the causal relation in Tamil. They have identified the causal markers or cue phrases and the syntactic patterns of the same. We have worked further and identified the features of these markers. Table 1, adapted from [15] summarizes the causal markers in Tamil and their properties.

Table 1. Causal markers and suffixes in Tamil

Causal Marker or Suffix	Lexical Marker / Suffix	Intra-sentential/ Inter-sentential	Intra-Clausal/ Inter-Clausal	Attaches to (or) follows	Remarks
-aal/-inaal	Suffix	Intra-sentential	Both	Noun phrase	
-ataal/ -atanaal/ -amaiyaal/ -apaTiyaal	Suffix	Intra-sentential	Inter-clausal	Verb	Cause is in subordinate clause and Effect is in matrix clause
kaaraNam	Lexical Marker	Both	Both	Noun Phrase	Effect precedes and Cause follows marker
kaaraNamaaka	Lexical Marker	Intra-sentential	Both	Noun Phrase/ Infinitive Verb	
kaaraNattaal	Lexical Marker	Intra-sentential	Inter-clausal	Relative Participle Verb	

Table 1. (*continued*)

aakaiyaal/ aatalaal/ aanapatiyaal	Discourse connective	Both	Inter-clausal	Sentence initial/ Clause initial	The intra-sentential form is nothing but a case of agglutination
atanaal/ itanaal	Discourse connective	Inter-sentential	-	Sentence initial	Can be decomposed to inherent pronouns *atu/itu* and causal suffix *–aal*

kaaRRu	*aTitt**ataal***	*kaakitankaL*	*paRantana.*
wind	blow-pst-CAUSE	papers	fly-pst

'Because the wind blew, the papers flew.'

The above example illustrates the cause-effect relation in Tamil. Here the cause is 'kaaRRu aTittatu/wind blew'. And the effect is 'kaakitankaL paRantana/papers flew'.

3 Automatic Cause – Effect Relation Identification

To identify and extract cause-effect relations in Tamil, we start with a set of linguistic cues, ranging from morphological suffixes to discourse connectives. To locate such linguistic cues, we need to do preprocessing of text. We have preprocessed the test for morph analysis, part-of-speech tagging (POS), chunking, clause tagging. Here, we have split the task into two stages - identification of the causal marker and identification of the arguments of the marker. We have used Conditional Random Fields (CRFs), a supervised machine learning method, for the automatic identification of cause-effect relations.

3.1 Conditional Random Fields

CRFs is an undirected graphical model, where the conditional probabilities of the output are maximized for a given input sequence [6]. We chose CRFs, because it allows linguistic rules or conditions to be incorporated into machine learning algorithm. CRFs make a first order Markov independence assumption and can be viewed as conditionally trained probabilistic finite state automata (FSA). The training of the CRFs requires iterative scaling techniques, where a quasi-Newton method such as L-BFGs is used. Here, we have used CRF++ [5], an open source toolkit for linear chain CRFs.

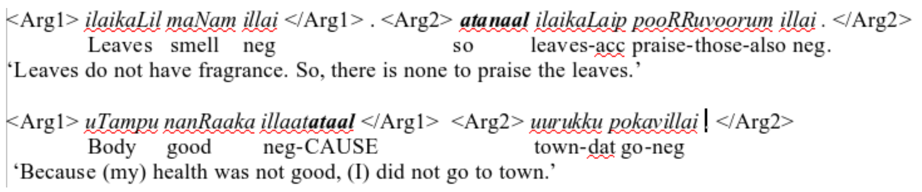
3.2 Corpus Annotation

Large-scale annotations of discourse structure like the Penn Discourse Tree Bank (PDTB) for English [13] and the Hindi Discourse Relation Bank (HDRB) [10] for Hindi make it easy for machine learning purposes in such languages. But, in the case of Tamil, since there is no Discourse Tree Bank, we identified a novel, *akal viLakku*, by *mu. varataraajan* for the training and testing corpus. It contains 8648 sentences of narrative discourse with dialogues interspersed. In-line annotation of the causal relations was done by a single annotator following a set of annotation guidelines formulated.

In PDTB, the assignment of the Arg1 and Arg2 labels to a discourse relation's arguments is syntactically driven. The two arguments to a discourse connective are simply labeled Arg2, for the argument that appears in the clause that is syntactically bound to the connective, and Arg1, for the other argument [12].

In HDRB, however, the Arg1/Arg2 label assignment is semantically driven, in that it is based on the "sense" of the relation to which the arguments belong. Thus, each sense definition for a relation specifies the sense-specific semantic role of each of its arguments, and stipulates one of the two roles to be Arg1, and the other, Arg2 [10]. We have labeled our arguments in a similar sense-specific manner. This is necessary because one causal marker may have the syntactic Arg2 as Cause and another causal marker may have the syntactic Arg1 as Cause. So, in this work, Cause is always tagged as Arg1 and Effect is tagged as Arg2.

The following examples of causal relations show how the tagging is done in the corpus in a sense-specific manner.



<Arg1> *ilaikaLil maNam illai* </Arg1> . <Arg2> ***atanaal*** *ilaikaLaip pooRRuvoorum illai* . </Arg2>
Leaves smell neg so leaves-acc praise-those-also neg.
'Leaves do not have fragrance. So, there is none to praise the leaves.'

<Arg1> *uTampu nanRaaka illaat**ataal*** </Arg1> <Arg2> *uurukku pokavillai* | </Arg2>
Body good neg-CAUSE town-dat go-neg
'Because (my) health was not good, (I) did not go to town.'

Fig. 1. Example of Tagging of Causal relations

3.3 Training and Testing

We used 6500 sentences consisting of 272 causal relationships tagged for training. The sentences had 13 different causal markers. This input data, was pre-processed with morphological analyzer [17], part-of-speech (POS) tagger [1], phrase chunker [14], and clause tagger [16]. We used the following features in CRFs.

In the first stage of identifying the causal markers, we train the system using the features of Word, POS, Chunk and Word suffix.

In the next stage we train the system to identify the arguments and their text spans. Here we have built 4 language models for each of the 4 boundaries – Arg2-START, Arg1-END, Arg1-START and Arg2-END in that order. The system was trained in 4 phases to produce the 4 models. In each phase we use the previously identified boundary also as a feature along with the features given in the table 2. We chose to first tag the boundaries (most often) close to the causal marker identified in stage 1 i.e., Arg2-START and Arg1-END.

We used 2148 sentences from the corpus consisting of 80 causal relations for testing. For testing the sentences were pre-processed similar to the training data. The system identified the causal markers in stage 1 and marked them. This output was input to stage 2. In both the stages we used CRFs as the machine learning algorithm.

It was observed that of all features, POS(b), Word Suffix(d), Type of Marker(f) and Sentence Position(i) had a significant impact on the results.

Table 2. Table of features

SNo	Features Used	Identification of Marker	Arg1-START	Arg1-END	Arg2-START	Arg2-END
a	Word	Y	N	Y	Y	Y
b	POS	Y	Y	Y	Y	Y
c	Chunk	Y	N	N	N	N
d	Word Suffix	Y	Y	Y	Y	Y
e	Causal Marker	N	Y	Y	Y	Y
f	Type of marker	N	Y	Y	Y	Y
g	Clause Type	N	Y	Y	Y	Y
h	Clause boundary	N	Y	Y	Y	Y
i	Sentence Position with respect to causal marker	N	Y	Y	Y	N
j	Combination of b and d	Y	N	N	N	N
k	Combination of i, f, d and b	N	Y	Y	Y	Y

3.4 Results and Discussion

Table 3 shows the results obtained by the system in stage 2. We observe that the precision of identification of Arg1-END and Arg2-START boundaries is better than the other two boundaries. This is due to the reason that these are nearer to the causal marker (for most of the markers). Also, since each boundary is successively identified, the error propagation of the previous model accounts for the lower precision of identification of the next boundary.

We noted that Arg1-START is generally at the start of a sentence and Arg2-END is generally at the end of a sentence. But there are cases where the cause-effect relation occurs in reported speech within quotes or in the subordinate clause of a complementizer. So, the machine tags two possible positions for these two boundaries (two Arg1-STARTs or two Arg2-ENDs for a single causal marker) in these sentences. This accounts for the lower precision of Arg1-START and Arg2-END. In fact, some of these cases are ambiguous even for a human annotator.

For instance, consider the following sentences.

(1.a) [*nee keeTT**ataal***]$_{\text{Arg1}}$ [*naan poneen enRu conneen*]$_{\text{Arg2}}$.
You ask-CAUSE I go-PST COMP say-PST
'I said that I went because you asked.'

(1.b) [*nee kuuppiTT**ataal***]$_{\text{Arg1}}$ [*naan poneen*]$_{\text{Arg2}}$ *enRu conneen.*
You call-CAUSE I go-PST COMP say-PST
'I said that I went because you called.'

Here, though both the sentences have the same syntactic structure and similar features, we interpret the scope of the effect (Arg2) differently in each case. This is because a cause denoted by the verb *keel*/ask has more probability of having an effect denoted by the verb *col*/say than having an effect denoted by the verb *po*/go in example 1.a. On the other hand, in example 1.b, cause denoted by verb *kuuppiTu*/call has more probability of having effect denoted by verb *po*/go than the verb *col*/say. Such semantics of verbs is not considered by the system and hence both the possibilities are tagged by the system.

With respect to some specific markers like *kaaraNamaaka,* the errors are more because the marker can occur in different patterns and the corpus size was too small for the machine to learn all these patterns. The system could not identify the exact boundaries when the arguments' text span was of more than two sentences. The system also failed to identify causal relations for which the causal marker was not specified explicitly. A good number of errors were due to structural interdependencies between causal relations at the discourse level. When there are such structures, there is a considerable overlap in the arguments of two causal relations leading to the improper identification of boundaries by the system. These are discussed in detail in the next section.

In 90% cases, atleast one boundary was identified correctly. The system identified atleast one argument (2 or 3 boundaries correct) properly in 82% of the cases. The system identified all 4 boundaries correctly in 55% of the cases for all causal markers.

With respect to the accuracy of identification of both arguments we get 75.35% as F-measure for all causal markers. In identifying the causal markers in stage 1, we get 92%.

Table 3. Results of extraction of causal relations using CRFs

	Total Causal Relations	System Tagged	Correctly Tagged	Precision (%)	Recall (%)	F-measure (%)
Arg1-START	80	71	52	73.24	65.00	68.87
Arg1-END	80	91	66	72.52	82.50	77.19
Arg2-START	80	94	74	78.72	92.50	85.05
Arg2-END	80	102	64	62.74	80.00	70.32
Average				71.80	80.00	75.35

4 Structural Interdependencies between Causal Relations

From the error analysis of our experiments, we found complex interdependencies and different structural patterns of causal relations. We found some structures similar to the structures discussed in [7] and some newer structures as well. We would like to highlight here that all these structures were observed not between any two discourse relations, as shown in [7], but within a single kind of discourse relation – the causal relation. These very unique patterns of interactions between the arguments of causal relations range from embedding and sharing to interleaving. Such variation in structures is possible in Tamil due to the free word order nature of Tamil. We generalized these patterns to get the possible structural interdependencies between causal relations in Tamil at the discourse level. We have also explored the possibility of such patterns in English in the following discussion.

4.1 Embedding within Itself – Cause (Arg1) within Effect (Arg2)

Within a single Causal relation, we observe an embedding of one argument within another in Tamil. The cause is expressed as an embedded adjunct. It takes the causal marker *–aal* which is suffixed to a NP. This is roughly equivalent to the markers *because of* and *due to* in English, in the sense that the cause is expressed as a NP and not as a clause. But, the difference is that both the markers *because of* and *due to* have a non-embedding structure of Marker-Arg1-Arg2 or Arg2-Arg1-Marker.

(2.a) [*appaa*	[*mutukuk*	*kaTTiy**aal***]$_{\text{Arg1}}$	*varuntikkoNTiruntaar.*]$_{\text{Arg2}}$
father	back	boil-CAUSE	suffer-PAST-PROG

'Father was suffering because of boils on his back.'

It appears that this is because of the free word order nature of Tamil. Example 2.a may be written as in 2.b where it has the trivial structure of Arg1-Marker-Arg2. Of course it can also be written as in 2.c where there is focus shift. Both 2.b and 2.c show trivial structures, though 2.a. has the most frequently occurring structure.

(2.b)[*mutukuk*	*kaTTiy**aal***]$_{\text{Arg1}}$	[*appaa*	*varuntikkoNTiruntaar.*]$_{\text{Arg2}}$
back	boil-CAUSE	father	suffer-PAST-PROG

'Because of boils on his back, father was suffering.'

(2.c)[*appaa*	*varuntiyatu*]$_{\text{Arg2}}$	[*mutukuk*	*kaTTiy**aal***].$_{\text{Arg1}}$
father	suffer-PAST	back	boil-CAUSE

'It was because of boils on his back that father was suffering.'

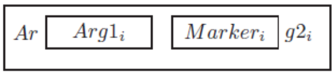


Fig. 2a. Cause (Arg1) embedded within its effect (Arg2)

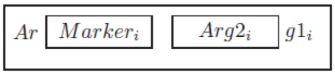


Fig. 2b. Effect (Arg2) embedded within its cause (Arg1)

4.2 Embedding within Itself – Effect (Arg2) within Cause (Arg1)

Though embedding of Arg2 within Arg1 was not observed in the errors, this is a logical possibility to be considered after encountering the structure in 4.1, for the sake of completeness (Fig.2.b). In English, we do not observe this internal embedding.

4.3 Between Two Causal Relations – Completely Independent: The Trivial Case

When we consider discourse structural interdependencies between two causal relations, we first discuss - for the sake of completeness - the trivial case of two totally independent Causal Relations occurring adjacent to each other. The structure is depicted as in Fig.3. The following example shows such a case.

(3)*onRu,* [*neTunkaalamaaka*	*makkaL*	*vaaznta*	*uur*]$_{\text{Arg1i}}$
one, longtime	people	live-RP	town
[***aakaiyaal*** *entak kiNaRRilum*	*niir*	*uppaaka irukkum.*]$_{\text{Arg2i}}$	

so all wells-LOC water salty be.
maRRonRu, [*mikap pazankaalattu uuramaippu*]$_{\text{Arg1j}}$
another very old township
[***aakaiyaal*** *terukkaL ellaam* *akalamaaka irukkum.*]$_{\text{Arg2j}}$
so streets all wide be.
‘Firstly, since it is a town where people have been living for a long time, all wells have salty water. Secondly, since it a very old township, all streets are wide.’

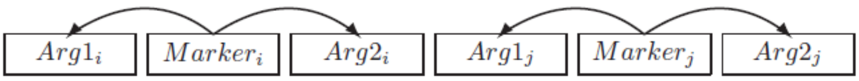


Fig. 3. Independent Causal Relations

4.4 Between Two Causal Relations – Containment

One most frequently occurring structural dependency is that of embedding or containment of the whole of a causal relation within one of the arguments of another causal relation. This argument may be the cause as illustrated by example 4.a or effect as in example 4.b. The structural dependencies are depicted in Fig. 4a and Fig. 4b.

(4.a)[*avaruTaiya manam keTTuppoovataRkuk* ***kaaraNam***]$_{\text{Arg2i}}$
his mind spoil-DAT reason
[[*enn-**aal***]$_{\text{Arg1j}}$ [*eeRpaTTa* *eemaaRRamtaan.*]$_{\text{Arg2j}}$]$_{\text{Arg1i}}$
I-CAUSE cause-RP disappointment
'The reason for the spoiling of his mind is the disappointment caused by me.’

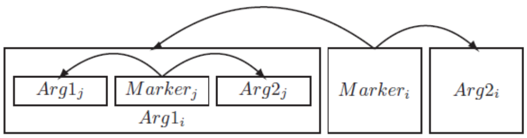


Fig. 4a. Containment: One Causal Relation within Cause(Arg1) of another

(4.b)[*ippootu pakkattu uurkaLilum paLLikaL uLLataal*]$_{\text{Arg1i}}$
now nearby towns-LOC schools be-CAUSE
[[*kalvi **kaaraNamaaka***]$_{\text{Arg1j}}$ [*varuvoorum*]$_{\text{Arg2j}}$ *kuRaintu viTTaarkaL.*]$_{\text{Arg2i}}$
education reason those who come reduced
'Since high schools have been established now in nearby towns as well, those who come for education have reduced.'

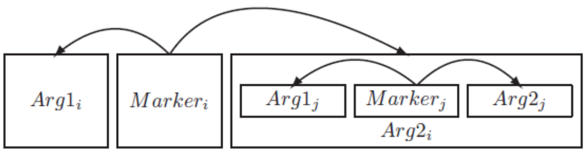


Fig. 4b. Containment: One Causal Relation within Effect (Arg2) of another

We mention embedding within Cause (Arg1) and embedding within Effect (Arg2) specifically here instead of including them both into embedding within an argument because the causal relation is asymmetrical. We observe such embedding of one causal relation within an argument of another causal relation in English as well.

4.5 Between Two Causal Relations – Sandwiching

Finding an independent causal relation between the arguments of another causal relation is very rare. It is difficult to consider that a totally independent Causal Relation occurs between the two arguments of another Causal Relation. Quite often, it is difficult to draw a line between a sandwiched Causal Relation and an embedded/contained Causal Relation. But example 5 is a clear case of sandwiching.

(5) [*cantiran*	*uNmaiyaakavee*	*maanam*	*uTaiyavan.*]$_{\text{Arg1i}}$	
Chandran	really	self-respect	have-person	
***atanaal**taan*	[*imaavatiy**aal***]$_{\text{Arg1j}}$	[*eemaaRRam*	*aTaintat*]$_{\text{Arg2j}}$ *aaka*	*uNarntatum*
so	Imavati-CAUSE	cheat	got	realized-after
[*kalluuri*	*viTutiya* *viTTee*	*ooTippooy viTTaan.*]$_{\text{Arg1i}}$		
college	hostel from	ran away.		

'Chandran really has self-respect. So, he ran away from the college hostel as soon as he realized that he got cheated by Imavati.'

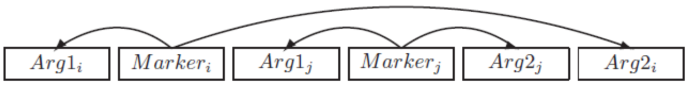


Fig. 5. Sandwiching: A causal relation between the arguments of another

This particular structure and the internal embedding structures discussed in sections 4.1 and 4.2 are newer structures not discussed in [7]. The possibility of this structure occurring in English is very rare.

4.6 Between Two Causal Relations – Complete Overlap/Shared Argument

An argument may be shared by two Causal Relations in different ways:

(a) $Arg1_i = Arg1_j$ (b) $Arg1_i = Arg2_j$ or $Arg2_i = Arg1_j$ (c) $Arg2_j = Arg2_i$

Here, the example is considered a shared argument only if Arg1 of one Causal Relation is Arg2 of another Causal Relation (as in case b above). If the shared argument is Arg1 of both the Causal Relations (case a) or Arg2 of both the Causal Relations (case c), then it is a case of multiple causes or multiple effects, respectively.

(6) [*vaNTikkaararkku*	*ippootu*	*varuvaay*	*kuRaintupoonat**aal***]$_{\text{Arg1i}}$
cart owners-DAT	now	income	reduced-CAUSE
[*avarkaL*	*kutiraikaLai*	*nanRaaka*	*vaittiruppatillai.*]$_{\text{Arg2i/Arg1j}}$
they	horses-ACC	well	keep-NEG

[***aakaiyaal***	*ippootu*	*uLLa*	*kutiraikaLum*	*paarppataRku*	*azakaaka*	*illai.*]$_{\text{Arg2j}}$
so	now	present	horses	see-DAT	beautiful	no

'As the income for cart owners has reduced now, they do not maintain the horses well. So, the horses, at present, do not look beautiful.

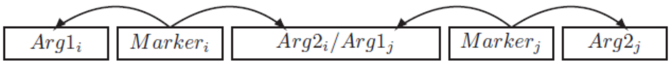


Fig. 6. Complete Overlap: Shared Argument

4.7 Between Two Causal Relations – Partial Overlap/Partially Shared Argument

This is similar to the structure in section 4.6, but one argument is shared partially as seen in example 7. These kinds of overlaps are seen in English as well.

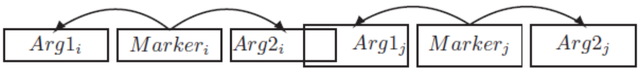


Fig. 7. Partial Overlap: Partly Shared Argument

(7) [*enakku ooyaata veelai;*]$_{\text{Arg1i}}$ [***atanaal*** [*cennaiyil kaTaikkup pooyp poruLkaLai*
I-DAT rest-no work so Chennai-LOC shop-DAT go things-ACC
vaanki koNTu uurkkup poovataRku vaayppum illai.]$_{\text{Arg2i}}$
buy-VBP town-DAT go-BEN chance no
atu tunpamaakavum irukkum.]$_{\text{Arg1j}}$ [***atanaal*** *uurkkup pookum pootellaam*
that painful-also be so town-DAT go time-always
ammaaviTam ruupaay nooTTukaLaik koTuttuviTTu tirumpuveen.]$_{\text{Arg2j}}$
mother-loc rupee notes-acc give-vbp return-past.
'I had lots of work. So, I did not have a chance to go shopping in Chennai, buy things and go to my hometown. That was also painful. So, whenever I went to my hometown, I gave my mother rupee notes and returned.'

4.8 Between Two Causal Relations – Interleaving: Pure Crisscross

Interleaving of two causal relations where the dependencies cross over is also observed as shown in example (8) and Fig. 8. This is found in English as well.

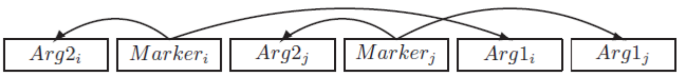


Fig. 8. Interleaving of causal relations

(8)[*meeRku naaTukaLil inta ozukkankaL iruppataRkuk* ***kaaraNam*** *uNTu.*]$_{\text{Arg2i}}$
western countries-LOC this discipline be-DAT reason Exist
[*inku illaamaikkuk* ***kaaraNam*** *uNTu.*]$_{\text{Arg2j}}$
here absence-DAT reason Exist
[*ankee piRarkku utavuvatee kaTavuLukku ukantatu ena ninaikkiRaarkaL.]* $_{Arg1i}$

there	others-DAT	help	God-DAT	liking	that think-present

[*inkee inta uNmai veLippaTaiyaaka illai.*]$_{\text{Arg1j}}$

here	this truth	explicit	no

'There is a reason why this discipline exists in western countries. There is a reason why this is absent here. There, religion has grown based on the belief that God likes us to help others. Here this truth is not explicit.'

4.9 Multiple Causes (Arg1s) or Multiple Effects (Arg2s)

Apart from the above noted structures, we found cases of multiple causes and multiple effects. Such cases might be due to shared arguments as noted in section 4.6(case a and case c), list structures or enumerations.

From the discussions in this section, we observe that structural interdependencies between causal relations in Tamil are very complex in comparison to English. This is due to the nature of the languages. Tamil has relatively free word order which allows for scrambling. The adjuncts in a clause can move freely within the clause, though long distance movement outside the clause is not possible. This allows for variations in structures, leading to structural complexities.

On analysis of the errors due to structural complexities and other reasons, we propose the following solutions to improve the system's performance. Firstly, the system can be improved by identifying and adding more features for certain markers like *kaaraNamaaka* and *-aal.* Secondly, when two causal relations are structurally interdependent, they do not always have the same causal marker. The features that significantly impact the machine learning and tagging for each marker are different. We can leverage on this fact and develop models specific to each marker or models specific to a certain group of markers of similar type (like intra-sentential or inter-sentential markers). It will give a simple but effective solution to the problem, thus improving the performance.

5 Conclusion

We presented a method for the identification and extraction of causal relations in Tamil using CRFs. The results are encouraging. We observed that the POS, Word suffix, Type of Marker and Sentence position had an impact on learning. The errors can be largely attributed to structural interdependencies between and within the causal relations. These structural patterns range from internal embedding to shared arguments to sandwiching. Though, most of these structures are found in both Tamil and English, the cases of internal embedding are unique in Tamil, due to its nature of free word order. If we can address this issue and develop specific models for each marker or for a group of markers, we can improve the performance of our system.

References

1. Arulmozhi, P., Devi, S.L.: HMM based POS Tagger for a Relatively Free Word Order Language. Journal of Research on Computing Science 18, 37–48 (2006)
2. Elwell, R., Baldridge, J.: Discourse Connective Argument Identification with Connective Specific Rankers. In: IEEE International Conference on Semantic Computing, August 4-7, pp. 198–205 (2008)

3. Girju, R.: Automatic Detection of Causal Relations for Question Answering. In: The Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003), Workshop on Multilingual Summarization and Question Answering - Machine Learning and Beyond (2003)
4. Khoo, C., Kornfilt, J., Oddy, R., Myaeng, S.H.: Automatic extraction of cause-effect information from newspaper text without knowledge-based inferencing. Literary & Linguistic Computing 13(4), 177–186 (1998)
5. Kudo, T.: CRF++, an open source toolkit for CRF (2005), `http://crfpp.sourceforge.net`
6. Lafferty, J., McCallum, A., Pereira, F.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: Proceedings of the 18th International Conference on Machine Learning (ICML 2001), pp. 282–289 (2001)
7. Lee, A., Prasad, R., Joshi, A., Dinesh, N., Webber, B.: Complexity of Dependencies in Discourse: Are Dependencies in Discourse More Complex Than in Syntax? In: Proceedings of the 5th International Workshop on Treebanks and Linguistic Theories, Prague, Czech Republic (December 2006)
8. Mann, W.C., Thompson, S.A.: Rhetorical structure theory: Toward a functional theory of text organization. Text 8(3), 243–281 (1988)
9. Marcu, D., Echihabi, A.: An Unsupervised Approach to Recognizing Discourse Relations. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002), Philadelphia, PA, July 7-12 (2002)
10. Oza, U., Prasad, R., Kolachina, S., Sharma, D.M., Joshi, A.: The Hindi Discourse Relation Bank. In: Proceedings of the Third Linguistic Annotation Workshop, Annual Meeting of the ACL, Suntec, Singapore, pp. 158–161. Association for Computational Linguistics, Morristown (2009)
11. Pechsiri, C., Sroison, P., Janviriyasopak, U.: Know-why extraction from textual data for supporting what question. In: Coling 2008: Proceedings of the Workshop on Knowledge and Reasoning For Answering Questions, Manchester, UK, ACL Workshops, pp. 17–24. Association for Computational Linguistics, Morristown (2008)
12. The PDTB Research Group: The Penn Discourse TreeBank 1.0. Annotation Manual, IRCS Technical Report IRCS-06-01, Institute for Research in Cognitive Science, University of Pennsylvania (March 2006)
13. Prasad, R., Dinesh, N., Lee, A., Miltsakaki, E., Robaldo, L., Joshi, A., Webber, B.: The Penn Discourse TreeBank 2.0. In: Proc. of LREC 2008 (2008)
14. Sobha, L., Vijay Sundar Ram, R.: Noun Phrase Chunker for Tamil. In: Proceedings of the First National Symposium on Modeling and Shallow Parsing of Indian Languages (MSPIL), IIT Mumbai, India, pp. 194–198 (2006)
15. Devi, S.L., Menaka, S.: Semantic Representation of Causality, National Seminar on Lexical Resources and Applied Computational Techniques on Indian Languages, Pondicherry University, October 4-5 (2010)
16. Vijay Sundar Ram, R., Devi, S.L.: Clause Boundary Identification Using Conditional Random Fields. In: Gelbukh, A. (ed.) CICLing 2008. LNCS, vol. 4919, pp. 140–150. Springer, Heidelberg (2008)
17. Viswanathan, S., Ramesh Kumar, S., Kumara Shanmugam, B., Arulmozi, S., Vijay Shanker, K.: A Tamil Morphological Analyser. In: Proceedings of the International Conference on Natural Language Processing (ICON), CIIL, Mysore, India (2003)
18. Wellner, B., Pustejovsky, J.: Automatically Identifiying the Arguments of Discourse Connectives. In: Proceedings of EMNLP-CoNLL (2007)

Comparing Approaches to Tag Discourse Relations

Shamima Mithun and Leila Kosseim

Concordia University,
Department of Computer Science and Software Engineering,
Montreal, Quebec, Canada
{s_mithun,kosseim}@encs.concordia.ca

Abstract. It is widely accepted that in a text, sentences and clauses cannot be understood in isolation but in relation with each other through discourse relations that may or may not be explicitly marked. Discourse relations have been found useful in many applications such as machine translation, text summarization, and question answering; however, they are often not considered in computational language applications because domain and genre independent robust discourse parsers are very few. In this paper, we analyze existing approaches to identify five discourse relations automatically (namely, *comparison, contingency, illustration, attribution*, and *topic-opinion*), and propose a new approach to identify *attributive* relations. We evaluate the accuracy of each approach with respect to the discourse relations it can identify and compare it to a human gold standard. The evaluation results show that the state of the art systems are rather effective at identifying most of the relations considered, but other relations such as *attribution* are still not identified with high accuracy.

1 Introduction

It is widely accepted that sentences and clauses in a text cannot be understood in isolation but in relation with each other. A text is not a linear combination of clauses but a hierarchial organized group of clauses placed together based on informational and interactional relations to one another. For example, in the sentence *"If you want the full Vista experience, you'll want a heavy system and graphics hardware, and lots of memory"*, the first and second clauses do not bear much meaning independently; they become more meaningful when we realize that they are related through the discourse relation *condition*.

In a discourse, different kinds of relations such as *contrast*, *causality*, *elaboration* may be expressed. The use of such discourse structures modelled by rhetorical predicates (described in section 2) have been found useful in many applications such as document summarization and question answering ([9, 7]). For example, [9] showed that rhetorical predicates can be used to select the content and generate coherent text in question answering with the help of schemata. Recently, [10] has demonstrated that rhetorical predicates can be useful in blog summarization. Rhetorical predicates have also been found useful for anaphora resolution [9] and machine translation [11].

Though rhetorical predicates are useful in many applications, their automatic identification remains a challenging task. Existing rhetorical predicate identification approaches (e.g. [9, 11]) are often domain or genre dependent. For example, in [9], predicates are identified based on the hierarchical structures and pre-stored relations in a knowledge base. In certain sub-languages, predicates are often identified by means of key words and other linguistic clues (e.g. *because*, *if*, *then*) or through verb frameworks [11]. With verb frameworks, characteristics of a verb are defined for the specified sub-language and each verb is associated with possible rhetorical predicates. [11] also used domain knowledge with verb frameworks to identify predicates.

In this paper, we focus on genre and domain independent intra-sentential rhetorical predicates identification approaches which can tag individual rhetorical predicates as opposed to performing a more complete discourse parse. Only intra-sentence predicates are considered because in many applications such as extractive summarization, question answering, and information retrieval, individual sentences are extracted from different documents or from different positions of a document to build a candidate sentence list. As a result, there is very little chance that inter-sentential relations will exist among candidate sentences. On the other hand, intra-sentential relations have already been found useful to organize texts and select content by utilizing schema in summarization and question answering [9, 10, 1]. Intra-sentential relations may enable a system to answer non-factoid questions such as "Why do people like Picasa?" by selecting clauses related through a causality; and [1] showed that 95% of the time, causality occurred within sentences in the corpus T (a gigaword newswire corpus of 4.7 million newswire documents[1]).

In this paper, we first introduce the set of rhetorical predicates which we have taken into consideration. Then we present different available approaches such as the SPADE parser [13], Jindal et al.'s [5] work, and Fei et al.'s [3] work that can be used to identify these rhetorical predicates. We have also developed an approach to identify the *attributive* predicate. We then evaluate the performance of each of these approaches using precision, recall, and F-Measure. We have also developed gold standards for the identification of each predicate to evaluate the effectiveness of these approaches. The evaluation results show that the current state of the art is acceptable to identify some predicates (e.g. *illustration*) but not others (e.g. *attribution*).

2 Rhetorical Predicates

Rhetorical predicates are the means which a speaker has to describe information. Rhetorical predicates describe different predicating acts a speaker can use and describe the structural relations between clauses in a text. Some examples

[1] Distributed by the Linguistic Data Consortium, `http://www.ldc.upenn.edu`

are *constituency* (that provides details about sub-parts), and *attributive* (that provides details about an entity or object).

Rhetorical predicates take clauses as arguments. Clauses represent the smallest units that stand in informational or interactional relationship with other parts of texts. In this framework, clauses are classified into rhetorical predicates based on their underlying information. Rhetorical predicates classify clauses into two broad categories:

1. *A clause that contains a relation with another clause.*
2. *A clause that provides information on its own.*

In the first case, rhetorical predicates describe the relation between clauses and thus express the relationship that unite them (e.g. the *evidence* predicate creates a relation with the stated fact in order to provide support) [9]. In the second case, rhetorical predicates characterize the structural purpose of a clause (e.g. the *attributive* predicate can describe the attribute of an object). Here, a single clause can characterize a predicate. This kind of discourse structure is not considered by most of the discourse theories except rhetorical predicates.

Our work was performed within the framework of developing a query-based summarizer for blogs. Hence, we considered the predicates that were most useful for this application [10]. We considered six types of rhetorical predicates, namely *comparison*, *contingency*, *illustration*, *attribution*, *topic-opinion*, and *attributive*. The *comparison*, *contingency*, *illustration*, and *attribution* predicates are also considered by most of the work in the field of discourse such as the PDTB research group [12] and [2]. We considered two additional classes of predicates: *attributive* and *topic-opinion*.

The *attributive* predicate, also included in Grimes' predicates [4], is considered because it describes attributes or features of an object or event and is often used in query-based summarization and question answering. We introduced the *topic-opinion* predicate because by analyzing the TAC-2008 corpus[2], we have found that the discourse structures (e.g. feelings, thoughts) captured by this predicate are often used in opinionated texts. In building our predicate model, we considered all main discourse structures listed in Mann and Thompson's Rhetorical Structure Theory (RST) taxonomy [6]. These discourse structures are also considered in Grimes' and Williams' predicate lists [9]. Description of these rhetorical predicates are given below:

1. **Comparison:** Gives a comparison and contrast among different situations - e.g. *Perhaps that's why for my European taste Starbucks makes great espresso while Dunkin's stinks.* The *comparison* predicates also subsume the *contrast*, *analogy*, and *preference* predicates.
2. **Contingency:** Provides cause, condition, reason, evidence for a situation, result or claim - e.g. *The meat is good because they slice it right in front of you.* The *contingency* predicate subsumes the *explanation*, *evidence*, *reason*, *cause*, *result*, *consequence*, *background*, *condition*, *hypothetical*, *enablement*, and *purpose* predicates.

[2] `http://www.nist.gov/tac`

3. **Illustration:** Is used to provide additional information or detail about a situation - e.g. *I have a special relationship with the lovely people who work in the Dunkin' Donuts in the Harvard Square T Station in Cambridge.* The *joint*, *list*, *disjoint*, and *elaboration* predicates are subclasses of the *illustration* predicate.
4. **Attribution:** Is used to convey reported speech both direct and indirect. This predicate can also be used to express feelings, thoughts, or hopes - e.g. *I said actually I think Zillow is great.*
5. **Topic-Opinion:** Can be used to express an opinion on a specific topic; an agent can express internal feeling or belief towards an object or an event - e.g. *The thing I love about their sandwiches is the bread.*
6. **Attributive:** Provides details about an entity or an event. It can be used to illustrate a particular feature about a concept - e.g. *Mary has a pink coat.*

As stated earlier, our study focused only on these predicates but other predicates would also be interesting to consider (e.g. *antithesis*).

3 Discourse Tagging

Several approaches to automatically identify the predicates described above have been proposed; the most notable ones are: the SPADE parser [13], Jindal et al.'s approach [5], and Fei et al.'s approach [3].

3.1 The SPADE Parser

The SPADE parser [13] was developed within the framework of RST. In SPADE, a large number of fine grained discourse relations are considered compared to those in RST. The SPADE parser identifies discourse relations within a sentence by first identifying elementary discourse units (EDU)s, then identifying discourse relations between two EDUs (clauses) by following the RST theory. For example, in the sentence below, the SPADE parser identifies two clauses:

a. *[Perhaps that's why for my European taste Starbucks makes great espresso]*
b. *[while Dunkin's stinks.]*

and assigns the relation *contrast* between these two clauses.

The parser consists of two components: the *discourse segmenter* and the *discourse parser*. The *discourse segmenter* divides sentences into clauses. It uses two components for this purpose namely a *statistical model*, which assigns a probability to the insertion of a discourse boundary after each word in the sentence, and a *segmenter* which uses the probabilities computed by the model for inserting discourse boundaries. Given a sentence, this model first finds the syntactic parse tree of the sentence. Then using both lexical and syntactic features of the parse tree it determines a probability of inserting a discourse boundary. Once the discourse boundaries of a sentence are determined the *discourse parser* creates a discourse tree for the sentence. The *discourse parser* also consists of two components: a *parsing model*, which assigns a probability to every potential candidate parse tree, and the *discourse parser*, which is an algorithm for finding the best discourse tree. To find the best discourse tree, it implements a bottom-up algorithm which searches through the space of all possible discourse trees using dynamic programming. In this process, between two clauses if more than one discourse relation is available then the relation with the highest probability score (that is calculated based on their syntactic and lexical information from the training corpus) is selected.

The SPADE parser can only identify discourse structures across clauses, and cannot identify those occurring within a clause. For example, in "*Dunkin Donuts' coffee tasted better than Starbucks*" a *comparison* structure is used, but would not be identified by SPADE. However, in our analysis, we found that *comparisons*, *topic-opinion*, and *attributive* do occur within a clause. To identify these kinds of structures, the taggers described in the next sections were considered.

3.2 Jindal et al.'s Approach

In order to label a clause as containing a *comparison* predicate, Jindal et al.'s approach [5] can be used. In this approach, using a set of keywords and annotated texts, the classifier first generates patterns for comparison sentence mining called sequences.

To build the sequence database, the classifier first considers the sentences which contain at least one predefined keyword. Then it creates a sequence using words which occur within a window of 3 words around the keyword. In the next step, these words are replaced with their part of speech (POS) tag and a class is associated with the sequence based on whether this sentence is a comparison or non-comparison sentence. For example, the sentence "With/IN Carmax/NNP you/PRP will/MD generally/RB always/RB pay/VB **more**/RBR than/IN from/IN going/VBG to/TO a/DT good/JJ used/VBN car/NN dealer/NN" contains the keyword "more" and the sequence will be stored in the database :

({RB}{RB}{VB}{more/RBR}{IN}{IN}{VBG}) *comparison*

After the database is constructed, class sequential rules (CSR) are generated. A CSR is a rule with sequences on the left and a class label on the right of the rule. The CSR rules are generated by combining sequences which are available in the sequence database. As CSR, those rules are accepted which meet the pre-specified *support* and *confidence* threshold value. The support and confidence of a rule are defined as follows:

$$Support\ of\ a\ rule = \frac{\#\ of\ instances\ containing\ this\ rule}{\#\ of\ instances\ in\ the\ sequence\ database}$$
$$Confidence\ of\ a\ rule = \frac{\#\ of\ instances\ containing\ this\ rule\ in\ this\ class}{\#\ of\ instances\ in\ the\ sequence\ database\ satisfying\ the\ rule}$$

A Naïve Bayes classifier is used using the CSR patterns as features to learn a 2-class classifier (comparison and non-comparison). To evaluate their approach, we have developed a classifier to identify the *comparison* predicate using their annotated dataset (see section 4).

3.3 Fei et al.'s Approach

The *topic-opinion* predicate indicates whether a sentence expresses an opinion towards a specific topic. Fei et al. [3] showed that the dependency relations of words defined by a dependency grammar are useful to find relations between a topic and subjective words.

Dependency relations refer the binary relations between two words where in this binary relation one word is the parent and the other word is the child. In this representation, one word can be associated with only one parent but with many children. In this way, when we create the dependency relations of a sentence it will be in a tree form (called a dependency tree). These dependency relations are useful to find relations (links) between subjective words and a topic. Different words of a sentence can be related using the dependency relations directly or based on the transitivity of these relations. For example, the dependency relation of the sentence *"Subway has bad food."* is shown in Figure 1.

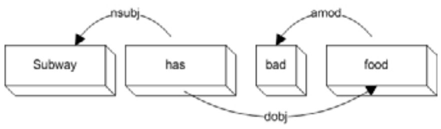


Fig. 1. Dependency Relations

The head of the arrow directs the child, the tail comes from the parent, and the tag of the arrow shows the dependency relation type. From Figure 1, we can see that both words *Subway* and *food* are children of the word *has.* The word *bad* and *food* are directly related using the dependency relation `amod`. *Subway* and *bad* are related based on the transitivity of the relations. With the help of dependency relations we can find that the topic *Subway* and the subjective word *bad* are related. Fei et al. [3] used 3 instances of dependency relations (shown in Figure 2) for opinion mining:

1. **Subjective Words that are Descendant of the Topic:** To identify whether subjective words (S-word) are descendent of the topic, subjective words should be in the modifier relation with the topic directly or based on some transitivity relations.
2. **Subjective Words and the Topic that have the Common Ancestor:** Under this category, [3] accepted instances where the same ancestor is the verb.
3. **Subjective Words that are Ancestors of the Topic:** To classify under this category according to [3], the subjective word needs to be a verb, and the topic needs to be in the subject or object of the verbs.

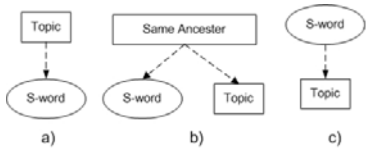


Fig. 2. Topic-opinion Dependency Relations Tree

In order to evaluate their approach, we have also built a tagger by following Fei et al.'s approach (see section 4).

3.4 Our Attributive Tagger

To our knowledge, no previous work has focused on tagging the *attributive* predicate. Hence, to identify these predicates, which typically occur within a clause, we developed our own tagger. Similarly to Fei et al.'s [3] work, we used dependency relations of words (using the Stanford parser[3]) to develop this classifier.

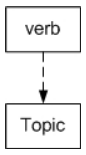


Fig. 3. Attributive Dependency Relations Tree

By analyzing TAC 2008 data, we have found that the dependency relation shown in Figure 3 can be used to identify the *attributive* predicate. From this figure, we can see that to be classified as an *attributive* predicate, the topic needs to be the descendant of a verb; however, the topic need not necessarily be directly related to the verb. From our TAC data analysis, we have also found that to classify as an *attributive* predicate, the topic needs to be in subject or object relation of the verb. We have devised a set of 5 heuristic rules by analyzing datasets containing 200 attributive sentences (sentences from TAC-2008). For example, in the sentence *"Picasa displays the zoom percentage"* there will be a dependency relation `nsubj` between the topic "*picasa*" and the verb "*displays*" (shown in Figure 4).

The 5 heuristics rules are:

1. The *verb* is directly associated with the *topic* using the dependency relation `nsubj`.
2. The *verb* is associated with a *noun* using the `nsubj` relation and that *noun* is linked with the *topic* using the dependency relation `nn`.
3. The *verb* is associated with a *noun* using the `nsubj` relation and that *noun* is linked with the *topic* using the dependency relation `prep`.

[3] Available at `nlp.stanford.edu/software/stanford-dependencies.shtml`

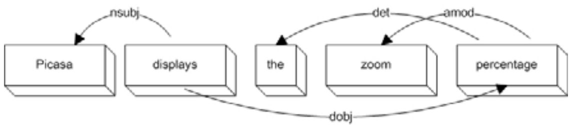


Fig. 4. Attributive Dependency Relations Example

4. The *verb* is associated with a *noun* using the `nsubj` relation and that *noun* is linked with the *topic* using the dependency relation `poss`.
5. The *verb* is directly associated with the *topic* using the dependency relation `dobj`.

4 Evaluation

This section describes the corpora and the evaluation results of the predicate taggers described above. This section also provides a comparison with a baseline and gold standard for each predicate.

4.1 Corpora

To evaluate the performance of the taggers, four different corpora have been used. The descriptions of these corpora are given below:

The SPADE Parser Corpus
To evaluate the SPADE parser, the publicly available RST Discourse Treebank 2002[4], which contains 385 Wall Street Journal articles from the Penn Treebank, was used. The dataset is divided into a training set of 347 articles (6132 sentences) and a testing set of 38 articles (991 sentences). In the corpus, for each document, a discourse tree was manually created by following Rhetorical Structure Theory (RST). In the evaluation, only discourse subtrees over individual sentences were utilized.

The Comparative Corpus
To evaluate the comparative classifier, the dataset developed by [5] was used. This corpus consists of 905 comparative and 4985 non-comparative sentences. Four human annotators labelled these data manually. This dataset consists of reviews, forum, and news articles from different sources.

The Topic-opinion Corpus
To evaluate the topic-opinion classifier, the corpus developed by [3] from the polarity dataset[5] was used. The polarity dataset originally includes 1000 positive and 1000 negative reviews on films. From this polarity dataset, [3] have

[4] Distributed by the Linguistic Data Consortium (`http://www.ldc.upenn`)
[5] `http://www.cs.cornell.edu/people/pabo/movie-review-data`

randomly annotated 400 sentences that contain both film terms and opinionated expressions from General Inquirer terms[6]. In this corpus of 400 sentences, in 262 sentences opinions are attached to the target. To annotate this corpus, 86 popular film terms from the dataset and online film glossary[7] were collected.

The Attributive Corpus

Since no standard dataset was available for the *attributive* predicates, we have manually created a corpus of 400 sentences from the TAC 2008 opinion summarization track. This corpus consists of 200 attributive sentences and 200 non-attributive sentences.

4.2 Results

For the evaluation, each approach was evaluated with its associated dataset and the performance was evaluated using precision, recall, and F-Measure scores. The SPADE parser's performance was evaluated on 18 discourse relations identification. On the other hand, the performance evaluation of all other classifiers was binary (e.g. attributive versus non-attributive).

Table 1. Performance of Different Predicate Identification Approaches

Rhetorical Predicate	**Clause Level**	**Classifier**	**Precision**	**Recall**	**F-Measure**
Comparison	Inter	SPADE	58%	31%	40%
Comparison	Intra	Jindal et al.'s	77%	81%	79%
		Authors'	66%	68%	67%
Contingency	Inter	SPADE	85%	76%	80%
Illustration	Inter	SPADE	79%	93%	85%
Attribution	Inter	SPADE	52%	83%	64%
Topic-opinion	Intra	Fei et al.'s	75%	66%	70%
		Authors'	66%	68%	67%
Attributive	Intra	Authors'	77%	76%	77%

Table 1 shows the results of the evaluation. The table indicates: a) the rhetorical predicates which have been identified; b) at what level these predicates occurred (within a clause or across two clauses); c) which classifier is used to identify the specified predicate; d) the evaluation results using precision, recall, and F-Measure.

From Table 1, we can see that to identify inter-clause *comparison*, *contingency*, *illustration*, and *attribution* predicates, the SPADE parser is used. As the evaluation of the SPADE parser was executed on 18 label relations and the performance for a specific predicate identification is not mentioned in [13], we have computed ourselves the performance of the SPADE parser for *contingency*,

[6] `http://www.wjh.havard.edu/~inquirer`

[7] `http://www.filmsite.org/filmterms.html`

comparison, *illustration*, and *attribution* predicates using the same corpus used by [13]. The performance of the SPADE parser to identify each of these predicates is shown in Table 1. The table shows the evaluation results of Jindal et al.'s approach (as published in [5]) and our implementation (authors') of their approach to identify the *comparison* predicate which occur within a clause. Table 1 also shows the evaluation results of Fei et al.'s approach (as published in [3]) and our implementation of their approach. Table 1 also shows the evaluation results of our approach to identify the *attributive* predicate.

Table 2. Baseline and Gold Standard Performance

		Baseline			**Gold Standard**		
Rhetorical Predicate	**Clause Level**	**P**	**R**	**F**	**P**	**R**	**F**
Comparison, Contingency, Illustration, Attribution	Inter	unknown	unknown	23%	unknown	unknown	77%
Comparison	Intra	94%	32%	48%	85%	92%	88%
Topic-opinion	Intra	70%	21%	32%	73%	77%	75%
Attributive	Intra	39%	67%	49%	76%	86%	81%

Table 2 shows the baseline and gold standard performance for identifying these rhetorical predicates using precision (P), recall (R), and F-Measure (F). The baseline and gold standard figures were computed as described below:

Baselines:
Inter-Comparison, Contingency, Illustration, Attribution: The SPADE baseline described in [13] was used. The baseline algorithm builds right branching discourse tree and labels with the most frequent relation learned from the training set.

Intra-Comparison**:** The baseline algorithm considers a sentence as a comparison if it contains any of the keywords of Jindal et al. [5].

Topic-opinion**:** Following [3], the baseline algorithm considers sentences as *topic-opinion* if they follow one of the two patterns below:

(RB)+JJ+(NN)+Target; ((RB)+JJ)+NN+Target

where, RB, JJ, and NN are part of speech (adverb, adjective, noun) and Target is the topic of the sentence.

Attributive**:** To be considered as an *attributive* predicate, the topic of the sentence needs to be associated with the verb using the dependency relation `subj`.

Gold Standards:
Inter-Comparison, Contingency, Illustration, and Attribution: The gold standard of [13] was used. It is computed as the agreement between two human annotators who independently annotated 53 articles of the RST Discourse Treebank corpus.

Intra-Comparison, Topic-opinion, and Attributive: The gold standards are computed as the agreement between two human annotators who annotated 100 sentences of the comparative, the topic-opinion, and the attributive corpus for each rhetorical predicate.

4.3 Analysis

In general, the state of the art approaches do much better at tagging rhetorical predicates compared to the baseline and do respectably well compared to the gold standard. As Table 1 shows, currently, the state of the art systems have difficulty tagging the rhetorical predicate *topic-opinion* - achieving an F-Measure of 70%. However, the gold standard is also very low (75%), leading us to believe that this predicate is hard to identify. The reason behind this could be it may not be marked explicitly in the text, or may be marked in a variety of ways. Moreover, sentiment identification, which is a sub-task of *topic-opinion* predicate tagging, is a complex task on its own. As a result, the F-Measure scores of the *attribution* predicate tagging, which also requires sentiment analysis, is also low. On the other hand, the rhetorical predicate *intra-comparison* is tagged satisfactorily by the state of the art systems, and the gold standard is high too. We believe that this rhetorical predicate is more explicitly marked linguistically and in a more stereotypical manner.

5 Conclusion and Future Work

In our work, we have identified a set of intra-sentential rhetorical predicates which can be expressed in factoid or opinionated texts and have analyzed domain and genre independent automatic approaches to identify these rhetorical predicates. We tried to use off-the-shelf approaches which have been developed for discourse analysis or for other purposes to identify intra-sentential discourse structures. In addition, we have introduced an automatic approach to identify the *attributive* predicate based on dependency relations. As a gold standard to evaluate the tagging of each predicate was not available, we have developed one and have used it to compare the performance of various approaches. The evaluation shows that these approaches are effective to identify some discourse structures (e.g. *illustration*) compared to others (e.g. *attribution*).

As the performance of our comparative and topic-opinion classifier is not very satisfactory, in the future we plan to conduct a manual analysis to find out why. To analyze our topic-opinion classifier's performance, we plan to evaluate its accuracy in sentiment analysis. In the future, we also plan to evaluate the usability of rhetorical predicate tagging for summarization.

Acknowledgement

The authors would like to thank the anonymous reviewers for their comments and Soricut et al. [13], Jindal et al. [5], and Fei et al.[3] for making their resources available.

This work was financially supported by NSERC.

References

[1] Blair-Goldensohn, S.J.: Long-answer Question Answering and Rhetorical-semantic Relations. Department of Computer Science, Columbia University, Columbia, USA (2007)

[2] Carlson, L., Marcu, D.: Discourse Tagging Reference Manual. University of Southern California Information Sciences Institute, ISI-TR-545 (2001)

[3] Fei, Z., Huang, X., Wu, L.: Mining the Relation between Sentiment Expression and Target Using Dependency of Words. In: PACLIC20: Coling 2006: Proceedings of the 20th Pacific Asia Conference on Language, Information and Computation, Wuhan, China, pp. 257–264 (November 2006)

[4] Grimes, J.E.: The Thread of Discourse. Cornell University, NSF-TR-1, NSF-GS-3180, Ithaca, New York (1972)

[5] Jindal, N., Liu, B.: Identifying Comparative Sentences in Text Documents. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, USA, pp. 244–251 (2006)

[6] Mann, W.C., Thompson, S.A.: Rhetorical Structure Theory: Toward a Functional Theory of Text Organisation. J. Text 3(8), 234–281 (1988)

[7] Marcu, D.: From Discourse Structures to Text Summaries. In: Proceedings of the ACL 1997/EACL 1997 Workshop on Intelligent Scalable Text Summarization, Madrid, Spain, pp. 82–88 (July 1997)

[8] Marcu, D., Echihabi, A.: An Unsupervised Approach to Recognizing Discourse Relations. In: ACL 2002: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, Philadelphia, Pennsylvania, pp. 368–375 (July 2002)

[9] McKeown, K.R.: Discourse Strategies for Generating Natural-Language Text. J. Artificial Intelligence 27(1), 1–41 (1985)

[10] Mithun, S., Kosseim, L.: A Hybrid Approach to Utilize Rhetorical Relations for Blog Summarization. In: Proceedings of TALN 2010, Montreal, Canada (July 2010)

[11] Mitkov, R.: How Could Rhetorical Relations be used in Machine Translation (And at Least Two Open Questions). In: Workshop on Intentionality And Structure In Discource Relations, pp. 86–89. Ohio State University Columbus, Ohio (1993)

[12] Prasad, R., Miltsakaki, E., Dinesh, N., Lee, A., Joshi, A., Robaldo, L., Webber, B.: The Penn Discourse Treebank 2.0. Annotation Manual. Institute for Research in Cognitive Science, University of Pennsylvania, IRCS-08-01 (2008)

[13] Soricut, R., Marcu, D.: Sentence Level Discourse Parsing using Syntactic and Lexical Information. In: NAACL 2003: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, Edmonton, Canada, pp. 149–156 (2003)

Semi-supervised Discourse Relation Classification with Structural Learning

Hugo Hernault, Danushka Bollegala, and Mitsuru Ishizuka

Graduate School of Information Science & Technology
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
hugo@mi.ci.i.u-tokyo.ac.jp, danushka@iba.t.u-tokyo.ac.jp,
ishizuka@i.u-tokyo.ac.jp

Abstract. The corpora available for training discourse relation classifiers are annotated using a general set of discourse relations. However, for certain applications, custom discourse relations are required. Creating a new annotated corpus with a new relation taxonomy is a time-consuming and costly process. We address this problem by proposing a semi-supervised approach to discourse relation classification based on Structural Learning. First, we solve a set of auxiliary classification problems using unlabeled data. Second, the learned classifiers are used to extend feature vectors to train a discourse relation classifier. By defining a relevant set of auxiliary classification problems, we show that the proposed method brings improvement of at least 50% in accuracy and F-score on the RST Discourse Treebank and Penn Discourse Treebank, when small training sets of ca. 1000 training instances are employed. This is an attractive perspective for training discourse relation classifiers on domains where little amount of labeled training data is available.

1 Introduction

Detecting the discourse relations underlying the different units of a text is crucial for several NLP applications, such as text summarization [1] or dialogue generation [2]. To date, only three major annotated corpora are available, the RST Discourse Treebank (RSTDT) [3], the Discourse Graphbank [4], and the Penn Discourse Treebank (PDTB) [5]. The RSTDT is based on the Rhetorical Structure Theory framework (RST) [6], and annotation is done using a set of 78 fine-grained discourse relations, usually grouped by researchers into a set of 18 more general relations [3]. In the Discourse GraphBank, annotation is done using a set of 11 discourse relations. Finally, in the PDTB, annotation is done in a hierarchical fashion, with 4 relations at the highest-level, and 20 at the most detailed level.

However, in some applications, we must extract discourse relations that are different from the ones defined in above-mentioned discourse theories. In [7] for instance, it is shown that the use of a RST discourse parser improves the detection of relevant information in clinical guidelines. Notably, certain RST discourse relations such as Temporal or Consequence are useful in the context of clinical information extraction. However, the majority of RST relations are too generic and not relevant enough for this task. For this application, capturing relations such as Permission, Obligation or Advice would be of greater interest. Thus, in the context of a specialized application, employing a discourse relation classifier trained on a custom set of discourse relations can be required.

A straightforward solution consists of creating a new corpus annotated with the desired set of discourse relations. However, this process is costly and time-consuming. Alternatively, it is interesting to tackle the discourse relation classification problem by employing a semi-supervised approach. While having at our disposition a small set of labeled examples, we propose to leverage freely-available unlabeled data, by employing Structural Learning [8]. In the proposed approach, unlabeled training data is employed to solve auxiliary classification tasks related to the main discourse classification problem. The unlabeled data can be obtained with a minimal effort, for instance on the web. By solving the auxiliary tasks, some information about the main discourse classification task is learnt, and encoded as new features in the main classifier's training and test feature vectors. We show that the proposed method brings a significant improvement in classification performance (F-score and accuracy), in particular when training sets of small to moderate (ca. 100 to 1000 instances) size are employed.

Our contributions in this paper are summarized as follows.

– We propose a set of auxiliary tasks related to the main discourse relation classification problem, and which can be solved using unlabeled data only. We show that incorporating these tasks into the main problem through Structural Learning brings significant improvement in F-score and classification accuracy of at least 50%, when small to moderate amounts of training data are used.
– The proposed method is evaluated on the RSTDT and PDTB corpus, and compared to a state-of-the-art semi-supervised discourse relation classification method [9].

2 Related Work

Most of the recent work on discourse relation classification have been based on either fully-supervised or unsupervised methods.

The first unsupervised approach to discourse relation classification was presented in [10]. In this work, the authors were the first to employ word pair features calculated from the two arguments of a relation. These features have the promise of capturing *implicit relations*, i.e. discourse relations not signaled by a discourse cue, such as *but*, *and* or *thus*. For instance, the presence of a word pair (*flashy, low-key*) indicates a Contrast relation.

Supervised methods have been employed to train discourse relation classifiers on the RSTDT. In [11], as a part of the sentence-level discourse sparser 'SPADE', a probabilistic model employing lexical and syntactic features is used for training a discourse relation classifier. In [12], for the same task, relation classification is done using a Support Vector Machines [13]-based classifier trained on a rich set of shallow lexical and syntactic features. In recent work [14], another RST parser based on a chart-parsing approach is presented. Here, discourse relation classification is performed using a neural network trained on syntactic and lexical features, including lexical heads.

Supervised methods have also been employed to learn discourse relation classifiers on the PDTB. In [15], an explicit discourse relation classifier is presented. Explicit relations are discourse relations signaled by a discourse cue, and the authors demonstrate that these can be classified accurately, with an F-score of 0.93. However, implicit discourse relations have been shown to be much more difficult to classify. In [16], implicit discourse relation classification on the PDTB is studied. The authors employ features such as word pairs, verb classes, modality, context, lexical features, and obtain a state-of-the-art accuracy of 0.446. In [17], for the same task, the authors also employ word pairs, as well as dependency paths, contextual information, and production rules in parse trees. They obtain an accuracy of 0.402.

Semi-supervised learning methods have been employed for a variety of tasks in NLP, such as named-entity recognition or text classification. In particular, [8] have presented the Structural Learning theory, which is based on the prediction of properties of the main classification task, using unlabeled data only. Their algorithm has been shown to perform at least as well as co-training [18] for several tasks. Structural Learning is conceptually similar to Multi-task learning [19], where related problems are learnt at the same time as the main classification problem, which enables inductive transfer and leads to a better model.

To the best of our knowledge, our previous work [9] corresponds to the first semi-supervised discourse relation classification method. In that work, a method based on the co-occurrence of features observed in unlabeled data was introduced. The degree of co-occurrence between feature pairs is first measured on a set of sentences extracted from Wikipedia[1], using the χ^2-measure [20]. Co-occurrence information is then used to extend the feature vectors of a discourse relation classifier, bringing additional information about features unseen during training. The feature set contains word pairs computed between the arguments of discourse relations, production rules from the parse trees, as well as lexical heads. This co-occurrence-based method brings significant increase in classification performance when training is done on small sets, containing few instances of certain discourse relations.

In this paper, we employ a feature set similar to [9], but propose a different semi-supervised learning method to tackle the discourse relation classification task. First, whereas the co-occurrence-based method employs unlabeled data to learn feature co-occurrences, the proposed method uses unlabeled data to solve auxiliary classification problems. Second, the co-occurrence-based method encodes co-occurrence information into a large set of new features, which is then appended to the original feature vectors. Because the size of the appended feature set depends on the number of unseen features during training, for small training sets, which correspond to a large number of unseen features, the process results

[1] `http://en.wikipedia.org`

in a considerable dimension increase, typically of ca. 10000. By contrast, in the proposed method, the information learned from unlabeled data is encoded into a compact set of new features, typically less than 100, and including these features into the classification problem does not increase dimension considerably. Nonetheless, because our experimental setting is similar, the proposed method can be directly evaluated against the feature co-occurrence-based discourse relation classification method introduced in [9].

3 Method

In this paper, we aim at learning a discourse relation classifier, given a set of labeled instances T and a set of unlabeled instances L, where typically $|L| >> |T|$. The main idea is to use the unlabeled instances to generate auxiliary tasks that are useful for discovering important properties about the structure of the main problem. If the auxiliary tasks are similar—or at least related—to the main discourse relation classification task, then we will benefit from solving them. For instance, consider the following Reason relation, holding between two discourse units in square brackets.

Reason: [Our research shows we sell more of our heavier issues] [because readers believe they are getting more for what they pay for.]

In discourse relation classification, it occurs often that a discourse relation can be predicted by observing the word pairs of its arguments. For instance, trivially, a word pair ($*$, *but*) is usually the indication of a Contrast relation. In our example, the word pair (*show*, *because*), which has been lemmatized to be made more general, is a strong indicator of the Reason relation. Intuitively, a task related to detecting the Reason relation will thus be the task of detecting the (*show*, *because*) word pair, when observing other features of the instance. A positive training instance of this new auxiliary task would be,

+(*show*, *because*): [Our research ____ we sell more of our heavier issues] [____ readers believe they are getting more for what they pay for.]

The original word pair has to be masked in order to make for an acceptable training instance. A negative training instance for this auxiliary task would be any instance not containing the word pair (*show*, *because*), such as,

−(*show*, *because*): [She has thrown extravagant soirees for crowds of people,] [but prefers more intimate gatherings.]

This auxiliary task, which is related to the task of predicting the Reason relation, can be learned using unlabeled data only. In Section 3.1, we detail the Structural Learning algorithm, and show how the information learned from solving these auxiliary tasks can be included into the main classification problem. Then, we present the features employed for this task in Section 3.2. Finally, we describe the auxiliary problem creation step in Section 3.3.

3.1 Structural Learning

In this section, we assume that we have at our disposition a training set consisting of T labeled instances $\left\{(\boldsymbol{x_t}, y_t)_{t=1}^{T}\right\}$, where the $\boldsymbol{x_t}$ are feature vectors and y_t the class labels. The feature space has dimension d.

First, we solve a set of auxiliary classification problems p_l for $l \in [1, \ldots, m]$, using linear classifiers, and find for each p_l the optimal weight vector $\hat{\boldsymbol{w_l}}$ such that,

$$\hat{\boldsymbol{w_l}} = \arg\min_{\boldsymbol{w}} \left(\sum_j L(\boldsymbol{w} \cdot x_j, p_l(x_j)) + \lambda ||\boldsymbol{w}||^2 \right) . \tag{1}$$

Here L is a loss function and λ a regularization coefficient.

Next, we stack the optimal weight vectors of each problem column-per-column, and create a matrix $W = [\hat{\boldsymbol{w_1}} \ldots \hat{\boldsymbol{w_m}}]$. In order to reduce the dimension of W, we perform on this matrix a singular value decomposition (SVD). It is noteworthy that whereas algorithms such as principal component analysis aim at reducing the dimension of the feature space, performing a SVD on W is a dimension reduction on the space of auxiliary classifiers, aimed at learning a compact representation of it.

Since typically discourse relation classifiers employ several types of heterogenous features, such as words, part-of-speech tags and word pairs, it is reasonable to perform a localized dimension reduction for each type of feature. Consequently, we perform a series of 'block' SVDs, for each type of feature employed. For each feature type f_i, $i \in [1, \ldots, n]$, whose index in the feature space starts at position s_i, and ends at position e_i, we create a feature type-specific structural parameter matrix θ_i so that,

$$U_i, D_i, V_i^T = \mathrm{SVD}(W_{[s_i:e_i,:]}) \tag{2}$$

$$\theta_i = U^T_{i[1:h,:]} \tag{3}$$

The number h is the number of structural features we wish to incorporate in our problem.

The complete structural parameter matrix $\theta = [\theta_1 \ldots \theta_n]$ has dimension $h \times d$, and it encodes the structure learnt by the auxiliary tasks in a low-dimension common space. We can now project each training and test feature vector of the main task on θ, and obtain a set of h new structural features, which are appended to their original feature vector. We obtain the training set,

$$\left\{ \left(\begin{bmatrix} \boldsymbol{x_t} \\ \theta \boldsymbol{x_t} \end{bmatrix}, y_t \right)_{t=1}^{T} \right\} \tag{4}$$

Finally, we rescale the extended features. As observed in [21], we found necessary to give relatively more weight to the structural features, which can be performed by rescaling them. This is done by finding a factor $k \in \mathbb{R}^{+}, k > 1$ so that,

$$\sum_{t=1}^{T} ||\theta \boldsymbol{x_t}|| = k \sum_{t=1}^{T} ||\boldsymbol{x_t}||. \tag{5}$$

This factor is found empirically, as the value that maximizes classification accuracy on a held-out dataset.

3.2 Features

We use three types of features, which have previously been employed successfully in several works presented in Section 2, including our co-occurrence-based semi-supervised method [9]: Word pairs, production rules from the parse tree, as well as features encoding the lexico-syntactic context at the border between two units of text [11]. Word pairs are lemmatized using the Wordnet-based lemmatizer of NLTK [22].

Figure 1 shows the parse tree for a sentence composed of two discourse units, which serve as arguments of a discourse relation we want to generate a feature vector from. Lexical heads have been calculated using the projection rules of [23], and annotated between brackets. Surrounded by dots is, for each argument, the minimal set of sub-parse trees containing strictly all the words of the argument.

We first extract all possible lemmatized word-pairs from the two arguments, such as (*Mr.*, *when*), (*decline*, *ask*) or (*comment*, *sale*). Next, we extract from left and right argument separately, all production rules from the sub-parse trees, such as NP $\mapsto$ NNP NNP, NNP $\mapsto$ "Sherry" or TO $\mapsto$ "to".

Finally, we encode in our features three nodes of the parse tree, which capture the local context at the connection point between the two arguments: The first node, which we call N_w, is the highest ancestor of the first argument's last word w, and is such that N_w's right-sibling is the ancestor of the second argument's first word. N_w's right-sibling node is called N_r. Finally, we call N_p the parent of N_w and N_r. For each node, we encode in the feature vector its part-of-speech (POS) and lexical head. For instance, in Figure 1, we have N_w = S(comment), N_r = SBAR(when), and N_p = VP(declined). In the PDTB, certain discourse relations have disjoint arguments. In this case, as well as in the case where the two arguments belong to different sentences, the nodes N_w, N_r, N_p cannot be defined, and their corresponding features are given the value zero.

3.3 Auxiliary Classification Problems

Following the intuition presented at the beginning of this section, we use as auxiliary tasks the prediction of word pairs in unlabeled data, when observing all other features. The creation of training data for the auxiliary task of predicting the presence of word pair (w_1, w_2) is done as follows:

1. We filter out unlabeled instances containing the word pair (w_1, w_2). These will serve as positive training examples for the auxiliary task.

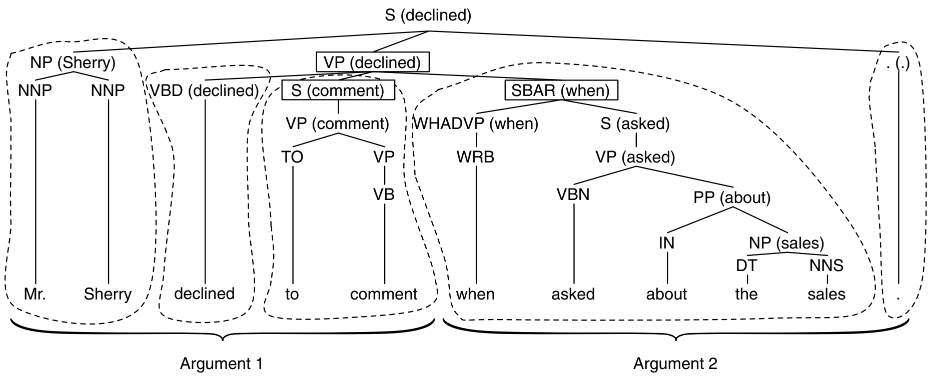


Fig. 1. Two arguments of a discourse relation, and the minimum set of subtrees that contain them—lexical heads are indicated between brackets

2. The remaining unlabeled instances, i.e. those which do not contain the word pair (w_1, w_2), will serve as negative training examples.
3. Since typically there are many more negative training examples than positive ones, there is a risk that the classifier might label every new test instance as belonging to the negative class. To prevent this issue, we cap the number of negative training examples. We empirically found that using a $2:1$ ratio of negative to positive training instances gave the best results. Using a lower ratio gave slightly worse results, while using higher ratios of negative training data did not significantly change the performance, but increased the training time of the auxiliary classifiers.
4. In positive and negative training instances, all word pair features are masked (set to zero). Although we could choose to keep certain word pairs unmasked, [8] recommend for optimal performance to mask (and predict) all features that have a good correlation to the labels of the auxiliary tasks (the other word pairs).

4 Experiments

In [8], it is shown that setting the number of structural features h between 20 and 100 does not change the results significantly. We select the intermediate value $h = 50$, which is used in all the following experiments. The factor used to rescale structural features is empirically set to five, which is consistent with the results of [21].

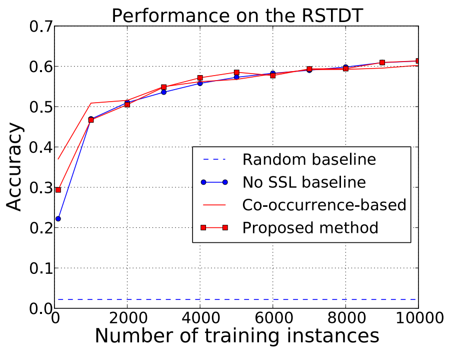
We employ as our unlabeled data the set of 100,000 unlabeled instances used in [9], which consists of sentences randomly extracted from Wikipedia and segmented into elementary discourse units automatically. The sentences have been parsed using the Stanford parser [24], in order to extract syntactic information. With 100,000 unlabeled training instances, it occurs often that the auxiliary classification task corresponding to the detection of a word pair will have very few positive training examples—typically around ten. To avoid incorporating into the structural features auxiliary problems whose classification performance is poor, we filter out auxiliary problems with less than 30 positive training instances. This finally results in solving 1358 auxiliary problems for RSTDT relation classification, and 1542 for PDTB.

We follow the common practice in discourse research for partitioning the discourse corpora into training and test set. For the RST classifier, the dedicated training and test sets of the RSTDT are employed. For the PDTB classifier, we conform to the guidelines of [25, 5]: The portion of the corpus corresponding to sections 2–21 of the WSJ is used for training the classifier, while the portion corresponding to WSJ section 23 is used for testing. This setting is identical to the one employed in [9].

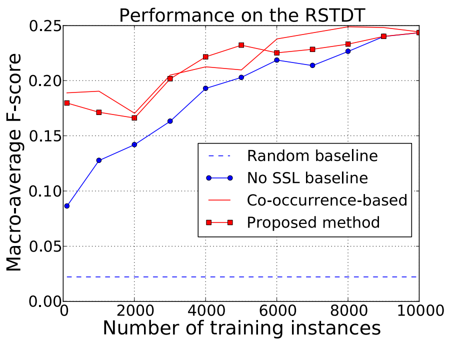
For RSTDT, we extract 25078 training vectors and 1633 test vectors. For PDTB we extract 49748 training vectors and 1688 test vectors. There are 41 classes (relation types) in the RSTDT relation classification task, and 29 classes in the PDTB task. For the PDTB, we select level-two relations, because they have better expressivity and are not too fine-grained. For our classifiers, we use the multi-class logistic regression (maximum entropy model) implemented in the Classias toolkit [27]. Regularization parameters are set to their default value of one and are fixed throughout the experiments described in the paper.

In the following experiments, we evaluate the performance of the proposed method against two baselines. The first is the 'random' baseline, in which classification decisions are made randomly. The second, noted *no SSL* in Figures 2 and 3, is the classifier trained with the same feature set, on the same training set as the proposed method, but for which no semi-supervised learning algorithm has been applied. As in [9], we employ macro-average F-score as the proposed evaluation metric. Indeed, since training sets can be imbalanced due to the prevalence of certain well-detected relations, such as ELABORATION or ATTRIBUTION in the case of the RSTDT, the micro-average F-score does not reflect accurately the classifier's performance on all classes. The macro-average F-score, which is the arithmetic mean of the F-score computed for each class, considers each class with equal importance.

We first measure the performance on the RSTDT when 100 to 10000 training instances are used. For each training set size, all classifiers are trained with the same instances. Results are indicated in Figure 2. We observe that the proposed method improves accuracy compared to the *no SSL* baseline only for 100 training instances. For both the proposed method and the co-occurrence-based method [9], above 2000 training instances, accuracy scores are as high as the *no SSL* baseline. However, we see a clear performance improvement over *no SSL* in terms of macro-average F-score. For 100 training instances, this baseline classifier has a macro-average F-score of 0.086. The classifier trained with the proposed method reaches a macro-average F-score of 0.180 (+108.34% score increase over the *no SSL* baseline), while the co-occurrence-based classifier obtains an F-score of 0.189 (+119% increase over *no SSL*). For 1000 training instances, the *no SSL* baseline has an F-score of 0.127, while the classifier trained with the proposed method reaches an F-score of 0.171 (+34.38% over *no SSL*). The co-occurrence-based classifier obtains a slightly higher F-score of 0.191 (+49.18% over *no SSL*). From 1000 to 9000 training instances, we observe in each case an F-score increase over *no SSL*, although the relative performance gain diminishes gradually. Finally, when 10000 training instances are used, both semi-supervised methods obtain the same F-score as *no SSL*, at around 0.244. As in the case of co-occurrence-based discourse relation classification [9], we observe that the proposed method is most efficient when small training sets are employed, whereas there is no performance gain when using larger sets of 10000 training instances.



(a) Accuracy

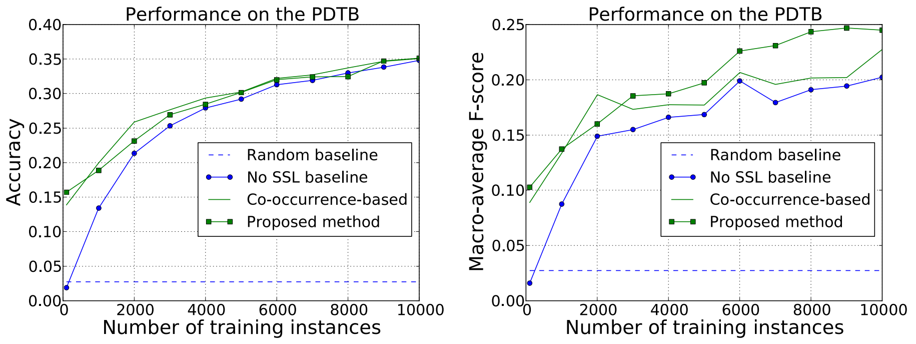


(b) Macro-average F-score

Fig. 2. Scores on the RSTDT, as a function of the number of training instances used

Similarly, we measure the performance of the proposed method on the PDTB. The results of this experiment are indicated in Figure 3. We observe a similar trend as in the case of the RSTDT experiments. For 100 training instances, the *no SSL* baseline has an extremely low accuracy of 0.019 and a macro-average F-score of 0.016. However, the classifier trained with the proposed method reaches a respective accuracy and F-score of 0.157 (+726.84% score change over the *no SSL* baseline) and 0.103 (+545.91% over *no SSL*). These scores are slightly higher than the co-occurrence-based classifier, which reaches respective accuracy and F-score of 0.139 (+630% over *no SSL*) and 0.089 (+459.12% over *no SSL*). When 1000 training instances are employed, using semi-supervised methods results in a clear improvement both in accuracy and F-score. The *no SSL* baseline obtains an accuracy of 0.134 and F-score of 0.087, while the proposed method reaches a respective accuracy and F-score of 0.189 (+40.75% over *no SSL*) and 0.137 (+56.91% over *no SSL*). In this case, the co-occurrence-based method obtains a respective accuracy and F-score of 0.199 (+48.73% over *no SSL*) and 0.134 (+52.69% over *no SSL*). On this dataset, the proposed method outperforms the co-occurrence-based method when more than 2000 training instances are used. Notably, for 9000 training instances, whereas *no SSL*'s macro-average F-score is 0.194, the proposed method reaches an F-score of 0.247 (+27.07% over *no SSL*), versus 0.202 (+3.96% over *no SSL*) for the co-occurrence-based classifier.

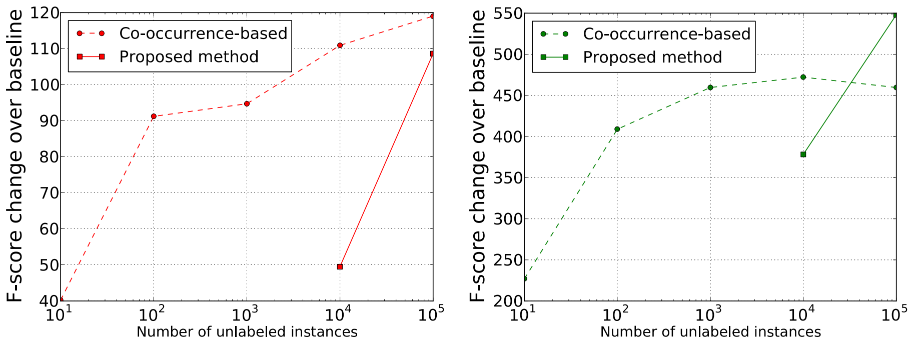
These scores are consistent with the results of Figure 2, with the exception that, for PDTB relation classification, the proposed method did improve the macro-average F-score when large training sets were used.



(a) Accuracy (b) Macro-average F-score

Fig. 3. Scores on the PDTB, as a function of the number of training instances used

An interesting property of a semi-supervised method is how its performance will be affected by the amount of unlabeled training data employed. After selecting a training set of 100 instances, we evaluate the performance of the proposed method when a variable amount of unlabeled training data is used. The results are shown in Figure 4. A first observation is that, when only 10, 100 or 1000 unlabeled instances are available, there is not enough data to train the auxiliary classifiers, and consequently the proposed method cannot be applied. On the other hand, the co-occurrence-based method performs well even with small amounts of unlabeled training data: With 10 unlabeled instances, this method increases the F-score for RSTDT by 40.2%, and by 227% for PDTB. For 10000 unlabeled training instances, it becomes possible to train the auxiliary classifiers. In this case, the proposed method scores lower than the feature co-occurrence-based method, with an F-score increase on the RSTDT of 110.9% for the co-occurrence-based method, vs. 49.5% for the proposed method. For the PDTB the F-score increase is 472.3% for the co-occurrence-based method, against 378% for the proposed method. Finally, when using the full set of $100,000$ unlabeled training instances, the performance of the proposed method increases dramatically, and becomes very close to the performance of the co-occurrence-based method. For RSTDT relation classification, we observe an F-score increase of 119.0% for the co-occurrence-based method, against 108.6% for the proposed method. However, for PDTB relation classification, the proposed method outperforms the co-occurrence-based method, with an F-score increase of 547.8% for the proposed method, against 459.7% for the co-occurrence-based method. These values confirm that, provided that we have a sufficient amount of unlabeled training data at our disposition, the proposed method performs at least as well as the co-occurrence-based discourse relation classification method [9].



(a) Performance on the RSTDT (b) Performance on the PDTB

Fig. 4. Effect of unlabeled data on the proposed method, for 100 training instances

Finally, we discuss some qualitative differences between the proposed method and the co-occurrence-based method [9]. First, whereas the co-occurrence-based method performs a large increase in the size of the feature space—dimension increase of ca. 15000 for a training set of 100 instances—the proposed method only adds a small, fixed number of features—set to 50 in our experiments. Then, the proposed method was shown to require more unlabeled data than the co-occurrence-based method, in order to train the auxiliary classification problems. Indeed, with 100 or 1000 unlabeled instances, most word pairs rarely occur in unlabeled data, which makes it impossible to train accurate auxiliary classifiers. Last, whereas the feature co-occurrence based method is independent from any classification problem or machine learning algorithm, the proposed method requires some human supervision in order to define relevant auxiliary tasks, and it requires employing linear classifiers to solve the auxiliary classification problems.

5 Conclusion

We presented a semi-supervised discourse relation classification method based on Structural Learning [8]. The method was evaluated on the RSTDT and PDTB, where it was shown to bring significant performance increase in accuracy and F-score, especially in the cases where small training sets of ca. 1000 instances were used. This is an interesting outlook for creating discourse relation classifiers on domains with little available training data.

The proposed method was compared to a feature co-occurrence-based method [9], and it was shown to perform comparably given the same amount of unlabeled data. Although the relative performance improvement over baseline classifiers is important, classification accuracy and macro-average F-score are rather low when large training sets are employed. We hypothesize that this is due to the poor detection of implicit relations, where the current state-of-the-art F-score is still modest. However, ongoing research has been focusing on finding appropriate features for this task [28,29], which has the promise of enabling us to improve classification performance.

References

1. Louis, A., Joshi, A., Nenkova, A.: Discourse indicators for content selection in summarization. In: Proc. of SIGDIAL 2010, pp. 147–156 (2010)
2. Piwek, P., Hernault, H., Prendinger, H., Ishizuka, M.: Generating dialogues between virtual agents automatically from text. In: Pelachaud, C., Martin, J.-C., André, E., Chollet, G., Karpouzis, K., Pelé, D. (eds.) IVA 2007. LNCS (LNAI), vol. 4722, pp. 161–174. Springer, Heidelberg (2007)
3. Carlson, L., Marcu, D., Okurowski, M.E.: Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory. In: Proc. of Second SIGdial Workshop on Discourse and Dialogue, vol. 16, pp. 1–10 (2001)
4. Wolf, F., Gibson, E.: Representing discourse coherence: A corpus-based study. Computational Linguistics 31, 249–287 (2005)
5. Prasad, R., Dinesh, N., Lee, A., Miltsakaki, E., Robaldo, L., Joshi, A., Webber, B.: The Penn Discourse TreeBank 2.0. In: Proc. of LREC 2008 (2008)
6. Mann, W.C., Thompson, S.A.: Rhetorical Structure Theory: Toward a functional theory of text organization. Text 8, 243–281 (1988)
7. Georg, G., Hernault, H., Cavazza, M., Prendinger, H., Ishizuka, M.: From rhetorical structures to document structure: Shallow pragmatic analysis for document engineering. In: Proc. of DocEng 2009, pp. 185–192. ACM, New York (2009)
8. Ando, R.K., Zhang, T.: A framework for learning predictive structures from multiple tasks and unlabeled data. J. Mach. Learn. Res. 6, 1817–1853 (2005)
9. Hernault, H., Bollegala, D., Ishizuka, M.: A semi-supervised approach to improve classification of infrequent discourse relations using feature vector extension. In: Proc. of EMNLP 2010, pp. 399–409 (2010)
10. Marcu, D., Echihabi, A.: An unsupervised approach to recognizing discourse relations. In: Proc. of ACL 2002, pp. 368–375 (2002)
11. Soricut, R., Marcu, D.: Sentence level discourse parsing using syntactic and lexical information. In: Proc. of NA-ACL 2003, vol. 1, pp. 149–156 (2003)
12. duVerle, D.A., Prendinger, H.: A novel discourse parser based on Support Vector Machine classification. In: Proc. of ACL 2009, pp. 665–673 (2009)
13. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, New York (1995)
14. Sagae, K.: Analysis of discourse structure with syntactic dependencies and data-driven shift-reduce parsing. In: Proc. of IWPT 2009, pp. 81–84 (2009)
15. Pitler, E., Raghupathy, M., Mehta, H., Nenkova, A., Lee, A., Joshi, A.: Easily identifiable discourse relations. In: Proc. of COLING 2008 (Posters), pp. 87–90 (2008)
16. Pitler, E., Louis, A., Nenkova, A.: Automatic sense prediction for implicit discourse relations in text. In: Proc. of ACL 2009, pp. 683–691 (2009)
17. Lin, Z., Kan, M.Y., Ng, H.T.: Recognizing implicit discourse relations in the Penn Discourse Treebank. In: Proc. of EMNLP 2009, pp. 343–351 (2009)
18. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: Proc. of COLT 1998, pp. 92–100 (1998)
19. Caruana, R.: Multitask learning: A knowledge-based source of inductive bias. In: Proc. of ICML 1993, pp. 41–48 (1993)
20. Plackett, R.L.: Karl Pearson and the chi-squared test. International Statistical Review / Revue Internationale de Statistique 51, 59–72 (1983)
21. Blitzer, J., McDonald, R., Pereira, F.: Domain adaptation with structural correspondence learning. In: Proc. of EMNLP 2006, pp. 120–128 (2006)

22. Loper, E., Bird, S.: NLTK: The natural language toolkit. In: Proc. of ACL 2002 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics, pp. 63–70 (2002)
23. Magerman, D.M.: Statistical decision-tree models for parsing. In: Proc. of ACL 1995, pp. 276–283 (1995)
24. Klein, D., Manning, C.D.: Fast exact inference with a factored model for natural language parsing. In: Advances in Neural Information Processing Systems, vol. 15. MIT Press, Cambridge (2003)
25. Prasad, R., Miltsakaki, E., Dinesh, N., Lee, A., Joshi, A., Robaldo, L., Webber, B.: The Penn Discourse Treebank 2.0 annotation manual. Technical report, University of Pennsylvania Institute for Research in Cognitive Science (2008)
26. Marcus, M.P., Marcinkiewicz, M.A., Santorini, B.: Building a large annotatedcorpus of English: The Penn Treebank. Computational Linguistics 19, 313–330 (1993)
27. Okazaki, N.: Classias: A collection of machine-learning algorithms for classification (2009), `http://www.chokkan.org/software/classias/`
28. Zhou, Z.M., Xu, Y., Niu, Z.Y., Lan, M., Su, J., Tan, C.L.: Predicting discourse connectives for implicit discourse relation recognition. In: Proc. of COLING 2010 (Posters), pp. 1507–1514 (2010)
29. Louis, A., Joshi, A., Prasad, R., Nenkova, A.: Using entity features to classify implicit discourse relations. In: Proc. of the SIGDIAL 2010, pp. 59–62 (2010)

Integrating Japanese Particles Function and Information Structure

Akira Ohtani

Osaka Gakuin University
Faculty of Informatics
2-36-1 Kishibe-minami, Suita, Osaka 564-8511 Japan
ohtani@ogu.ac.jp

Abstract. This paper presents a new analysis of the discourse functions of Japanese particles *wa* and *ga*. Such functions are integrated with information structures into the constraint-based grammar under the HPSG framework. We examine the distribution of these particles and demonstrate how the thematic-rhematic dichotomy of the constituent can be determined by the informational status of one or more of its daughter constituents through various linguistic constraints. We show that the relation between syntactic constituency and information structure of a sentence is not a one-to-one mapping as a purely syntax-based analysis assumes, and then propose the multi-dimensional grammar which expresses mutual constraints on the thematic-rhematic interpretation, syntax and phonology.

1 Introduction

Information Structure (IS) plays a crucial role for ensuring coherence in discourse. In many languages, intonation is the primary means of conveying IS. The mini-dialogue in (1), where **bold face** corresponds to the so called B-accent (L+H*) and SMALL CAPITALS indicate the word bearing the so called A-accent (H*), illustrates the connection between IS and accent in English.

(1) Speaker Q: So tell me about the people in the White House.
Anything I should know?

Speaker A*1*: Yes. [$_\theta$ The **president**] [$_\rho$ hates the Delft CHINA SET].
Don't use it. (Engdahl and Vallduví [1]:5, ex.3, Modified.)

The information conveyed by a sentence is split into new information *rheme* (ρ *focus*) and information already present in the discourse *theme* (θ, *topic*).

It has been observed that languages adopt different means to encode their IS: English employs prosody, Catalan relies on word order, and Greek uses both. In addition to those means, Japanese utilizes morphology.

(2) Speaker A*2*:	[$_\theta$ Daitooryoo-*wa*]	[$_\rho$ maisen-no	SYOKKI-*ga*	okonomi	desu].
	president-θ	Meissen-GEN	china.set-NOM	like	

'The president likes the Meissen china set.'

In (2) theme and rheme are identified by the particles, *wa* and *ga*, respectively.

However, the distribution of these particles does not correspond to the thematic-rhematic dichotomy of sentences. For example, *wa*-marked constituents can be construed as rheme if they bear new information, as in (3A).

(3) Speaker Q: What is the president eating?

Speaker A: [$_\theta$ Daitooryoo-*wa*] [$_\rho$ CHOKOREETO-*wa*] meshiagatte imasu...
president-θ chocolate-ρ eating is
'It is (at least) chocolates that the president is eating.'

When CHOKOREETO 'chocolate' is being focused, the phrase receives a high pitch (either on *chokoreeto* or on *wa*) and is construed as rheme.

Another example comes from the domain of rheme. Let us consider (4) in which CHOKOREETO is pitch-accented.

(4) Speaker Q: What is the president doing?

Speaker A: [$_\theta$ Daitooryoo-*wa*] [$_\rho$ CHOKOREETO-*wa* meshiagatte imasu]...
president-θ chocolate-ρ eating is
'It is (at least) eating chocolates that the president is doing.'

(4A) implies that the president are eating chocolates, and he may also be doing something else, such as sending text messages. That is, *chokoreeto-wa meshiagatte imasu* 'eating chocolates' not just *chokoreeto-wa* carries a rheme interpretation. Like English, a pitch accent serves to mark rheme for more than just the accented word. The domain is extended beyond the constituent marked with *wa*.

As it can be seen in (3A) and (4A), there are two usages of the particle *wa*, i.e. *theme* and *rheme*. Japanese is the language in which the order of sentence constituents other than verbs is relatively free. However, a scrambled word order is not possible when a sentence contains a *wa*-marked subject indicating theme followed by a *wa*-marked object indicating rheme. Compare (3A) and (5A).

(5) Speaker A: #CHOKOREETO-*wa* daitooryoo-*wa* meshiagatte imasu...
chocolate-ρ president-θ eating is

Analysis of these discourse functions of the particles is not only linguistically interesting but also important for computational applications, for example, the automatic identification of the coherence of a text and machine generation of contextually-appropriate particles in Japanese. While *wa* and *ga* provide useful materials in relation to the study of focus structure (Erteschik-Shir [2]) and information packaging theory (Vallduví [3]), the existing linguistic analyses are mostly informal and not sufficiently detailed for computational applications.

In this paper, we attempt to remedy the situation by providing a new formal analysis of the discourse functions of Japanese particles *wa* and *ga*. We argue that the relation between the syntactic constituency and the information structure of a sentence is not a one-to-one mapping as purely syntax-based analysis assumes.

The paper is organized as follows. Section 2 summarizes how theme and rheme are identified in this work. Section 3 formalizes the constraints on Japanese thematic-rhematic dichotomy based on the previous constraint-based approach to English. Section 4 discusses some implications of our analysis and shows some applications of the formalization. Section 5 concludes this paper.

2 Theme/Rheme Dichotomy and *WA*/*GA*-Marked Subject

Japanese is a language in which *theme* and *rheme* are identified by the use of particles. In the case of subjects, these are either marked with *wa* or *ga*. In this section, we first summarize how theme and rheme are identified in the context of this paper since their definitions vary considerably among linguists.

Erteschik-Shir [2] studies the interface between discourse structure and syntax defining a grammatical level of focus structure in which theme and rheme (described as *topic* and *focus*, respectively) constituents are marked. Theme is distinguished from rheme by Reinhart's [4] theme test.

(6) Speaker Q: Tell me about x
Speaker A: ... x ... (X = TOPIC) (Erteschik-Shir [2]:14, ex.11)

Erteschik-Shir defines theme as the subject of the predication. Thus, in (7A) *the president* is the theme, and the predicate represents the assertion made about the theme.

(7) Speaker Q: Tell me about the president.
Speaker A: [$_\theta$ The **president**] is eating chocolates.
(As for the president, he is eating chocolates.)

The part 'president' bears a theme-associated B-accent. Theme is old information in the sense that it has already been introduced in discourse.

Rheme is determined by using question-answer pairs to identify the constituent which answers a *wh*-question.

(8) a. Speaker Q: Who is eating chocolates?
Speaker A: [$_\rho$ The PRESIDENT] is eating chocolates.
(It is the president who is eating chocolates.)
b. Speaker Q: What is the president eating?
Speaker A: The president is eating [$_\rho$ CHOCOLATES].
(It is chocolates that the president is eating.)
c. Speaker Q: What is the president doing?
Speaker A: The president [$_\rho$ is eating CHOCOLATES].
(It is eating chocolates that the president is doing.)

In (8a) and (8b), the part 'president' and 'chocolates' receive a high pitch (H*). In (8c), the nuclear stress (the A-accent) appears on the right hand periphery of the clause. A pitch accent can serve to mark rheme, i.e. new information (of more than just the accented word).[1]

[1] Vallduví [3] proposes a three-way partition of information structure of a sentence. The information conveyed by a sentence is split into new information *focus* (rheme) and information already present in the discourse *ground*. The latter is further divided into *link* (topic, theme) and *tail*. In (8a) the part 'ate chocolates' is the tail, but we ignore this information throughout this paper.

Based on the definitions and the diagnostics shown above, let us examine the interpretation of *wa*-marked and *ga*-marked subjects in Japanese.

As can be seen in (9A), the *wa*-marked subject has a *theme* interpretation.

(9) Speaker Q: Tell me about the president.

Speaker A:	[$_\theta$ Daitooryoo-*wa*]	chokoreeto-o	meshiagatte	imasu.
	president-θ	chocolate-ACC	eating	is

‘As for the president, he is eating chocolates.’

Theme marking differs from English in that *daitooryoo-wa* ‘the president’ goes unstressed. *Wa*-marked constituents can also carry a rheme interpretation, as in (10A), if the phrase receives a high pitch (either on *daitooryoo* or on *wa*).

(10) Speaker Q: Who is eating chocolates?

Speaker A:	[$_\rho$ DAITOORYOO-*wa*]	chokoleeto-o	meshiagatte	imasu...
	president-ρ	chocolate-ACC	eating	is

‘It is (at least) the president who is eating chocolates.’

Ga-marked constituents are construed as rheme if they bear new information.

(11) Speaker Q: Who is eating chocolates?

Speaker A:	[$_\rho$ DAITOORYOO-*ga*]	chokoleeto-o	meshiagatte	imasu.
	president	chocolate-ACC	eating	is

‘It is (only) the president who is eating chocolates.’

The part ‘daitooryoo’ is marked by the boundary tone (H%), which marks rheme and adds its own semantic contribution.

There is a difference between *wa*-marked and *ga*-marked rheme. In (10A) and (11A), ‘daitooryoo’ belongs to a set specified in the discourse, i.e. the people in the White House. In (11A), he has to be the only member of the set who is eating chocolates, if the First Lady is also eating chocolates then the statement becomes false. In (10A), on the other hand, he is not required to be the only member of the set. In other words, members who are eating chocolates have to be listed exhaustively in (11A) but non-exhaustively in (10A).

Here we do not go into detail about these discourse (and/or semantic) functions, but we do make a distinction between *wa*-marked and *ga*-marked rheme in Section 3, and mention the properties of the particle *ga* by referring to those functions in Section 4.

3 Theme/Rheme *WA* and Japanese Information Structure

The constraint-based grammar formalism is well-suited for representing IS which interacts with syntax and phonology in principled ways. In this section, we consider how IS, which is a crucial factor for thematic-rhematic dichotomy, is formally represented in Japanese multi-dimensional grammar under the computationally applicable framework of HPSG (Sag et al. [5]) and its extension (Engdahl and Vallduví [1]).

IS is represented as the INFO(RMATION)-ST(RUCTURE) feature within lexical and the phrasal *signs'* value of CON(TE)X(T) feature as following:[2]

(12)
$$\begin{bmatrix} \text{PHON} & \begin{bmatrix} \ldots & \\ \text{ACCENT} & \textit{accent} \end{bmatrix} \\ \text{SYNSEM} & \begin{bmatrix} \text{SYN} & \begin{bmatrix} \text{HEAD} & \textit{head} \\ \text{VAL} & [\ \ldots\ ] \end{bmatrix} \\ \text{SEM} & [\ \ldots\ ] \\ \text{ARG-ST} & \langle\ \ldots\ \rangle \\ \text{CONX} & \begin{bmatrix} \ldots & \\ \text{INFO-ST} & \begin{bmatrix} \text{THEME} & [\ \ldots\ ] \\ \text{RHEME} & \{\ \ldots\ \} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{DTRS} & [\ \ldots\ ] \end{bmatrix}$$

As in (12), PHON(OLOGY) and CONX are enriched with ACCENT and INFO-ST, respectively. The feature INFO-ST directly represents the THEME and RHEME dichotomy of a sentence.

Let us look at first how pitch accent type and informational status constrain each other by referring to the examples in Section 1. The skeletal lexical signs of the part {*daitooryoo-wa*}, {*chokoreeto-wa*}, {*okonomi desu*, *meshiagatte imasu*}, and {*syokki-ga,*} are posited as (13a), (13b), (13c), and (13d), respectively.

(13) a. $\boxed{1}\begin{bmatrix} \text{PHON} & [\text{ACCENT} \quad U] \\ \text{HEAD} & [\text{PFORM} \quad \textit{wa}] \\ \text{INFO-ST} & [\text{THEME} \quad \boxed{1}] \end{bmatrix}$ b. $\boxed{1}\begin{bmatrix} \text{PHON} & [\text{ACCENT} \quad A] \\ \text{HEAD} & [\text{PFORM} \quad \textit{wa}] \\ \text{INFO-ST} & [\text{RHEME} \quad \{\boxed{1}\}] \end{bmatrix}$

c. $\boxed{1}\begin{bmatrix} \text{PHON} & [\text{ACCENT} \quad U] \\ \text{INFO-ST} & [\ \ ] \end{bmatrix}$ d. $\boxed{1}\begin{bmatrix} \text{PHON} & [\text{ACCENT} \quad A] \\ \text{INFO-ST} & [\text{RHEME} \quad \{\boxed{1}\}] \end{bmatrix}$

(13a)–(13d) show that the value of ACCENT and the value of INFO-ST constrain each other. This is expressed by means of structure-sharing between INFO-ST and the sign itself. (13a) and (13b) introduce *U*(nmarked) and *A*(-accented) *wa*-marking words, which are construed as theme and rheme, respectively. (13c) says about itself that the value of INFO-ST is not specified if the value of ACCENT is unmarked without any morphological marking.

It is worth noting that (13d) shows the presence of A-accent is sufficient to identify the informational contribution of the lexical sign as rheme, and vice versa; the constraint in (13b) seems redundant. However, we have observed *wa*-marked (i.e. [HEAD|PFORM *wa*]) rheme and *ga*-marked (i.e. [HEAD|CASE *nom*]) rheme are different in the discourse (and/or semantic) functions which we left in Section 2 for future work. Here we simply maintain this distinction without enriching CONX (and/or SEM(ANTICS)).

Thus, IS in Japanese, as in Catalan-type languages, depends crucially on morpho-syntactic devices such as *wa*-marking.[3] Furthermore, as in English-type

[2] The relevant units of linguistic information are called *signs*. The signs are often abbreviated by omitting features and type designations that can be readily inferred.

[3] Regarding (13a) and (13b), we do not describe either such phonological aspects or a morphological process which removes the case particles *ga* and *o* obligatorily and *ni* optionally, while keeping other case particles and postpositions intact.

languages, the prosodic phenomenon of sentence accent is also essential. (13a)–(13d) show that CONX|INFO-ST value, i.e. discourse is not only an independent level of linguistic information but also interacts with the other grammatical levels, i.e. PHON and SYN(TAX), simultaneously.

A Japanese sentence is restricted to at most one *wa*-marked theme phrase, which, if present, appears in sentence-initial position; however, multiple elements within the sentence can carry a rheme interpretation by receiving a high pitch in situ even if they are marked with *wa*.[4] Following what is commonly accepted in the linguistic literature on the *wa*-marking topicalization (e.g. Hoji [6], Saito [7]), we assume that *wa*-marked elements for rheme and theme are licensed by the following two lexical rules, rheme-*substitution* and theme-*addition*.

(14) a. Rheme-substitution Lexical Rule

$$\begin{bmatrix} \text{HEAD} & \textit{verb} \\ \text{ARG-ST} & \langle \boxed{a} \oplus \boxed{1} \oplus \boxed{b} \rangle \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} \text{HEAD} & \textit{verb} \\ \text{ARG-ST} & \left\langle \boxed{a} \oplus \boxed{2}\,\text{PP}\begin{bmatrix} \text{ACCENT} & A \\ \text{PFORM} & \textit{wa} \\ \text{INFO-ST} & [\text{RHEME} \ \ \{\boxed{2}\}] \end{bmatrix} \oplus \boxed{b} \right\rangle \end{bmatrix}$$

where $\boxed{a}$ *and* $\boxed{b}$ *are possibly empty lists of* SYNSEM *objects,* $\boxed{1}$ *and* $\boxed{2}$ *are identical other than their* ACCENT, PFORM *and* INFO-ST *values, and* $\oplus$ *stands for list concatenation.*

b. Theme-addition Lexical Rule

$$\begin{bmatrix} \text{HEAD} & \begin{bmatrix} \textit{verb} & \\ \text{VFORM} & \textit{assert} \end{bmatrix} \\ \text{INFO-ST} & [\ ] \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \textit{verb} & \\ \text{VFORM} & \textit{assert} \end{bmatrix} \\ \text{VAL} & \begin{bmatrix} \text{TOPIC} & \left\langle \boxed{1}\,\text{PP}\begin{bmatrix} \text{ACCENT} & U \\ \text{HEAD} & [\text{PFORM} \ \ \textit{wa}] \\ \text{INFO-ST} & [\text{THEME} \ \ \boxed{1}] \end{bmatrix} \right\rangle \end{bmatrix} \\ \text{INFO-ST} & [\text{THEME} \ \ \boxed{1}] \end{bmatrix}$$

In (14a), a *wa*-marked rheme phrase is introduced into the ARG(UMENT)-ST(RUCTURE) of the verb by substituting one of its argument(s). In (14b), on the other hand, the TOPIC feature is introduced as the value of VAL(ENCE) which describes the categories of the constituent. The value of TOPIC is a singleton list in order to subcategorize for at most one *wa*-marked theme phrase.

It is worth noting that a theme phrase is prohibited from appearing both in a conditional clause and in a relative clause, whereas a rheme phrase is not. The theme phrase requires modality at the sentential ending. These characteristics

[4] This also matches the characterization of *link* (theme) and *focus* (rheme), since under Vallduví's system [3] the Information Structure of a sentence is restricted to at most one *link* but any number of *focus* and *tail* elements, and moreover the property of theme is consistent with his original conception of *link* as exclusively sentence-initial.

can be straightforwardly explained by assuming that the specific head (i.e. [V(ERB)FORM assert(ive)]) has a different valence specification for TOPIC.

In addition to lexical signs there are phrasal signs which result from combining signs according to the grammar rules. The phrasal signs for verb phrases and sentences are licensed by the following two rules, respectively.

(15) a. Head-Subject-Complement Rule

$$\begin{bmatrix} \textit{phrase} & \\ \text{COMPS} & \langle\ \rangle \end{bmatrix} \rightarrow \boxed{1} \ldots \boxed{n}\ \text{H}\begin{bmatrix} \textit{word} & \\ \text{COMPS} & \langle \boxed{1}, \ldots, \boxed{n} \rangle \end{bmatrix}$$

A phrase can consist of a lexical head preceded by all of its complements.

b. Head-Topic Rule (to be revised)

$$\begin{bmatrix} \textit{phrase} & \\ \text{TOPIC} & \langle\ \rangle \end{bmatrix} \rightarrow \boxed{1}\quad \text{H}\begin{bmatrix} \textit{phrase} & \\ \text{TOPIC} & \langle \boxed{1} \rangle \end{bmatrix}$$

A phrase can consist of a phrasal head preceded by its topic.

These grammar rules are well-formedness conditions on possible phrases of which D(AUGH)T(E)RS represent the immediate constituent structure. (15b) ensures that a phrase subcategorized as TOPIC appears in a sentence-initial position.

There are additional constraints specifying how INFO-ST of a phrase is constrained by INFO-ST of one or more of its daughters.

(16) INFO-ST Instantiation Principle (in Japanese)

Either (i) if a daughter's INFO-ST is instantiated, then the mother inherits this instantiation (*for narrow rheme*),

(Engdahl and Vallduví [1]:12, ex.15, Slightly modified.)

or (ii) if the non-agentive highest ranking argument's RHEME is instantiated, then the RHEME of the mother is the *sign* itself (*for wide rheme*).

(Chung et.al. [8]:397, ex.37, Slightly modified.)

Phrasal signs must satisfy the INFO-ST Instantiation Principle (IIP).

Now we examine how these constraints predict the INFO-ST value of the previous examples, which are repeated below with slight modification.

(17) Speaker Q: Who is eating chocolates? (=(10Q))

Speaker A$_1$: [$_\rho$ DAITOORYOO-*wa*] chokoleeto-o meshiagatte imasu.
president-ρ chocolate-ACC eating is

'It is (at least) the president who is eating chocolates.' (=(10A))

Speaker A$_2$: [$_\rho$ DAITOORYOO-*ga*] chokoleeto-o meshiagatte imasu.
president-NOM chocolate-ACC eating is

'It is (only) the president who is eating chocolates.' (=(11A))

(18) Speaker Q$_1$: What is the president eating? (=(3Q))

Speaker Q$_2$: What is the president doing? (=(4Q))

Speaker A: $[_{\theta}$Daitooryoo-*wa*$]$ $[_{\rho_2}$ $[_{\rho_1}$ CHOKOREETO-*wa*$]$ meshiagatte imasu$]$.
president-θ chocolate-ρ eating is

ρ_1: 'It is (at least) chocolates that the president is eating.' (=(3A))
ρ_2: 'It is (at least) eating chocolates that the president is doing.' (=(4A))

The lexical rules, rheme-substitution and theme-addition, in (14a) and (14b) operate on the verb *meshiagatte imasu* 'eat' in (19) give rise to corresponding verbs in (20a) and (21) which are responsible for (17A$_1$) and (18A), respectively.

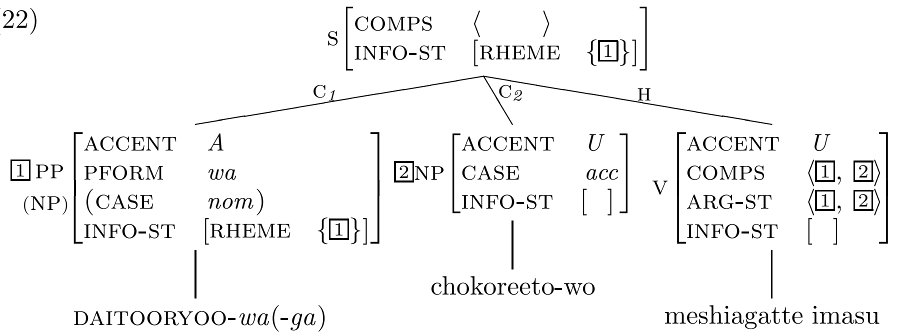
(19)
$$\begin{bmatrix}\text{HEAD} & \textit{verb}\\ \text{ARG-ST} & \langle \boxed{1}\,\text{NP}[\textit{nom}],\ \boxed{2}\,\text{NP}[\textit{acc}]\,\rangle \\ \text{INFO-ST} & [\ \ ]\end{bmatrix}$$

(20) a.
$$(19) \rightarrow \begin{bmatrix}\text{HEAD} & \textit{verb}\\ \text{ARG-ST} & \left\langle \boxed{1}\,\text{PP}\begin{bmatrix}\text{ACCENT} & \textit{A}\\ \text{PFORM} & \textit{wa}\\ \text{INFO-ST} & [\text{RHEME}\ \ \{\boxed{1}\}]\end{bmatrix},\ \boxed{2}\,\text{NP}[\textit{acc}]\right\rangle\end{bmatrix}$$

b.
$$(19) \rightarrow \begin{bmatrix}\text{HEAD} & \textit{verb}\\ \text{ARG-ST} & \left\langle \boxed{1}\,\text{NP}[\textit{nom}],\ \boxed{2}\,\text{PP}\begin{bmatrix}\text{ACCENT} & \textit{A}\\ \text{PFORM} & \textit{wa}\\ \text{INFO-ST} & [\text{RHEME}\ \ \{\boxed{2}\}]\end{bmatrix}\right\rangle\end{bmatrix}$$

(21)
$$(20b) \rightarrow \begin{bmatrix}\text{HEAD} & \textit{verb}\\ \text{VAL} & \begin{bmatrix}\text{TOPIC} & \left\langle \boxed{3}\,\text{PP}\begin{bmatrix}\text{ACCENT} & \textit{U}\\ \text{HEAD} & [\text{PFORM}\ \ \textit{wa}]\\ \text{INFO-ST} & [\text{THEME}\ \ \boxed{3}]\end{bmatrix}\right\rangle\end{bmatrix}\\ \text{ARG-ST} & \left\langle \boxed{1}\,\text{NP}[\textit{nom}],\ \boxed{2}\,\text{PP}\begin{bmatrix}\text{ACCENT} & \textit{A}\\ \text{PFORM} & \textit{wa}\\ \text{INFO-ST} & [\text{RHEME}\ \ \{\boxed{2}\}]\end{bmatrix}\right\rangle\\ \text{INFO-ST} & [\text{THEME}\ \ \boxed{3}]\end{bmatrix}$$

The signs and the rules described in (12)-(15) and (20a) lead to the following simplified representation of (17:A$_1$) (and (17:A$_2$)), in which values of the DTRS attribute are presented in the constituent tree notation whose arcs are labelled H(EAD-DTR), C(OMP-DTRS) or T(OPIC-DTR), and the HEAD feature is omitted.

(22)

S [COMPS ⟨ ⟩ ; INFO-ST [RHEME {1}]]
C1
C2
H
1 PP (NP) [ACCENT A ; PFORM wa ; (CASE nom) ; INFO-ST [RHEME {1}]]
2 NP [ACCENT U ; CASE acc ; INFO-ST [ ]]
V [ACCENT U ; COMPS ⟨1, 2⟩ ; ARG-ST ⟨1, 2⟩ ; INFO-ST [ ]]
DAITOORYOO-wa(-ga)
chokoreeto-wo
meshiagatte imasu

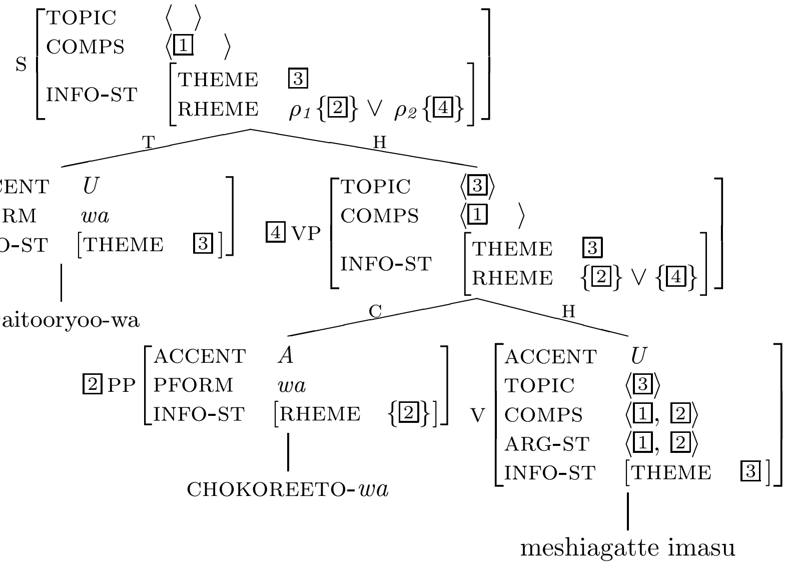
Since we assume that in topic-prominent languages like Japanese and Korean, the subject is the highest ranking argument on the ARG-ST list, HPSG's Argument Realization Principle (ARP) requires only a minor revision as shown in (23).

(23) Argument Realization Principle (in Japanese)
The elements on the ARG-ST list is realized as COMPS.

The verbs in (19)-(21) have the realization COMPS$\langle \boxed{1}, \boxed{2} \rangle$ and the appropriate elements referred in particular rules in (15) are canceled from the relevant valence specifications of the head daughter (in head-complement phrases, in this case).

In (22), PP DAITOORYOO-*wa* is focused not only phonologically (A-accent) but also morphologically (*wa*-marking). This is sufficient to identify the informational contribution of such an element as rheme. Consequently (16i) is available within IIP and then the rheme value $\boxed{1}$ is inherited by the mother constituent.

IIP also addresses the contextual ambiguity in (18A). This sentence, with an A-accent on the *wa*-marked object, can be interpreted either with *narrow rheme* (ρ_1) on the object postposition phrase or with *wide rheme* (ρ_2) on the whole verb phrase depending on the context in (18Q$_1$) and (18Q$_2$), respectively. The tree (24) shows how the value of INFO-ST of the sentence follows from IIP.

(24)

S
TOPIC ⟨ ⟩
COMPS ⟨1 ⟩
INFO-ST THEME 3 RHEME ρ1{2} ∨ ρ2{4}
T
H
3 PP ACCENT U PFORM wa INFO-ST [THEME 3]
Daitooryoo-wa
4 VP TOPIC ⟨3⟩ COMPS ⟨1 ⟩ INFO-ST THEME 3 RHEME {2} ∨ {4}
C
H
2 PP ACCENT A PFORM wa INFO-ST [RHEME {2}]
CHOKOREETO-wa
V ACCENT U TOPIC ⟨3⟩ COMPS ⟨1, 2⟩ ARG-ST ⟨1, 2⟩ INFO-ST [THEME 3]
meshiagatte imasu

Since the RHEME value of PP CHOKOREETO-*wa* is instantiated, two disjunctive options are available within IIP. (16i) requires the mother, i.e. VP to inherit the INFO-ST value of its daughter, i.e. [RHEME $\boxed{2}$]. (16ii) allows that RHEME value of the mother is $\boxed{4}$ VP itself, since the non-agentive highest ranking argument's RHEME is instantiated, and therefore the value is $\boxed{4}$. Consequently, either $\boxed{2}$ (ρ_1) or $\boxed{4}$ (ρ_2) propagates to the INFO-ST value of its mother, i.e. S by (16i).

$\boxed{2}$PP is canceled from the COMPS, other valence specifications are passed up from the head daughter to its mother, and then the TOPIC is discharged by $\boxed{3}$PP.

It is worth noting that the COMPS list remains unsaturated. This is because the additional theme phrase, i.e. 'daitooryoo-*wa*' does not cancel off the subject, i.e. $\boxed{1}$. We think this is a correct prediction since theme-addition sentences allow a resumptive pronoun which may cause cancellation whereas rheme-substitution sentences do not.[5] The relation between a theme element and an unrealized element is a matter of context. We do not go into enough detail about such predictions to require the introduction of lexical rules for *wa*-marked elements.

In this section, we have represented the constraint-based formalization of the Japanese IS system by following the thematic-rhematic dichotomy discussed in the previous section. To capture the functions of the particles, in particular *wa* which is instantiated as not only theme but also rheme, we introduced the new lexical rules and grammar rules for Japanese while making a minor revision to the HPSG constraints that play a role in English system.

4 Rheme *GA* Projection and Obligatory Interpretation

In this section, we discuss the implications of our analysis and show more applications of the formalization of Japanese IS by referring to the functions of the particle *ga*. The nominative case particle *ga* is often associated with rheme. When a sentence with a *ga*-marked subject is uttered out-of-the-blue, the whole sentence bears new information as (25A_1).

(25) Speaker Q: What happened?

Speaker A_1: [$_\rho$ Daitooryoo-*ga* chokoreeto-o kai mashita].
president-NOM chocolate-ACC bought
'The president bought chocolates.'

Speaker A_2: #[$_\rho$ DAITOORYOO-*ga* chokoreeto-o kai mashita].
'#[$_\rho$ The PRESIDENT bought chocolates].'

IIP constrains the value of RHEME of a sentence if its complement is instantiated. However, the wide rheme on the whole sentence (ρ) in (25A_1) is the case that no complement's RHEME is instantiated, and therefore IIP does not constrain the INFO-ST value of the sentence.

It is worth noting that no set from which 'daitooryoo' is picked out is presupposed in (25A_1). This is different from the rheme we have determined and formalized in the previous section. Thus, we distinguish such an interpretation from the rheme constrained by IIP, and then suppose that the *all-rheme* reading in (25A_1) is licensed by Chafe's definition [9]: 'every utterance contains new information.' When a sentence does not contain any parts which make an informational contribution, it receives an all-rheme reading.

Selkirk [10] claims that an A-accent on the external argument in English cannot project the rheme value up to the mother, as can be seen in (25A_2). Japanese also shows the same sensitivity and this is confirmed by the suitability or unsuitability of the answer to a *wh*-question in (25Q).

(26) Speaker A_3: #[$_\rho$ DAITOORYOO-*ga* hashiri mashita].
president-NOM ran
'#[$_\rho$ The PRESIDENT ran].'

[5] For a more detailed discussion of these sentences, see Hoji [6].

Speaker A_4: [$_\rho$ DAITOORYOO-*ga* nakunari mashita].
president-NOM died
'[$_\rho$ The PRESIDENT died].'

The rheme of the pitch-accented subject of the transitive verb in (25A_2) and unergative verb in (26A_3) cannot be projected up, whereas that of the unaccusative verb in (26A_4) can, and then the sentence includes an all-rheme reading. These readings are predicted by (16ii), i.e. one of the disjunctive options of IIP.

However, unlike English, Japanese prohibits rheme on an oblique argument from being projected up to its mother phrase. As can be seen in (27A_1) and (27A_2), it is only when the object CHOKOREETO-O receives a high pitch that the VP (or S) can receive the wide rheme reading.

(27) Speaker Q: What did the president do?
Speaker A_1: # Daitooryoo-wa [$_\rho$ MARIA-NI chokoreeto-o age mashita].
president-θ Maria-DAT chocolate-ACC gave
'The president [$_\rho$ gave chocolates to MARIA].'
Speaker A_2: Daitooryoo-wa [$_\rho$ Maria-ni CHOKOREETO-O age mashita].
president-θ Maria-DAT chokolate-ACC gave
'# The president [$_\rho$ gave CHOCOLATES to Maria].'

(27A_1) cannot be a felicitous reply to a VP-rheme question in (27Q). Even in scrambled examples in (28), the felicity of (27A_1) and (27A_2) holds.

(28) Speaker A_1′: # Daitooryoo-wa [$_\rho$ chokoreeto-o MARIA-NI age mashita].
Speaker A_2′: Daitooryoo-wa [$_\rho$ CHOKOREETO-O Maria-ni age mashita].

This is rather expected, considering the cross-linguistic properties of the language. Chung et.al [8] claim that the only difference in IIP between Korean and English is that the non-agentive ranking argument that allows wide rheme projection in (16ii) is the *highest* in Korean whereas the *lowest* in English. As is well known, Korean and Japanese have almost identical grammar systems. Thus, the present formalization requires only a minor revision to the English IIP.

What is interesting is that, unlike English, the sentence-initial *ga*-marked phrase of an individual-level predicate as in (29A_1) and the so-called multiple nominative construction (MNC) as in (29A_2) in Japanese (and Korean) can only be construed as rheme without any pitch accent (e.g. Kuroda [11], Kuno [12]).

(29) Speaker A_1: [$_\rho$ Hisho-*ga*] yuunoo desu.
secretary-NOM efficient is
'It is (only) the secretary who is efficient.'
Speaker A_2: [$_\rho$ Daitooryoo-ga] hisyo-ga yuunoo desu.
president-NOM secretary-NOM efficient is
'It is (only) the president whose secretary is efficient.'

Various subject-sensitive phenomena such as honorification, binding, control, and so on indicate that the immedicate preverbal nominative NP carries the subject properties whereas the initial nominative NP does not.

Extensive past study reveals that the sentence-initial *ga*-marked phrase is the realization of rheme. Kuno [12] distinguishes two usages of *ga*, which are referred to as *descriptive* as in ($25A_1$) and *exhaustive listing* as in ($29A_1$) and ($29A_2$). The evidence that such a phrase as in ($29A_2$) is solely rheme comes from several tests. For example, only the first *ga*-marked phrase can be *wh*-questioned as in ($30Q_1$) while the second one cannot as shown in ($30Q_2$).

(30) Speaker Q_1:

Dare-ga	hisyo-ga	yuunoo	desu	ka?
who-NOM	secretary-NOM	efficient	is	Q

'(lit.) Who is it whose secretary is efficient?'

Speaker Q_2:

*Daitooryoo-ga	dare-ga	yuunoo	desu	ka?
president-NOM	who-NOM	efficient	is	Q

We posit that the particle *ga* has three lexical signs; in addition to the nominative case marker (i.e. [CASE *nom*]) described in Section 3 and repeated here as in both (31a) and (31b), it also serves as a rheme marker as in (31c).

(31) a. $\boxed{1}\left[\begin{array}{ll}\text{ACCENT} & U\\ \text{CASE} & nom\\ \text{INFO-ST} & [\ ]\end{array}\right]$ b. $\boxed{1}\left[\begin{array}{ll}\text{ACCENT} & A\\ \text{CASE} & nom\\ \text{RHEME} & \boxed{1}\end{array}\right]$ c. $\boxed{1}\left[\begin{array}{ll}\text{ACCENT} & A \vee U\\ \text{MARKING} & ga\\ \text{RHEME} & \boxed{1}\end{array}\right]$

(31a) specifies only the nominative value of the head. The preverbal *ga*-marked phrase in ($29A_2$) and the unmarked subject of all-rheme sentence in ($25A_1$) are constrained by it. Both (31b) and (31c) mark rheme, but the difference between these signs is that only (31c) interacts with the new grammar rule in (32a) and therefore the sentence-initial *ga*-marked phrase construed as rheme obligatorily.

(32) a. Head-Specifier Rule

$$\left[\begin{array}{ll}\textit{phrase} & \\ \text{RHEME} & \{\boxed{1}\}\end{array}\right] \rightarrow \boxed{1}\left[\begin{array}{ll}\textit{phrase} & \\ \text{MARKER} & ga\\ \text{RHEME} & \{\boxed{1}\}\end{array}\right] \quad \text{H}\left[\begin{array}{ll}\textit{phrase} & \\ \text{SPR} & \langle\boxed{1}\rangle\end{array}\right]$$

b. Head-Topic Rule (final)

$$\left[\begin{array}{ll}\textit{phrase} & \\ \text{TOPIC} & \langle\ \rangle\end{array}\right] \rightarrow \boxed{1} \quad \text{H}\left[\begin{array}{ll}\textit{phrase} & \\ \text{TOPIC} & \langle\boxed{1}\rangle\\ \text{SPR} & \langle\ \rangle\end{array}\right]$$

Because of the introduction of the new valence feature SP(ECIFIE)R, Head-Topic Rule is revised. (33) shows that theme is higher than rheme as (32b) constraints.

(33) Speaker A_1:

[$_\theta$ Kankoku-wa]	[$_\rho$ daitooryoo-ga]	hisyo-ga	yuunoo	desu.
Korea-θ	president-NOM	secretary-NOM	efficient	is

'As for Korea, it is (only) the president whose secretary is efficient.'

Speaker A_2:#[$_\rho$ Kankoku-ga][$_\theta$ daitooryoo-wa] hisyo-ga yuunoo desu.

Some of the important constraints in MNC are (i) the consecutive phrases need to be in a certain semantic relation, and (ii) the specifier value of the non-initial nominative requires to be unsaturated. Compare ($29A_2$) with (34).

(34) Speaker A_1:

[$_{\text{NP}}$ Daitooryoo-no	hisyo-ga]	yuunoo	desu.
president-GEN	secretary-NOM	efficient	is

'It is (only) the president whose secretary is efficient.'

Speaker A$_2$:	*Daitooryoo-ga	[$_{\text{NP}}$ Kim-no	hisyo-ga	]	yuunoo desu.
	president-NOM	Kim-GEN	secretary-NOM		efficient is

To account for these phenomena, we assume the following lexical rule.

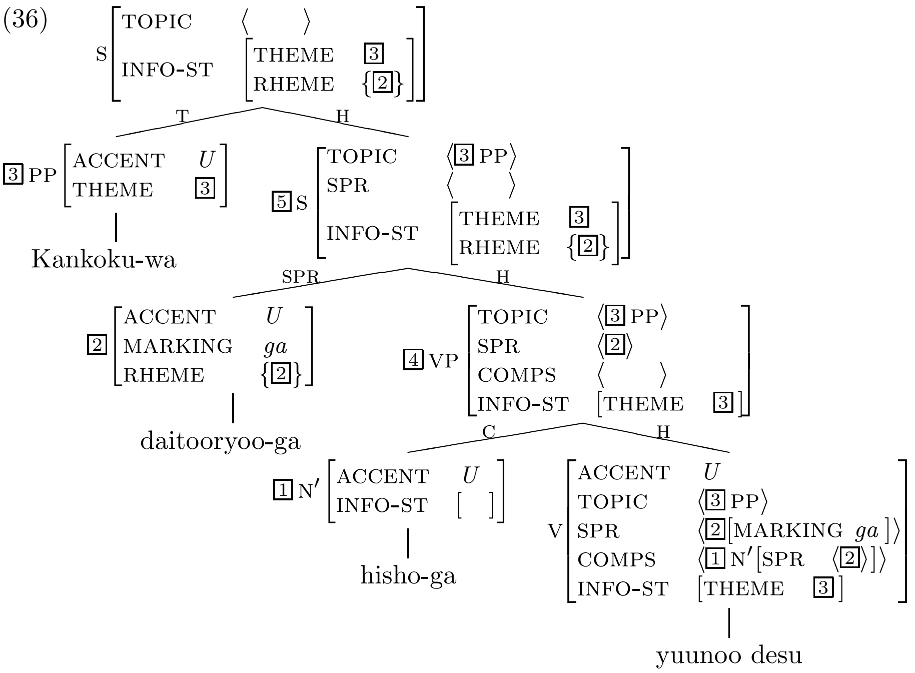
(35) Rheme-addition Lexical Rule

$$\begin{bmatrix}\text{HEAD} & \textit{individual-level-pred}\\ \text{COMPS} & \langle \text{NP}[\textit{nom}]\rangle\end{bmatrix} \rightarrow \begin{bmatrix}\text{HEAD} & \textit{individual-level-pred}\\ \text{SPR} & \langle \boxed{1}[\text{MARKING } \textit{ga}\,]_i\rangle\\ \text{COMPS} & \left\langle \text{N}'\begin{bmatrix}\text{CASE} & \textit{nom}\\ \text{SPR} & \langle\boxed{1}\rangle\end{bmatrix}_j\right\rangle\\ \text{CONX} & \begin{bmatrix}\text{RESTR} & R\,(i,j)\\ \text{INFO-ST} & [\text{RHEME} \quad \boxed{1}]\end{bmatrix}\end{bmatrix}$$

In (35), the value of RESTR(ICTION) is construed as placing semantic condition, e.g. *subordinate Relation* between the entity of the first *ga*-marked element (i) and the entity of the second nominative element (j) that is subcategorized for by the individual-level predicate. The second element differs from ordinary NPs in that it cannot be fully saturated. This incompleteness accounts for the impossibility of replacing the element with a fully saturated *wh*-word as in (30Q$_2$).

Let us examine how the lexical signs posited in (31) and the sign derived by (35) interact with the grammar-rules introduced in (32).

(36)


S [TOPIC ⟨ ⟩, INFO-ST [THEME 3, RHEME {2}]]
T
H
3 PP [ACCENT U, THEME 3]
Kankoku-wa
5 S [TOPIC ⟨3 PP⟩, SPR ⟨ ⟩, INFO-ST [THEME 3, RHEME {2}]]
SPR
H
2 [ACCENT U, MARKING ga, RHEME {2}]
daitooryoo-ga
4 VP [TOPIC ⟨3 PP⟩, SPR ⟨2⟩, COMPS ⟨ ⟩, INFO-ST [THEME 3]]
C
H
1 N′ [ACCENT U, INFO-ST [ ]]
hisho-ga
V [ACCENT U, TOPIC ⟨3 PP⟩, SPR ⟨2[MARKING ga]⟩, COMPS ⟨1 N′[SPR ⟨2⟩]⟩, INFO-ST [THEME 3]]
yuunoo desu

(36) represents the constituent tree notation of (33A$_1$). (32b) ensures that the theme phrase ‘Kankoku-wa’ subcategorized for by TOPIC appears in the sentence-initial position, which is higher than the rheme ‘daitooryoo-ga’. (32a) constrains the phrase selected by SPR to project the rheme value up to the mother, which results in the obligatory rheme interpretation for the first *ga*-marked phrase. Thus, Japanese employs base-generated gapless thematic-rhematic construction.

5 Conclusions and Future Work

This paper addresses fundamental questions regarding the multi-dimensional grammar for Japanese. IS is an integral part of the grammar, which interacts in principled ways with syntax, morphology, and phonology. We have shown that, unlike English but in a similar manner to Catalan, Japanese informational interpretation has the following characteristics:

– The sentence is restricted to at most one morphologically *wa*-marked but phonologically unmarked theme phrase, which, if present, appears in the sentence-initial position.
– The obligatory rheme with marker *ga* and either with or without phonological marking is also encoded by syntactically higher position only preceded by the theme.
– The optional rheme interpretation of one or more phrases marked with either *wa* or *ga* can be carried by receiving a high pitch in situ.

By assuming HPSG and its extension on IS, we have outlined a constraint-based system which can explain that the distribution of the particles *wa/ga* and dichotomy of *theme/rheme* is not in a one-to-one mapping relation, but the relevant grammar modules of syntax and phonology are mutually constrained.

We did not go into detail on the difference between *wa*-marked and *ga*-marked optional rhemes, and the relation among consequitive *ga*-marked phrases in MNC. We have proposed a base-generated multiple occurrence mechanism for them but will leave the semantic constraints for future work.

Acknowledgements

We are indebted to two anonymous CICLing reviewers, Mark Steedman, Michinao Matsui, Takeo Kurafuji, Robert Logie, and Pilar Valverde for their invaluable comments on an earlier version of this paper. All remaining inadequacies are our own. This research is partially supported by the Human Communication Research Centre in the School of Informatics at the University of Edinburgh and Grant-in-Aid for Scientific Research (C), 21500152 of the Japan Society for the Promotion of Science (JSPS).

References

1. Engdahl, E., Vallduví, E.: Information packaging in HPSG. In: Grover, C., Vallduví, E. (eds.) Edinburgh Working Papers in Cognitive Science. Studies in HPSG, vol. 12, pp. 1–32. Centre for Cognitive Science, University of Edinburgh (1996)
2. Erteschik-Shir, N.: The Dynamics of Focus Structure. Cambridge University Press, Cambridge (1997)
3. Vallduví, E.: The Informational Component. Garland, New York (1992)
4. Reinhart, T.: Pragmatics and linguistics: An analysis of sentence topic. Philosophica 27, 53–94 (1981)

5. Sag, I.A., Wasow, T., Bender, E.M.: Syntactic Theory: A Formal Introduction, 2nd edn. CSLI Publication, Stanford (2003)
6. Hoji, H.: Logical Form Constraints and Configurational Structures in Japanese. PhD thesis, University of Washington (1985)
7. Saito, M.: Some Asymmetries in Japanese and Their Theoretical Implications. PhD thesis, Massachusetts Institute of Technology (1985)
8. Chung, C., Kim, J.B., Sells, P.: On the role of argument structure in focus projections. In: Cihlar, J., Franklin, A., Kaiser, D., Kimbara, I. (eds.) Papers from the 39th Annual Meeting of the Chicago Linguistic Society (CLS39): The Main Session, vol. 1, pp. 386–404. University of Chicago, Chicago (2007)
9. Chafe, W.L.: Givenss, contrastiveness, definiteness, subjects, topics and points of view. In: Li, C. (ed.) Subject and Topic, pp. 25–56. Academic Press, New York (1976)
10. Selkirk, E.O.: Sentence prosody: Intonation, stress, and phasing. In: Goldsmith, J.A. (ed.) The Handbook of Phonological Theory, pp. 550–569. Blackwell, Malden (1995)
11. Kuroda, S.Y.: Generative Grammatical Studies in the Japanese Linguistics. PhD thesis, Massachusetts Institute of Technology (1965)
12. Kuno, S.: The Structure of the Japanese Language. The MIT Press, Cambridge (1973)

Assessing Lexical Alignment in Spontaneous Direction Dialogue Data by Means of a Lexicon Network Model

Alexander Mehler[1], Andy Lücking[2], and Peter Menke[2]

[1] Goethe University Frankfurt am Main
mehler@em.uni-frankfurt.de
[2] Bielefeld University
{andy.luecking,peter.menke}@uni-bielefeld.de

Abstract. We apply a network model of lexical alignment, called *Two-Level Time-Aligned Network Series*, to natural route direction dialogue data. The model accounts for the structural similarity of interlocutors' dialogue lexica. As classification criterion the directions are divided into effective and ineffective ones. We found that effective direction dialogues can be separated from ineffective ones with a hit ratio of 96% with regard to the structure of the corresponding dialogue lexica. This value is achieved when taking into account just nouns. This hit ratio decreases slightly as soon as other parts of speech are also considered. Thus, this paper provides a machine learning framework for telling apart effective dialogues from insufficient ones. It also implements first steps in more fine-grained alignment studies: we found a difference in the efficiency contribution between (the interaction of) lemmata of different parts of speech.

1 Motivation

According to the *Interactive Alignment Model* [1, *IAM*], mental representations of dialogue partners on all linguistic levels become more and more similar, i.e. *aligned*, during their communicative interaction. Since the linguistic levels – phonetic, lexical, syntactic, semantic, situation model – are interconnected, alignment propagates through these levels. Via this spreading of alignment, global alignment, that is, alignment of situation models, can be a result of local alignment on lower levels. Thus, the IAM provides an account to the ease and efficiency of dialogical communication beyond explicit negotiation. Part of the efficiency of communication is the fulfillment of the dialogue task or purpose. Consequently, we would expect that *more aligned dialogues are more successful* – a proposition we make productive below.

The central mechanism that is acknowledged within the IAM is *priming*.[1] Priming is typically understood and modeled as spreading activation in neural networks. Two varieties of activation have to be distinguished:

[1] But see [2] for an argument that priming cannot be the process that implements alignment.

1. A linguistic form /x/ activates its corresponding mental representation x within the interlocutors. We simply call this *activation*.
2. A representation y, which is activated by a form /y/, also activates representations which are related to y. The kind of relation depends on the linguistic level of which y is an element. For example, if y is the phonological representation of *can*, the phonologically similar representation *pan* may be co-activated. Since the contents of many cans can be heated in pans, the semantic representations of both forms also trigger each other. The mediated activation of a representation x by a form /y/ is termed *co-activation*.

The linguistic forms produced and perceived in a dyad do not only prime their corresponding representations, they also co-activate a set of related representations. A model that captures the structure of dialogue lexica of speakers is a network model of interlinked nodes. The nodes of this model represent linguistic elements of a certain kind. Since we are concerned with lexical alignment in this paper, the nodes in our model represent lemmata. In order to give an impression of the phenomena we are interested in, consider the following score of a dialogue extract:[2]

A:	*street lights*		*lamps*
B:		*street lamps*	

A and *B* talk about the same (plural) referent, what we will call the *topic* of a contribution. The term *A* proposes (*street lights*) is corrected by *B* (*street lamps*). *B*'s correction is then partly taken up by *A* (*lamps*). From the perspective of alignment, the dialogue lexica of the interlocutors contain three related nouns which are linked among each other in corresponding ways. The interlocutors finally align on the repeated use of a certain noun, namely *lamp*.

Observable evidence for alignment like the *lamp* example is ubiquitous in human communication. This notwithstanding, a correlation between the type of communication and extent of alignment has been reported. [4] found that speakers in a task-oriented dialogue setting are more receptive to priming than speakers in a spontaneous dialogue setting. The authors used common linear regression as the statistical analysis tool. Recently, [5] developed a network-based framework to model alignment in dialogue, the so-called *Two-Level Time-Aligned Network* (TiTAN) model. The TiTAN model has already been applied to strictly task-oriented dialogue data [6]. In this paper, we use the TiTAN model to assess alignment in more spontaneous dialogue data, namely direction dialogues. We do that by classifying dialogues for being effective or ineffective according to their main function, that is, direction giving.

In the following Section 2 we shortly point out two shortcomings of previous approaches to measure priming or alignment which are overcome by the TiTAN model introduced in Section 3. After that, the TiTAN model is applied to natural

[2] The extract is taken from dialogue no. 24 around second 600 of the collection from [3] – see Section 4 for some more details. In its German original form, the sequence of nouns is *Straßenlampen* – *Straßenlaternen* – *Laternen*.

language direction data. The data and the results are described in Section 4, which is followed by a conclusion that summarizes our findings.

2 Related Work

The approach followed here diverges in two respects from related work that tries to measure priming or alignment.

The earliest work on assessing alignment-related properties of (written) texts in quantitative terms is the lexical adaption model proposed by [7]. In a nutshell, Church measured the frequency of primed words in comparison to unprimed ones in the second half of split documents. A related measurement of the recurrence of syntactic patterns was conducted by [4], who account for the repetition of phrase structure rule instances within the Switchboard [8] and the HCRC Map Task [9] corpora.

A priming assessment that relates counting repeated elements to task achievement was implemented by [10]. They trained a *Support Vector Machine* (SVM) to predict task success from lexical and syntactic repetition in the HCRC Map Task corpus. Thus, the study is also precursor for the efficiency of aligned dialogue hypothesis pursued in the empirical part of this paper. The SVM is applied to time stamps in the data, indicating the proportion of variance that can be explained by the model.

The accounts for assessing priming effects in natural language data so far underlie two restrictions:

1. They focus on the *repetition* of elements, that is, they do not account for co-activation and linked representations.
2. They operate on fairly *arbitrary temporal units* that were artificially imposed on the data.

The model proposed by [5], the *Two-Level Time-Aligned Network* (*TiTAN*) model, avoids both afore-mentioned restrictions. The temporal units which carry the alignment process are dialogue turns, genuine components of conversations. The network structure allows for capturing co-activation of related elements. The next section explains how the TiTAN model of direction givings looks like.

3 Modeling Dialogue Lexica as TiTAN Series

During their conversation, interlocutors establish a so called *dialogue lexicon* [1] of commonly or differently used words. On the one hand, they may reuse words that their partner used the same way or at least similarly within their conversation. Alternatively, interlocutors may use the same words but for different things or may introduce new words that were not used before. Sameness (and conversely difference) of word usage, thus, is detected according to the extensional criterion of aboutness. What words are about is called their *topic* in the following. By speaking about similar word usages we refer to the similarity of the lexical contexts of words [11]. In the present scenario, this context is identified with the basic structural unit of dialogues, that is, the turn [12].

From this point of view, the generation of a dialogue lexicon is conceived as a process in which a lexical network grows turn by turn based on the word usages of the dialogue partners. In such a *dialogue lexicon network*, vertices denote lexical items while the strength of their edges denote the number of contexts in which these words co-occurred until the corresponding point in time. In this sense, the generation of a dialogue lexicon appears as a *time series* that emits lexical networks at its different time points. It is this combined notion of complex networks [13] and time series that is used to model the build-up of dialogue lexica as a result of dialogical communication. In this section, we briefly recapitulate this model in terms of so called *Two-Layer Time-Aligned Network* (TiTAN) series [5] and introduce its instantiation in the context of direction givings.

Generally speaking, a TiTAN series is a time series $\{L_t \,|\, t \in \quad \}$ of indexed graphs L_t that model the dialogue lexicon of a dyadic conversation at time t. Each of these graphs L_t is partitioned into two layers, A and B, representing each interlocutors' part of the dialogue lexicon. In order to instantiate the notion of a TiTAN series in the framework of direction givings, we start with formalizing dialogue lexica before we explain how TiTAN series are serialized.

Formally speaking, the dialogue lexicon of a dyadic conversation among two interlocutors A and B at time t is modeled as a labeled graph $L_t = (V, E_t, \mathcal{L})$. In this graph, the vertex set V is partitioned into non-empty disjunct subsets V_A and V_B whose elements denote the words used by interlocutor A and B, respectively, to perform the task of direction giving. The vertices in V are labeled by the surjective function $\mathcal{L} : V \to L_V$ where, in our case, the set of labels L_V consists of lemmata. Analogously, the edge set E_t is partitioned into three non-empty disjunct subsets E_{AB_t}, E_{A_t}, E_{B_t} where all edges $\{v, w\} \in E_{AB}$ end at vertices $v \in A, w \in B$, while all edges $\{x, y\} \in E_X$, $X \in \{A, B\}$, end at vertices $x, y \in V_X$. E_A and E_B capture intrapersonal lexical relations, while edges in E_{AB} are used to link lexical items shared among the interlocutors. The subgraphs $L_{A_t} = (V_A, E_{A_t}, \mathcal{L})$ and $L_{B_t} = (V_B, E_{B_t}, \mathcal{L})$ are called the A- and B-layer, respectively, of the two-layer graph $L_t = (V, E_t)$ at time t. They are denoted by the projection functions $\pi_A(L_t) = L_{A_t}$ and $\pi_B(L_t) = L_{B_t}$. In terms of our application area, layer A represents the dialogue lexicon of interlocutor A, layer B represents the dialogue lexicon of interlocutor B, while the graph L_t provides a unified model of their overall dyadic dialogue lexicon.

The networks defined so far model linguistic units and their relations. However, they do not distinguish between seldomly and frequently intantiated relations. This asymmetry is accounted for by assigning weights to the edges in L_t. Thus, dialogue lexica are modeled as weighted labeled graphs $L_t = (V, E_t, \mu_t, \mathcal{L})$ that are indexed by the point in time $t \in \quad$ at which they are spanned. Recall that t is derived from the dialogue turns of the interlocutors and, thus, from a dialog-inherent time-related ordering. In this sense, a TiTAN series is serialized according to the contributions of the interlocutors manifested and organized as turns. As a two-layer graph, L_t is divided into the subgraphs

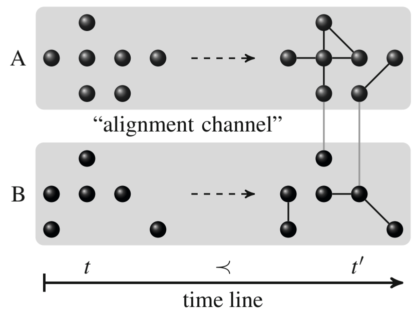
$\pi_A(L_t) = L_{A_t} = (V_A, E_{A_t}, \mu_{A_t}, \mathcal{L})$ and $\pi_B(L_t) = L_{B_t} = (V_B, E_{B_t}, \mu_{B_t}, \mathcal{L})$ according to the distribution of L_t over the agents A and B at time t. The spanning of edges within L_t is done as follows [5, p. 1453]:

– *Intrapersonal links:* if at time t, agent $X \in \{A, B\}$ uses a word form as an instance of lemma $l \in L_V$ to express the current turn's topic $T = T(t)$, intrapersonal links between vertex $v \in V_X, \mathbb{L}_X(v) = l$ are generated, and all other vertices w in layer L_{X_t} whose lemma $\mathcal{L}(w)$ was used by X in the same or any preceding turn to speak about the same topic T. If any of these edges $e = \{v, w\}$ already exists, its weight is incremented by 1, that is, $\mu_t(e) = \mu_{t-1}(e) + 1$. Initially, all edges have a weight of 1.
– *Interpersonal links:* if at time t, agent $X \in \{A, B\}$ uses a word form as an instance of lemma $l \in L_V$ to express the topic $T = T(t)$, which has been expressed by the dialogue partner $Y \neq X$ in any preceding turn on the same topic by means of the same lemma, an interpersonal link $\{v, w\} \in E_t$ between $v \in V_A$ and $w \in V_B$ is generated for which $\mathcal{L}(v) = \mathcal{L}(w) = l$, given that this link does not already exist. Otherwise, its weight is increased by 1.

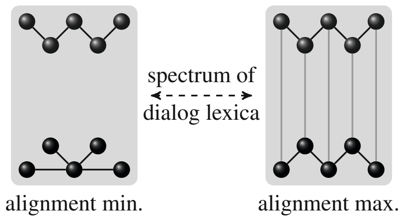
With the passing of time, this process generates a series of dialogue lexica L_t that are indexed by the corresponding time point t. Figure 1(a) provides a schematic visualization of this construction process of a TiTAN series. The starting point is given by completely unlinked dialogue lexica L_{A_0} and L_{B_0} of the interlocutors A and B. Following the afore-given construction procedure, the lexica are networked turn-wise by adding intra- and interpersonal links. A TiTAN, thus, allows for modeling for each turn the degree of structural coupling of the dialogue lexica of both interlocutors. It finally results in a dialogue lexicon that manifests the degree of lexical alignment at the end of the conversation of both interlocutors. To see this, look at Figure 1(b), which shows two extreme values of dialogue lexica: the lower bound is given by two layers L_{A_t} and L_{B_t} of the overall dialogue lexicon L_t that are completely disconnected and internally structured in completely different ways. Such a situation occurs if both agents always use different words or denote the same topics always differently. The upper bound is set by two isomorphic layer graphs that are fully linked. This scenario corresponds to a dialogue in which both agents always use the same words the same way. Due to thematic progression of natural dialogues, constraints by stylistics and verbal economy, and psychological factors of various kinds, neither of these extremal points is to be expected to be realized by dialogical conversation. They delineate, however, theoretical boundary values that make lexical alignment a measurable property [5].

In the framework of task-related conversations like direction giving, alignment is supposed to be bound up with communicative success, i.e. efficiency [1, p. 172].

The question arises how to measure whether a dialogical interchange is efficient or not. Using TiTAN series to represent such dialogues, we hypothesize that the class of effective directions can be separated from the class of ineffective directions in terms of the topology of the final state of the dialogue lexica L_t. In other words, we hypothesize that the way lexical items are connected and clustered in a dialogue lexicon informs about the status of the corresponding



(a) Illustration of a *Two-layer Time-Aligned Network* series. Initially, the lexica of interlocutors A and B are unlinked (left of dashed arrows). Dialogue turn by dialogue turn, the interlocutors' lexica are networked such that a dialogue lexicon emerges that is spanned by intra- and interpersonal links across the alignment channel (right of dashed arrows).



(b) The extrema of the graph-theoretical modeling of lexical alignment: completely independent (left) vs. fully linkage of identical lexica (right). Note that alignment minimum and maximum are theoretical extrema that are not to be expected to be found in real dialogues, which are supposed to populate the spectrum of dialog lexica.

Fig. 1. (Co-)Activation of representations within the dialogue networks of interlocutors A and B: A TiTAN series illustration (a) and structural extrema (b)

direction giving. If this is true, it should be possible to utilize complex network theory [13] to represent dialogue lexica by topological indices that are finally input to unsupervised learning of the class of effective and ineffective directions. This is the way, we proceed in this paper. More specifically, we apply *Quantitative Network Analysis* (QNA) [14,15] to represent and classify dialogue lexica by means of complex network theory. In the present area of application, QNA involves three steps of modeling:

1. *Quantitative graph modeling*: initially, each dialogue lexicon is represented by a vector of topological indices that model its network structure.
2. *Feature selection:* in the next step, a genetic search is performed to find salient features within the vectors that best separate effective and ineffective dialogues. Note that this process of feature selection may stop at a local maximum as it does not necessarily find an optimal feature subset.
3. *Classification:* based on the appropriately projected feature vectors, a hierarchical agglomerative clustering is performed together with a subsequent partitioning that is informed about the number of target classes. We use *complete linkage* together with the *Mahalanobis distance* to perform this step. Note that we use MATLAB to make any of these computations. Note also that the Mahalanobis distance is used to handle correlations between features.

To sum up, QNA takes the set of input dialogues together with the parameter space of linkage methods and distance measures to find out the feature subset that best separates the data according to the underlying classification. In the present study, we utilize a subset of indices of complex network theory together with a subset of indices that were invented to model dialogue lexica [5]. See [5] and [15] for a summary of this quantitative graph model. All in all, 50 topological indices were computed per input dialogue to model its structural characteristics. Note that we exclude simple frequency oriented indices (e.g., the number of vertices or edges). In Section 4, we discuss eight instantiations of this model by experimenting with a set of 25 dialogues about directions.

4 Experimentation

4.1 Data

The speech data the TiTAN model is applied to are taken from the *Bielefeld Speech and Gesture Alignment Corpus* (SaGA) [3]. The primary data of the SaGA corpus are made up of 25 direction dialogues. After finishing a simulated bus ride through a virtual town, one participant explains the route taken and some sights passed to a second participant.

Video and audio recordings were made of the experiments, and on their basis, an orthographic transcription of speech on the level of words has been created. Typical phenomena of spontaneous speech (for example, clitics, elisions, assimilations, and spontaneous neologisms) were transcribed according to guidelines in order to ensure consistency.

These transcriptions were tagged with part-of-speech and lemma information by a system consisting of the eTagger of the eHumanities Desktop [16], a central trigram HMM tagger that has been trained on the German Negra Corpus.[3] The *Stuttgart-Tübingen Tag Set* (STTS, cf. [17]) was used, along with pre- and post-processing mechanisms that are specialized in the handling of the phenomena of spoken language mentioned above. Preprocessing methods map recurring word form variants to their standardized counterparts before tagging. Postprocessing mechanism apply several heuristics to unrecognized words that help to identify neologisms – for example those that had been constructed from two or more known words (e.g., "Peitschenlampe'" – *whip lamp*, constructed from the nouns "Peitsche" / *whip* and "Lampe" / *lamp*). Still, there were word forms that could not be detected or handled automatically. These tokens have been manually corrected after applying the tagger.

Since we are not concerned with well-formedness or related grammatical notions, but rather with regularites of word use, the syntactically fine-grained POS of the STTS are too detailed. Thus, we mapped the STTS onto the functionally basic types N(oun), V(erb), ADJ(ective), ADV(erb) JUNC(tors), PREP(ositions), DET(erminers), PRO(nouns), and PART(icles). A fourth type, called REST, collects the remainder of POS like interjections and fragments. These basic categories are used in the construction of dialogue lexica.

[3] `http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/`

For the construction of a TiTAN series, information about turns and their respective topic are required. As a consequence, the turn boundaries needed for the construction of a TiTAN were annotated manually for all of the 25 dialogues. The topics we acknowledge are derived from the stages of the route through the virtual town of the primary data. The SaGA town and its 12 stages (topics) are shown in Figure 2. In addition, there is a 13th topic called *SaGA* which indicates turns that are about (large parts of) the whole virtual town. The label *META* is used to classify turns that do not relate to the route, but rather negotiate discourse issues or interpersonal concerns.

Each dialogue has been rated with respect to whether the interlocutors converge on a suitable description of the SaGA route. The criterion is whether the addressee has been put into the position to find the way from the sculpture to the fountain without going astray. We distinguish three cases or classes: 1. The direction is correct; 2. The direction is partially correct, but sufficient for the purpose to cross the SaGA town; 3. The direction is full of holes and useless. If, for instance, a participant mistakes the conifers in the park for leaf trees but apart from that gives a right direction, the dialogue is classified into the second class. Class 1 and 2 are grouped together into “correct” directions. In sum, there are 17 wrong and 8 correct directions. For each of these 25 dyads, a separate dialogue lexicon network has been built according to Section 3.

One might object that the occurrence of alignment is independent from the validity of the given direction. Note, however, that the classification of dialogues is not concerned with their correctness in the first place. In particular class 2 above accounts for directions that are false strictly speaking, but nonetheless carry enough information to let the addressee find the way. Finally, the class 3 dialogues are clearly faulty. *So what is the root of the matter?* [1, p. 172] emphasize that “alignment of situation models is central to successful dialogue”. No matter whether the situation models are correct or not, a precondition is that the dialogue participants have situation models at all! There are reasons to assume that this is the problem with class 3 dialogues. The direction givers’ models of the SaGA town are fragmentary – the models contain gaps. It is questionable whether fragmentary models can be conceived of as situation *models* at all. As it stands, we are aware of these theoretical obstacles, but regard our classification approach as feasible.

4.2 Evaluation

In this section, we describe the experimental scenarios by which we test our classification hypothesis introduced in Section 3. This hypothesis says that the efficiency of a direction giving in dialogical communication can be detected based on the topology of the final state of the corresponding dialogue lexicon. As described in Section 3, we test this hypothesis in the framework of *Quantitative Network Analysis* (QNA). More specifically, we test 8 different variants of this hypothesis (as summarized in Table 1):

– *Variant* [*N*]*:* we start with considering nouns only. The idea behind this approach is that nouns are mainly used by interlocutors to refer to the

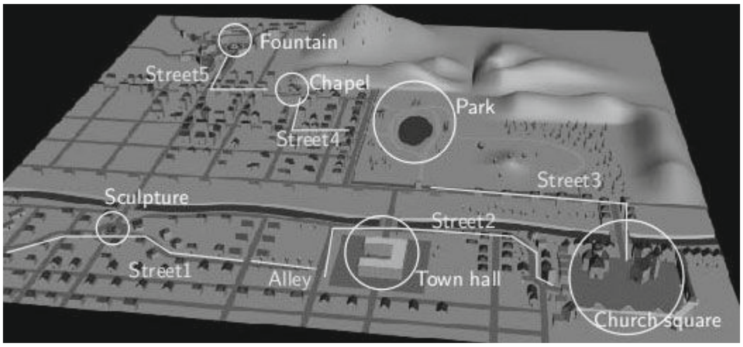


Fig. 2. Overview of the virtual SaGA town, with topics marked

reference universe of the direction giving – see the entities of the virtual SaGA town marked in Figure 2. We expect that the efficiency of directions is more easily identified by means of the nominal subnetworks of the corresponding dialogue networks.

– *Variant* $[N|A]$*,* $[N|V]$ *and* $[N|V|A]$*:* alternatively, we experiment with additionally considering adjectives and verbs. The reason to take these POS into account is that many of their instances have a descriptive meaning in relation to the reference universe of the direction giving (as, e.g., the verb *to turn* in *Turn to the left*).

In addition to these four variants, we consider those subnetworks that exclude words with a meta-communicative function (see the Rows 1–4 and the column *Meta*, which codes whether words are included that are tagged by *META* according to Section 4.1). These are words (as, e.g., *to think* in *Let me think*) that do not have a referential meaning regarding the reference universe of the direction giving, but serve, for example, to organize the dialogue. In our corpus of 25 direction givings, we have annotated 5,561 word forms with a meta-communicative function in relation to the overall set of 45,190 word forms that we were manually annotated. Thus, more than 10% of the word forms were used for meta-communicative reasons. From this perspective, one may expect an effect of excluding or including this class of words.

Table 1 summarizes the results of our findings. It shows that the best performing variant is based on selecting nouns without any meta-communicative function (see Row 1). This variant produces an F-score of more than 96%. The F-score (or F-measure) is the harmonic mean of recall and precision of the computed classification in relation to the correct classification of the data into 17 ineffective and 8 effective dialogues. An F-score of 96% means that nearly all dialogues have been classified correctly. If we additionally consider verbs, the F-score decreases to 92% (Row 2). The loss of classification is even higher if we separately consider the network of adjectives and that of verbs and adjectives (Row 3 and 4). Thus, although adjectives and verbs have denotational meanings

Table 1. Summary of the results of differently parameterized *Quantitative Network Analyses* (QNA). Row no. 9 shows the average F-score of these variants. The last column denotes the number of features output by the genetic search of the best performing subset of features as part of QNA (see Section 3).

No.	Setting	Meta	Procedure	F-score	Features
1.	$[N]$	yes	QNA	.96057	16 / 50
2.	$[N\|V]$	yes	QNA	.92	18 / 50
3.	$[N\|A]$	yes	QNA	.91651	20 / 50
4.	$[N\|V\|A]$	yes	QNA	.88171	21 / 50
5.	$[N]$	no	QNA	.92194	21 / 50
6.	$[N\|V]$	no	QNA	.87771	18 / 50
7.	$[N\|A]$	no	QNA	.88171	22 / 50
8.	$[N\|V\|A]$	no	QNA	.91651	24 / 50
9.	average over non-random approaches			.9096	20
10.	random baseline known-partition			.58668	
11.	random baseline equi-partition			.58583	

in the dialogues analyzed here, they do not help to separate the class of effective and ineffective direction givings to the same degree as nouns only.

These results seem to be contra-intuitive. Denotations of orientations and movements should be key ingredients of a successful direction giving. However, they are relational in character as they depend on the things they relate. Regarding situation models, a precondition for relational specification is that the objects in question are (correctly) spread out on the mental model. This in turn requires that the objects are available to the interlocutor. Objects are typically denoted by nouns or noun phrases. If the direction giver can name the things he wants to talk about, he can relate them to each other or to the direction follower. Thus, correct $[N|V|A]$-dialogues depend on correct $[N]$-dialogues. Besides this logical relationship, however, verbs, and adjectives may be the source for errors beyond nominal expressions. The decreasing F-score of $[N|V|A]$, $[N|V]$, and $[N|A]$ variants in comparison to the $[N]$ variant is very probably due to the asymmetrical status of the $[N]$ partitions of the dialogues in relation to their adjective- and verb-based partitions.

What happens if we additionally consider words with meta-communicative functions? As shown by the rows 5 through 8 in Table 1, there is a negative effect of including meta-communicative words. However, the differences being observed are rather marginal so that we conclude that there is only a small effect of either including or excluding this class of words. Meta-communicative acts typically provide information that the addressee has either understood the direction or that he could not follow. Thus, meta-communicative turns are used to convey a sort of binary information. As this information does not relate to the direction proper, it may be the reason for the lack of classificatory power being observed.

In order to further assess the quality of our results, we computed two random baselines (Row 10 and 11 in Table 1): the baseline called *known-partition* has information about the number of instances of the target categories. That is, by knowing that there are 17 ineffective and 8 effective dialogues, this baseline randomly generates two subsets of these cardinalities to compute the corresponding F-score. By repeating this procedure 1,000 times, we get an expected F-score of about 58%. This score is a little bit smaller if we consider the second random baseline that assumes equal sizes of the target categories (in our case 12 and 13). Obviously, all topology-related classifiers clearly outperform these two baselines. Thus, we can conclude, at least until any future falsification, that the efficiency of a direction giving is encoded into the *structure* of its dialogue lexicon.

5 Conclusion

In this paper, we applied *Two-Layer Time-Aligned Network* (TiTAN) series in the context of direction givings. Based on this graph model, we implemented several classifiers that solely explore the structure of dialogue lexica to assess their efficiency. By example of a corpus of 25 dialogues, we have shown that topological indices of dialogue lexica can indeed reveal this status. We also observed that lexical units with meta-communicative functions have a small effect on classification. This is in support of the observation that lexical manifestations of dialogue organization have a some impact on the efficiency of direction givings. Furthermore, we observed that the networking of nouns has the highest classificatory power, while the subnetworks of adjectives and verbs are less informative. One reason for this finding may be the outstanding referential meaning of nouns in conjunction with their semantic specificity. There are several POS that we did not consider here. Apart from adverbs, this relates to instances of closed POS. In future work will consider these classes and their role in the organization of dialogue lexica too.

Acknowledgements

This work has been supported by the German Research Foundation (DFG), which granted the Collaborative Research Center 673, and the German Ministry of Education and Research, which funds the *Linguistic Networks* project.

References

1. Pickering, M.J., Garrod, S.: Toward a mechanistic psychology of dialogue. Behavioral and Brain Sciences 27(2), 169–190 (2004)
2. Schiller, N.O., de Ruiter, J.P.: Some notes on priming, alignment, and self-monitoring. Behavioral and Brain Sciences 27, 208–209 (2004)
3. Lücking, A., Bergmann, K., Hahn, F., Kopp, S., Rieser, H.: The Bielefeld speech and gesture alignment corpus (SaGA). In: Multimodal Corpora: Advances in Capturing, Coding and Analyzing Multimodality, Malta, 7th International Conference for Language Resources and Evaluation (LREC 2010), pp. 92–98 (2010)

4. Reitter, D., Keller, F., Moore, J.D.: Computational modelling of structural priming in dialogue. In: Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, New York, NY, pp. 121–124 (2006)
5. Mehler, A., Lücking, A., Weiß, P.: A network model of interpersonal alignment in dialogue. Entropy 12(6), 1440–1483 (2010)
6. Mehler, A., Weiß, P., Menke, P., Lücking, A.: Towards a simulation model of dialogical alignment. In: Smith, A.D.M., Schoustra, M., de Boer, B., Smith, K. (eds.) The Evolution of Language. Proceedings of the 8th International Conference (EVOLANG8), pp. 238–245. World Scientific, Singapore (2010)
7. Church, K.W.: Empirical estimates of adaptation: The chance of two noriegas is closer to $p/2$ than p^2. In: Proceedings of Coling 2000, Saarbrücken, Germany, pp. 180–186 (2000)
8. Wheatley, B., Doddington, G., Hemphill, C., Godfrey, J., Holliman, E., McDaniel, J., Fisher, D.: Robust automatic time alignment of orthographic transcriptions with unconstrained speech. In: Proc. ICASSP 1992, vol. 1, pp. 533–536 (1992)
9. Anderson, A.H., Bader, M., Gurman Bard, E., Boyle, E., Doherty, G., Garrod, S.: The HCRC map task corpus. Lang. Speech 34, 351–366 (1991)
10. Reitter, D., Moore, J.K.: Predicting success in dialogue. In: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL), Praque, Czech Republic, pp. 808–815 (2007)
11. Miller, G.A., Charles, W.G.: Contextual correlates of semantic similarity. Language and Cognitive Processes 6(1), 1–28 (1991)
12. Schegloff, E.A.: Sequence Oeganization in Interaction. Cambridge University Press, Cambridge (2007)
13. Newman, M.E.J.: The structure and function of complex networks. SIAM Review 45, 167–256 (2003)
14. Mehler, A.: Structural similarities of complex networks: A computational model by example of wiki graphs. Appears in: Applied Artificial Intelligence (2008)
15. Mehler, A., Pustylnikov, O., Diewald, N.: Geography of social ontologies: Testing a variant of the Sapir-Whorf Hypothesis in the context of Wikipedia. Computer Speech and Language (2010)
16. Gleim, R., Waltinger, U., Ernst, A., Mehler, A., Esch, D., Feith, T.: The ehumanities desktop - an online system for corpus management and analysis in support of computing in the humanities. In: Proceedings of the Demonstrations Session of the 12th Conference of the European Chapter of the Association for Computational Linguistics EACL 2009, Athens (March 2009)
17. Schiller, A., Teufel, S., Thielen, C.: Guidelines für das Tagging deutscher Textcorpora mit STTS. Technical Report, IMS-CL, Universität Stuttgart (1995)

Towards Well-Grounded Phrase-Level Polarity Analysis

Robert Remus and Christian Hänig

University of Leipzig
Natural Language Processing Group
Department of Computer Science
04103 Leipzig, Germany
{rremus,chaenig}@informatik.uni-leipzig.de

Abstract. We propose a new rule-based system for phrase-level polarity analysis and show how it benefits from empirically validating its polarity composition through surveys with human subjects. The system's two-layer architecture and its underlying structure, i.e. its composition model, are presented. Two functions for polarity aggregation are introduced that operate on newly defined semantic categories. These categories detach a word's syntactic from its semantic behavior. An experimental setup is described that we use to carry out a thorough evaluation. It incorporates a newly created German-language data set that is made freely and publicly available. This data set contains polarity annotations at word-level, phrase-level and sentence-level and facilitates comparability between different studies and reproducibility of our results.

1 Introduction

With the advancing integration of the Internet into our everyday life, the amount of user generated content grows rapidly. People blog about their experiences, discuss in fora, author product reviews or twitter short messages. They do not stick to certain topics but write about everything of interest, e.g. holidays or recent purchases. In "Web 2.0", people express their *opinion* directly and frankly without being asked to do so and hence, this content has an immeasurable value for market research. While for marketing purposes, sentence- or even document-level analysis may suffice, a more *fine-grained analysis* is essential for deeper investigations established in business environments (e.g. quality assurance, competition analysis).

Whereas most approaches to sentiment analysis focus on two- or three-way classification of words (cf. [1]), sentences (cf. [2,3]) or complete documents (cf. [4]), with both rule-based or machine learning techniques, they all make a general assumption: each sentence or document deals with exactly one *topic*. This should be true for most product reviews, but within fora discussions or blog entries, people often address multiple topics. Typical sentences like *Ich mag X, aber Y sieht komisch aus.* (*I like X, but Y looks strange.*) contain more than one topic. Thus, in order to perform a thorough and fine-grained analysis, one has to delve deeper into the text – down to the *phrase-level*. At this level, we are able to extract relations (cf. [5]) between a topic and its associated polarity (e. g. *X* and *mag* (*like*), *Y* and *komisch* (*strange*)), which is crucial for topic-centric sentiment analysis.

We therefore propose a new rule-based system for phrase-level polarity analysis and illustrate its concepts in German. Although this has been done before for English, this work's novelty lies in its idea to overcome the nowadays predominant approach to rule-based polarity analysis. Instead of solely relying on *intuition* when modeling *polarity composition*, we either *empirically validate* its general notions by carrying out surveys with human subjects or base them upon findings of other fields of research, e.g. psychology.

1.1 Related Work

The body of sentiment analysis literature is large and nowadays literally encompasses more than a thousand studies. Many of them were extensively surveyed in [6]. However, work specifically tailored to phrase-level polarity analysis is quite sparse. That is partly because this task is possibly even harder than document- or sentence-level polarity analysis as there is *less context*, and partly because there is *no publicly available data set* which is fine-grained enough for training, testing and evaluation purposes on phrase-level in the sense of *phrase* as a *syntactic constituent*.

[7] laid the groundwork by extending [8]'s English-language MPQA corpus annotation scheme with annotations for contextual polarity. MPQA annotations among other features were then used in a two-step machine learning approach to first classify expressions in being neutral or polar, then to classify them being positive, negative or both. [9] later expanded their previous work by in-depth comparison of different machine learning algorithms and feature combinations. Their research differs from ours in that they focus on phrases in the sense of *word n-grams*, whereas we focus on *syntactic constituents*.

[10] chunked sentences into syntactic constituents and analysed them using the *Dictionary of Affect in Language* (DAL) and *n-gram techniques*. Perhaps closest to our research is [11]'s and [12]'s work: both "compose" sentence polarity out of phrase polarity according to the *semantic compositional principle*, whereas we exclusively aim to determine phrase polarity. The major drawback is that unfortunately both [11] and [12] rely on only commercially available parsers and not publicly available lexical resources. Thus, their algorithms and results are neither easily reproducible nor extendible. Additionally, the general notions and aggregation rules they build up on lack empirical validation.

1.2 Outline

This paper is outlined as follows: in the following Section we introduce the polarity composition model. A comprehensive evaluation is presented in Section 3, along with a now publicly available German-language data set for evaluation purposes. Finally, we draw conclusions in Section 4.

2 Polarity Composition Model

The polarity composition model (PCM) is a two-layered structure consisting of a word-level and a phrase-level polarity analysis, the latter depending on the former.

2.1 Word-Level Polarity Analysis

The goal of word-level polarity analysis is to identify the word form's *prior polarity*, i.e. its semantic orientation without any given context (cf. [9]). To determine a word form's prior polarity, it is searched for in [13]'s *SentiWS*. SentiWS is a publicly available German-language dictionary listing positively and negatively connotated words plus their part of speech (POS) tag, their inflections where available and a value $v \in [-1, 1]$ that expresses their polarity's *intensity*. A $v > 0$ suggests positive polarity, a $v < 0$ negative polarity. The bigger $|v|$ is, the more "expressive" is the word form.

The search incorporates POS tags, which act as a light-weight *word sense disambiguation* (cf. [14]). If the search is successful, the found word form is assigned the corresponding weight and a *category*.

Categories separate the word forms' syntactic functions from their semantic tasks. From our point of view, there are *potential polar categories* and *modifying categories*. Potential polar categories comprehend adjectives and adverbs, nouns and verbs. Modifying categories span negations as well as weakening and strengthening intensifiers. The word form's POS tag is mapped to an appropriate category, involving manually compiled lists of negations and intensifiers. Table 1 summarises the categories and their abbreviations as used here.

Table 1. Categories and their abbreviations

Abbr.	**Category**
ADJ	Adjectives, adverbs
N	Nouns
V	Verbs
NEG	Negations
INC	Strengthening Intensifiers
DEC	Weakening Intensifiers

A sub-category not yet addressed in this work is that of *relative adjectives* (cf. [15]) such as *klein* (*small*) – *groß* (*big*) or *niedrig* (*low*) – *hoch* (*high*). This category is particularly interesting, because its members show a somewhat irregular behavior: whereas *großer Verlust* (*big loss*) is negatively connotated in the context of economics, it might be positively connotated when talking about a person's body weight. In both cases *großer* (*big*) just intensifies the polarity of *Verlust* (*loss*). But one might imagine someone saying *Ganz groß!* (*This is big!*). In that case *groß* shows positive polarity on its own. Thus, relative adjectives are clearly *domain-dependent*.

2.2 Phrase-Level Polarity Analysis

Phrase-level polarity analysis builds up on word-level polarity analysis' results and aims to recognise the word form's *contextual polarity* (cf. [9]). To achieve this, the local context has to be taken into account – e.g. the (linguistic) *phrase structure*. The phrase structure with regard to polarity is determined by negations as well as weakening and strengthening intensifiers, i.e. the modifying categories we introduced earlier. [16] calls them *Contextual Valence Shifters* (CVS). Typical phrase types are verbal phrases (VPs), noun phrases (NPs) and prepositional phrases (PPs), among others.

Negations. The most prominent CVS are *negations*, e.g. *nicht* (*not*) as in *nicht schlecht* (*not bad*). But negations are not necessarily limited to *nicht*, they might as well appear in the form of words like *kein* (*no*), *niemals* (*never*), *ohne* (*without*) etc.

Currently, the best practice in sentiment analysis literature is "flipping" the polarity of a polar word form when proceeded by a negation (cf. for example [17]), i.e. turning a positive polarity negative and vice versa. In contrast to that, we believe in *affective asymmetries* as recently discussed in psychology (cf. [18]): positivity and negativity are not exact opposites, but have their very own characteristics. Thinking of the antonyms *gut* (*good*) and *schlecht* (*bad*) and assuming both have opposite polarities, *nicht gut* (*not good*) would have the same polarity as *schlecht* (*bad*) and *nicht schlecht* (*not bad*) would have same polarity as *gut* (*good*). Is that true? We believe, that

$$\left[X\ is\ \left\{\begin{matrix} good. \\ bad. \end{matrix}\right\}\right] \text{ is } \left\{\begin{matrix} \text{“more positive”} \\ \text{“more negative”} \end{matrix}\right\} \text{ than } \left[X\ is\ \left\{\begin{matrix} not\ bad. \\ not\ good. \end{matrix}\right\}\right].$$

Weakening and Strengthening Intensifiers. Besides negations, weakening and strengthening *intensifiers* are the most important CVS according to [16]. Examples are *sehr* (*very*), *viel* (*much*) and *selten* (*rarely*), *wenig* (*little*) etc. They mostly behave like we expect them to – they weaken or strengthen polar word forms:

$$\left[X\ is\ \left\{\begin{matrix} very\ good. \\ very\ bad. \end{matrix}\right\}\right] \text{ is } \left\{\begin{matrix} \text{“more positive”} \\ \text{“more negative”} \end{matrix}\right\} \text{ than } \left[X\ is\ \left\{\begin{matrix} good. \\ bad. \end{matrix}\right\}\right].$$

and, analogously

$$\left[X\ is\ \left\{\begin{matrix} rarely\ good. \\ rarely\ bad. \end{matrix}\right\}\right] \text{ is } \left\{\begin{matrix} \text{“less positive”} \\ \text{“less negative”} \end{matrix}\right\} \text{ than } \left[X\ is\ \left\{\begin{matrix} good. \\ bad. \end{matrix}\right\}\right].$$

Formal Construct. In order to account for these considerations and flexibly model the discussed phrase-internal dependencies, a *formal construct* is needed. Inspired by [11] and [12], who "composed" the polarity of complex lexical units out of atomic lexical units in a bottom-up fashion following the *compositional principle*, we now define such a construct's formal syntax and semantics.

Syntax A rule r has the form

$$r := [(d, f, p)\, \mathrm{CAT}_i \ldots \mathrm{CAT}_j]$$

with either $\mathrm{CAT}_k \in \{\mathrm{ADJ, N, V, NEG, INC, DEC}\}$ a category or a rule by itself, "…" an optional marker for discontinuity, $d \in \{\rightarrow, \leftarrow\}$ a direction, $f \in \{a_+, a_*\}$ an aggregation function and $p \in \mathbb{Q}$ a parameter obligatory for $a_\times$. One can *visualize* a rule as shown in Figure 1. Because of CAT_k's *recursive definition*,

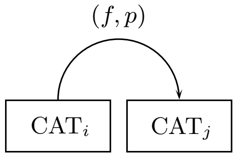


Fig. 1. Visualisation of the formal construct

it is possible to compile rules of arbitrary complexity, e.g.

$$[(d, f, p)\, [(d', f', p')\, \mathrm{CAT}_i\, \mathrm{CAT}_j]\ \mathrm{CAT}_k]$$

Moreover syntax allows for *one optional* category $\mathrm{CAT}_k \in \{NEG, INC, DEC\}$, expressed by surrounding the category in question with cambered brackets as in $[(d, f, p)\, [(d', f', p')\, \{\mathrm{CAT}_i\}\, \mathrm{CAT}_j]\, \mathrm{CAT}_k]$.

The motivation behind (d, f, p) and its values is as follows: d was introduced to allow for modifications not only of the *phrase head*, but also of parts of the phrase, e.g. in a VP like *funktioniert nicht gut* (*doesn't work well*) where *nicht* relates to *gut* instead of the phrase head *funktionieren* (*work*). f was introduced to allow for different kinds of interactions between potential polar and modifying categories. Finally, p was introduced to allow for flexible rule modeling. p values ideally should be empirically motivated, e.g. through small experiments like the following: we asked 8 judges, whether they perceive *kein schönes Auto* (*not a nice car*) or *kein sehr schönes Auto* (*not a very nice car*) as "more negative". 75% of the judges perceived the former as "more negative" as the latter. For that reason the factor p of NEG modifying ADJ is bigger than NEG modifying INC (cf. paragraph Modelling). Except for this specific instance, the remaining p values were chosen *intuitively*, but will be addressed in future work.

Semantics: Central to the interpretation of a rule r is the definition of the aggregation functions a_+ and $a_\times$. a_+ combines the polarity values $v_1, v_2 \in [-1, 1]$ of possibly complex word forms w_1, w_2 as shown in Equation 1.

$$a_+(v_1, v_2) = \begin{cases} v_1 + v_2, & \text{if } v_1, v_2 \geq 0 \text{ or } v_1, v_2 \leq 0 \\ \min\{v_1, v_2\} - \max\{v_1, v_2\} & \text{else} \end{cases} \tag{1}$$

If v_1, v_2 are *both* positive or *both* negative, a_+ acts as an ordinary addition. However, if their polarity differs, we subtract the larger from the smaller value, i.e. the result is *always negative*. Although this may seem contra-intuitive at first sight, we carried out two small surveys to verify its benefits.

We randomly extracted a 100 word pairs from a corpus containing 16.7 million sentences. Each word pair contains one word form that is positively connotated according to SentiWS, and one that is negatively connotated. Examples are *übertriebene Pflege* (*exaggerated care*) or *mächtiges Problem* (*mighty problem*). We then asked 9 judges to judge these word pairs regarding their overall polarity. Their judgements reach an inter-annotator agreement of $\kappa = 0.71$ and are therefore considered significant. 73% of all word pairs were judged being of negative polarity, 17.78% were judged positive and 9.22% were judged as being neutral or unclear. Thus, if a_+ is calculated as given in Equation 1 the sign of the polarity is correct in at least 73% of all cases and wrong in 27% of all cases at most.

Although this survey verifies our hypothesis of phrases containing a positive and a negative word as being most probably negative, some cases still remain worthy of further discussion. Especially the influence of the polarity's intensity on the phrase's overall polarity requires additional studies as currently even weakly negative terms turn the polarity of positive phrases into negative.

To verify $\min\{v_1, v_2\} - \max\{v_1, v_2\}$'s usefulness itself we then selected all word pairs that were judged being negative by at least 7/9 raters and were judged being positive by at most 1/9 raters. 65 word pairs fulfilled these criteria. We then asked 11 judges to judge whether the whole word pair is *more expressive* than the head of the word pair alone, i.e. we asked whether *übertriebene Pflege* is more expressive than *Pflege* (*care*). Although the inter-annotator agreement was lower ($\kappa = 0.5$), the judges decided for the whole word pair being more expressive 69% of the time. 22% of the time they decided that it was not more expressive and in 9% of the time they could not make a differientation. Thus, if a_+ is calculated as given in Equation 1 the calculated polarity intensity is "correct" in at least 69% of all cases.

$a_\times$ combines the polarity values $v_1, v_2 \in [-1, 1]$ of possibly complex word forms w_1, w_2 and a parameter p in a multiplication-like fashion as shown in Equation 2.

$$a_\times(v_1, v_2, d, p) = \begin{cases} p \cdot v_1 \cdot v_2, & \text{if } d \in \{\rightarrow, \leftarrow\}, v_1, v_2 \neq 0 \\ p \cdot v_2, & \text{if } d = \rightarrow, v_1 = 0, v_2 \neq 0 \\ p \cdot v_1, & \text{if } d = \leftarrow, v_1 \neq 0, v_2 = 0 \\ p & \text{else} \end{cases} \tag{2}$$

Once a_+ or $a_\times$ is calculated, both functions are *normalised* through a as shown in Equation 3 in order to ensure comparability of their results. The bigger $|a|$ is, the more "expressive" is the composition of w_1 and w_2.

$$a(a_.) = \begin{cases} \min\{a_., 1, 0\} & \text{if } a_. > 0 \\ \max\{a_., -1, 0\} & \text{if } a_. < 0 \end{cases} \tag{3}$$

The categories function as *wildcards*. The phrase *kein Erfolg* (*no success*) could be captured by the negation category NEG and the noun category N in the rule

$[(d, f, p)\,[(d', f', p')\,\mathrm{NEG}\,\mathrm{N}]]$ that then could be interpreted by applying the aggregation functions according to the tripels (d, f, p) and (d', f', p'). If there's an optional category, the rule is interpreted as if there were two, so from the rule including an optional category $[(d, f, p)\,[(d', f', p')\,\{\mathrm{CAT}_i\}\,\mathrm{CAT}_j]\,\mathrm{CAT}_k]$ we derive $[(d, f, p)\,[(d', f', p')\,\mathrm{CAT}_i\,\mathrm{CAT}_j]\,\mathrm{CAT}_k]$ and $[(d, f, p)\,\mathrm{CAT}_j\,\mathrm{CAT}_k]$. The discontinuity marker allows for matching non-adjacent categories, e.g. matching *zuverlässiger und hilfsbereiter Freund* (*reliable and helpful friend*) by a rule like $[(d, f, p)\,[(d', f', p')\,\mathrm{ADJ}\ldots\,\mathrm{ADJ}]\,\mathrm{N}]$.

Modeling: Although the rule base is included in [19] and is fairly self-explanatory, it is important to mention some considerations regarding the PCM. We decided to use $a_\times$ whenever modeling aggregations in which modifying categories take part in. If $a_\times$ is used for modeling a negation that *directly* modifies a polar category, p is set to -0.75. If a negation modifies another modifying category, p is set to -0.4, to account for our earlier discussion of affective asymmetries and directions of modifications. Intensifiers behave *symmetrically*: if $a_\times$ is used for modeling a strengthening intensifier, p is set to 1.5, thus increasing its strength by 50%. If used for a weakening intensifier, p is set to 0.5, thus decreasing its strength by 50%. So far we mostly relied on intuition when choosing p's values. They are certainly subject to debate, but we hope to verify them or choose more appropriate values in future work.

Aggregations in which only potential polar categories participate are modeled using a_+.

Example: To illustrate the PCM's behavior, the polarity composition of the VP *funktioniert bei Regen nicht sehr gut* (*doesn't work very well when it's raining*) is now presented in detail. The word-level polarity analysis assigns the category V and the weight 0.004 to *funktioniert*, NEG to *nicht*, INC to *sehr* and ADJ and 0.372 to *gut*. The phrase-level polarity analysis now matches the VP to an appropriate rule: $[(\leftarrow, a_+)\mathrm{V}\ldots[(\rightarrow, a_\times, 1.5)\,[(\rightarrow, a_\times, -0.4)\mathrm{NEG}\,\mathrm{INC}]\;\mathrm{ADJ}]]$ (cf. Figure 2). The rule matches because it allows for categories in between V and NEG, e.g. *Regen* – N – through the marker for discontinuity. Rules are processed *inside out*, so $[(\rightarrow, a_\times, -0.4)\mathrm{NEG}\,\mathrm{INC}]$ is interpreted first. $a_\times(0.0, 0.0, \rightarrow, -0.4) = -0.4$ (cf. Equation 2) is calculated and hence the expression *nicht sehr*

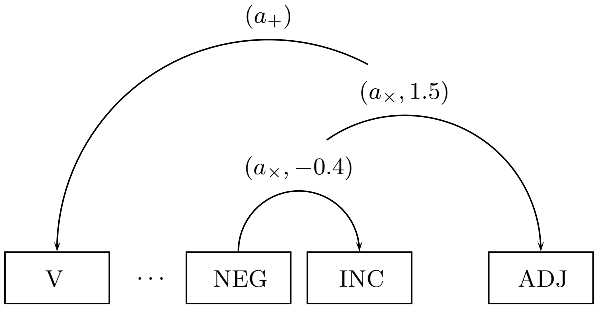


Fig. 2. Visualisation of the example

(*not very*) is now weighted -0.4. Next $[(\rightarrow, a_\times, 1.5) - 0.4\,\mathrm{ADJ}]$ is interpreted and $a_\times(-0.4, 0.372, \rightarrow, 1.5) = -0.2232$ is calculated. The expression *nicht sehr gut* (*not very good*) is now weighted -0.2232. Last $[(\leftarrow, a_+)\mathrm{V} \ldots - 0.2232]$ is interpreted and according to Equation 1 $a_+(0.004, -0.2232) = -0.2272$ is calculated. The whole VP *funktioniert bei Regen nicht sehr gut* is now weighted -0.2272 and is therefore considered being of *negative polarity.*

3 Evaluation

For evaluation purposes our experimental setup consists of an implementation similar to [19], and a newly created German-language data set included in [19]. As our approach heavily relies on POS tags and identified phrase structure, we employ the *Stanford POS Tagger* (cf. [20,21]) and *Stanford Parser* (cf. [22,23]) which also includes a *tokenizer.* Both provide state of the art performance for German as well as a few other languages and are publicly available.

3.1 Data Set

Unfortunately, there is no German-language data set publicly available, that provides polarity annotations at word-level, phrase-level and sentence-level. Therefore, in order to evaluate the PCM's performance we created a new one and made it freely available (cf. [19]). 1,000 sentences containing the term *W212*[1] and 1,000 sentences containing the the term *Rost* (*rust*) were randomly selected from a corpus comprising 16.7 million sentences. This corpus consists of posts to a variety of internet fora focussing on automobiles. All 2,000 sentences were then manually categorised as being positive, negative or neutral, i.e. showing no overt polarity.

Out of these 2,000 sentences, for each polarity category 133 sentences containing the term *W212* and 33 sentences containing the term *Rost* were selected, leaving 166 sentences of positive, 166 sentences of negative and 166 sentences of neutral polarity. Sentences not parseable, mostly because they were ill-formed or too long, were removed, resulting in a final data set of 477 sentences. The minimal sentence length is 4 word forms, the maximum is 41 word forms and the approximate average is 15.96 word forms.

The whole data set was *tokenized* and *parsed.* All resulting word forms and phrases were then annotated by two raters. They were allowed to annotate positive and negative polarity, as well as no or ambiguous polarity. In addtion, raters were allowed to “override” and replace sentence annotations, if they disagreed with the given polarity. The inter-annotator agreement, a free-marginal variant of Cohen's Kappa we also used earlier (cf. [24], [25]) is $\kappa = 0.76$ for word-level polarity and $\kappa = 0.58$ for phrase-level polarity.

So whereas the word-level polarity shows “substantial agreement” according to [26], the phrase-level polarity only shows “moderate agreement”. These values meet our expectations: it is quite easy to agree on the polarity of isolated word

[1] W212 denotes Mercedes Benz's E-Class model manufactured since 2009.

forms, but it is not as easy to agree on longer chunks, or phrases. Because of that, we report all our results compared to the first rater's and compared to the second rater's annotations, as well as compared to their *consensus*.

3.2 Results

In order to evaluate the PCM, we created two experimental setups. Within the first setup, errors originating from the POS tagger, parser or SentiWS are not taken into account. The second setup measures the performance of the complete workflow including the pre-processing steps which are explained earlier. In both scenarios, we provide results of a baseline algorithm for comparison. The baseline algorithm simply sums up the polarity values $v_1, \ldots, v_k$ of the word forms $w_1, \ldots, w_k$ contained within the target phrase. Its output is compared to the consensus of Rater 1 and 2.

Phrase-Level. The results of the phrase-level evaluation are given in Table 2. In this scenario, only 3,177 of the total 3,597 phrases are regarded, the remaining 420 are discarded due to erroneous pre-processing.

Table 2. Precision P, recall R and f-measure $F_{\alpha=0.6}$ for the phrase-level polarity analysis compared to $\text{Rater}_{1,2}$'s annotations and their consensus. Errors that are either POS tagger-, parser- or SentiWS-based were ignored.

	Positive			**Negative**			**Overall**		
	P	R	F	P	R	F	P	R	F
Rater 1	0.90	0.89	0.89	0.88	0.87	0.88	0.89	0.88	0.89
Rater 2	0.77	0.87	0.81	0.81	0.90	0.85	0.79	0.89	0.83
Consensus	0.91	0.91	0.91	0.95	0.92	0.94	*0.93*	*0.91*	*0.92*
Baseline	0.70	0.78	0.73	0.77	0.75	0.76	0.74	0.76	0.75

When comparing the consensus to the baseline, our PCM achieves significant improvements for precision, recall and f-score – 25.7%, 19.7% and 22.7%, respectively. It is notable that the difference between the baseline and the results achieved by our PCM drops when being compared to a single rater. The results are still very competitive, achieving improvements of 10.7% and 18.7%, respectively. The drop reflects the difficulty in defining an *indisputable* evaluation data set for sentiment analysis. Therefore, it is our opinion that it is adequate to compare approaches considering undisputed annotations, i.e. the consensus, in the first place.

We will analyse common error types of the PCM and discuss potential solutions to them in Section 3.3.

System-Level. At system-level, all 3,597 phrases are taken into account. The results are given in Table 3. As for the token-level (cf. [13]) and the phrase-level, the precision values for negative phrases are higher than they are for positive phrases. This may be a direct consequence of the tendency to use positive expressions not only in positive contexts, but also in neutral or even negative ones.

Table 3. Precision P, recall R and f-measure $F_{\alpha=0.6}$ for the system-level polarity analysis compared to Rater$_{1,2}$'s annotations and their consensus

	Positive			**Negative**			**Overall**		
	P	R	F	P	R	F	P	R	F
Rater 1	0.53	0.60	0.56	0.68	0.54	0.62	0.59	0.57	0.58
Rater 2	0.45	0.60	0.50	0.61	0.55	0.59	0.52	0.58	0.54
Consensus	0.51	0.67	0.56	0.70	0.62	0.67	*0.59*	*0.64*	*0.61*
Baseline	0.45	0.59	0.49	0.65	0.56	0.61	0.53	0.58	0.55

Examples include phenomena like ironic and sarcastic propositions like *Na toll!* (*Oh great!*), or set phrases like *Guten Morgen!* (*Good morning!*).

The relative improvements achieved by our model at system-level are 11.3% for precision, 10.3% for recall and 10.9% for the f-score.

3.3 Discussion

After presenting the results of our experiments, we thoroughly analyse common error types.

– Phrases containing *irony and sarcasm* are not yet addressed by the PCM, consequently, misclassifications occur. A set phrase or idiom like *Gute Nacht* (*Good night*) is classified as positive due to the positive prior polarity of *Gute* (*Good*). But when used in the sense of *aus und vorbei* (*it's over*), this phrase can also be of negative polarity. In order to avoid such errors, an additional classification into sarcastic and non-sarcastic sentences (cf. [27]) may be essential.
– *Colloquial language* is very typical for textual resources obtained from Internet fora or blogs. They contain many neologisms and arbitrary phrase constructions that have to be dealt with. Negations like the number 0 as in *0 Probleme* (*0 problems*) represent only the tip of the iceberg. To address this problem, a continuous editing of the language resources along with an enrichment by contextual information (e.g. matching *0* either as a number or as a negation) is necessary.
– *Interactions between nouns and verbs denoting presence or absence* are not yet supported by our model. Nevertheless, for correct analyses of sentences like *Rost ist nicht vorhanden* (*Rust is not present*) this is important. Similarly, *interactions between nouns and nouns denoting presence or absence* as in *ein Auto, das am besten durch Abwesenheit von Geräuschen und Vibrationen charakterisiert wird* (*a car best characterized by the absence of noise and vibration*) exist, too. An introduction of appropriate categories (i.e. ABS (absence) and PRE (presence)) and their integration into the PCM possibly helps to solve this problem.
– The lack of a distinction between *relative and absolute adjectives* (cf. [15]) entails several erroneous analyses (see Section 2.1). Analogous to words denoting presence or absence, the establishment of a new category for relative adjectives might be a sufficient extension of our model.

Contrary to the error types that have been mentioned so far that may be addressed by extending the PCM, *disputatious polarities* present a far more complex problem: sometimes, even human raters can not fully agree whether a lexical unit, e.g. a phrase, is of negative or positive polarity, or no polarity at all. Some phrases in our data set were first independently marked with identical polarities by the raters, but subsequent discussions then lead to uncertainty about their "correct" polarity. Especially comparitive utterances like *es hat weniger Rost als das andere Auto* (*it has less rust than the other car*) are difficult to judge.

4 Conclusions and Future Work

We proposed a new rule-based system for polarity analysis that benefits from empirical validation of its notions, thereby achieving competitive results. In doing so, we pointed out new ways to look at very interesting aspects of sentiment analysis, e.g. the fact that polarity composition and its underlying aggregations may sometimes be contra-intuitive at first sight. While our experiments were carried out using German-language resources, we believe many, if not all, of our insights are easily transferable to other languages. We strongly encourage others to utilise our findings and make extensive use of surveys and the like for verifying decisions taken in polarity analysis systems to be right or wrong.

Future research directions include investigating in sub-categories such as the above mentioned *relative adjectives* and interactions between them and other categories. Also we would like to examine interdependencies of adjacent lexical units on higher levels, e.g. phrase to phrase and sentence to sentence or phrase to sentence, sentence to document and vice versa (cf. [28]). Keeping these recommendations in mind, we believe it is promising to further study polarity composition, aggregation functions and their application.

Acknowledgments

We gratefully acknowledge the effort of all participants in our surveys and the numerous insightful discussions we had. Special thanks goes to the anonymous reviewers whose useful comments and suggestions considerably improved the original paper.

References

1. Hatzivassiloglou, V., McKeown, K.: Predicting the Semantic Orientation of Adjectives. In: Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 174–181 (1997)
2. Kim, S., Hovy, E.: Determining the Sentiment of Opinions. In: Proceedings of the 20th International Conference on Computational Linguistics (COLING), pp. 1367–1373 (2004)
3. Kim, S., Hovy, E.: Automatic Detection of Opinion Bearing Words and Sentences. In: Proceedings of the 2nd International Joint Conference on Natural Language Processing, IJCNLP (2005)

4. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? Sentiment Classification using Machine Learning Techniques. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 79–86 (2002)
5. Hänig, C., Schierle, M.: Relation Extraction based on Unsupervised Syntactic Parsing. In: Proceedings of the Conference on Text Mining Services, pp. 65–70. University of Leipzig, Leipzig (2009)
6. Pang, B., Lee, L.: Opinion Mining and Sentiment Analysis. Foundations and Trends in Information Retrieval 2(1-2), 1–135 (2008)
7. Wilson, T., Wiebe, J., Hoffmann, P.: Recognizing Contextual Polarity in Phrase-level Sentiment Analysis. In: Proceedings of the Conference on Human Language Technology (HLT) and Empirical Methods in Natural Language Processing (EMNLP), pp. 347–354 (2005)
8. Wiebe, J., Wilson, T., Bruce, R., Bell, M., Martin, M.: Learning Subjective Language. Computational Linguistics 30(3), 277–308 (2004)
9. Wilson, T., Wiebe, J., Hoffmann, P.: Recognizing Contextual Polarity: an Exploration of Features for Phrase-level Sentiment Analysis. Computational Linguistics 35(3), 399–433 (2009)
10. Agarwal, A., Biadsy, F., Mckeown, K.: Contextual Phrase-level Polarity Analysis Using Lexical Affect Scoring and Syntactic N-grams. In: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics, pp. 24–32 (2009)
11. Neviarouskaya, A., Prendinger, H., Ishizuka, M.: Compositionality Principle in Recognition of Fine-grained Emotions from Text. In: Proceedings of the 3rd International Conference on Weblogs and Social Media (ICWSM), pp. 278–281 (2009)
12. Moilanen, K., Pulman, S.: Sentiment Composition. In: Proceedings of the International Conference on Recent Advances in Natural Language Processing, RANLP (2007)
13. Remus, R., Quasthoff, U., Heyer, G.: SentiWS – a Publicly Available German-language Resource for Sentiment Analysis. In: Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC), pp. 1168–1171 (2010)
14. Wilks, Y., Stevenson, M.: The Grammar of Sense: Using Part-of-Speech Tags as a First Step in Semantic Disambiguation. Natural Language Engineering 4(2), 135–143 (1998)
15. Lyons, J.: Semantics, vol. 2. Cambridge University Press, Cambridge (1977)
16. Polanyi, L., Zaenen, A.: Contextual Valence Shifters. In: Shanahan, J., Qu, Y., Wiebe, J. (eds.) Computing Attitude and Affect in Text: Theory and Application. The Information Retrieval Series, vol. 20, pp. 1–9. Springer, Heidelberg (2006)
17. Das, S., Chen, M.: Yahoo! for Amazon: Sentiment Extraction from Small Talk on the Web. Management Science 53(9), 1375–1388 (2007)
18. Cacioppo, J., Larsen, J., Smith, N., Berntson, G.: What Lurks below the Surface of Feelings? In: Manstead, A., Frijda, N., Fischer, A. (eds.) Feelings and Emotions: the Amsterdam Symposium, pp. 223–242. Cambridge University Press, Cambridge (2004)
19. Remus, R., Hänig, C.: Software and data set accompanying this work (2011), `www.CICLing.org/2011/software/70`
20. Toutanova, K., Manning, C.: Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-speech Tagger. In: Proceedings of Joint Conference on Empirical Methods in Natural Language Processing (EMNLP) and Very Large Corpora (VLC), pp. 63–71 (2000)

21. Toutanova, K., Klein, D., Manning, C., Singer, Y.: Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network. In: Proceedings of the Human Language Technologies: North American Chapter of the Association for Computational Linguistics (HLT-NAACL), pp. 173–180 (2003)
22. Klein, D., Manning, C.: Accurate Unlexicalized Parsing. In: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL), pp. 423–430 (2003)
23. Klein, D., Manning, C.: Fast Exact Inference with a Factored Model for Natural Language Parsing. Advances in Neural Information Processing Systems 15, 3–10 (2003)
24. Cohen, J.: A Coefficient of Agreement for Nominal Scales. Educational and Psychological Measurement 20, 37–46 (1960)
25. Randolph, J.: Free-Marginal Multirater Kappa (multirater κfree): An Alternative to Fleiss Fixed-Marginal Multirater Kappa. Presented at the Joensuu Learning and Instruction Symposium (2005)
26. Landis, J., Koch, G.: The Measurement of Observer Agreement for Categorical Data. Biometrics 33(1), 159–174 (1977)
27. Tsur, O., Davidov, D., Rappoport, A.: ICWSM – A Great Catchy Name: Semi-Supervised Recognition of Sarcastic Sentences in Online Product Reviews. In: Proceedings of the Forth International AAAI Conference on Weblogs and Social Media, Washington D.C., USA, pp. 162–169 (2010)
28. McDonald, R., Hannan, K., Neylon, T., Wells, M., Reynar, J.: Structured Models for Fine-to-Coarse Sentiment Analysis. In: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 432–439 (2007)

Implicit Feature Identification via Co-occurrence Association Rule Mining

Zhen Hai, Kuiyu Chang, and Jung-jae Kim

School of Computer Engineering, Nanyang Technological University, Singapore
{haiz0001, askychang, jungjae.kim}@ntu.edu.sg

Abstract. We apply a network model of lexical alignment, called *Two-Level Time-Aligned Network Series*, to natural route direction dialogue data. The model accounts for the structural similarity of interlocutors' dialogue lexica. As classification criterion the directions are divided into effective and ineffective ones. We found that effective direction dialogues can be separated from ineffective ones with a hit ratio of 96% with regard to the structure of the corresponding dialogue lexica. This value is achieved when taking into account just nouns. This hit ratio decreases slightly as soon as other parts of speech are also considered. Thus, this paper provides a machine learning framework for telling apart effective dialogues from insufficient ones. It also implements first steps in more fine-grained alignment studies: we found a difference in the efficiency contribution between (the interaction of) lemmata of different parts of speech.

1 Motivation

According to the *Interactive Alignment Model* [1, *IAM*], mental representations of dialogue partners on all linguistic levels become more and more similar, i.e. *aligned*, during their communicative interaction. Since the linguistic levels – phonetic, lexical, syntactic, semantic, situation model – are interconnected, alignment propagates through these levels. Via this spreading of alignment, global alignment, that is, alignment of situation models, can be a result of local alignment on lower levels. Thus, the IAM provides an account to the ease and efficiency of dialogical communication beyond explicit negotiation. Part of the efficiency of communication is the fulfillment of the dialogue task or purpose. Consequently, we would expect that *more aligned dialogues are more successful* – a proposition we make productive below.

The central mechanism that is acknowledged within the IAM is *priming*.[1] Priming is typically understood and modeled as spreading activation in neural networks. Two varieties of activation have to be distinguished:

1. A linguistic form /x/ activates its corresponding mental representation x within the interlocutors. We simply call this *activation*.

[1] But see [2] for an argument that priming cannot be the process that implements alignment.

2. A representation y, which is activated by a form /y/, also activates representations which are related to y. The kind of relation depends on the linguistic level of which y is an element. For example, if y is the phonological representation of *can*, the phonologically similar representation *pan* may be co-activated. Since the contents of many cans can be heated in pans, the semantic representations of both forms also trigger each other. The mediated activation of a representation x by a form /y/ is termed *co-activation*.

The linguistic forms produced and perceived in a dyad do not only prime their corresponding representations, they also co-activate a set of related representations. A model that captures the structure of dialogue lexica of speakers is a network model of interlinked nodes. The nodes of this model represent linguistic elements of a certain kind. Since we are concerned with lexical alignment in this paper, the nodes in our model represent lemmata. In order to give an impression of the phenomena we are interested in, consider the following score of a dialogue extract:[2]

A:	*street lights*		*lamps*
B:		*street lamps*	

A and *B* talk about the same (plural) referent, what we will call the *topic* of a contribution. The term *A* proposes (*street lights*) is corrected by *B* (*street lamps*). *B*'s correction is then partly taken up by *A* (*lamps*). From the perspective of alignment, the dialogue lexica of the interlocutors contain three related nouns which are linked among each other in corresponding ways. The interlocutors finally align on the repeated use of a certain noun, namely *lamp*.

Observable evidence for alignment like the *lamp* example is ubiquitous in human communication. This notwithstanding, a correlation between the type of communication and extent of alignment has been reported. [4] found that speakers in a task-oriented dialogue setting are more receptive to priming than speakers in a spontaneous dialogue setting. The authors used common linear regression as the statistical analysis tool. Recently, [5] developed a network-based framework to model alignment in dialogue, the so-called *Two-Level Time-Aligned Network* (TiTAN) model. The TiTAN model has already been applied to strictly task-oriented dialogue data [6]. In this paper, we use the TiTAN model to assess alignment in more spontaneous dialogue data, namely direction dialogues. We do that by classifying dialogues for being effective or ineffective according to their main function, that is, direction giving.

In the following Section 2 we shortly point out two shortcomings of previous approaches to measure priming or alignment which are overcome by the TiTAN model introduced in Section 3. After that, the TiTAN model is applied to natural language direction data. The data and the results are described in Section 4, which is followed by a conclusion that summarizes our findings.

[2] The extract is taken from dialogue no. 24 around second 600 of the collection from [3] – see Section 4 for some more details. In its German original form, the sequence of nouns is *Straßenlampen – Straßenlaternen – Laternen.*

2 Related Work

The approach followed here diverges in two respects from related work that tries to measure priming or alignment.

The earliest work on assessing alignment-related properties of (written) texts in quantitative terms is the lexical adaption model proposed by [7]. In a nutshell, Church measured the frequency of primed words in comparison to unprimed ones in the second half of split documents. A related measurement of the recurrence of syntactic patterns was conducted by [4], who account for the repetition of phrase structure rule instances within the Switchboard [8] and the HCRC Map Task [9] corpora.

A priming assessment that relates counting repeated elements to task achievement was implemented by [10]. They trained a *Support Vector Machine* (SVM) to predict task success from lexical and syntactic repetition in the HCRC Map Task corpus. Thus, the study is also precursor for the efficiency of aligned dialogue hypothesis pursued in the empirical part of this paper. The SVM is applied to time stamps in the data, indicating the proportion of variance that can be explained by the model.

The accounts for assessing priming effects in natural language data so far underlie two restrictions:

1. They focus on the *repetition* of elements, that is, they do not account for co-activation and linked representations.
2. They operate on fairly *arbitrary temporal units* that were artificially imposed on the data.

The model proposed by [5], the *Two-Level Time-Aligned Network* (*TiTAN*) model, avoids both afore-mentioned restrictions. The temporal units which carry the alignment process are dialogue turns, genuine components of conversations. The network structure allows for capturing co-activation of related elements. The next section explains how the TiTAN model of direction givings looks like.

3 Modeling Dialogue Lexica as TiTAN Series

During their conversation, interlocutors establish a so called *dialogue lexicon* [1] of commonly or differently used words. On the one hand, they may reuse words that their partner used the same way or at least similarly within their conversation. Alternatively, interlocutors may use the same words but for different things or may introduce new words that were not used before. Sameness (and conversely difference) of word usage, thus, is detected according to the extensional criterion of aboutness. What words are about is called their *topic* in the following. By speaking about similar word usages we refer to the similarity of the lexical contexts of words [11]. In the present scenario, this context is identified with the basic structural unit of dialogues, that is, the turn [12].

From this point of view, the generation of a dialogue lexicon is conceived as a process in which a lexical network grows turn by turn based on the word usages of the dialogue partners. In such a *dialogue lexicon network*, vertices denote lexical items while the strength of their edges denote the number of contexts in which these words co-occurred until the corresponding point in time. In this sense, the generation of a dialogue lexicon appears as a *time series* that emits lexical networks at its different time points. It is this combined notion of complex networks [13] and time series that is used to model the build-up of dialogue lexica as a result of dialogical communication. In this section, we briefly recapitulate this model in terms of so called *Two-Layer Time-Aligned Network* (TiTAN) series [5] and introduce its instantiation in the context of direction givings.

Generally speaking, a TiTAN series is a time series $\{L_t \mid t \in \quad \}$ of indexed graphs L_t that model the dialogue lexicon of a dyadic conversation at time t. Each of these graphs L_t is partitioned into two layers, A and B, representing each interlocutors' part of the dialogue lexicon. In order to instantiate the notion of a TiTAN series in the framework of direction givings, we start with formalizing dialogue lexica before we explain how TiTAN series are serialized.

Formally speaking, the dialogue lexicon of a dyadic conversation among two interlocutors A and B at time t is modeled as a labeled graph $L_t = (V, E_t, \mathcal{L})$. In this graph, the vertex set V is partitioned into non-empty disjunct subsets V_A and V_B whose elements denote the words used by interlocutor A and B, respectively, to perform the task of direction giving. The vertices in V are labeled by the surjective function $\mathcal{L} : V \rightarrow L_V$ where, in our case, the set of labels L_V consists of lemmata. Analogously, the edge set E_t is partitioned into three non-empty disjunct subsets E_{AB_t}, E_{A_t}, E_{B_t} where all edges $\{v, w\} \in E_{AB}$ end at vertices $v \in A, w \in B$, while all edges $\{x, y\} \in E_X$, $X \in \{A, B\}$, end at vertices $x, y \in V_X$. E_A and E_B capture intrapersonal lexical relations, while edges in E_{AB} are used to link lexical items shared among the interlocutors. The subgraphs $L_{A_t} = (V_A, E_{A_t}, \mathcal{L})$ and $L_{B_t} = (V_B, E_{B_t}, \mathcal{L})$ are called the A- and B-layer, respectively, of the two-layer graph $L_t = (V, E_t)$ at time t. They are denoted by the projection functions $\pi_A(L_t) = L_{A_t}$ and $\pi_B(L_t) = L_{B_t}$. In terms of our application area, layer A represents the dialogue lexicon of interlocutor A, layer B represents the dialogue lexicon of interlocutor B, while the graph L_t provides a unified model of their overall dyadic dialogue lexicon.

The networks defined so far model linguistic units and their relations. However, they do not distinguish between seldomly and frequently intantiated relations. This asymmetry is accounted for by assigning weights to the edges in L_t. Thus, dialogue lexica are modeled as weighted labeled graphs $L_t = (V, E_t, \mu_t, \mathcal{L})$ that are indexed by the point in time $t \in \quad$ at which they are spanned. Recall that t is derived from the dialogue turns of the interlocutors and, thus, from a dialog-inherent time-related ordering. In this sense, a TiTAN series is serialized according to the contributions of the interlocutors manifested and organized as turns. As a two-layer graph, L_t is divided into the subgraphs $\pi_A(L_t) = L_{A_t} = (V_A, E_{A_t}, \mu_{A_t}, \mathcal{L})$ and $\pi_B(L_t) = L_{B_t} = (V_B, E_{B_t}, \mu_{B_t}, \mathcal{L})$ according to the distribution of L_t over the agents A and B at time t. The spanning of edges within L_t is done as follows [5, p. 1453]:

- *Intrapersonal links:* if at time t, agent $X \in \{A, B\}$ uses a word form as an instance of lemma $l \in L_V$ to express the current turn's topic $T = T(t)$, intrapersonal links between vertex $v \in V_X, \mathbb{L}_X(v) = l$ are generated, and all other vertices w in layer L_{X_t} whose lemma $\mathcal{L}(w)$ was used by X in the same or any preceding turn to speak about the same topic T. If any of these edges $e = \{v, w\}$ already exists, its weight is incremented by 1, that is, $\mu_t(e) = \mu_{t-1}(e) + 1$. Initially, all edges have a weight of 1.
- *Interpersonal links:* if at time t, agent $X \in \{A, B\}$ uses a word form as an instance of lemma $l \in L_V$ to express the topic $T = T(t)$, which has been expressed by the dialogue partner $Y \neq X$ in any preceding turn on the same topic by means of the same lemma, an interpersonal link $\{v, w\} \in E_t$ between $v \in V_A$ and $w \in V_B$ is generated for which $\mathcal{L}(v) = \mathcal{L}(w) = l$, given that this link does not already exist. Otherwise, its weight is increased by 1.

With the passing of time, this process generates a series of dialogue lexica L_t that are indexed by the corresponding time point t. Figure **????** provides a schematic visualization of this construction process of a TiTAN series. The starting point is given by completely unlinked dialogue lexica L_{A_0} and L_{B_0} of the interlocutors A and B. Following the afore-given construction procedure, the lexica are networked turn-wise by adding intra- and interpersonal links. A TiTAN, thus, allows for modeling for each turn the degree of structural coupling of the dialogue lexica of both interlocutors. It finally results in a dialogue lexicon that manifests the degree of lexical alignment at the end of the conversation of both interlocutors. To see this, look at Figure **????**, which shows two extreme values of dialogue lexica: the lower bound is given by two layers L_{A_t} and L_{B_t} of the overall dialogue lexicon L_t that are completely disconnected and internally structured in completely different ways. Such a situation occurs if both agents always use different words or denote the same topics always differently. The upper bound is set by two isomorphic layer graphs that are fully linked. This scenario corresponds to a dialogue in which both agents always use the same words the same way. Due to thematic progression of natural dialogues, constraints by stylistics and verbal economy, and psychological factors of various kinds, neither of these extremal points is to be expected to be realized by dialogical conversation. They delineate, however, theoretical boundary values that make lexical alignment a measurable property [5].

In the framework of task-related conversations like direction giving, alignment is supposed to be bound up with communicative success, i.e. efficiency [1, p. 172].

The question arises how to measure whether a dialogical interchange is efficient or not. Using TiTAN series to represent such dialogues, we hypothesize that the class of effective directions can be separated from the class of ineffective directions in terms of the topology of the final state of the dialogue lexica L_t. In other words, we hypothesize that the way lexical items are connected and clustered in a dialogue lexicon informs about the status of the corresponding direction giving. If this is true, it should be possible to utilize complex network theory [13] to represent dialogue lexica by topological indices that are finally input to unsupervised learning of the class of effective and ineffective directions.

This is the way, we proceed in this paper. More specifically, we apply *Quantitative Network Analysis* (QNA) [14,15] to represent and classify dialogue lexica by means of complex network theory. In the present area of application, QNA involves three steps of modeling:

1. *Quantitative graph modeling*: initially, each dialogue lexicon is represented by a vector of topological indices that model its network structure.
2. *Feature selection:* in the next step, a genetic search is performed to find salient features within the vectors that best separate effective and ineffective dialogues. Note that this process of feature selection may stop at a local maximum as it does not necessarily find an optimal feature subset.
3. *Classification:* based on the appropriately projected feature vectors, a hierarchical agglomerative clustering is performed together with a subsequent partitioning that is informed about the number of target classes. We use *complete linkage* together with the *Mahalanobis distance* to perform this step. Note that we use MATLAB to make any of these computations. Note also that the Mahalanobis distance is used to handle correlations between features.

To sum up, QNA takes the set of input dialogues together with the parameter space of linkage methods and distance measures to find out the feature subset that best separates the data according to the underlying classification. In the present study, we utilize a subset of indices of complex network theory together with a subset of indices that were invented to model dialogue lexica [5]. See [5] and [15] for a summary of this quantitative graph model. All in all, 50 topological indices were computed per input dialogue to model its structural characteristics. Note that we exclude simple frequency oriented indices (e.g., the number of vertices or edges). In Section 4, we discuss eight instantiations of this model by experimenting with a set of 25 dialogues about directions.

4 Experimentation

4.1 Data

The speech data the TiTAN model is applied to are taken from the *Bielefeld Speech and Gesture Alignment Corpus* (SaGA) [3]. The primary data of the SaGA corpus are made up of 25 direction dialogues. After finishing a simulated bus ride through a virtual town, one participant explains the route taken and some sights passed to a second participant.

Video and audio recordings were made of the experiments, and on their basis, an orthographic transcription of speech on the level of words has been created. Typical phenomena of spontaneous speech (for example, clitics, elisions, assimilations, and spontaneous neologisms) were transcribed according to guidelines in order to ensure consistency.

These transcriptions were tagged with part-of-speech and lemma information by a system consisting of the eTagger of the eHumanities Desktop [16], a central trigram HMM tagger that has been trained on the German Negra Corpus.[3] The *Stuttgart-Tübingen Tag Set* (STTS, cf. [17]) was used, along with pre- and post-processing mechanisms that are specialized in the handling of the phenomena of spoken language mentioned above. Preprocessing methods map recurring word form variants to their standardized counterparts before tagging. Postprocessing mechanism apply several heuristics to unrecognized words that help to identify neologisms – for example those that had been constructed from two or more known words (e.g., "Peitschenlampe'" – *whip lamp*, constructed from the nouns "Peitsche" / *whip* and "Lampe" / *lamp*). Still, there were word forms that could not be detected or handled automatically. These tokens have been manually corrected after applying the tagger.

Since we are not concerned with well-formedness or related grammatical notions, but rather with regularites of word use, the syntactically fine-grained POS of the STTS are too detailed. Thus, we mapped the STTS onto the functionally basic types N(oun), V(erb), ADJ(ective), ADV(erb) JUNC(tors), PREP(ositions), DET(erminers), PRO(nouns), and PART(icles). A fourth type, called REST, collects the remainder of POS like interjections and fragments. These basic categories are used in the construction of dialogue lexica.

[3] `http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/`

Fig. 1. Overview of the virtual SaGA town, with topics marked

For the construction of a TiTAN series, information about turns and their respective topic are required. As a consequence, the turn boundaries needed for the construction of a TiTAN were annotated manually for all of the 25 dialogues. The topics we acknowledge are derived from the stages of the route through the virtual town of the primary data. The SaGA town and its 12 stages (topics) are shown in Figure 1. In addition, there is a 13th topic called *SaGA* which indicates turns that are about (large parts of) the whole virtual town. The label *META* is used to classify turns that do not relate to the route, but rather negotiate discourse issues or interpersonal concerns.

Each dialogue has been rated with respect to whether the interlocutors converge on a suitable description of the SaGA route. The criterion is whether the addressee has been put into the position to find the way from the sculpture to the fountain without going astray. We distinguish three cases or classes: 1. The direction is correct; 2. The direction is partially correct, but sufficient for the purpose to cross the SaGA town; 3. The direction is full of holes and useless. If, for instance, a participant mistakes the conifers in the park for leaf trees but apart from that gives a right direction, the dialogue is classified into the second class. Class 1 and 2 are grouped together into "correct" directions. In sum, there are 17 wrong and 8 correct directions. For each of these 25 dyads, a separate dialogue lexicon network has been built according to Section 3.

One might object that the occurrence of alignment is independent from the validity of the given direction. Note, however, that the classification of dialogues is not concerned with their correctness in the first place. In particular class 2 above accounts for directions that are false strictly speaking, but nonetheless carry enough information to let the addressee find the way. Finally, the class 3 dialogues are clearly faulty. *So what is the root of the matter?* [1, p. 172] emphasize that "alignment of situation models is central to successful dialogue". No matter whether the situation models are correct or not, a precondition is that the dialogue participants have situation models at all! There are reasons to assume that this is the problem with class 3 dialogues. The direction givers' models of the SaGA town are fragmentary – the models contain gaps. It is questionable whether fragmentary models can be conceived of as situation *models* at all. As it stands, we are aware of these theoretical obstacles, but regard our classification approach as feasible.

4.2 Evaluation

In this section, we describe the experimental scenarios by which we test our classification hypothesis introduced in Section 3. This hypothesis says that the efficiency of a direction giving in dialogical communication can be detected based on the topology of the final state of the corresponding dialogue lexicon. As described in Section 3, we test this hypothesis in the framework of *Quantitative*

Network Analysis (QNA). More specifically, we test 8 different variants of this hypothesis (as summarized in Table 1):

– *Variant* [*N*]*:* we start with considering nouns only. The idea behind this approach is that nouns are mainly used by interlocutors to refer to the reference universe of the direction giving – see the entities of the virtual SaGA town marked in Figure 1. We expect that the efficiency of directions is more easily identified by means of the nominal subnetworks of the corresponding dialogue networks.
– *Variant* [*N*|*A*], [*N*|*V*] *and* [*N*|*V*|*A*]*:* alternatively, we experiment with additionally considering adjectives and verbs. The reason to take these POS into account is that many of their instances have a descriptive meaning in relation to the reference universe of the direction giving (as, e.g., the verb *to turn* in *Turn to the left*).

In addition to these four variants, we consider those subnetworks that exclude words with a meta-communicative function (see the Rows 1–4 and the column *Meta*, which codes whether words are included that are tagged by *META* according to Section 4.1). These are words (as, e.g., *to think* in *Let me think*) that do not have a referential meaning regarding the reference universe of the direction giving, but serve, for example, to organize the dialogue. In our corpus of 25 direction givings, we have annotated 5,561 word forms with a meta-communicative function in relation to the overall set of 45,190 word forms that we were manually annotated. Thus, more than 10% of the word forms were used for meta-communicative reasons. From this perspective, one may expect an effect of excluding or including this class of words.

Table 1 summarizes the results of our findings. It shows that the best performing variant is based on selecting nouns without any meta-communicative function (see Row 1). This variant produces an F-score of more than 96%. The F-score (or F-measure) is the harmonic mean of recall and precision of the computed classification in relation to the correct classification of the data into 17 ineffective and 8 effective dialogues. An F-score of 96% means that nearly all dialogues have been classified correctly. If we additionally consider verbs, the F-score decreases to 92% (Row 2). The loss of classification is even higher if we separately consider the network of adjectives and that of verbs and adjectives (Row 3 and 4). Thus, although adjectives and verbs have denotational meanings

Table 1. Summary of the results of differently parameterized *Quantitative Network Analyses* (QNA). Row no. 9 shows the average F-score of these variants. The last column denotes the number of features output by the genetic search of the best performing subset of features as part of QNA (see Section 3).

No.	**Setting**	**Meta**	**Procedure**	**F-score**	**Features**
1.	$[N]$	yes	QNA	.96057	16 / 50
2.	$[N\|V]$	yes	QNA	.92	18 / 50
3.	$[N\|A]$	yes	QNA	.91651	20 / 50
4.	$[N\|V\|A]$	yes	QNA	.88171	21 / 50
5.	$[N]$	no	QNA	.92194	21 / 50
6.	$[N\|V]$	no	QNA	.87771	18 / 50
7.	$[N\|A]$	no	QNA	.88171	22 / 50
8.	$[N\|V\|A]$	no	QNA	.91651	24 / 50
9.	average over non-random approaches			.9096	20
10.	random baseline known-partition			.58668	
11.	random baseline equi-partition			.58583	

in the dialogues analyzed here, they do not help to separate the class of effective and ineffective direction givings to the same degree as nouns only.

These results seem to be contra-intuitive. Denotations of orientations and movements should be key ingredients of a successful direction giving. However, they are relational in character as they depend on the things they relate. Regarding situation models, a precondition for relational specification is that the objects in question are (correctly) spread out on the mental model. This in turn requires that the objects are available to the interlocutor. Objects are typically denoted by nouns or noun phrases. If the direction giver can name the things he wants to talk about, he can relate them to each other or to the direction follower. Thus, correct $[N|V|A]$-dialogues depend on correct $[N]$-dialogues. Besides this logical relationship, however, verbs, and adjectives may be the source for errors beyond nominal expressions. The decreasing F-score of $[N|V|A]$, $[N|V]$, and $[N|A]$ variants in comparison to the $[N]$ variant is very probably due to the asymmetrical status of the $[N]$ partitions of the dialogues in relation to their adjective- and verb-based partitions.

What happens if we additionally consider words with meta-communicative functions? As shown by the rows 5 through 8 in Table 1, there is a negative effect of including meta-communicative words. However, the differences being observed are rather marginal so that we conclude that there is only a small effect of either including or excluding this class of words. Meta-communicative acts typically provide information that the addressee has either understood the direction or that he could not follow. Thus, meta-communicative turns are used to convey a sort of binary information. As this information does not relate to the direction proper, it may be the reason for the lack of classificatory power being observed.

In order to further assess the quality of our results, we computed two random baselines (Row 10 and 11 in Table 1): the baseline called *known-partition* has information about the number of instances of the target categories. That is, by knowing that there are 17 ineffective and 8 effective dialogues, this baseline randomly generates two subsets of these cardinalities to compute the corresponding F-score. By repeating this procedure 1,000 times, we get an expected F-score of about 58%. This score is a little bit smaller if we consider the second random baseline that assumes equal sizes of the target categories (in our case 12 and 13). Obviously, all topology-related classifiers clearly outperform these two baselines. Thus, we can conclude, at least until any future falsification, that the efficiency of a direction giving is encoded into the *structure* of its dialogue lexicon.

5 Conclusion

In this paper, we applied *Two-Layer Time-Aligned Network* (TiTAN) series in the context of direction givings. Based on this graph model, we implemented several classifiers that solely explore the structure of dialogue lexica to assess their efficiency. By example of a corpus of 25 dialogues, we have shown that topological indices of dialogue lexica can indeed reveal this status. We also observed that lexical units with meta-communicative functions have a small effect on classification. This is in support of the observation that lexical manifestations of dialogue organization have a some impact on the efficiency of direction givings. Furthermore, we observed that the networking of nouns has the highest classificatory power, while the subnetworks of adjectives and verbs are less informative. One reason for this finding may be the outstanding referential meaning of nouns in conjunction with their semantic specificity. There are several POS that we did not consider here. Apart from adverbs, this relates to instances of closed POS. In future work will consider these classes and their role in the organization of dialogue lexica too.

Acknowledgements

This work has been supported by the German Research Foundation (DFG), which granted the Collaborative Research Center 673, and the German Ministry of Education and Research, which funds the *Linguistic Networks* project.

References

1. Pickering, M.J., Garrod, S.: Toward a mechanistic psychology of dialogue. Behavioral and Brain Sciences 27(2), 169–190 (2004)
2. Schiller, N.O., de Ruiter, J.P.: Some notes on priming, alignment, and self-monitoring. Behavioral and Brain Sciences 27, 208–209 (2004)
3. Lücking, A., Bergmann, K., Hahn, F., Kopp, S., Rieser, H.: The Bielefeld speech and gesture alignment corpus (SaGA). In: Multimodal Corpora: Advances in Capturing, Coding and Analyzing Multimodality, Malta, 7th International Conference for Language Resources and Evaluation (LREC 2010), pp. 92–98 (2010)

4. Reitter, D., Keller, F., Moore, J.D.: Computational modelling of structural priming in dialogue. In: Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, New York, NY, pp. 121–124 (2006)
5. Mehler, A., Lücking, A., Weiß, P.: A network model of interpersonal alignment in dialogue. Entropy 12(6), 1440–1483 (2010)
6. Mehler, A., Weiß, P., Menke, P., Lücking, A.: Towards a simulation model of dialogical alignment. In: Smith, A.D.M., Schoustra, M., de Boer, B., Smith, K. (eds.) The Evolution of Language. Proceedings of the 8th International Conference (EVOLANG8), pp. 238–245. World Scientific, Singapore (2010)
7. Church, K.W.: Empirical estimates of adaptation: The chance of two noriegas is closer to $p/2$ than p^2. In: Proceedings of Coling 2000, Saarbrücken, Germany, pp. 180–186 (2000)
8. Wheatley, B., Doddington, G., Hemphill, C., Godfrey, J., Holliman, E., McDaniel, J., Fisher, D.: Robust automatic time alignment of orthographic transcriptions with unconstrained speech. In: Proc. ICASSP 1992, vol. 1, pp. 533–536 (1992)
9. Anderson, A.H., Bader, M., Gurman Bard, E., Boyle, E., Doherty, G., Garrod, S.: The HCRC map task corpus. Lang. Speech 34, 351–366 (1991)
10. Reitter, D., Moore, J.K.: Predicting success in dialogue. In: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL), Praque, Czech Republic, pp. 808–815 (2007)
11. Miller, G.A., Charles, W.G.: Contextual correlates of semantic similarity. Language and Cognitive Processes 6(1), 1–28 (1991)
12. Schegloff, E.A.: Sequence Oeganization in Interaction. Cambridge University Press, Cambridge (2007)
13. Newman, M.E.J.: The structure and function of complex networks. SIAM Review 45, 167–256 (2003)
14. Mehler, A.: Structural similarities of complex networks: A computational model by example of wiki graphs. Appears in: Applied Artificial Intelligence (2008)
15. Mehler, A., Pustylnikov, O., Diewald, N.: Geography of social ontologies: Testing a variant of the Sapir-Whorf Hypothesis in the context of Wikipedia. Computer Speech and Language (2010)
16. Gleim, R., Waltinger, U., Ernst, A., Mehler, A., Esch, D., Feith, T.: The ehumanities desktop - an online system for corpus management and analysis in support of computing in the humanities. In: Proceedings of the Demonstrations Session of the 12th Conference of the European Chapter of the Association for Computational Linguistics EACL 2009, Athens (March 2009)
17. Schiller, A., Teufel, S., Thielen, C.: Guidelines für das Tagging deutscher Textcorpora mit STTS. Technical Report, IMS-CL, Universität Stuttgart (1995)

Construction of Wakamono Kotoba Emotion Dictionary and Its Application

Kazuyuki Matsumoto and Fuji Ren

Faculty of Engineering, University of Tokushima
2-1 Minami-josanjima, Tokushima, 770-8506, Japan
{matumoto,ren}@is.tokushima-u.ac.jp

Abstract. Currently, we can find a lot of weblogs written by young people. In the weblogs, they tend to describe their undiluted emotions or opinions. To analyze the emotions of young people and what causes those emotions, our study focuses on the specific Japanese language used among young people, which is called *Wakamono Kotoba*. The proposed method classifies *Wakamono Kotoba* into emotion categories based on superficial information and the descriptive texts of the words. Specifically, the method uses literal information used for *Wakamono Kotoba*, such as Katakana, Hiragana, and Kanji, etc., stroke count, and the difficulty level of the Kanji as features. Then we classified *Wakamono Kotoba* into emotion categories according to the superficial similarity between the word and the *Wakamono Kotoba* registered in the dictionary with an annotation of its emotional strength level. We also proposed another method to classify *Wakamono Kotoba* into emotion categories by using the co-occurrence relation between the words included in the descriptive text of the *Wakamono Kotoba* and the emotion words included in the existing emotion word dictionary. We constructed the *Wakamono Kotoba* emotion dictionary based on these two methods. Finally, the applications of the *Wakamono Kotoba* emotion dictionary are discussed.

Keywords: Wakamono kotoba, emotion dictionary, emotion corpus.

1 Introduction

Currently, a variety of languages are used to convey text information on the Internet. Internet terminology is an example and it is usually used only on the Internet. Another example is *Wakamono Kotoba*, a Japanese language used by the younger generation ranging from teenagers to those in their late 20's. They are also frequently used on the Internet. Without processing *Wakamono Kotoba* correctly, it would be difficult to extract the living opinions of younger people from the information on the Web. To precisely recognize *Wakamono Kotoba* in a sentence, it is necessary to register these words in the dictionary. However, the process is not efficient.

Therefore, technology to automatically recognize unknown *Wakamono Kotoba* words from a sentence and to analyze such words semantically would be useful.

Unknown word processing is an important research topics in the field of natural language processing. Unknown words have been analyzed based on the superficial features such as the type of character (Katakana, Hiragana, Kanji, etc.) or the word examples [1–4]. However, even though an unknown word can be recognized and classified into rough categories, it is still difficult to classify those words semantically.

In this paper, we propose a method to estimate the emotions expressed with *Wakamono Kotoba* and consider the possibility of using the method to estimate the writer's emotion from sentences that include unknown words that are not registered in the analytical dictionaries, such as *Wakamono Kotoba* words.

1.1 Existing Research

There is already a lot of research dealing with *Wakamono Kotoba*, especially in the field of linguistics, [5] but not in the field of engineering. Harada et al. [6] studied the technology to generate *Wakamono Kotoba* words from existing words. Their method can create new *Wakamono Kotoba* words by abbreviating existing words according to a rule. However, *Wakamono Kotoba* is much affected by the time period background and has various types; therefore, a simple rule cannot cover the various kinds of *Wakamono Kotoba.*

Matsumoto et al. [7] collected sentences that included *Wakamono Kotoba* and constructed an emotion corpus. They conducted an experiment to estimate emotion from the sentences that included *Wakamono Kotoba* based on the emotion corpus. Emotion can be estimated from the sentences that include *Wakamono Kotoba* words that are in the corpus; however, there is a problem in that emotion cannot be estimated from the sentences that include new *Wakamono Kotoba* words that are not in the corpus. While *Wakamono Kotoba* words are being created every day, many of them are disappearing. Therefore it is inefficient to register all *Wakamono Kotoba* in a dictionary.

In this paper, we conducted a questionnaire to identify the emotions of the active *Wakamono Kotoba* and constructed a basic *Wakamono Kotoba* dictionary based on the questionnaire results. With this dictionary, we proposed a method to estimate the emotions of the *Wakamono Kotoba* words not registered in the dictionary.

1.2 The Definition of Wakamono Kotoba

In this paper, we define *Wakamono Kotoba* as follows:

(A) Words introduced in commercially available books or Web pages as *Wakamono Kotoba.*
(B) Words often used on the Web.

We identified words satisfying both the (A) and (B) conditions as *Wakamono Kotoba.*

To fulfill condition (A) words shown in dictionaries of slang words or new words were chosen as *Wakamono Kotoba* candidates. Then, to fulfill condition

(B), these candidate words were searched using Yahoo! Blog search[1] and sorted in order of the search results (hit frequency), and, finally, the words ranked over the median were defined as *Wakamono Kotoba.* The median me of the search hit frequency is calculated by Equation 1. m indicates the rank in the hit results, n shows the total number of words, and X_m and X_{m+1} describe the hit numbers of the words ranked m-th and $(m+1)$-th. Only the words ranked over the me were used in this study.

$$me = \begin{cases} X_m & \text{if } n \text{ is odd } : m = \frac{n+1}{2} \\ \frac{X_m + X_{m+1}}{2} & \text{if } n \text{ is even} : m = \frac{n}{2} \end{cases} \tag{1}$$

2 Wakamono Kotoba Emotion Dictionary

The *Wakamono Kotoba* emotion dictionary that we created includes annotations of emotional strength for each *Wakamono Kotoba* with four-dimensional vector. Table 1 shows a part of the dictionary. For example, the *Wakamono Kotoba* word "*Nenige*" expresses the emotion of the vector [0,-3,-1,0]. This means that the word conveys the emotion of Hate with 3 degree and Sadness with 1 degree.

In this study, we used eight basic kinds of emotions identified by Plutchik [8], such as "Anticipation," "Surprise," "Trust," "Hate," "Joy," "Sadness," "Fear" and "Anger" and defined four kinds of opposing emotional pairs with a four-dimensional vector. This dictionary can be used to estimate the emotion of a sentence that includes *Wakamono Kotoba* and as training data to estimate the emotion of an unknown or new *Wakamono Kotoba* word.

The dictionary was constructed in the following steps:

1) Selection of *Wakamono Kotoba*
   Words satisfying both conditions (A) and (B) were registered in the list as *Wakamono Kotoba.*
2) Judgment of Part of Speech The part speech of each *Wakamono Kotoba* was manually judged.
3) Questionnaire to estimate the emotion of each *Wakamono Kotoba* word
   A questionnaire was used to ask if the examinees know the *Wakamono Kotoba* word and what emotion each *Wakamono Kotoba* expresses.
4) Registration of *Wakamono Kotoba* in the dictionary
   Based on the result of the questionnaire, the emotion vector is annotated for each *Wakamono Kotoba*, and the words are registered in the *Wakamono Kotoba* emotion dictionary.

We used the questionnaire to ask two examinees about the emotions of *Wakamono Kotoba* words.

[1] `http://blog-search.yahoo.co.jp/`

Table 1. Example of *Wakamono Kotoba* emotion dictionary

Word	Anticipation ↕ Surprise	Trust ↕ Hate	Joy ↕ Sadness	Fear ↕ Anger
nenige	0	-3	-1	0
rasukaru	0	-1	0	0
chapatsu	0	-2	0	0

3 Proposed Method

3.1 Emotion Estimation Method Based on Superficial Feature of Wakamono Kotoba

One of the features of the *Wakamono Kotoba* used by young people is the humor of the impressions of the words; [9–11] such words are often spoofs of known words. For this reason, we thought that, if the superficial features were similar, the emotional impressions would also be similar between the words.

Examples of *Wakamono Kotoba*'s superficial features and their emotional impressions are described below.

Ex.1 Recently, I am *Yaseborikku* (getting slim).
Ex.2 Recently, I am *Honeborikku* (getting bony).

"*Yaseborikku*" in Ex. 1 means people who are thin, and "*Honeborikku*" in Ex. 2 means big-boned or bony. These two words are similar in usage; however, their meanings are different. Both "*Yaseborikku*" and "*Honeborikku*" are *Wakamono Kotoba* words, derived from "*Metaborikku*," which is the abbreviation for metabolic syndrome, and both words share negative impressions (getting bony).

That is, words derived from words with negative impressions tend to take on the negative impressions of the original words. In this study, we focused on the superficial similarity based on the hypothesis that words with similar superficial features would share similar emotions. We supposed that the literal features would affect the impression of physical appearance and that the pronunciation (reading) feature would affect the auditory impression. Therefore, we also decided to use pronunciation (reading) information as a feature. The superficial information used for estimation is as follows:

f_1: Number of characters
f_2: Sum of the stroke counts of the characters
f_3: Sum of the difficulty level of the Kanji characters
f_4: Number of Hiragana
f_5: Number of Katakana
f_6: Number of letters
f_7: Number of numeric characters
f_8: Number of Kanji characters
f_9: Sum of the stroke counts of readings

The similarity degree between each superficial feature is calculated by the cosine similarity (shown in Equation 2), which is often used for vector similarity calculations.

V_1 and V_2 are the feature vectors consisting of the features f_1 to f_9 that are extracted from the *Wakamono Kotoba.*

$$Sim\_cos(f_1, \cdots, f_9) = \frac{V_1 \cdot V_2}{|V_1| \times |V_2|} \tag{2}$$

Some of the common approaches to calculate superficial similarity are character N-gram or edit distance. We decided to use two kinds of cosine similarities: 1) character bi-gram(Sim_bi_gram) and 2) pronunciation (reading) bi-gram ($Sim\_read\_bi\_gram$). We also used the Jaro-Winkler Distance [12](Sim_jwd) of the character type. We used the Jaro-Winkler Distance because the method can reflect more detailed difference than the Levenstein Distance with respect to similarity.

The final similarity is calculated by Equation 3 as the sum of these similarities. When the emotion vector annotated for the *Wakamono Kotoba* maximizes the value of Equation 3, the vector is identified as the emotion vector of the inputted *Wakamono Kotoba* (see Fig. 1).

$$\begin{aligned} Sim = Sim\_cos(f_1, \cdots, f_9) + Sim\_bi\_gram \\ + Sim\_read\_bi\_gram + Sim\_jwd \end{aligned} \tag{3}$$

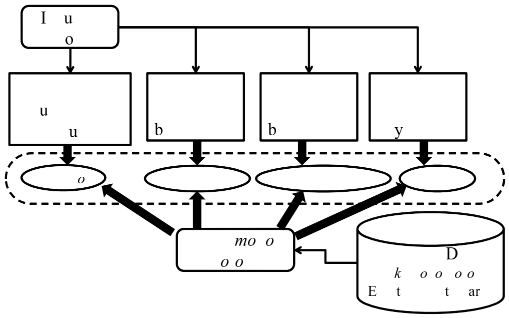


Fig. 1. Flow of calculating similarity of *Wakamono Kotoba*

3.2 Emotion Estimation Based on Semantic Information

From the superficial features of *Wakamono Kotoba*, we might judge that the emotions are similar only if the characters used or the number of characters in the word are similar. However, even though their characters or pronunciation are the same, many words express different meanings, causing ambiguity in the word senses. Therefore, we proposed a method to calculate an emotion vector using the relation of the content words included in the descriptions for the *Wakamono Kotoba* and the basic emotion word. Following is the description for the *Wakamono Kotoba* word "*Muchaburi*."

"*Muchaburi*" (2007) ⋯
A type of lead-in to a joke with a planned punch line, where the person who composes the sketch gradually builds up the joke differently from the prepared one and <u>forces</u> the final punch line to the other person <u>in a screwed-up condition</u> ⋯. (Japanese Slang Dictionary[2])

In this example, the descriptions of the *Wakamono Kotoba* word includes words explaining their meanings or situations of use.

Therefore, we considered the relation between the words included in the description and the emotion words to estimate the emotion of the *Wakamono Kotoba*. The MI-score is a co-occurrence scale calculated with the word frequency in the corpus as the denominator. MI-score which is co-occurrence scale is calculated with frequency of word in whole corpus as denominator. Because the interpretations for Wakamono Kotoba are usually not so long, the co-occurrence rate between the words used in the interpreations and the basic emotion words is expected to be low. As the result, the value of the MI-score would become low and it would be difficult to recognize the distinctive difference among each MI-score. Therefore, we thought that the MI-score method was not effective for our experiment. If we use this scale, because that if the co-occurrence between the word include in interpretation and basic emotion word, the MI-score value become extremely low, the method is not effective.

We used a contextual similar word database [13] as an index to judge the similarity of the contexts where the words were used. This database listed a maximum of 500 words in descending order of the contextually similarity used on the Web to one million words. Table 2 shows an example of the basic emotion words. These words were extracted from a Japanese-English parallel corpus[14, 15], from which a maximum of 10 kinds of typical words were selected for each emotion category.

The flow of emotion estimation using the relation degree of the words in the description of the *Wakamono Kotoba* word and the basic emotion words is as follows:

Step 1. Each description of a *Wakamono Kotoba* word is divided into morphemes and classified as content words or the function words.
Step 2. According to the results of Step 1, a content word vector is created for each *Wakamono Kotoba* word.
Step 3. The set of contextually similar words included in each word in the basic emotion word list is extracted from the contextually similar word database. Then the top 100 or 200 words in each set of the contextually similar words are set as the relation word vector $CW(EW_j)$.

[2] `http://zokugo-dict.com/`

Table 2. Example of basic emotion words

Emotion Category	Words
Anticipation	*tanoshimi, kitai, kibou*, etc.
Surprise	*azen, bikkuri, kyotan*, etc.
Trust	*shinyu, doui , koui, okiniiri, suki, taisetsu*, etc.
Hate	*keishi, urusai, kirai, hinann*, etc.
Joy	*kofuku, arigato, tanoshii, shiawase, egao*, etc.
Sadness	*shitsuren, rakutan, namida, hikan*, etc.
Fear	*sakuran, touwaku, shinpai, osoroshii*, etc.
Anger	*kennka , gekido, hysteria, fungai, rippuku* , etc.

Step 4. By extracting the top 100 or 200 words from the set of the contextually similar words, the relation word vector $CW(KW_j)$ is created for the content word vector of each *Wakamono Kotoba* word.

Step 5. The relation degree score (e_j) is calculated based on the number of corresponding words in between the relation word vectors of the content words and the basic emotion words (Equation 4).

$$e_j = \frac{|CW(KW_i) \cap CW(EW_j)| \times 2}{|CW(KW_i)| + |CW(EW_j)|} \tag{4}$$

Step 6. An emotion vector is created based on the relation degree score.

Equation 5 shows how to convert to the four-dimension emotion vector. The function of $sgnMax$ returns a signed value of the larger absolute value in either e_n or e_m . The sign for Anticipation, Trust, Joy, Fear is set as plus (+), and the sign for Surprise, Hate, Sadness, or Anger is set as minus (-).

$$E_{n,m} = \frac{sgnMax(|e_n|, |e_m|)}{|e_n| + |e_m|} \times 3 \tag{5}$$

4 Experiment for Emotion Determination of Wakamono Kotoba

In this section, the evaluation experiments were conducted for two kinds of emotion estimation for *Wakamono Kotoba*: 1) using superficial features and 2) using semantic information.

The targets for evaluation were 756 *Wakamono Kotoba* words with annotated emotion vectors identified using the questionnaire. The evaluation was conducted by calculating how much the output emotion vector corresponds with the emotion vector as the correct answer. Match score of EV and EV' is calculated with Equation 6.

$$MatchScore(EV, EV') = \frac{\sum_{i=1}^{4} Weight(e_i, e'_i)}{4} \tag{6}$$

The function of weight returns the weight according to each matching pattern. Table 3 shows the matching patterns and the corresponding weights.

Table 3. Matching patterns and correspondent weights

Matching Pattern		$Weight(e_i, e'_i)$
The sign of e_i is same as the sign of e'_i	$e_i = e'_i = 0$	0.5
	$e_i = e'_i \neq 0$	1.0
	$e_i \neq e'_i$	$3/(\lvert e_i - e'_i \rvert + 1)$
The sign of e_i is different from the sign of e'_i		0

4.1 Preliminary Experiment: Evaluation of Polarity Estimation

As a preliminary experiment, we evaluated the performance of the estimation of emotion polarity: positive, negative, and neutral. Positive emotions are Anticipation, Joy, and Trust. Negative emotions are Sadness, Hate, Fear, and Anger. A neutral emotion is Surprise.

In the method based on superficial features, we sorted *Wakamono Kotoba* in order of the superficial similarity, focused on the top k-th and evaluated the most frequent polarity as the estimation result (k-nearest neighbor method). Leave-one-out Cross Validation was used as the experiment method. Fig. 2 shows the estimation result when k was changed from 1 to 50. Table 4 shows the estimation accuracy of the emotion polarity when the semantic information was used.

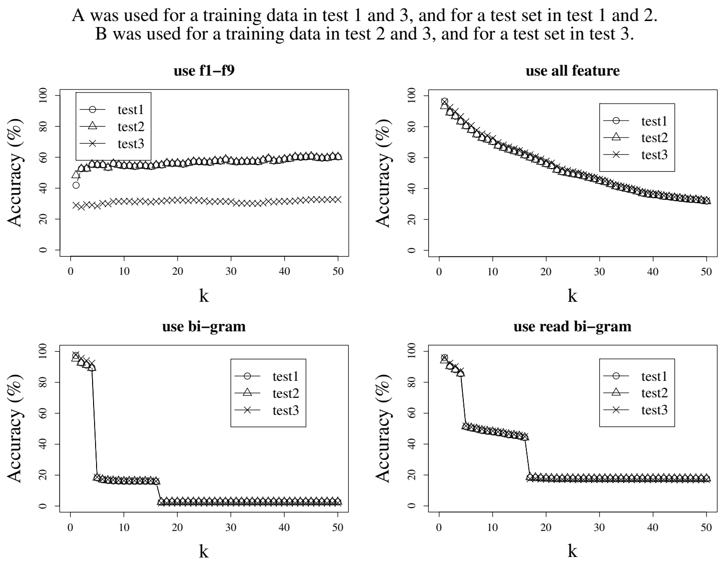


Fig. 2. Result of polarity estimation (use superficial features)

Table 4. Result of polarity estimation (use semantic information)

	top100	top200
Accuracy (%)	35.88	37.07

4.2 Experiment 1: Superficial Features

We set the correct threshold value of the *MatchScore* for each condition and conducted experiments based on the superficial features of *Wakamono Kotoba*. Leave-one-out Cross Validation was used as the experiment method. Table 5 shows the accuracy in each experiment condition when the correct threshold was set as 0.5.

Table 5. Accuracy of emotion estimation (*MatchScore* Threshold 0.5)

Test no.	Test data / Train data	Feature	Accuracy(%)
test 1	A / A	Character bi-gram	34.4
		Read bi-gram	33.2
		$f_1 - f_9$	38.2
		All features	**49.5**
test 2	A / B	Character bi-gram	32.6
		Read bi-gram	31.4
		$f_1 - f_9$	36.1
		All features	**46.7**
test 3	B / A	Character bi-gram	31.5
		Read bi-gram	30.2
		$f_1 - f_9$	34.6
		All features	**44.2**

The experimental results by each feature are shown in Fig. 3. These figures showed that the best result was obtained in test 1. The result might indicate that the training data and the test data were created by the same examinee.

We obtained a better result in test 2 (using A as the test set) than in test 3 (using B as the test set). Because the number of *Wakamono Kotoba* annotated emotion vectors was larger in test data A than in the test data B, the emotion vector given by the examinees varied greatly. As result, test data A might have become unfit as training data.

The experimental results showed that higher accuracy was obtained when superficial features other than character bi-gram or reading bi-gram were used. In fact, features such as the character type and its stroke count, difficulty level, and reading information played important roles in estimating the emotion of *Wakamono Kotoba*.

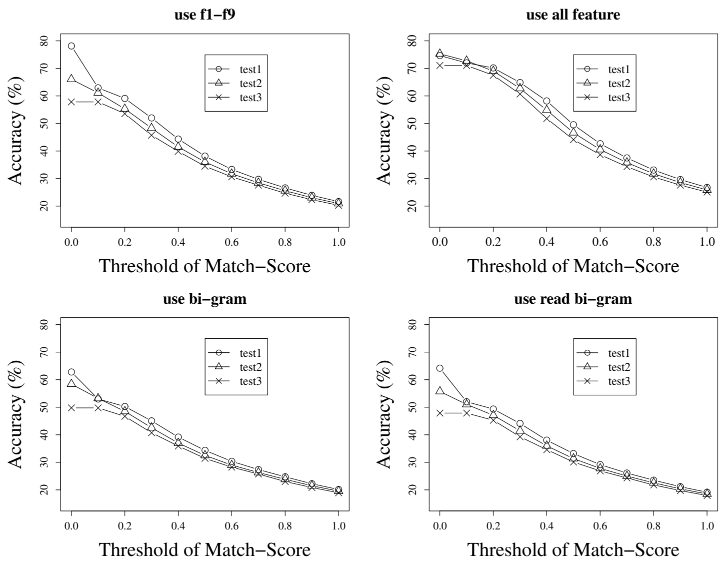


Fig. 3. Result of emotion estimation (use superficial features)

4.3 Experiment 2: Semantic Information

Fig. 4 shows the results of the experiment using description vectors. The accuracy rates using the top 100 contextually similar words and using the top 200 contextually similar words did not show a distinctive difference.

However, the result from using the top 100 words was higher than the result from using the top 200 words when the threshold value of $MatchScore$ was low. In total, the accuracy of the method using the superficial features was higher than another method using the semantic information.

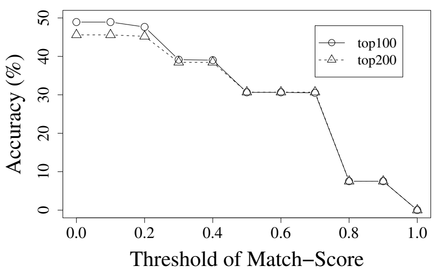


Fig. 4. Result of emotion estimation (use semantic information)

5 Discussion

From the experimental results, we found that it was difficult to use semantic information effectively because many *Wakamono Kotoba* words had complex meanings, so the description often included noise words.

On the other hand, using the superficial features, our method estimated the emotion of *Wakamono Kotoba* words with a certain level of accuracy. However, when the correct threshold of $MatchScore$ was 0.6 to 0.7, the accuracy was higher in the method using semantic features.

If the training data of *Wakamono Kotoba* is classified in advance according to the semantic features, the accuracy might be higher than in the method using only superficial features.

6 Application for Emotion Estimation from Sentences

Our *Wakamono Kotoba* emotion estimation method can be applied to the emotion estimation of a sentence that includes *Wakamono Kotoba*.

First, *Wakamono Kotoba* is extracted from the inputted sentence. After the emotion of the extracted *Wakamono Kotoba* is estimated, the emotion is estimated from the other parts of the sentence excluding the *Wakamono Kotoba*, using the existing method (word-based or grammar-based). By integrating the two results, it will be possible to estimate the sentence emotion with higher accuracy than with the existing method.

7 Conclusion

In this paper, we proposed two types of emotion estimation methods for *Wakamono Kotoba* using superficial features and semantic information and then conducting the evaluation experiments. According to the experimental results, the method using superficial feature was more effective for emotion estimation than the method using semantic information. The results also suggested that our method would be applicable for emotion estimation from the sentences that include unknown *Wakamono Kotoba* because our method was effective to estimate emotion with a certain accuracy without considering the sense of the *Wakamono Kotoba*.

On the other hand, to make the process applicable to unknown *Wakamono Kotoba*, it is also necessary to detect *Wakamono Kotoba* in the sentence automatically. Because it is difficult to judge whether a word is *Wakamono Kotoba* or another unknown word without background information, in the future, we would like to propose a method to judge whether a word is *Wakamono Kotoba* or not automatically using the appearance log or user profile on weblogs.

Acknowledgement

This research has been partially supported by Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Challenging Exploratory Research, 21650030.

References

1. Nagata, M.: A Part of Speech Estimation Method for Japanese Unknown Words using a Statistical Model of Morphology and Context. In: Proceedings of ACL 1999, pp. 277–284 (1999)
2. Sasada, T., Mori, S., Kawahara, T.: Kana-Kanji Conversion by Using Unknown Word-Pronunciation Pairs with Contexts. Journal of Natural Language Processing 13(2), 131–153 (2010); The Association for Natural Language Processing
3. Asahara, M., Matsumoto, Y.: Japanese Unknown Word Identification by Character-based Chunking. In: Proceedings of the 20th International Conference on Computational Linguistics, pp. 459–465 (2004)
4. Nakagawa, T., Matsumoto, Y.: Guessing Parts-of-Speech of Unknown Words Using Global Information. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics(COLING/ACL 2006), pp. 705–712 (2006)
5. Kuwamoto, Y.: Production and Stability of Wakamono Kotoba. The Bulletin of Akita National College of Technology 38, 113–120 (2003)
6. Harada, T., Kameda, H.: Younger's Word and their Processing Method. IEICE Technical Report 101(712), 17–24 (2002)
7. Matsumoto, K., Konishi, Y., Sayama, M., Ren, F.: Wakamono Kotoba Emotion Corpus and Its Application for Emotion Estimation. In: Proceedings of 2010 International Conference on Advanced Intelligence. Progress of Advanced Intelligence, vol. 2, pp. 89–99 (2010)
8. Plutchik, R.: The Emotion: Facts, Theories, and a New Model. Random House, New York (1962)
9. Yamaguchi, N.: Wakamono Kotoba ni Mimi wo Sumaseba. Kodansha publishers (2007)
10. Kitahara, Y.: Afureru Shingo. Taishukan Publisher (2009)
11. Yonekawa, A.: Wakamonogo wo Kagakusuru. Meiji Shoin (1998)
12. Winkler, W.E.: Overview of Record Linkage and Current Research Directions. Research Report Series (2006)
13. Kazama, J., Torisawa, K.: Inducing Gazetteers for Named Entity Recognition by Large-scale Clustering of Dependency Relations. In: Proceedings of ACL 2008: HLT, pp. 407–415 (2008)
14. Minato, J., Matsumoto, K., Ren, F., Kuroiwa, S.: Corpus-based Analysis of Japanese-English Emotional Expressions. In: Proceedings of IEEE NLP-KE 2007, pp. 413–418 (2007)
15. Matsumoto, K., Ren, F.: Estimation of Word Emotions Based on Part of Speech and Positional Information. Computers in Human Behavior (2010), doi:10.1016/j.chb.2010.10.028

Temporal Analysis of Sentiment Events – A Visual Realization and Tracking

Dipankar Das[1], Anup Kumar Kolya[1], Asif Ekbal[2], and Sivaji Bandyopadhyay[1]

[1] Department of Computer Science and Engineering, Jadavpur University, Kolkata 700 032, India
[2] Department of Information Engineering and Computer Science, University of Trento, Italy
{dipankar.dipnil2005,anup.kolya,asif.ekbal}@gmail.com, sivaji_cse_ju@yahoo.com

Abstract. In recent years, extraction of temporal relations for events that express sentiments has drawn great attention of the Natural Language Processing (NLP) research communities. In this work, we propose a method that involves the association and contribution of sentiments in determining the event-event relations from texts. Firstly, we employ a machine learning approach based on Conditional Random Field (CRF) for solving the problem of Task C (identification of event-event relations) of TempEval-2007 within TimeML framework by considering *sentiment* as a feature of an event. Incorporating sentiment property, our system achieves the performance that is better than all the participated state-of-the-art systems of TempEval 2007. Evaluation results on the Task C test set yield the F-score values of 57.2% under the strict evaluation scheme and 58.6% under the relaxed evaluation scheme. The positive or negative coarse grained sentiments as well as the Ekman's six basic universal emotions (or, fine grained sentiments) are assigned to the events. Thereafter, we analyze the temporal relations between events in order to track the sentiment events. Representation of the temporal relations in a graph format shows the shallow visual realization path for tracking the sentiments over events. Manual evaluation of temporal relations of sentiment events identified in 20 documents sounds satisfactory from the purview of event-sentiment tracking.

Keywords: Temporal Relations, CRF, TempEval-2007, TimeML, Sentiment Event, Visual Tracking.

1 Introduction

The kinds of states which change and thus might need to be located in time are referred as events in the present context. The event entities are represented by finite clauses, nonfinite clauses, nominalizations, event-referring nouns, adjectives and even some kinds of adverbial clauses. In general, the events are described in different newspaper texts, stories and other important documents where occurrence time of events, temporal location and ordering of the events are specified. Several earlier methods have been proposed for detecting and tracking events from text archives [1].

On the other hand, text does not only contain informative contents, but also some attitudinal private information, including emotional states. Human emotion described in texts is an important cue for our daily communication too. But, the identification of emotional states from texts is not an easy task as emotion is not open to any objective observation or verification [2]. Nowadays, in the Natural Language Processing (NLP) communities, several research activities on sentiment and/or emotion analysis are in full swing. Sentiment of people is important as it has great influence on our society. Our main motivation to investigate the insides of event-sentiment relation lies with the facts that though events and sentiments are closely coupled with each other from social, psychological and commercial perspectives, there has been very little attention regarding their detection The identification of the temporal relations between two events by taking the sentiment feature into account is also crucial to analyze and track human sentiments. This is also important in a wide range of other NLP applications that include temporal question answering, document summarization, current information retrieval systems etc.

Mishne and de Rijke [3] proposed a system, *MoodViews*[1] to analyze the temporal change of sentiment. MoodViews analyzes multiple sentiments by using 132 sentiments used in LiveJournal[2]. Although our concept for the sentiment graph is similar to MoodViews, we focus on temporal relations between events associated with similar or different types of sentiments. With respect to information visualization, Havre et al. proposed a system called *ThemeRiver* [4] that visualizes thematic flows along with timeline. Although our approach is different from ThemeRiver, we focus on visualization of sentiment flows on events based on temporal expressions. The temporal sentiment identification from social events has been carried out in [5]. In their task, the authors have analyzed the temporal trends of sentiments and topics from a text archive that has timestamps in Weblog and news articles and produces two kinds of graphs, *topic graph* that shows temporal change of topics associated with a sentiment and *sentiment graph* that shows temporal change of sentiments associated with a topic. In contrast, our present task incorporates the knowledge of temporal relations (e.g. AFTER, BEFORE, OVERLAP) instead of timestamps for temporal sentiment tracking. In addition to that, we also analyze the role of sentiment in identifying temporal relations between the events.

Let us consider the following example:

"The prime minister of India told Friday that he has ***talked*** with top commander of Indian military force and ***sent*** a team to *recover the host of Taj Hotel* ***hijacked***."

For example, in the above sentence, the native speakers can quickly identify the ordering of the three events, namely '***hijacking***', '***talking***' and '***sending***' as: ***hijacking→ talking→sending*** even though the temporal relations such as '***before***', '***after***' or '***overlap***' never appeared in the text. But, the above example also shows the presence of underlying sentiments (as shown in underlined script) scribed in the sentence. The TempEval-2007 challenge addressed the question of identifying temporal relations by using a common corpus on which research systems competed to find temporal relations [6]. Our present aim is not only to identify the temporal

[1] http://moodviews.com/

[2] http://www.livejournal.com/

relations from the events but also to identify the sentiments associated with the events, to determine the contribution of sentiment in identifying temporal relations as well as to track the sentiments over the events based on temporal relations.

The present system identifies temporal relations between the events by considering the contribution of the sentiment property into account. We propose a machine learning approach based on Conditional Random Field (CRF) [7] for solving the problem of Task C (identification of event-event relations) of TempEval-2007 within TimeML framework. The task is to identify the temporal relations between the events that occur in two consecutive sentences and to classify the event pairs into their respective temporal classes. Incorporating sentiment property into the set of other standard features, the proposed system outperforms all the participated state-of-the-art systems of TempEval 2007 with the F-score values of 56.87% under the *strict* evaluation scheme and 59.20% under the *relaxed* evaluation scheme. The positive or negative coarse-grained sentiments as well as the Ekman's [8] six basic universal emotions or fine-grained sentiments (*happiness, sadness, anger, fear, surprise* and *disgust*) are assigned to the events. Based on the temporal relations, the events from each of the documents are represented using a graph that shows the shallow path for identifying the sentiment changes over events. Manual evaluation of temporal relations of sentiment events identified in 20 documents sounds satisfactory from the purview of event-sentiment tracking.

The rest of the paper is organized as follows. Section 2 describes the sentiment based temporal relation identification using CRF. The evaluation schemes and results are discussed in Section 3. The tagging of the events with sentiments, generation and tracking of the event-sentiment relational graph and the evaluation of the event-sentiment tracking system along with the associated results are discussed in Section 4. Finally Section 5 concludes the paper.

2 CRF Based System for Identifying Temporal Relations

The Task C at TempEval-2007 was involved with the automatic identification of temporal relations holding between verb events in adjacent sentences. There are two types of events such as, *main-event* and *sub-ordinate event*. The *main-event* is determined from multiple sentences by following very shallow, syntactic-based criteria within the scope of a sentence. It has been observed that syntactically *subordinate events* are dominated by *main-event* in coordination relations and the *main-event* is the first one in the sentence string. The events expressions were annotated in the source in accordance with the TimeML standard [9]. For all the tasks, data were provided for training and testing that includes annotations identifying: (1) sentence boundaries, (2) all temporal referring expression as specified by TIMEX3, (3) all events as specified in TimeML and (4) selected instances of temporal relations, as relevant to the given task. For task C, a restricted set of event terms, whose stems occurred twenty times or more in the TimeBank corpus, was identified. This set is referred to as the Event Target List or ETL. Furthermore, only the event expressions that occur within the ETL are considered. In the training and test data, TLINK annotations for these temporal relations are provided. The only difference being that in the test data the relation type is withheld.

2.1 Pair-Wise Classification Using CRF

In the present approach, the identifications and classifications of temporal relations are based on a supervised machine learning algorithm, Conditional Random Field (CRF) that can include arbitrary set of features and still can avoid over fitting in a principled manner. We consider the temporal relation identification task as a pair-wise classification problem in which the EVENT target pairs are modelled. In the present task, we report only Task C that identifies temporal relations between the events that appear in two adjacent sentences. The events and temporal expressions (TE) were annotated in the source in accordance with the TimeML standard [9]. The set of temporal relations to be predicted includes: OVERLAP, BEFORE, AFTER, BEFORE-OR-OVERLAP, OVERLAP-OR-AFTER and VAGUE.

The main advantage of CRF is the assumption of conditional independence of the observed data. In generative approach, this might be too restrictive for a considerable number of object classes. Unlike ME, CRF does not suffer from the label bias problem. CRF can include arbitrary set of features and has the ability of automatic feature induction.

2.2 Features

We use the gold-standard TimeBank features for training and testing the CRF model. The features are extracted automatically from the respective datasets but we mainly use the various combinations of the following features:

(i). **Event Class**: This is denoted by the 'EVENT' tag and used to annotate those elements in a text that mark the semantic events described by it.
(ii). **Event Stem**: This feature extracts the stem of the head event.
(iii). **Event and Time Strings**: This feature is used to denote the actual event strings and time.
(iv). **Part of Speech (POS) of Event Terms**: POS information is very useful to identify the even-event relations. The features values may be either of ADJECTIVE, NOUN, VERB, and PREP.
(v). **Event Tense**: This feature is useful to capture the standard distinctions among the grammatical categories of verbal phrases. The tense attribute can have values, namely PRESENT, PAST, FUTURE, INFINITIVE, PRESPART, PASTPART, or NONE
(vi). **Event Aspect**: It denotes the aspect of the events. The aspect attribute may take values**,** PROGRESSIVE, PERFECTIVE and PERFECTIVE PROGRESSIVE or NONE
(vii). **Event Polarity**: Polarity of an event instance is represented by the boolean value, POSITIVE or NEGATIVE.
(viii). **Event Modality**: The modality attribute is only present if there is a modal word that modifies the instance.
(ix). **Type of Temporal Expression**: It represents the temporal relationship holding between events, times or between an event and a time of the event.
(x). **Temporal Signal**: This is used to represent the temporal prepositions.
(xi).**Temporal Relation between the Document Creation Time and Temporal Expression in the Target Sentence**: The value of this feature could be "*greater than*", "*less than*", "*equal*", or "*none*".

Incorporating Sentiment Feature

The sentiment is an important cue that effectively describes the events associated with it. The binary classification of the sentiments as well as the fine-grained categorization of Ekman's six emotions is utilized to qualify the event properties. The sentiment attributes are identified for each of the sentences. Here, we mainly employ the word to sentence level emotion tagging module [11] for identifying the sentiment properties of events. The features for positive and negative sentiments as well as a representative feature with respect to all the six emotions are assigned for classifying the event pairs into the temporal classes.

The sentiment or emotional verbs play an important role in identifying the temporal relations. To accomplish the goal, we include a special feature for sentiment verbs that are identified using SentiWordNet [12] or WordNet Affect lists [14] or VerbNet [15]. The verbs of SentiWordNet or WordNet Affect are identified using the Part-of-Speech (POS) information. On the other hand, VerbNet associates the semantics of a verb with its syntactic frames and combines traditional lexical semantic information such as thematic roles and semantic predicates, with syntactic frames and selectional restrictions. Verb entries in the same VerbNet class share common syntactic frames, and thus they are believed to have the same syntactic behavior. The VerbNet files containing the verbs with their possible subcategorization frames and membership information are stored in XML file format. For example, the emotional verbs "*love*" and "*enjoy*" are members of the *admire-31.2-1* class and "*enjoy*" also belongs to the class *want-32.1-1*. The XML files of VerbNet are preprocessed to build up a general list that contains all member verbs and their available syntax information retrieved from VerbNet. The main criterion that is considered for selecting the frames is the presence of "*emotional_state*" type predicate associated with the frame semantics.

We obtain the training and testing datasets from the TempEval-2007 evaluation task. The datasets are preprocessed for the specified CRF format. Thereafter, we extract features in the form of vectors from the annotated training data. The feature vectors consisting of the available features for each <*main-event*, *main-event*> and <*main-event*, *next subordinate event*, *previous subordinate event*, *main-event*> pair in the TimeBank corpus are identified. Now, we have a training data in the form (*Wi, Ti*), where, *Wi* is the i^{th} pair along with its feature vector and *Ti* is its corresponding TempEval relation class. All the feature vectors are extracted from the training data. The temporal relations are annotated by one of the labels, such as BEFORE, BEFORE-OR-OVERLAP, OVERLAP, OVERLAP-OR-AFTER, AFTER or VAGUE. We have trained CRF using the automatically extracted feature vectors and by defining the appropriate feature template. The models are created from the training set and the feature template. The same feature extraction methodology is again repeated for the test data. An unknown instance <*main-event*, *main-event*> or <*main-event*, *next subordinate event*, *previous subordinate event*, *main-event*> is assigned the appropriate output label, i.e., OVERLAP, BEFORE, AFTER, BEFORE-OR-OVERLAP, OVERLAP-OR-AFTER and VAGUE, depending upon the probabilities, learned in the CRF model. The output label predicted by the CRF is matched against the reference label.

3 Evaluation of Temporal Event Identification System

For TempEval -2007, the tasks were defined in such a way that a simple pair-wise comparison is possible since it was not required to create a full temporal graph and judgments are made in isolation. There are three basic temporal relations (BEFORE, OVERLAP, and AFTER) as well as three disjunctions over this set (BEFORE-OR-OVERLAP, OVERLAP-OR-AFTER and VAGUE). The organizers used two scoring schemes: **strict** and **relaxed**.

The strict scoring scheme only counts exact matches as success. For example, if the key is OVERLAP and the response is BEFORE-OR-OVERLAP then this is counted as ‘failure’. The standard definitions of precision and recall are followed: *Precision* = *Rc* / *R* and *Recall* = *Rc* / *K*, where, *Rc* is the number of correct answers in the response, *R* is the total number of answers in the response and *K is* the total number of answers in the key. For the relaxed scoring scheme, precision and recall are defined as *Precision* = *Rcw* / *R* and *Recall* = *Rcw* / *K*, where, *Rcw* reflects the weighted number of correct answers. The *F-score* is measured as follows where *Pr* = *Precision* and *Re* = *Recall*: *F-score*=2**Pr***Re* / (*Pr* + *Re*). We have developed a number of CRF models based on the features and/or feature templates included into it. We have a training data in the form (*Wi, Ti*), where, *Wi* is the i^{th} pair along with its feature vector and *Ti* is its corresponding TempEval relation class. Models are built based on the training data and the feature template. During evaluation, we obtain the highest performance for the following feature templates as shown in Figure 1. In the figure, w_i : Current <event, event> pair, $w_{(i-n)}$: Previous nth <event, event> pair, $w_{(i+n)}$: Next nth <event, event> pair, t_{i-1}: previous <event, event> pair.

$w_{(i-3)}$
$w_{(i-2)}$
$w_{(i-1)}$
w_i
w_{i+1}
$w_{(i+2)}$
$w_{(i+3)}$
Combination of w_{i-1} and w_i
Combination of w_i and w_{i+1}
Dynamic output tag (t_{i-1}) of the previous pair
Feature vector of w_i of other features

Fig. 1. Best Feature Template of the CRF based System for <*main-event, next sub-event, prev sub-event, main-event*> relation

The test data consists of 20 articles from TimeBank [8]. The performance is assessed with three evaluation metrics, namely precision (P), recall (R) and F-score (FS). The systems are evaluated in terms of two scoring schemes, ‘*strict*’ and ‘*relaxed*’. The *strict* scoring scheme counts only exact matches, while the *relaxed* one gives credit to partial semantic matches too. Evaluation results [13] show that the system performs better with the context size of seven (i.e., previous three, current and the next three <*main-event*, *next-subordinate-event*, *previous- subordinate-event*, *main-event*> pairs), tense, aspect and temporal class features. It shows the precision, recall and F-score values of 43.8% 43.8% and 43.8% respectively under the *strict* evaluation scheme and 46.9, 46.9% and 46.9%, respectively under the *relaxed* evaluation scheme. For the sub-ordinate event, the system demonstrated the precision, recall and F-score values of 55.1%, 55.1% and 55.1%, respectively for the strict evaluation scheme and 56.9%, 56.9% and 56.9%, respectively for the relaxed evaluation scheme. Table 1 shows the results by incorporating the sentiment feature into the system. For the *main-event*, sentiment feature in CRF based system performs better with the margins of 1.4 percentage F-scores in the *strict* evaluation scheme and 1.5 percentage F-scores in the relaxed evaluation scheme.

Table 1. Precision (P), Recall (R) and F-scores of CRF based system

Techniques	**Strict**			**Relaxed**		
	P	**R**	**FS**	**P**	**R**	**FS**
CRF (*main event*)	0.438	0.438	0.438	0.469	0.469	0.469
CRF (*main event*) +**Sentiment Feature**	0.452	0.452	0.452	0.484	0.484	0.484
CRF (*subordinate event*)	0.551	0.551	0.551	0.569	0.569	0.569
CRF (*subordinate event*) +**Sentiment Feature**	0.572	0.572	0.572	0.586	0.586	0.586

The incorporation of sentiment feature also shows that the CRF is most effective to handle the *subordinate-event* in association with the knowledge regarding the sentiment property of the events. It shows the overall performance improvement of 2.1 and 1.7 percentage points over the earlier model in the strict and relaxed evaluation scheme, respectively. The system also exhibits superior performance for the *subordinate-event* over the *main-event*.

4 Event-Sentiment Tagging

Opinion or Emotion or Sentiment analysis is a recent sub discipline at the crossroads of information retrieval [16] and computational linguistics [17]. Information is concerned not only with the subject or topic or event of a document but also with the sentiment or emotion it expresses. It has a rich set of applications such as tracking users' sentiments about products or events or about political candidates as expressed in online forums, customer relationship management, stock market prediction, social networking etc. Not only the classification of reviews [18] or newspaper articles [19] or blogs [20] but the Question-Answering systems [21] and current Information Retrieval systems [22] are also increasingly incorporating sentiment analysis within their scopes.

In the present task, we use the TimeML corpus for assigning coarse grained sentiments such as positive or negative and fine grained sentiments such as Ekman's six emotions (*happiness, sadness, anger, fear, surprise* and *disgust*) at the sentence level. Event-sentiment relational graph for each of the documents is generated based on the temporal relations identified using a CRF based supervised event-event relation identification system. The generation of the event-sentiment relational graph using an open source graphical system helps to visualize and track the changes of sentiments between consecutive and remote events of a document. The sentiment change between two consecutive events is termed as sentiment twist and change between rare or remote events containing one or more intermediate events is termed as sentiment transition. The manual evaluation of the event-sentiment system sounds satisfactory.

4.1 Tagging of Sentiments to Events

The investigation mainly focused on analyzing the impacts of coarse grained and fine grained sentiments on events that are present in the TempEval-2007 TimeML corpus. Ekman's [8] six basic emotion types, such as *happiness, sadness, anger, fear, surprise,* and *disgust* are considered as fine-grained sentiment whereas two different valences, positive and negative are considered as coarse grained and are assigned to the sentences that contain events. Other sentences are considered as neutral. The sentiment tagging systems [10] [11] work at two levels, word level followed by sentence level.

The CRF based machine learning approach that incorporates several singleton features (e.g. *POS*, *words from SentiWordNet* [12], *question words*, *reduplication*, *punctuation markers* or *special symbols* (!, @, $, ?)) as well as different combination of *context features* (e.g., *unigram*, *bigram* etc.) are employed for word level emotion tagging. The system [10] demonstrates F-score of 72.27% for English SemEval 2007 affect sensing news corpus containing 2,500 development word tokens. Incorporation of error analysis and equal distribution of emotion tags with the non-emotion tag improves the word level emotion tagging and an overall F-score of 83.65% is achieved by the system on 1,500 test word tokens of the news corpus.

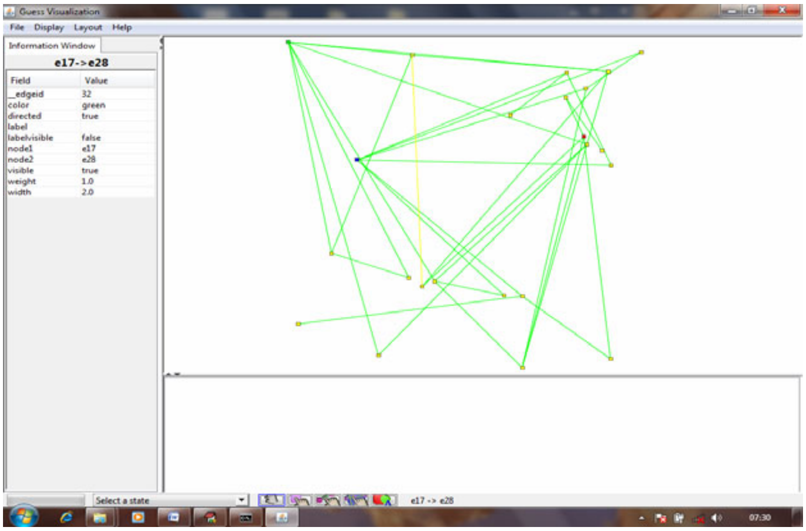
The sentential emotions and valence tags are assigned based on the word level emotion tagged constituents. The system calculates six different emotion tag weights from SentiWordNet and applies the tag weights on the word level emotion tagged data to acquire sentence level emotion scores for each emotion type. A sentence level emotion tag that has the maximum emotion score is assigned to each sentence. The system shows the F-scores of 66.66%, 59.33%, 62.32%, 62.70%, 65.89% and 62.67% for *happy sad, anger, disgust*, *fear* and *surprise* emotion classes, respectively on 200 test sentences. On the other hand, the polarity information of the emotion tag weights helps in calculating the valence score for each of the sentences. The total emotion tag weights acquired from the different emotion tags in a sentence are treated as the valence or coarse grained sentiments (*positive* and *negative*) of the sentence. It has been observed that the system achieves an average F-score of 66.41% for coarse grained sentiment tagging on 250 test sentences [11].

As there is no emotion-annotated information available in TempEval-2007 corpus, each test sentence of the corpus was annotated with single emotion tag and evaluated by us successively. The systems [10] [11] have been applied on 20 test articles of the TempEval event corpus. Manual evaluation shows that the system achieves an average F-score of 64.23% for emotion tagging with respect to all emotion classes. But, the sentences of the TempEval corpus is annotated with positive and negative sentiments and the evaluation yields an average F-score of 66.23%.

4.2 Generation of Event-Sentiment Relational Graph

The Ekman's six different emotions along with positive, negative or neutral sentiments are tagged with the sentences containing one or more potential events. If we consider the positive, negative and neutral valences as coarse grained sentiment events and Ekman's six emotions as the fine-grained sentiments, by hypothesis, the temporal relations also exist among the sentiment events and the relations between each of the events are represented using a directed graph. The temporal relations between each of the successive sentiment or neutral events have already been identified by the CRF based system as described in Section 2.

An open source graphical tool [3] has been used to represent the temporal relations among the events. The tool uses an XML file schema to store the information regarding nodes as well as the edges of a graph (as shown in Figure 2). For each of the documents of TempEval-2007 event corpus, a separate graph is generated. The sentiment of each sentence is assigned to its containing event and each event is represented using a graphical node. The event nodes that are of similar sentiments are connected to their corresponding sentiment hubs based on their annotated sentential sentiment tags.



e17->e28

Field	Value
__edgeid	32
color	green
directed	true
label	
labelvisible	false
node1	e17
node2	e28
visible	true
weight	1.0
width	2.0

Fig. 2. Snapshot of a Sentiment-Event Directed Graph for a document

4.3 Tracking of Event-Sentiment Relational Graph

The tracking of sentiments includes sentiment twist and sentiment transition. The sentiment twist between two consecutive events and sentiment transition among more than two events are identified by arriving at the corresponding sentiment hub. The sentiment change or tracking of sentiments between two consecutive events or

[3] http://www.hpl.hp.com/research/idl/projects/graphs/guess.html

sentiment twist is identified from the AFTER, BEFORE and OVERLAP temporal relations. The ambiguities of the OVERLAP relations are identified by the notion of BEFORE-OR-OVERLAP, OVERLAP-OR-AFTER relations. It has been observed that the maximum length of an event chain or sentiment transition in the TempEval 2007 corpus is four. The number of instances of the sentiment transitions is less than the number of instances of the sentiment twists in the TimeML corpus. Hence, the sentiment transition or tracking of sentiment is identified based on the sentiment twists of the intermediate event pairs in an event chain.

4.4 Evaluation of Event-Sentiment Tracking

The sentiment events are identified from 20 test documents. Apart from sentiment twist, the maximum number of participating events in a sentiment transition is 3~ 4. The results of the event-sentiment tracking are shown in Table 2. In the present experiments, we have considered all the six emotions of Ekman's (*happiness, sadness, anger, fear, surprise* and *disgust*) but the results only consider the single average emotion instead of six. Results show that the performance of the system is comparatively better in case of identifying sentiment tracking between coarse grained sentiments rather than coarse to fine grained sentiments and vice versa. It has to be mentioned that though the system performs satisfactory in identifying the event sentiment tracking path, most of the errors have occurred due to the misleading characteristics of the system in assigning neutral tags to the sentences with implicit sentiments. The rest of the errors occur in detecting the immediate sentiment changes of reverse polarity (+ve/-ve) in the sentiment twists. It shows that the complementary sentiment changes in sentiment twists are not always reliable without proper reasoning but in case of sentiment transitions, the reversibility may occur by changing sentiments in the intermediate event nodes in an event chain.

Table 2. Results of F-scores of the Event-Sentiment Tracking

Source-Destination	**Twist/Transition between # Events**		
Pair	**Two**	**Three**	**Four**
+ve → -ve	0.7032	0.6924	0.6821
+ve → neutral	0.7365	--	--
+ve → Emotion	0.6414	0.6277	0.6087
-ve → +ve	0.7065	0.6833	0.6724
-ve → neutral	0.7151	--	--
-ve → Emotion	0.6075	0.5877	0.5729
neutral → +ve	0.7011	--	--
neutral → -ve	0.6898	--	--
neutral → Emotion	0.6337	--	--
Emotion → +ve	0.6227	0.6077	0.5802
Emotion → -ve	0.6393	0.6207	0.6162
Emotion → neutral	0.6210	--	--

5 Conclusion

In this paper, we have reported our work on temporal relation identification under the TempEval 2007 evaluation exercise. The Task C of TempEval-2007was involved with the identification of six relations between the events in two consecutive sentences. Evaluation results show that the CRF based system outperforms all the state-of-the-art participating systems by including sentiment as the feature with all other available features of the TimeBank corpus. The sentiment tagged events, their visualization and tracking as well the evaluation show a promising venue of research. In future, we would like to employ the system in identifying sentiment changes from lengthy event chains in order to investigate the potential reasoning behind the sentiment change.

Acknowledgments. The work reported in this paper was supported by a grant from the India-Japan Cooperative Programme (DSTJST) 2009 Research project entitled "Sentiment Analysis where AI meets Psychology" funded by Department of Science and Technology (DST), Government of India.

References

1. Wayne, C.L.: Multilingual Topic Detection and Tracking: Successful Research Enabled by Corpora and Evaluation. In: Language Resources and Evaluation Conference (LREC), pp. 1487–1494 (2000)
2. Quirk, R., Greenbaum, S., Leech, G., Svartvik, J.: A comprehensive Grammar of the English Language. Longman, New York (1985)
3. Mishne, G., de Rijke, M.: MoodViews: Tools for Blog Mood Analysis. In: AAAI 2006 Spring Symposium on Computational Approaches to analyzing Weblogs, AAAI-CAAW 2006 (2006)
4. Havre, S., Hetzler, E., Whitney, P., Nowell, L.: ThemeRiver: Visualizing Thematic Changes in Large Document Collections. IEEE Transactions on Visualization and Computer Graphics 8(1), 9–20 (2002)
5. Fukuhara, T., Nakagawa, H., Nishida, T.: Understanding Sentiment of People from News Articles: Temporal Sentiment Analysis of Social Events. In: ICWSM 2007, Boulder, Colorado, USA (2007)
6. Verhagen, M., Gaizauskas, R., Schilder, F., Hepple, M., Katz, G., Pustejovsky, J.: SemEval-2007 Task 15: TempEval Temporal Relation Identification. In: Proceedings of the 4th International Workshop on Semantic Evaluations (semEval 2007), pp. 75–80. Association for Computational Linguistics, Prague (2007)
7. Lafferty, J., McCallum, A.K., Pereira, F.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: Proceedings of 18th International Conference on Machine Learning (2001)
8. Ekman, P.: Facial expression and emotion. American Psychologist 48(4), 384–392 (1993)
9. Pustejovsky, J., Castano, J., Ingria, R., Sauri, R., Gaizauskas, R., Setzer, A., Katz, G., Radev, D.: TimeML: Robust Specification of Event and Temporal Expressions in Text. In: Proceedings of the Fifth International Workshop on Computational Semantics (IWCS-5), Tilburg (January 2003)
10. Das, D., Bandyopadhyay, S.: Sentence Level Emotion Tagging on Blog and News Corpora. Journal of Intelligent System (JIS) 19(2), 125–134 (2010)

11. Das, D., Bandyopadhyay, S.: Sentence Level Emotion Tagging. In: Proceedings of the International Conference on Affective Computing & Intelligent Interaction, Amsterdam, Netherlands, pp. 375–380 (2009), doi:10.1109/ACII.2009.5349598
12. Esuli, A., Sebastiani, F.: SENTIWORDNET: A Publicly Available Lexical Resource for Opinion Mining. In: LREC 2006 (2006)
13. Kolya, A.K., Ekbal, A., Bandyopadhyay, S.: A Supervised Machine Learning Approach for Event Event Relation Identification. In: PACLIC 2010, Sendai, Japan, November 4-7 (2010)
14. Strapparava, C., Valitutti, A.: WordNet-Affect: an affective extension of WordNet. In: Proceedings of the 4th International Conference on Language Resources and Evaluation, pp. 1083–1086 (2004)
15. Kipper-Schuler, K.: VerbNet: A broad-coverage, comprehensive verb lexicon. Ph.D. thesis, Computer and Information Science Dept., University of Pennsylvania, Philadelphia, PA (2005)
16. Sood, S., Vasserman, L.: ESSE: Exploring Mood on the Web. In: Proceedings of the 3rd International AAAI Conference on Weblogs and Social Media (ICWSM) Data Challenge Workshop (2009)
17. Wiebe, J., Wilson, T., Rebecca, F., Bell, M., Martin, M.: Learning Subjective Language. Computational Linguistics 30, 277–308 (2004)
18. Turney, P.D.: Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pp. 417–424 (2002)
19. Lin, K.H.-Y., Yang, C., Chen, H.-H.: What Emotions News Articles Trigger in Their Readers? In: Proceedings of SIGIR, pp. 733–734 (2007)
20. Yang, C., Lin, K.H.-Y., Chen, H.-H.: Emotion classification Using Web Blog Corpora. In: IEEE, WIC, ACM International Conference on Web Intelligence, pp. 275–278 (2007)
21. Cardie, C., Wiebe, J., Wilson, T., Litman, J.D.: Combining Low-Level and Summary Representations of Opinions for Multi-Perspective Question Answering. In: New Directions in Question Answering, pp. 20–27 (2003)
22. Pang, B., Lee, L.: Opinion mining and sentiment analysis. In: Foundations and Trends in Information Retrieval 2, vol. 1(2), pp. 1–135 (2008)

Highly-Inflected Language Generation Using Factored Language Models

Eder Miranda de Novais, Ivandré Paraboni, and Diogo Takaki Ferreira

School of Arts, Sciences and Humanities, University of São Paulo (USP / EACH)
Av. Arlindo Bettio, 1000 - São Paulo, Brazil
{eder.novais,ivandre,diogo.ferreira}@usp.br

Abstract. Statistical language models based on n-gram counts have been shown to successfully replace grammar rules in standard 2-stage (or 'generate-and-select') Natural Language Generation (NLG). In highly-inflected languages, however, the amount of training data required to cope with n-gram sparseness may be simply unobtainable, and the benefits of a statistical approach become less obvious. In this work we address the issue of text generation in a highly-inflected language by making use of factored language models (FLM) that take morphological information into account. We present a number of experiments involving the use of simple FLMs applied to various surface realisation tasks, showing that FLMs may implement 2-stage generation with results that are far superior to standard n-gram models alone.

Keywords: Text Generation, Surface Realisation, Language Modelling.

1 Introduction

In Natural Language Generation (NLG) systems, surface realisation can be viewed as the task that takes as an input a set of features representing a sentence specification (or *what* to say), and produces the corresponding output string (*how* to say it) [1]. In the case of symbolic generation, the input to surface realisation will normally have to be provided in a high level of detail, making the surface realisation module[1] difficult to adapt to applications that are not linguistically-motivated by design.

By contrast, with the more recent use of statistical methods in NLG, the issue of input specification to surface realisation has become in many ways more manageable. Standard 2-stage (or 'generate-and-select') architectures as in [3][2] complement an underspecified input by overgenerating a large number of alternative realisations (often including ungrammatical sentences) and selecting the most likely output according to a statistical language model.

Language models may however suffer from data sparseness, and statistical NLG relies heavily on large corpora as training data. Early work in the field [3]

[1] See [2] for details on a typical NLG architecture.

[2] Given the limited availability of NLP resources for our target language, in this paper we do not discuss state-of-art grammar acquisition approaches such as [4].

implemented 2-stage generation with the aid of a 250-million words corpus of English texts. Presently, several billion-words English corpora are not uncommon, but in order to achieve comparable results for more inflected languages (for which the effects of data sparseness will be presumably much more dramatic) the training corpus would have to be in principle much larger.

Since such large (and good-quality) corpora may not always be obtainable, alternative ways of coping with data sparseness in highly-inflected languages are called for. To this end, one possible strategy that has emerged in recent years within the language modelling community is the use of language models that take additional (e.g., morphological) features into account besides the usual n-gram counts.

In this work we investigate the use of so-called Factored Language Models [11] as an alternative to standard n-gram counts in order to generate text in a highly inflected language. To this end, we present a number of experiments focussing on individual surface realisation subtasks, and we compare the results of different models applied to the 'select' stage of the generation process.

The reminder of this paper is organised as follows. Section 2 introduces previous work on 2-stage NLG and the surface realisation issues that are the focus of this paper. Section 3 describes our general approach and the language models to be used in the experiments. Section 4 described the experiments themselves and their results, which are further discussed in Section 5. Section 6 presents our conclusions and future work.

2 Background

The work in [5] presented a number of experiments in 2-stage surface realisation in which generation decisions were made with the aid of simple n-gram models built from a 40-million words corpus of Brazilian Portuguese newspaper articles [6]. The experiments intended to simplify the input requirements to a surface realisation module (and hence free the underlying application from the burden of making certain linguistic decisions) in a number of individual surface realisation tasks, ranging from synonymy to sentence linearization.

Generally speaking, results in [5] were positive but still insufficient for practical purposes. This was particularly the case of three subtasks, which were deemed problematic for the n-gram approach: the lexicalization of verb phrases, the ordering of noun modifiers and the issue of verb-complement agreement (particularly in passive voice.) In this paper the three issues are revisited.

Verb phrase (VP) lexicalization [7] is the task of choosing the appropriate wording for VP heads (i.e., verbs) from a synset conveying a potentially large number of alternative realisations[3]. For instance, given the goal of producing an output string for the input concept 'say', the system will consider a synset conveying synonymous words such as 'say', 'inform', 'tell' and so on. The task becomes even more complex if, for instance, we do not have a means to choose

[3] The lexicalization of Portuguese NPs, by contrast, was found to be less problematic in [5] and will not be further discussed in this paper.

among several competing synsets [8]. As we discuss later, applying random selection to this task is not a viable alternative, and even a purely frequency-based approach tends to perform poorly.

The ordering of noun modifiers corresponds to the linearization of a set of content words conveying a noun head and its modifiers (adjectives, prepositional phrases etc.) For instance, given the content words {blue, small, car} we would like to produce the NP 'the small blue car'. The ordering of noun modifiers is a complex task even in the case of English modifiers, which tend to occur before the NP head [9,10]. In languages such as Portuguese, however, the ordering of modifiers is far more flexible.

Finally, the verb-complement agreement task consists of establishing the correct morphological features of verb complements. For instance, in 'They are nice guys' the complement ('nice') agrees in number with the subject ('They'.) In more inflected languages, verb-complement agreement may also involve gender agreement, and the task becomes even more challenging (from an n-gram perspective) if we consider the potentially long distances between the agreeing terms. In [5] this was found to be particularly the case of sentences in passive voice, which may be explained by the observation that passive voice usage (and the corresponding n-grams) was less common in the training data.

The accuracy scores reported in [5] for the above surface realisation subtasks were as follows: VP lexicalization achieved up to 40% accuracy using a frequency-based approach; the ordering of noun modifiers achieved up to 65% accuracy using a standard 2-gram language model, and verb-complement agreement in passive voice obtained up to 68% accuracy using a 3-gram language model. In what follows we intend to improve on all these results.

3 Current Work

Despite the overall positive results of the n-gram approach in [5] for our three working issues (namely, VP lexicalization, ordering of noun modifiers and verb-complement agreement), one may ask how these results may be improved without resorting to a very large (and otherwise unavailable) corpus. One possible strategy, and which will be followed in this paper, is the use of *Factored Language Models* (FLMs) [11], which are widely used in speech recognition, machine translation and many other statistical NLP applications.

FLMs generalize the n-gram approach by incorporating information from various sources (besides n-gram counts.) In a FLM, each word w_t is represented as a vector of k factors $\{f_t^1, f_t^2, ...f_t^k\}$. Factors may represent any word feature deemed useful (e.g., to model a highly-inflected language) such as morphological classes, roots, semantics etc. For details on FLMs and the related issue of generalized parallel backoff, we refer to [11].

The motivation for using FLMs in highly-inflected language processing is self-explanatory, as illustrated even by a simple example in English: if a bigram as 'he failed' does not occur in the training data, but the bigram 'he finished' does, we may still take advantage of, say, part-of-speech information, to obtain some probability estimate of 'failed' given that the previous word is a pronoun.

The benefits of using FLMs are of course more evident in the case of a highly-inflected language, for which not only verbs but many other word classes may vary in gender, number, tense etc. However, taking a large number of factors into account may be costly, and the resulting models may be difficult to use in practice (assuming of course that we had sufficient computational resources to build such large models in the first place.) Given our goal of applying FLMs to practical language generation, in this work we will focus on simple FLMs conveying a limited set of factors, and which can be applied under conditions of reasonable efficiency.

For the purpose of comparison to previous work, all language models below were built from the same training data used in [5]. From this corpus, standard interpolated 2-gram and 3-gram language models were built to be used in our current baseline systems. In addition to that, from a tagged version of the same corpus, we also built four factored language models that are meant to be tested against the n-gram approach. The factored models take into account words, lemmas, part-of-speech, gender and number information, although the best-performing alternative in each experiment turned out to use only a subset of the original factors[4]. All factored models use Kneser-Ney smoothing when applicable.

For the first two tasks (VP lexicalization and ordering of noun modifiers) we will use a simple factored model that takes the previous words (W) into account and, in case the bigram (or trigram) is not found, attempts to use the previous lemma counts (L) instead. We will refer to the bigram and trigram versions of this model respectively as FLM_1 and FLM_2, which are represented as follows.

$$FLM_1 : p(W_t|W_{t-1}, L_{t-1}).$$

$$FLM_2 : p(W_t|W_{t-1}, L_{t-1}, W_{t-2}, L_{t-2}).$$

Regarding our third task (verb-complement agreement), a pilot experiment suggested - perhaps rather intuitively - the need to take gender and number information into account. Thus, in our bigram FLM_3 and its trigram version FLM_4 the probability of the current word is estimated by considering as factors the previous words (W), gender (G) and number (N) information:

$$FLM_3 : p(W_t|W_{t-1}, G_{t-1}, N_{t-1}).$$

$$FLM_4 : p(W_t|W_{t-1}, G_{t-1}, N_{t-1}, W_{t-2}, G_{t-2}, N_{t-2}).$$

In the next section we apply the above $FLM_1...FLM_4$ models to the three surface realisation tasks under consideration as part of a standard 2-stage NLG system for a highly-inflected language.

[4] In particular, in an earlier version of this work, a model using part-of-speech information was found to outperform our current system for task 2 (ordering of modifiers), but the difference turned out to be non-significant for our present purposes.

4 Evaluation

In what follows we describe the results of a number of experiments focussing on the three surface realisation tasks introduced in the previous section (VP lexicalization, ordering of noun modifiers and verb-complement agreement.) To this end, surface realisation decisions made with the aid of factored language models ($FLM_1...FLM_4$) will be compared with a number of baseline systems deemed relevant to each task. These include the standard n-gram approach and additional Random and Frequency-based strategies when relevant. All results reported for the baseline systems are taken from [5], and are presently reproduced for ease of comparison. The factored language models and instructions on how to use them are available from www.CICLing.org/2011/software/75.

The test data used in our experiments consisted of the original input for each task and corresponding Reference sets described in [5]. Evaluation proper is carried out by comparing the strings produced by each system (FLM or baseline) to the equivalent in each Reference set, using the following four evaluation metrics:

– Edit-distance (Levenshtein's distance): the number of insert, delete and substitute operations needed to make both strings identical. Zero edit-distance stands for an identical match.
– Accuracy: the number of exact matches between the two strings, being assigned the value 1 if both strings are identical, or 0 otherwise.
– NIST [12] and BLEU [13] scores: standard Machine Translation evaluation metrics based on n-gram counts. BLEU scores range from 0 to 1, being 1 equal to 100% accuracy. NIST scores (which give more weight to rare n-grams than in the case of BLEU scores) do not have an upper bound, but higher values stand for higher correctness as well.

4.1 Verb Phrase Lexical Choice

Our first experiment addresses the use of FLMs applied to the task of lexicalization of Verb Phrase (VP) head constituents. Given the goal of realising a particular concept as a verb, the experiment consists of choosing the appropriate verb surface form from a synset conveying, on average, 10 synonymous for each concept. To this end, the experiment used the same 40-sentences Reference set built for the task in [5]. Keeping all the other sentence constituents unchanged, the Reference set was overgenerated allowing only the main VP head constituents to vary according to the synonymous found in each synset, producing 400 alternative surface realisation forms.

In order to select the most likely surface realisation form for each sentence, four baseline systems taken from [5] are considered: standard bigram and trigram language models, a Frequency-based approach that selects the most common word found in the corpus for each verb concept, and a Random strategy that selects a random synonym out of each synset. The baseline systems are compared against FLM_1 (bigram) and FLM_2 (trigram) models described in the previous section (i.e., those that take the previous words and their lemmas into account.)

Table 1 shows the results of this experiment (keeping in mind that best results correspond to *lower* Edit-distance and *higher* Accuracy, NIST and BLEU scores.)

Table 1. Verb Phrase lexical choice

System	Edit-distance	Accuracy	NIST	BLEU
Standard 2-gram	4.98	0.13	8.731	0.863
Standard 3-gram	3.88	0.30	8.877	0.893
FLM_1	3.00	0.53	9.009	0.928
FLM_2	1.93	0.65	9.119	0.947
Frequency-based	3.25	0.42	8.891	0.907
Random	5.38	0.13	8.659	0.852

4.2 Ordering of Noun Modifiers

The second experiment applies FLMs to the ordering of pre- and post-modifiers such as adjectives, prepositional phrases (PPs) etc. within noun phrases (NPs.) To this end, we reused the Reference set developed for the task in [5], conveying 40 sentences in standard subject-verb-complement structure.

The NPs under consideration conveyed one or two modifiers each, with a single (i.e., non-ambiguous) ordering evenly distributed across the data (i.e., before or after the head noun, or in both positions simultaneously.) In addition to that, two kinds of NP were considered: with and without PP attachments as in 'the far end of the corridor'. The use of a PP attachment increases the challenge of finding the correct ordering using a statistical model as some alternatives under consideration (e.g., '*the end of the corridor far') may leave a potentially wide gap between modifier and head noun[5].

Given that certain orderings would not naturally occur in our data (e.g., placing an adjective between the noun and a prepositional phrase), and also to enable a comparison with related work [5], this experiment did not test every possible ordering of modifiers, that is, we limited ourselves to a number of combinations that were critical to our approach. For that reason we will call this a constrained experiment, which will be subsequently complemented by an unconstrained version that takes full free order into account.

The constrained experiment consisted of producing different orderings for the constituents of NPs found in the subject position, and comparing a number of strategies for selecting the most likely surface realisation for each case. Keeping all other sentence constituents unaltered, 128 alternative sentences were produced (an average of 3.2 alternatives for each Reference sentence,) and subsequently selected according to three baseline systems: the standard bigram and trigram models, and a Random selection strategy[6]. These baseline

[5] The use of PPs in this way is nevertheless ubiquitous in our target language.

[6] Unlike the previous experiment (VP lexical choice,) the task of choosing the ordering of noun modifiers is not a matter of domain or genre preference, but rather a decision guided by linguistic constraints. For that reason, a comparison to a frequency-based strategy would not be applicable.

systems are compared once again to the FLM_1 (bigram) and FLM_2 (trigram) models. The results are shown in Table 2.

Table 2. Ordering of noun modifiers (constrained)

System	Edit-distance	Accuracy	NIST	BLEU
Standard 2-gram	5.15	0.65	7.244	0.814
Standard 3-gram	5.35	0.63	7.201	0.795
FLM_1	1.55	0.85	7.341	0.929
FLM_2	2.75	0.80	7.314	0.888
Random	9.73	0.27	6.831	0.559

As the data used in the above experiment was limited to a subset of possible alternative orderings (namely, the most challenging alternatives from a 2-stage generation perspective,) we decided to make the task more difficult and test how the FLM models would perform under free word order. We developed an unconstrained version of the same experiment in which all possible orderings of NP constituents in the Reference set were considered, making 352 alternatives in total (an average of 8.8 alternatives for each sentence.) Results for the FLM models and the Random baseline system are presented in Table 3.

Table 3. Ordering of noun modifiers (unconstrained)

System	Edit-distance	Accuracy	NIST	BLEU
FLM_1	3.93	0.70	7.304	0.772
FLM_2	4.43	0.60	7.224	0.754
Random	11.85	0.20	6.369	0.346

4.3 Verb-Complement Agreement

In our final experiment we evaluate the use of FLMs to establish agreement between verb and complement constituents in passive voice usage. As a Reference set we used the 120 sentences in passive voice considered in [5]. The sentences are divided in three groups conveying 0, 1 or 2 intermediate adverb constituents (hence varying the distance within the dependency chain), with complement gender (male/female) and number (singular/plural) attributes evenly distributed. The following is an example of each group, keeping in mind that the original (Portuguese) text would require the complement term ('concerned') to agree with the noun head ('visitors') in both gender and number.

(a) The visitors are concerned by the news.
(b) The visitors are very concerned by the news.
(c) The visitors are not very concerned by the news.

The same evaluation procedure adopted in the previous experiments was repeated. Keeping all other sentence constituents unchanged, the sentences in the

Reference set were overgenerated whilst allowing only the gender and number of the complement term to vary, making four alternative surface realisations for each sentence. The resulting 480 output sentences (4 alternatives per sentence) were evaluated according to the bigram, trigram and Random baseline strategies, and also by the FLM_3 (bigram) and FLM_4 (trigram) models. The results are shown in Table 4.

Table 4. Verb-complement agreement constraints (passive voice)

System	Edit-distance	Accuracy	NIST	BLEU
Standard 2-gram	0.45	0.67	7.132	0.886
Standard 3-gram	0.42	0.68	7.167	0.892
FLM_3	0.30	0.75	7.171	0.920
FLM_4	0.25	0.75	7.169	0.920
Random	0.95	0.30	6.473	0.754

5 Discussion

The experiments presented in the previous section illustrate a number of ways in which simple factored language models may free an underlying application from (some of) the burden of providing detailed input specification to the surface realisation module.

The lexicalization of Portuguese VPs was previously shown [5] to be a challenging task with maximum accuracy of 42% for a frequency-based strategy. Presently, we notice that both FLM_1 and FLM_2 now easily outperform all the baseline systems in Table 1, including the frequency-based approach. However, we acknowledge that these results may be still considered insufficient for practical purposes (i.e., with maximum 65% accuracy for the trigram factored model), and that more training data may be ultimately required.

Regarding the ordering of noun modifiers task, previous work has shown maximum accuracy of 65% using a standard bigram model. This was once again outperformed by the use of factored language models, with up to 85% accuracy in the case of the bigram FLM (Table 2). Interestingly, however, in both standard and factored language models, the use of bigrams slightly outperforms the trigram version of the same model. Although the difference in this case was not found to be significant, this may suggest that the ordering of noun modifiers does not require higher-order language models.

In the complementary (unconstrained) experiment on ordering of noun modifiers all permutations of NP constituents were considered, which nearly tripled the size of the search space. Even in this case, however, we were able to show (cf. Table 3) that FLMs were superior to random selection, and also outperformed standard n-gram models even in the much simpler (constrained) task in previous Table 2. Put together, the two experiments seem to suggest that we may ultimately be able to devise (e.g., with some additional training data) a surface realisation module that simply takes as an input a bag of content words with no particular order, and produces as a result a well-formed output NP *with no implemented grammar rules*.

Finally, in the verb-complement agreement task (Table 4), FLMs also showed improvement over standard n-gram models, with no significant difference between models of order 2 and 3. These results are still below those obtained for active voice agreement in [5], suggesting that alternative (factored) models are called for in order to achieve optimal performance in the task.

6 Conclusions

In this paper we have revisited a number of surface realisation subtasks in highly-inflected language generation. Building upon previous work, we presented a series of experiments involving the use of simple factored language models that take morphological information into account as an alternative to standard n-gram counts in a 2-stage NLG architecture.

The experiments addressed the issue of VP lexicalization, the ordering of noun modifiers and verb-complement agreement. In all cases, our results suggest that FLMs may indeed outperform n-gram models using only a fraction of the training data that would normally be required by the standard n-gram approach in a highly-inflected language.

Our experiments represent a first step towards the design of a trainable surface realisation module for the Brazilian Portuguese language, which should ideally generate text from an input conveying minimal linguistic knowledge, and without the need to implement complex grammar rules (e.g., such as those required to establish the ordering of noun modifiers.)

As future work, we intend to collect a larger data set to build a more robust version of each of our current FLMs, and propose new models (particularly for the lexicalization and verb-complement agreement tasks) based on these insights. Other aspects of the current work that need improvement include a thorough investigation of models that take part-of-speech information into account (which seem particularly relevant to the ordering of modifiers task) and an extension of these to other surface realisation tasks so as to obtain a fully functional system.

Acknowledgments. The authors acknowledge support by FAPESP and the University of São Paulo.

References

1. Gatt, A., Reiter, E.: SimpleNLG: A realization engine for practical applications. In: European Natural Language Generation Workshop, ENLG 2009 (2009)
2. Reiter, E.: An Architecture for Data-to-Text Systems. In: European Natural Language Generation Workshop (ENLG 2007), pp. 97–104 (2007)
3. Langkilde, I.: Forest-based statistical sentence generation. In: Proceedings of ANLP-NAACL 2000, pp. 170–177 (2000)

4. Belz, A.: Automatic Generation of Weather Forecast Texts using Comprehensive Probabilistic Generation-Space Models. Natural Language Engineering 14(4), 431–455 (2008)
5. de Novais, E.M., Dias Tadeu, T., Paraboni, I.: Improved Text Generation Using N-gram Statistics. In: Kuri-Morales, A., Simari, G.R. (eds.) IBERAMIA 2010. LNCS (LNAI), vol. 6433, pp. 316–325. Springer, Heidelberg (2010)
6. Nunes, M.G.V., Vieira, F.M.C., Zavaglia, C., Sossolote, C.R.C., Hernandez, J.: A construcao de um lexico para o portugues do Brasil: licoes aprendidas e perspectivas. II Encontro para o processamento de portugues escrito e Falado, 61–70 (1996)
7. Reiter, E., Sripada, S.: Human Variation and Lexical Choice. Computational Linguistics 28(4) (2002)
8. Bangalore, S., Rambow, O.: Corpus-based lexical choice in natural language generation. In: 38th Meeting of the ACL, Hong Kong, pp. 464–471 (2000)
9. Malouf, R.: The order of prenominal adjectives in natural language generation. In: Proceedings of ACL 2000, Hong Kong (2000)
10. Mitchell, M.: Class-Based Ordering of Prenominal Modifiers. In: Proceedings of the 12th European Workshop on Natural Language Generation, Athens, pp. 50–57 (2009)
11. Bilmes, J., Kirchhoff, K.: Factored Language Models and Generalized Parallel Backoff. In: Proceedings of HLT-NAACL 2003, vol. 2 (2003)
12. NIST: Automatic Evaluation of Machine Translation Quality using n-gram Co-occurrence Statistics (2002), `http://www.nist.gov/speech/tests/mt/doc/ngram-study.pdf`
13. Papineni, S., Roukos, T., Ward, W., Zhu, W.: Bleu: a method for automatic evaluation of machine translation. In: ACL 2002, pp. 311–318 (2002)

Prenominal Modifier Ordering in Bengali Text Generation

Sumit Das, Anupam Basu*, and Sudeshna Sarkar

Indian Institute of Technology, Kharagpur
Department of Computer Science and Engineering
Kharagpur, West Bengal, India – 721302
{sumitdas,anupam,sudeshna}@cse.iitkgp.ernet.in

Abstract. In this paper, we propose a machine learning based approach for ordering adjectival premodifiers of a noun phrase (NP) in Bengali. We propose a novel method to learn the pairwise orders of the modifiers. Using the learned pairwise orders, longer sequences of pronominal modifiers are ordered following a graph based method. The proposed modifier ordering approach is compared with an existing approach using our own dataset. We have achieved approximately 4% increment in the *F-measure* with our approach indicating an overall improvement. The modifier ordering approach proposed here can be implemented in a Bengali text generation system resulting in more fluent and natural output.

1 Introduction

Natural Language Generation (NLG) systems should produce text which is meaning-preserving and fluent. Ordering prenominal modifiers is an important task in text generation, because wrongly ordered modifiers in NPs affect the meaning and fluency of the generated text. In English, prenominal modifiers can occur almost in any order, depending on the context. Some orders are more marked than the others, but strictly speaking none are ungrammatical. For example, for the NP in (1), (1a) is more fluent than the other two orders.

1. (a) *charming young blond* ***lady***
   (b) * *blond young charming* ***lady***[1]
   (c) * *blond charming young* ***lady***

Though there exists some consensus that the prenominal modifier ordering is partly governed by the semantic constraints, but the exact semantic constraints are not known. Few early studies [1,2] manually analyzed small corpora, based on which they placed the modifiers in broad semantic classes. They defined rules to impose order among the modifier classes. The modifiers were ordered according to the order of the classes to which they belong. The recent works on

* Prof. Anupam Basu is the Director (Hon.) of Society for Natural Language Technology Research (SNLTR), Kolkata, West Bengal, India.

[1] The '*' marked NPs are not fluent.

modifier ordering has moved away from the rule based linguistic approaches to machine learning based approaches. Prenominal modifier ordering has applications in several other fields, such as, machine translation, question-answering, text summarization etc.

Previous research on modifier ordering was mostly done for English. Till date no rule-based or machine learning based work has been reported on this topic for Bengali. Modifier ordering is necessary for improving the fluency and naturalness of Bengali text. For example, for the NP in (2), (2a) is most fluent. The other orders sound a bit unreal and better suitable in some specific context.

2. (a) *nAnArakama bichitra sAmudrika* ***prAnI***[2] (many peculiar sea creatures)
   (b) * *bichitra sAmudrika nAnArakama* ***prAnI*** (peculiar sea many creatures)
   (c) * *sAmudrika bichitra nAnArakama* ***prAnI*** (sea peculiar many creatures)

Out of 129808 NPs with adjectival premodifiers, extracted from a 2.21 million token Bengali corpus (taken from CIIL corpus[3]), 4651 (approximately 3.6%) contain multiple premodifiers. Thus, in Bengali text a significant portion of the NPs have multiple premodifiers. Here, we make an assumption that in an NP with multiple adjectival modifiers, all of them modify the head noun. This assumption is valid because an NP with multiple modifiers, all modifying the head noun, is far more common than an NP with multiple modifiers where one of them modifies another.

In this paper, we follow a novel machine learning based approach for carrying out the prenominal modifier ordering in Bengali. For ordering multiple premodifiers in an NP, first the pairwise order of the modifiers are determined empirically from corpus. We propose a three stage process to build a model for pairwise ordering of modifiers. Then the sequential chain of the modifiers is generated from the known pairwise orders by following a graph based method. Our prenominal modifier ordering approach can be implemented in the referring expression generation module or the surface realization component of a Bengali NLG system to improve the quality of the generated text.

The remainder of this paper is organized as follows: In Section 2, we provide a brief overview of the existing works. In Section 3, we explain how we prepared our data set. We discuss a baseline model for modifier ordering in Section 4. Detailed description of our proposed approach is given in Section 5. In Section 6, we discuss the performance evaluation measures and the results obtained. In Section 7, concluding remarks and some future directions are provided.

2 Related Works

The early research works on prenominal modifier ordering were mainly rule based linguistic approaches [3,4,5]. They performed manual analysis of small corpora

[2] In this paper, Bengali graphemes are written using Roman Script in ITRANS notation. They are written in italics font.

[3] A part of the EMILE/CIIL corpus developed at Central Institute of Indian Languages (CIIL), Mysore.

and explored various factors that influence the ordering. They conjectured that ordering of prenominal modifiers is a function of certain semantic constraints relating different aspects of the modifiers to the nouns they modify. They grouped the modifiers into broad semantic classes, such as, color, size, nationality etc. and defined rules for ordering these classes. For example, Goyvaerts [6] proposed the following order for English noun modifiers: $quality \prec size/length/shape \prec old/new/young \prec color \prec nationality \prec style \prec gerund \prec denominal$[4]. There is no general consensus on the different criteria used to arrive at these semantic classes. Furthermore, it is not clear how to map the modifiers to these semantic classes. This justifies the exploration of machine learning based approaches which use a large amount of direct corpus evidence and do not suffer from the drawbacks of rule-based linguistic approaches.

Among the recent computational works, Shaw and Hatzivassiloglou [7] first proposed a corpus-based statistical approach for this problem. Three methods e.g., direct evidence, transitivity and clustering were proposed by them for ordering pair of modifiers in English. Briefly, direct evidence method predicts the modifier ordering from the statistics directly obtained from the corpus. In transitivity method, the modifiers in the unseen pairs are ordered by finding the transitive closures of the prenominal modifiers. In clustering method, an order similarity metric is used to group the modifiers into classes. The modifiers are ordered according to the order of the classes in which they belong. This work can order pair of modifiers only. The highest prediction accuracy of 90.67% was reached when the model was trained and tested on a medical corpus. However, they experienced a large difference in accuracy (54%) when the Wall Street Journal (WSJ) corpus [8] was used. Thus, they concluded that the modifier ordering is domain dependent.

Malouf [9] proposed a number of statistical and machine learning based techniques for ordering modifiers. The reported accuracy in this work ranged from 78.28% to 89.73% when used on 100 million tokens of BNC[5]. He achieved the best result (89.73%) by combining memory-based learning and positional probability. This approach was also limited to pairwise modifier ordering.

Mitchell‘s [10] approach grouped the modifiers into classes based on their positional distribution with the head nouns and the other modifiers. He mentioned a preferable order among the different positional classes proposed. Unlike the previous approaches, this work can order more than two prenominal modifiers. Mitchell evaluated his model in different domains and achieved token precision of 89.57% and type precision of 94.17%. The recall values were much low (63.47% and 58.18%). Overall, the prediction accuracy of his approach was 74.14%.

Dunlop [11] introduced a novel modifier ordering approach adapted from multiple sequence alignment (MSA) techniques which are generally used in computational biology. The approach utilized simple features within the raw text, and did not require any semantic information. The ordering of more than two prenominal modifiers can be predicted by this method. The model was trained and tested

[4] $A \prec B$ stands for “A precedes B ”.

[5] `http://www.natcorp.ox.ac.uk/`

with different corpora. This work reported 88.9% token accuracy, 88.2% type precision, 88.1% type recall and 88.2% type f-measure for pairwise modifier ordering. For longer modifier sequence ordering, the reported token accuracy is 86.7%, token precision is 86.7%, token recall is 86.7% and token f-measure is 86.7%.

3 The Data Set Preparation

We have taken a set of 174814 unlabeled Bengali sentences (approximately 2.21 million tokens) from CIIL corpus to extract the prenominal modifier sequences which are used by the modifier ordering methods discussed here. We have identified the noun phrases (NP) by using a shallow parser for Indian Languages[6]. The word level morphological analysis, parts-of-speech (POS) tags and chunk boundaries with type-casted chunk labels in the SSF[7] structures generated by the shallow parser are used here for modifier sequence extraction. We have extracted the simplex NPs from these NPs by matching them with a set of regular expressions. These regular expressions consist of the syntactic categories and the different forms of a simplex NP in Bengali. A simplex NP is a maximal NP that includes sequence of premodifiers, followed by a head noun. The premodifiers can be cardinal numbers (QC), ordinal numbers (QO), quantifiers (QF), possessive pronouns (PRP), adjectives (JJ) and non-head nouns (NN). The adjectival premodifier sequences followed by a head noun are extracted from the simplex NPs by dropping the QCs, QOs, QFs, PRPs and NNs. After this, the inflected forms of the adjectival modifiers and the head nouns are transformed to their base forms by using the morphological information generated by the shallow parser. This increases the count of the modifiers and the head nouns. There are 3289 modifier sequence types in the 4651 multiple modifier sequences extracted from the corpus. 77.77% of these 3289 sequence types occur only once. Our subsequent analysis operates on this data. We randomly held out 10% of our corpus (approximately 17481 sentences) for test purpose, and used the remaining 90% for training. The modifier sequences extracted from the training corpus are used to train our proposed modifier ordering model. The sequences obtained from the test corpus are used for evaluation.

4 Word Bigram as Baseline Model

Given a number of words, the task of generating the most natural word sequence is a well known problem in statistical language processing. The n-gram language model is the most straightforward and used solution for this type of problems. The prenominal modifier ordering problem can be mapped to this more general word ordering problem. We use a bigram language model as the baseline model for ordering modifiers. To determine the correct order for a sequence of modifiers, we generate all possible orderings and choose the one with the highest

[6] `http://ltrc.iiit.ac.in/analyzer/bengali/`

[7] `http://ltrc.iiit.ac.in/analyzer/bengali/shallow-parser-ben-3.0.fc8/doc/ssf-guide-4oct07.pdf`

probability. We build a backoff word bigram model using the SRILM statistical language model toolkit [12]. The unlabeled training sentences are used to build the bigram model. We evaluate the model by using the sequences, containing two or more premodifiers followed by a head noun, extracted from the test corpus. The result of this experiment is somewhat disappointing. Among the 502 test sequences, the order is correctly predicted for 353. The reason for such poor performance is that the frequencies of the modifier sequences are very low in Bengali text. So, the bigram model is not a reliable solution for this problem.

5 Methodology

Since the word bigram based approach does not perform very well for ordering modifiers, so we choose to pursue a more focused solution for this problem. To order multiple premodifiers in an NP we mainly perform the following two tasks.

1. Build a machine learning based model for pairwise modifier ordering. For ordering the modifiers of the seen modifier pairs, use direct corpus evidences, while for the unseen pairs transitivity and semantic cluster based methods are used to impose order.
2. Determine the correctly ordered chain of multiple premodifiers if that can be generated unambiguously from the known pairwise orders.

Brief description of the techniques used to carry out these tasks are given below.

5.1 Pairwise Ordering

We follow a three stage process to build the pairwise modifier ordering model. The stages are direct evidence, transitivity and semantic clustering. The sequences, containing premodifiers followed by a head-noun, extracted from the training corpus are used to build the model. From these sequences, first we extract only the modifier strings by dropping the trailing head-noun. Then from those modifier sequences we obtain the ordered modifier pairs. For example, for the NP "*nAnArakama bichitra sAmudrika* ***prAnI***", the modifier sequence is "*nAnArakama bichitra sAmudrika*". From this the three ordered modifier pairs ⟨*nAnArakama, bichitra*⟩, ⟨*nAnArakama, sAmudrika*⟩ and ⟨*bichitra, sAmudrika*⟩ are generated. An NP with n premodifiers has ${}^{n}C_2$ ordered modifier pairs. From these ordered modifier pairs we construct a $m \times m$ matrix $ModPair$, where m is the number of distinct modifier types. $ModPair[a, b]$ gives the number of occurrence of the ordered pair $\langle a, b\rangle$ in the training corpus. From 4099 multiple modifier sequences extracted from the training corpus, we produce 4119 ordered modifier pairs. There are 2042 different modifier types in the extracted data. So, from this we can infer that the data is quite sparse. The three stages for building the pairwise modifiers ordering model are described in the following subsections.

Direct Evidence. The simplest strategy followed here for pairwise ordering is direct evidence method proposed by Shaw and Hatzivassiloglou [7]. To order a pair of modifiers e.g., $\{a, b\}$, we rely on the direct corpus evidences. If $ModPair[a, b] > ModPair[b, a]$ then the order is "$a \prec b$". For $ModPair[a, b] < ModPair[b, a]$, the order is "$b \prec a$". If $ModPair[a, b] = ModPair[b, a]$ then no ordering preference can be given.

This method is highly accurate for the seen pairs, but for the unseen pairs it can't give any ordering preferences. Only 0.07% elements of the matrix $ModPair$ have nonzero values. Thus the matrix is too sparse and the overall accuracy of direct evidence is low. Out of 549 ordered modifier pairs extracted from test corpus, 379 are correctly ordered by this method. In order to overcome the sparseness of the data, we need a method which can predict the order of the modifiers of an unseen pairs from the seen modifier pairs.

Transitivity. To overcome the limitation of the direct evidence method, Shaw and Hatzivassiloglou [7] proposed a more generalized approach that orders the modifiers of the unseen pairs too. They compute the transitive closure of the ordering relation "$\prec$". That is, if "$a \prec c$" and "$c \prec b$", then by transitivity "$a \prec b$". For example, the modifiers "*bichitra*" and "*jaiba*" never occur together in the training data. So, they can't be ordered by the direct evidence method. However, the orders "*bichitra* $\prec$ *sAmudrika*" and "*sAmudrika* $\prec$ *jaiba*" are given by the direct evidence method. Therefore, by transitivity we can predict the order "*bichitra* $\prec$ *jaiba*".

The transitivity method can reach to contradictory ordering inferences. For example, along with the above mentioned orders, we found the following ordered modifier pairs in our corpus: ⟨*jaiba, rAsAYanika*⟩, ⟨*rAsAYanika, nAnAbidha*⟩, ⟨*nAnAbidha, sAmudrika*⟩ and ⟨*sAmudrika, khanija*⟩. Using transitivity we can infer the order "*jaiba* $\prec$ *sAmudrika*". This is contradictory to our previous inference. To quantify the relative strengths of these transitive inferences, Shaw and Hatzivassiloglou [7] proposed a graph based solution. A directed graph is formed with the distinct modifier types found in the training corpus. If the pair $\{a, b\}$ occurs n times in the training corpus among which the number of occurrence of the ordered pair $\langle a, b\rangle$ is m, then the directed edge "$a \rightarrow b$" is assigned a weight given by the following formula:

$$-log(1 - \sum_{r=m}^{n} {}^{n}C_{r} \times \frac{1}{2}^{n})$$

Clearly, more the number of occurrences of an ordered pair for a pair of modifiers, greater the weight of corresponding edge in the graph. Now, the pairwise modifier ordering problem is turned into the problem of finding the minimum weight path between the graph nodes. For this, Floyd-Warshall's [13] all pairs shortest path algorithm is used. For the pair $\{a, b\}$, the minimum weight path among a to b and b to a is calculated. The ordering preference for the pair $\{a, b\}$ is whichever is minimum among these two. If they are equal then no ordering inference can be drawn for the pair $\{a, b\}$. Transitivity method fills up the $ModPair$ matrix to 14.26%. Out of 549 test modifier pairs, 405 are correctly ordered by this method.

Semantic Clustering. As discussed in Section 2, earlier linguistic approaches put the modifiers into semantic classes and generalized the modifier ordering task to the ordering of the corresponding semantic classes. They determined features depending on which the modifiers are assigned to different classes. But the major problem with those approaches is that they don't agree on the features and classes used. Here, we propose a computational approach to cluster the modifiers into a number of classes so that semantically related modifiers become the member of the same class. This idea is motivated by the work of automatic adjective clustering by Hatzivassiloglou [14]. After the classes are formed, the pairwise order among the modifier classes are found by using the ordered modifier pairs extracted from the training corpus. Using the semantic clustering method, the ordering of those modifier pairs which could not be ordered either by direct evidence method or by transitivity method is done.

We produce the different modifiers-nouns pairs from the sequences, extracted from the training corpus, consisting of one or more prenominal modifiers followed by a head noun. 119359 such modifier-noun pairs are produced from 115292 such sequences. Using the modifier-noun pairs, we find the distribution of the nouns a modifier modifies. This is based on the expectation that the modifiers describing the same property tend to modify approximately the same set of nouns. Therefore, we infer that the modifiers having similar distribution are semantically similar. Here, we use the cosine vector measure to calculate the similarity between the modifiers. The modifiers are represented as real-valued vectors in a multi-dimensional space. The vector spaces consist of the nouns with which the modifier collocates in the produced modifier-noun pairs. Table 1 gives an example of five nominal modifiers represented as vectors in noun spaces.

Table 1. Modifier Vectors in Noun Spaces

	gA.DI	*rAstA*	*jalasrota*	*chele*	*prastAba*
ba.Da	3	2	0	4	0
ra∼Ngina	1	0	0	0	0
sundara	1	4	0	2	1
tIbra	0	0	3	0	0
bhaYa∼Nkara	0	2	2	0	1

The modifier *ba.Da* occurs thrice prenominally with the noun *gA.DI*, twice with *rAstA*, four times with *chele* and never occurs with *jalasrota* and *prastAba*. For two n-dimensional vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ the cosine similarity is given by the following formula:

$$\cos(\boldsymbol{x},\boldsymbol{y}) = \frac{\boldsymbol{x}\cdot\boldsymbol{y}}{|\boldsymbol{x}||\boldsymbol{y}|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} {x_i}^2 \sum_{i=1}^{n} {y_i}^2}}$$

Using this cosine similarity values the semantic dissimilarity among the modifiers are calculated as follows:

$$dissimilarity(\boldsymbol{x}, \boldsymbol{y}) = 1 - \cos(\boldsymbol{x}, \boldsymbol{y})$$

The 15073 modifiers occurring prenominally in the training corpus are partitioned into disjoined classes so that the modifiers with high dissimilarity values are placed in different classes. We use non-hierarchical K-means algorithm for clustering the modifiers. Here, we used 10 clusters. Table 2 shows some of the adjectival modifiers belonging to these clusters as found by the K-means algorithm.

Table 2. Clusters Containing Semantically Related Modifiers

Cluster	Modifier
1	*ra∼Ngina, kAryapayogI, kAryakara, suDola, tapasbI, raktAkta*
2	*puruShattama, bichchhurita, shaktishAlI, gamanashIla*
3	*sAmudrika, nirdhArita, prAkRRitika, sAmAjika*
4	*pratibhAbAna, parishramI, byAkula*
5	*charama, jamakAlo, nirlipta, chakachake*
6	*natuna, barjita, sAbadhAnI, chalati, purano*
7	*uchchatama, be.NTe, sarbAdhika, janapriYa*
8	*baddha, jIrNa, sarala, gunI*
9	*asAmAjika, bikaTa, bijAtIYa,jAntaba*
10	*jalaja, spaShTa, AbachhA, meghalA*

After the classes of premodifiers are induced, for each pair of classes we decide a pairwise order. For two classes C_i and C_j, we extract all pairs of premodifiers $\{x, y\}$ with $x \in C_i$ and $y \in C_j$. If we have evidence (either by direct evidence or by transitivity) that $x \prec y$, then one point is added in favor of $C_i \prec C_j$; whereas, one point is added in favor of $C_j \prec C_i$ if $y \prec x$. Once all such modifier pairs are considered, the pairwise order of the pair $\{C_i, C_j\}$ is the one for which the calculated score is greater. Now the order of an unseen modifier pair $\{a, b\}$, where $a \in C_i$ and $b \in C_j$, is the order of the class pair $\{C_i, C_j\}$. The drawback of this method is that if no order can be defined for the class pair $\{C_i, C_j\}$ or the class of any of the modifiers a or b is not found then the order of the pair $\{a, b\}$ cannot be inferred. Out of 549 test modifier pairs 440 are correctly ordered by the semantic clustering method when combined with direct evidence and transitivity methods.

5.2 Sequence Ordering

Our goal is to correctly order multiple premodifiers of a given NP. For the NPs with two modifiers, this is done by using the pairwise modifier ordering model. But for the NPs with more than two premodifiers, we need to extend the pairwise ordering method. For the NPs with more than two premodifiers, we generate all possible modifier pairs. The preferred orders of those modifier pairs are generated using the pairwise ordering method. Some or all of the pairwise orders are known by this. Now, using the known pairwise orders we try to build an ordered modifier chain. If that can be generated without ambiguity then the generated chain of modifiers is the correct order of the modifiers for the given NP. For example, in the NP "*nAnArakama bichitra sAmudrika* ***prAnI***", there are three modifier pairs, e.g., $\{nAnArakama, bichitra\}$, $\{bichitra, sAmudrika\}$, and $\{nAnArakama, sAmudrika\}$. If the pairwise ordering model gives the ordering preferences as "$nAnArakama \prec bichitra$", "$bichitra \prec sAmudrika$", and "$sAmudrika \prec nAnArakama$" then no ordered modifier chain is possible. So, the premodifiers of this NP cannot be ordered correctly. On the other hand, if the pairwise orders, known from the pairwise ordering model, are "$nAnArakama \prec bichitra$" and "$bichitra \prec sAmudrika$" then ordered chain can be formed. The order of the prenominal modifiers is "$nAnArakama \prec bichitra \prec sAmudrika$".

To generate the ordered chain from known pairwise orders, we use the idea of topological sort of the nodes of a directed graph. For a given NP with multiple premodifiers, the nodes of the graph are the premodifiers; the directed edges are along the known pairwise modifier orders. The algorithm for generating the ordered modifier chain is given in Algorithm 1.

Algorithm 1. Algorithm for generating ordered chain of modifiers

```
Input: The directed graph formed by the modifier of an NP
Output: Ordered chain of modifiers if a chain can be formed
$L$ ← Empty list that will contain the ordered modifier nodes;
$S$ ← Set of all modifier nodes with indegree 0;
while $S$ contains only 1 node do
    remove a node $n$ from $S$;
    insert $n$ in $L$;
    foreach node $m$ with an edge $e$ from $n$ to $m$ do
        remove edge $e$ from graph;
        if $m$ has no other incoming edges then
            insert $m$ into $S$;
        end
    end
end
if graph has edges or $L$ does not contain all the graph nodes then
    no ordered chain of modifiers is possible;
end
else
    get the modifiers from $L$ sequentially generating the ordered modifier chain;
end
```

6 Evaluation

We use *precision*, *recall* and *F-measure* as the evaluation metrics for our prenominal modifier ordering approach. These are widely used evaluation measures in NLP and IR research. If X is the set modifier sequences found in the test corpus and Y is the set of modifier sequences generated by the system when used on the test modifier sequences, then *precision*, *recall* and *F-measure* are defined as follows:

$$Precision = \frac{X \cap Y}{Y} \qquad Recall = \frac{X \cap Y}{X}$$

$$F\text{-}measure = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

We use 10 fold cross-validation to separate out the test corpus from the training corpus. The modifier sequences, containing multiple modifiers, are extracted from the test corpus to evaluate the prenominal modifier ordering approach proposed here. 502 such test modifier sequences are obtained. As discussed in Section 3, during the preparation of data we transformed the modifier tokens to their corresponding types, using morphological analyzer. So the calculated *precision*, *recall* and *F-measure* values are for the modifier types.

The baseline model, discussed in Section 4, predicts the ordering of 361 test modifier sequences among which 353 are correct. Our method combines the power of transitive inference and clustering with the direct corpus evidence. Due to these, greater number of test sequences is ordered correctly by our approach. Among the 442 test sequences, for which our modifier ordering model predicts some order, 415 are correct. Table 3 compares the *precision*, *recall* and *F-measure* of the baseline model and our approach. The *precision* of our approach is less than that of baseline model. But, as our approach correctly orders much greater number of test modifier sequences, the *recall* and *F-measure* values are higher for it.

Table 3. Comparison of *Precision*, *Recall* and *F-measure* of Our Approach with the Baseline Model

	Precision	*Recall*	*F-measure*
Baseline	97.78%	70.32%	81.81%
Our Approach	93.89%	82.69%	87.93%

All the previous works on prenominal modifier ordering evaluated their systems for pair of modifiers. In the same line, we also test our approach on modifier pairs. From 502 test modifier sequences 549 modifier pairs are generated which are used to evaluate the pairwise modifier ordering model. We evaluate our approach for ordering more than two prenominal modifiers too. To our knowledge, only Mitchell [10] and Dunlop [11] tested their systems for more than two modifiers.

The existing prenominal modifier ordering methods, tested for English, can be implemented for Bengali. To test how effective are they for Bengali, we implement the Mitchell's approach. We train and test it with our data set. Four prenominal positions and nine positional classes for the Bengali modifiers are considered in building this model. Mitchell's approach clusters the modifiers based on their positional distribution with the head nouns. But in Bengali, the word order is not so rigid. So, the clustering the noun modifiers based on their positional distribution is not so effective for prenominal modifier ordering in Bengali. Table 4 compares the *precision*, *recall* and *F-measure* values of our model with that of Mitchell's model when evaluated on our data set. The results for pairwise ordering as well as for sequence ordering are shown. Both the *precision* values of our approach is bit less than those of Mitchell's approach. But, the *recall* values are much higher for our approach. The *F-measure* values for our approach also show improvements over the *F-measure* values of Mitchell's approach.

Table 4. Comparison of *Precision*, *Recall* and *F-measure* of Our Approach with the Mitchell's Approach

		Mitchell	**Our Approach**
Pairwise Ordering	*Precision*	95.79%	92.55%
	Recall	74.68%	80.14%
	F-measure	83.93%	85.9%
Sequence Ordering	*Precision*	94.64%	93.89%
	Recall	73.9%	82.69%
	F-measure	83.99%	87.93%

7 Conclusions

Here, we have discussed an approach to automatically order multiple premodifiers of the NPs in Bengali. To order multiple modifiers, our approach first learns the pairwise modifier orders through three stages: direct evidence, transitivity and semantic clustering. Then, by using the pairwise orders, longer prenominal modifier sequences are generated. We tested an existing modifier ordering approach on our data set. Our modifier ordering approach shows considerable improvements over that existing approach.

The approach described here, builds a machine learning based model for ordering prenominal modifiers. It orders the modifiers of an NP as they appear in the training corpus. Since Bengali is a free-word-order language, the other modifier orders which do not appear in the training corpus may also sound fluent and natural. Thus, for ordering of prenominal modifiers in Bengali, linguistic properties of the modifiers can be used. The ontological knowledge of the noun modifiers in Bengali can be used for better ordering of the prenominal modifiers. In the future, we will train and test our modifier ordering model with Bengali corpus in different domains to test its domain dependency. We will implement our approach in a Bengali text generation system to test the improvements in fluency of the generated text.

Acknowledgments

We would like to thank anonymous reviewers for their valuable comments. We would also like to thank Mr. Plaban Kumar Bhowmik, Mr. Tirthankar Dasgupta and Mr. Sanjay Chatterjee for their valuable advice and support. This work is supported by the project Sanyog - Phase II, funded by Media Lab Asia and Society for Natural Language Technology Research (SNLTR), and conducted in Communication Empowerment Laboratory, Indian Institute of Technology, Kharagpur.

References

1. Danks, J.H., Glucksberg, S.: Psychological scaling of adjective orders. Journal of Verbal Learning and Verbal Behavior 10, 63–67 (1971)
2. Martin, J.: Semantic determinants of preferred adjective order. Journal of Verbal Learning and Verbal Behavior 8, 697–704 (1969)
3. Danks, J.H., Schwenk, M.A.: Prenominal adjective order and communication context. Journal of Verbal Learning and Verbal Behavior 11, 183–187 (1972)
4. Martin, J.: Adjective order and juncture. Journal of Verbal Learning and Verbal Behavior 9, 379–383 (1970)
5. Bacharach, V.R., Maisto, A.A.: Prenominal adjective order and visual discrimination in children. Journal of Experimental Child Psychology 17, 495–506 (1974)
6. Goyvaerts, D.L.: An introductory study on the ordering of a string of adjectives in present-day english. Philologica Pragensia 11, 12–28 (1968)
7. Shaw, J., Hatzivassiloglou, V.: Ordering among premodifiers. In: Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics, pp. 135–143. Association for Computational Linguistics, Morristown (1999)
8. Marcus, M.P., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of english: The penn treebank. Computational Linguistics 19, 313–330 (1993)
9. Malouf, R.: The order of prenominal adjectives in natural language generation. In: ACL 2000: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, pp. 85–92. Association for Computational Linguistics, Morristown (2000)
10. Mitchell, M.: Class-based ordering of prenominal modifiers. In: ENLG 2009: Proceedings of the 12th European Workshop on Natural Language Generation, pp. 50–57. Association for Computational Linguistics, Morristown (2009)
11. Dunlop, A., Mitchell, M., Roark, B.: Prenominal modifier ordering via multiple sequence alignment. In: HLT 2010: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 600–608. Association for Computational Linguistics, Morristown (2010)
12. Stolcke, A.: Srilm - an extensible language modeling toolkit, pp. 901–904 (2002)
13. Aho, A.V., Hopcroft, J.E., Ullman, J.D.: The Design and Analysis of Computer Algorithms. Addison-Wesley Publishing Company, Reading (1974)
14. Hatzivassiloglou, V., Mckeown, K.R.: Towards the automatic identification of adjectival scales: Clustering adjectives according to meaning. In: Proceedings of the 31st Annual Meeting of the ACL, pp. 172–182 (1993)

Bootstrapping Multiple-Choice Tests with THE-MENTOR

Ana Cristina Mendes, Sérgio Curto, and Luísa Coheur

Spoken Language Systems Laboratory - L^{2}F/INESC-ID
Instituto Superior Técnico, Technical University of Lisbon
R. Alves Redol, 9 - 2°– 1000-029 Lisboa, Portugal
{ana.mendes,sergio.curto,luisa.coheur}@l2f.inesc-id.pt

Abstract. It is very likely that, at least once in their lifetime, everyone has answered a multiple-choice test. Multiple-choice tests are considered an effective technique for knowledge assessment, requiring a short response time and with the possibility of covering a broad set of topics. Nevertheless, when it comes to their creation, it can be a time-consuming and labour-intensive task. Here, the generation of multiple-choice tests aided by computer can reduce these drawbacks: to the human assessor is attributed the final task of approving or rejecting the generated test items, depending on their quality.

In this paper we present THE-MENTOR, a system that employs a fully automatic approach to generate multiple-choice tests. In a first offline step, a set of lexico-syntactic patterns are bootstrapped by using several question/answer seed pairs and leveraging the redundancy of the Web. Afterwards, in an online step, the patterns are used to select sentences in a text document from which answers can be extracted and the respective questions built. In the end, several filters are applied to discard low quality items and distractors are named entities that comply with the question category, extracted from the same text.

1 Introduction

Multiple-choice tests are an effective technique for knowledge assessment, requiring a short response time and with the possibility of covering a broad set of topics. Typically, these tests consist in a number of test items, each composed by two parts: a question and a group of suggested answers. Respondents are supposed to identify the correct answer among the incorrect ones (called distractors). The following is an example of a multiple-choice test item with one correct answer and two distractors:

Q. *"What is the largest ocean?"*

1. *Atlantic* `(distractor)`
2. *Pacific* `(correct answer)`
3. *Indian* `(distractor)`

The manual creation of multiple-choice test items is a time consuming trial and error process; in this context, computer aided multiple-choice tests generation can help reducing the amount of time allocated to this task.

In this paper, we hypothesize that the process of generating multiple-choice tests with only one correct answer per question can rely on the bootstrap of a set of question/answer (Q/A) seed pairs. Here, we describe our approach to automatically generate multiple-choice tests and present THE-MENTOR, a system that generates multi-choice tests about a free text document. THE-MENTOR is composed by two main components which perform the following tasks:

Learning lexico-syntactic patterns – A set of Q/A seeds is used to bootstrap patterns that relate questions with answers. Patterns are learned from the Web, and we exploit its redundancy to create plausible patterns. Moreover, we perform verb expansion and allow several types of patterns, according to the precision of the match against the original seeds. The decision of accepting different types of patterns resulted from the fact that, if patterns are too specific (`strong` patterns), they will not frequently match and not many tests will be generated; if patterns are too generic (`weak` patterns), the quality of the generated tests decreases.

Generation of test items – The retrieved patterns extract sentences where answers can be found and from which the respective questions can be built. In order to discard low quality items, several filters are applied. Distractors are named entities that comply with the question category, extracted from the same text.

Afterwards, the user can evaluate the quality of the generated test items through a web interface. The multiple-choice test will be composed by the test items the user considered as having quality.

This paper is organized as follows: in Section 2 we describe the pattern learning task; in Section 3 we describe how to generate multiple-choice test items. In Section 4 we show the evaluation results; in Section 5 we present related work; in Section 6 we present our conclusions and point to future work directions.

2 Pattern Learning

The first task in the generation of multiple-choice test items by THE-MENTOR has to do with learning lexico-syntactic patterns and is performed off-line, that is, before the user specifies the text document based on which the test will be generated.

The algorithm for pattern learning is based on the bootstrapping technique presented in [1], and involves the following two stages. First, we use a seed pair – composed by a natural language question and its correct answer – to bootstrap patterns that relate questions and answers. We call B-PATTERNS to the patterns extracted from the bootstrap process. Second, the B-PATTERNS are validated using a validation pair – also a natural language question and its correct answer – as input. The validation pair will help to remove those patterns that are too specific to the seed pair. Therefore, each seed pair has a validation pair associated. These pairs are automatically grouped, knowing that although their constituents are lexically distinct, they must share the same syntactic structure as well as the same category. For example, the seed question *"Who is the President of France?"* cannot be validated by the question *"What is the capital of France?"* since, although both share the same syntactic structure (*WHNP VBZ NP*), they have a different focus: the former searches for the name of an individual (HUMAN:INDIVIDUAL) and the later for the name of a city (LOCATION:CITY).

The syntactic analysis of questions is made using the Berkeley Parser [2] trained on the QuestionBank [3], a treebank of 4,000 parse-annotated questions. In what concerns the classification of questions, we use a machine learning-based classifier fed with features derived from a rule-based classifier [4]. Regarding the question categories, we use Li and Roth's two-layer taxonomy [5], consisting of a set of six coarse-grained categories and fifty fine-grained ones.

Finding patterns. The algorithm to find B-PATTERNS starts by generating permutations of a set comprising the seed answer, the phrasal nodes of the seed question (excluding the *Wh*-phrase), and a wildcard * which stands as a placeholder for one or more words and adds diversity into the generated patterns. For instance, considering the question *"Who painted the Birth of Venus ?"* and the sentence *Botticelli has painted the Birth of Venus*, a wildcard is required to match the verb *has*. Since we do not allow the wildcard to be the first or the last element in the query, the total number of permutations is $n! - 2(n-1)!$, in which n is the number of elements to be permuted. In addition, the reason why we use phrasal nodes instead of question tokens as it is done in [1], is because they represent a single unit of meaning, and therefore should not be broken down into parts (except for verb phrases). For example, considering the previous question *"Who painted the Birth of Venus ?"*. it does not make sense to divide the noun-phrase *the Birth of Venus*, since it would generate several meaningless permutations, like *Birth the painted Botticelli * of Venus*.

After the permutations have been created, each is enclosed in double quotes and sent to Google search[1]. The double quotes ensure that each search result contains the exact quoted permutation, with every word in the exact same order in which it appears in the original query. The snippets retrieved from the search engine are then broken down into sentences, and if there exists a sentence that matches the respective permutation, we rewrite it as a pattern. Consider again the question "[$_{\text{WHNP}}$ Who] [$_{\text{VBD}}$ painted] [$_{\text{NP}}$ the Birth of Venus]", and suppose a sentence *Botticelli has painted the Birth of Venus* that matches the permutation *Botticelli * painted the Birth of Venus*. The resulting pattern would then be "{*ANSWER*} has *VBD NP*", created by replacing each phrasal node with the respective tag, and the seed answer with the tag "{ANSWER}".

Pattern Validation. While many of the learned patterns are generic enough to be applied to other questions, there are others specific to the seed pair. For instance, the pattern "*NP* was *VBD* around 1486 by {*ANSWER*}", extracted from the sentence *The Birth of Venus was painted around 1486 by Botticelli.* Since this pattern only works for the seed question (and possibly for a small number of works of art of that same year), it should be filtered-out. To eliminate these elements, we use a different algorithm which requires the use of a validation pair.

The validation algorithm works by testing each generated pattern against the validation pair, and calculating its precision. The precision is considered to be the ratio between the number of times the pattern matched the retrieved snippets and the

[1] In this work we use Google as the search engine. However, there is no technical reason that prohibits this system to use another search engine.

number of times the pattern was expected to match (that is, the maximum number of snippets retrieved by the search engine). For example, using the aforementioned pattern and the validation pair *"Who painted Guernica?"/Picasso*, the query *Guernica was painted around 1486 by Picasso* would be issued, resulting in zero results – and thus zero precision. This would cause the pattern to be ruled-out, as the algorithm dictates that each pattern must have precision larger than a threshold in order to be retained.

2.1 Handling Verbs

Our method for learning patterns implies that every phrase (except the *Wh*-phrase) must be present in the B-PATTERNS. From now on, we call `strong` patterns to the B-PATTERNS that contain all phrases (and their contents) of the seed question. Whereas this is the expected behaviour for Noun- and Prepositional-phrases that should be stated *ipsis verbis* in the sentence fragments that will generate the patterns as they are in the seed question, the same does not apply for Verb-phrases. The pattern generator should be flexible enough to capture a pattern in the sentence *Botticelli finished painting The Birth of Venus in 1485.*, even if the surface word that corresponds to the verb is not the same as the one present in *"Who painted the Birth of Venus?"*.

Being so, we allow the verbs in the pattern to be in a different inflexion than the main verb in the seed question. Moreover, in case the seed question has an auxiliary verb, the sentence fragment does not need to contain it, since these are most probable to appear in interrogative sentences than on declaratives ones. To create these patterns, we pick the main verb of the question and conjugate it in its multiple inflexions. Afterwards, a new query is sent to the search engine with the several inflexions, and without the presence of the auxiliary verb (if it exists in the question).

The B-PATTERNS generated by verb inflection are named `inflected` patterns.

2.2 Allowing Weak Patterns

There are, however, some patterns that should not be disregarded, even if they do not contain all question phrases and cannot be handled by allowing the multiple inflexion of verbs. These patterns arise from sentence that, despite not completely rephrasing the question, capture the existing relation between it and the answer. For instance, the pattern "*NP*, by $\{ANSWER\}$" should be recovered from the sentence *The Birth of Venus, by Botticelli*, even if it does not include the verb (in this case, *painted*). These patterns are different from the `strong` and `inflected` patterns, not only because of how they were created, but also because they will trigger distinct strategies in the test item generation. To generate this type of patterns (called `weak` patterns), the procedure is similar as referred, just we do not allow the Verb Phrases in the question to be present on the permutations.

Table 1 shows a set of patterns[2] generated from, and validated with, the snippets retrieved by the search engine, for questions with flatten syntactic structure *WHNP VBD NP* and category HUMAN:INDIVIDUAL.

[2] Here, as well as throughout the entire paper, the Penn Treebank II Tags [6] are used.

Table 1. Example of extracted patterns with respective precision and type

Question	HUMAN:INDIVIDUAL-*WHNP VBD NP*	
Precision	**Pattern**	**Type**
0.625	{*ANSWER*}'s *NP*	`W`
0.25	{*ANSWER*} began *VBG* NP	`I`
0.625	*NP VBD* by {*ANSWER*}	`S`

3 Building Mutiple-Choice Tests

The next task in the generation of multiple-choice test items by THE-MENTOR is done online. The user specifies the target documents and the system parses the text and applies the learned patterns in order to obtain Q/A pairs (as well as distractors). This method also involves several strategies for filtering the obtained Q/A pairs in order to discard low quality pairs.

3.1 Extracting Question/Answer Pairs

Our algorithm for extracting Q/A pairs relies on matching lexico-syntactic information from the B-PATTERNS against the parsed sentences of the target document. Each match is done at two levels in the sentence parse tree: at the word level, since most of the patterns include tokens to separate the syntactic components, and at the syntactic level. For that purpose, we have developed a tree matching algorithm (out of the scope of this paper) to find all the occurrences of a given pattern on the syntactic tree of the parsed sentences.

After the extraction of the sentence fragments where questions and their respective answers are stated, we apply a set of filters to refine the proceeding generation of multiple-choice test items, namely:

Forcing Question/Answer Category Matching. The extracted fragments in which the answer does not comply with the expected category can be discarded. Consider, for example, the Q/A pair: *"Who was François Rabelais?"-An important 16th century writer*. Here, the answer agrees with the semantic class expected by the question (HUMAN:INDIVIDUAL), indicated by the word *writer*.
Thus, we test the answer in order to check if at least one of its words belongs to the question category. By using WordNet's lexical hierarchy, a word is associated with a higher-level semantic concept, which represents itself a question category. To do so, we have manually grouped a set of WordNet synsets into fifty clusters, each representing a question category. For example the category HUMAN:INDIVIDUAL is related with the synsets `person`, `individual`, `someone`, `somebody` and `mortal`. The words *actor*, *leader* and *writer* are hyponyms of (at least) one of these synsets. Since WordNet can be seen as a directed acyclic graph, with synsets as vertices and lexical relations – such as hypernym – as edges, we employ a breadth-first search on the translated synset's hypernym tree, in order to find a synset that pertains to any of the pre-defined clusters.

Discarding Anaphoric References. A group of simple regular expressions is used to invalidate questions that contain anaphoric references and others, which we empirically know that will not result in quality multiple-choice test items. Thus, questions like *"What is it?"*, *"Where is there?"* or *"What is one?"* are discarded.

3.2 Generation of Test Items

Given that we successfully discover and extract fragments in the target document that match B-PATTERNS, the generation of multiple-choice test items is straightforward and performed according to the type of the fragment extracted from the sentence (`strong`, `inflected` and `weak`). Since we keep track of both the set of questions (that share the syntactic form) for which we discovered the patterns, and the sentences that generated them, the generation goes as follows: both `strong` and `inflected` patterns result in a direct unification of all extracted fragment components with the B-PATTERNS components. However, within `inflected` patterns, the verb is inflected with the tense and person existing in the question and the auxiliary in the question is also used. In what concerns the `weak` patterns, we perform the unification of the fragment components with the respective pattern components, and for all the components that do not appear in the fragment, the components in the question are used.

Regarding the generation of distractors, we search in the text for named entities whose type agree with the category of the question. For that purpose, and to take advantage of the rich taxonomy of question categories utilized, we developed and use several strategies to recognize named entities from texts. These strategies include the usage of regular expressions (to extract numerical entities), gazetteers (to extract locations) and a machine learning-based recognizer (for persons, locations, organizations and others). We choose the named entities nearer to the sentence that originated the Q/A pair, however not in the same sentence. As an example, if we consider the sentence *This resource briefly explores the telegraph invented by Samuel Morse.*, that originates the Q/A pair *"Who invented the telegraph?"-Samuel Morse*, since its category is HUMAN:INDIVIDUAL, we will search and use as distractors the named entities of type PERSON in the nearer sentences.

4 Experiments

Our approach takes as input natural language questions and their correct answers. In our experiments, we used 139 natural language Q/A pairs, some taken from an on-line trivia, others manually created. All questions are factoids pertaining to 10 categories – ENTITY:CURRENCY (5 Q/A pairs), ENTITY:SPORT (3), ENTITY:LANGUAGE (4), HUMAN:INDIVIDUAL (28), LOCATION:CITY (11), LOCATION:COUNTRY (24), LOCATION:MOUNTAIN (2), LOCATION:OTHER (27), LOCATION:STATE (12) and NUMERIC:DATE (23). To allow comparisons with other systems, the used Q/A pairs are available in `http://qa.l2f.inesc-id.pt/wiki/index.php/Resources`

The pairs were automatically grouped (according to their category and syntactic structure), in order to create the seed/validation pairs. This step resulted in a set of

Table 2. Example of seed-validation pairs

Group – HUMAN:INDIVIDUAL-`WHNP VBD NP`
S *"Who wrote Odyssey?"/Homer*
V *"Who painted Guernica ?"/Picasso*
Group – LOCATION:COUNTRY-`WHPP VBD NP VBN`
S *"In which country was Bjorn Borg born?"/Sweden*
V *"In which country was the match invented?"/France*

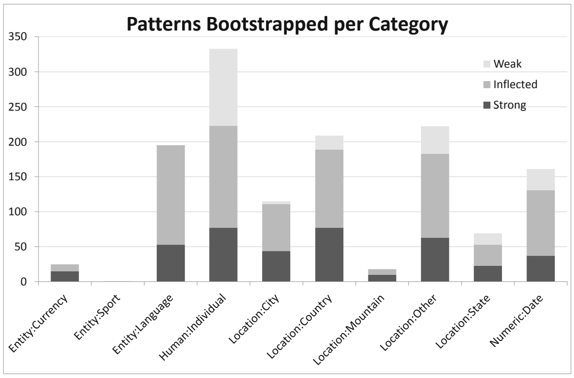


Fig. 1. Distribution of B-PATTERNS per category

668 seed/validation pairs, which, along with the different syntactic structures and categories, led to a total of 20 groups. Examples of seed/validation pairs are presented in Table 2, with a reference to the group to which they belong.

The 16 top ranked snippets retrieved by the web search engine Google were used to learn the patterns, according to the process described in Section 2. The learning task resulted in 1348 B-PATTERNS, from which 1126 were unique.

The distribution of patterns according to the category is presented in Figure 1. We noticed that the more seed/validation pairs exist, the more B-PATTERNS were bootstrapped by category. An exception to this was the category ENTITY:LANGUAGE, for which a small number of pairs (12) gave rise to a large number of patterns (all belonging to the type `strong` and `inflected`). The highest number of patterns were discovered for the category HUMAN:INDIVIDUAL, which was the category with more seed/validation pairs. Moreover, a great share of patterns of this category are of type `weak`: almost one third. The ratio between `weak` patterns and the total amount of patterns is less for the other categories: in five categories this ratio lower then 5%.

To evaluate the generated test items we used a similar model of test item review to that of [7]. If an item makes no sense (like *"Who left alone much?"-the new British rulers*) it is *discarded*; otherwise, it is marked as *worthy*. A *worthy* item is evaluated according to the degree of review needed, and classified as: **minor**, if it requires up to minimal corrections (like article introduction or spelling corrections), for example

Table 3. Number of generated question and answer pairs for each pattern type

Type	Degree of Review			Discarded	Total
	Min.	*Mod.*	*Maj.*		
`S`	14	26	11	8	**59**
`I`	2	7	8	28	**45**
`W`	0	2	55	53	**110**
Total	16	35	74	89	**214**

"Who was François Rabelais?"-An important 16th century writer; **moderate**, if it requires the removal/insertion or reordering of words, or if a set of distractors is not applicable, for example, in: *"Who was Eugène Viollet-le-Duc?"-the associated architect* the words "*the associated*" should be removed; **major**, if it requires a deep grammatical correction, for example in: *"Who became Philip I?"-the Spanish king* the question should be reformulated to *"What did Philip I become?"*.

4.1 Evaluation

We used the Wikipedia article about the "History of Portugal" [3] as target document in the evaluation of our approach for the generation of the test items.

Results for each type of pattern – `strong`, `inflected` and `weak` – are shown in Table 3.

A total of 806 sentences from the referred article tried to match with every of the learned patterns. Considering the type of the involved patterns, 59 were `strong`, 45 `inflected` and 110 `weak` patterns extracted Q/A pairs. Although more patterns were activated, they did not pass the filtering phase.

As expected, most of the Q/A pairs extracted from the `strong` patterns generated items considered as *worthy*. This type of patterns generated the pairs that needed small or no revision. However, a tendency exists for augmenting the degree of review needed when lowering the constraints imposed by the patterns (measured by the existence of the question components in the pattern). The extracted question/answer pairs are distributed through six of the aforementioned categories:

– *Human:Individual*: generated the highest number of pairs (138), mostly using `strong` patterns. Greatly contributed to the total of minor revision items (11 of 14);
– *Location:City*: generated the second highest number of pairs: 27, in which 13 were *worthy*;
– *Location:Country*: generated only eight pairs, only two considered *worthy*;
– *Location:Other*: generated 18 pairs. Since this category is more generic that the previous two *Locations*, it could generate better test items than the others (for instance, a Q/A pair *"Where is Lisbon?"-Europe* is allowed, but not *"In which country is Lisbon?"-Europe*);
– *Location:State*: generated seven pairs. All of them were discarded.
– *Numeric:Date*: generated 13 pairs, three of which needing minor revisions.

[3] `http://en.wikipedia.org/wiki/History_of_Portugal`

Although no patterns of the other four categories matched, this result was somewhat expected: firstly, due to the nature of the document target in use: an article about the history of a country; secondly, the patterns belonging to these categories existed in higher number than the others (an exception being the category ENTITY:LANGUAGE).

The B-PATTERN that generated more patterns was the `weak` pattern "*NP with* {*ANSWER*}", however for the 72 generated, all of them are either discarded, or need major revision. The one that was most successful, with a higher number of *worthy* generated items with minor revision when compared to the total number of generated patterns (11 in 42), was "{*ANSWER*} *VBD NP*", both from the category HUMAN:INDIVIDUAL.

Concerning the distractors, mostly they were appropriate to the generated test item. Moreover, and since they are in agreement with the question, if its *Wh*-phrase has to be reviewed/replaced, the distractor will probably have to be changed too.

4.2 Discussion

The approach used within THE-MENTOR receives as input a set of natural language Q/A pairs and generates test items in order to create multiple-choice tests. With this approach, several test items were generated automatically that can help the creation of multiple-choice tests, originated from a small set of seed pairs. These seeds can be easily found and built, for instance, using the test sets made available in evaluation campaigns for QA systems (like TREC or CLEF).

As results suggested, there is a relation between the types of the B-PATTERNS and the test items they generate: STRONG patterns generated better Q/A pairs, however in a lower number, and WEAK patterns generated Q/A pairs with lower quality, but still most of them can be used after some revision. However, a similar relation could not be spotted for the category type and the generated test items. It was anticipated that the WEAK patterns would lead to the worse results, however we consider that they are able to capture important information. We believe that their posterior generation into Q/A pairs and the automatic filtering phase should be improved.

Also, our approach relies in the lexico-syntactic information stated on the patterns. Even if with this we are neglecting information that could be valuable in the matching of sentences and generation of the tests, for instance semantic information, we could still generate a large set of test items, most of which can be used.

5 Related Work

There are not many examples in the literature of systems that focus on the generation of multiple-choice tests. An exception is the computer-aided environment for generating multiple-choice test items, described in [8]. Authors present a system that relies heavily on natural language processing techniques and resources, built on the notion of key-terms (terms about which the test items should be generated). The system performs three main tasks: it starts by identifying and extracting the key-terms from the source corpora, by using regular expressions that match nouns and noun-phrases; afterwards, question generation rules are applied only to sentences of SV(O) structure and the generated questions filtered to assure grammatical correctness; lastly, concepts semantically related with the answer are retrieved from the WordNet [9]. After the generation, a *post-editing* phase exists in which the test items are revised by human assessors. This system was later adapted to the medical domain [10].

Authors [11] and [12] describe two other systems for multiple-choice test generation. However, the type of questions they output are different from the ones of the aforementioned system and from our work: the fill-in-the-blank (or cloze) questions, are built with blank spaces to be filled by the appropriate option. The first system, called WebExperimenter, obtains distractors from several sources/techniques such as WordNet, edit distance or mutual information, and uses a machine learning classifier to decide the correct position of the blank in the question. WebExperimenter was later adapted to assist the learning of English as a second language [13]. The second system was originally built with the purpose of measuring the English proficiency of non-native speakers, and works by selecting and replacing a word (authors focused uniquely on verbs) in a correct English sentence with a blank. Distractors are chosen in order to maintain the same characteristics of the correct choice, and picked from a thesaurus. The correctness of each distractor is assessed through a web-based verification: if the sentence restored from the blanked sentence and the distractor exists in the web, the distractor is assumed to be correct. Following this line, several systems, like REAP [14] and FAST [15], put their efforts in the improvement of cloze questions.

Although the literature in multiple-choice test generation is not extensive, this task can easily borrow and adapt techniques employed in Question-Answering (QA). These have influenced our work, hence, here we briefly describe some of these systems.

A good parcel of the research in question answering relies on the usage of patterns, namely to bridge the gap between the question and the sentence in which the answer can be found. The main idea is that the answer to a given question will probably occur in sentences that contain a rewrite of the original question. For example, given the question *"Who painted the Birth of Venus?"*, a possible rewrite is *painted the Birth of Venus*, which is very likely to appear after the answer *Botticelli*. However, there is also a strong possibility that there are words separating the rewrite and the answer. If we find these sequences, we are able to create patterns that will allow us to find the answers to similar questions. For instance, *ANSWER, who REWRITE* is a pattern that can be extracted from the sentence *Botticelli, who painted the Birth of Venus*.

In the QA track of the TREC-10, the winning system – described in [16] – presents an extensive list of surface patterns and draws the attention of the community to the potential of this technique. Posterior work of [1] details a pattern-learning algorithm, that can be summarized in the following: first, a question (part of it) and its answer are submitted to Altavista; second, the 1000 top documents are downloaded and those containing both the answer and the question are retained; finally, the longest matching substrings are extracted and the question and the answer are replaced by tokens `<QUESTION>` and `<ANSWER>`. Our pattern-learning algorithm is similar to this, although we accept patterns that do not match both the question and the answer. Moreover, our patterns are syntactically-based. Somehow related with the work of [1] is the work of [17] and [18]. The former searches for possible answers in snippets by analysing substrings that have similar contexts of already known answers and uses genetic algorithms in the process;

the later bases the performance of the QA system AskMSR on manually created rewrite rules which are likely substrings of declarative answers to questions. The authors also felt the need to produce less precise rewrites, since the correct ones did not match any document. On the Dutch language, [19] explores the question rewriting process for questions that have as answer type *person* and *location*. The authors use syntactic information in the question analysis, but the rewrite rules are hand built, like in [20], which presents an extensive list of regular expressions.

6 Conclusions and Future Work

Here we presented THE-MENTOR, a system that automatically generates multiple-choice test items, composed by a question, a correct answer and a set of distractors. First, it exploits the redundancy of large corpora sources to bootstrap frequent patterns. Each pattern is assumed to bridge the gap between a question and its answer. Afterwards, given a target document, it extracts question/answer pairs from the sentences that match the patterns, as well as the distractors in their surroundings, and builds test items.

By using mainly syntactic information complemented by verb conjugation and WordNet information, the approach we described allowed us to achieve an set of patterns that, after applied to a medium sized target document, could generate a large amount of question/answer pairs, most of which can be used without or after some revision.

As future work, we intend to generate patterns using semantic features, rather than only lexico-syntactic ones. Moreover, we would like to evaluate this approach in texts of different nature. To use dependency grammar is also in our plans in order to allow the system to learn long distance dependencies. When it comes to distractor extraction, we are considering using other sources besides the target document.

Acknowledgments

Ana Cristina Mendes is supported by a PhD fellowship from Fundação para a Ciência e a Tecnologia (SFRH/BD/43487/2008).

References

1. Ravichandran, D., Hovy, E.: Learning surface text patterns for a question answering system. In: ACL 2002: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, pp. 41–47. Association for Computational Linguistics, Morristown (2002)
2. Petrov, S., Klein, D.: Improved inference for unlexicalized parsing. In: Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference, pp. 404–411. Association for Computational Linguistics, Rochester (2007)
3. Judge, J., Cahill, A., van Genabith, J.: Questionbank: creating a corpus of parse-annotated questions. In: ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics, pp. 497–504. Association for Computational Linguistics, Morristown (2006)

4. (omitted for blind review purposes)
5. Li, X., Roth, D.: Learning question classifiers. In: Proceedings of the 19th International Conference on Computational Linguistics, pp. 1–7. Association for Computational Linguistics, Morristown (2002)
6. Bies, A., Ferguson, M., Katz, K., Macintyre, R., Contributors, M., Tredinnick, V., Kim, G., Marcinkiewicz, M.A., Schasberger, B.: Bracketing guidelines for treebank ii style penn treebank project (1995)
7. Mitkov, R., Ha, L.A.: Computer-aided generation of multiple-choice tests. In: Proceedings of the First Workshop on Building Educational Applications using Natural Language Processing, Edmonton, Canada (2003)
8. Mitkov, R., Ha, L.A., Karamanis, N.: A computer-aided environment for generating multiple-choice test items. Nat. Lang. Eng. 12, 177–194 (2006)
9. Fellbaum, C. (ed.): WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
10. Karamanis, N., Ha, L.A., Mitkov, R.: Generating multiple-choice test items from medical text: a pilot study. In: INLG 2006: Proceedings of the Fourth International Natural Language Generation Conference, pp. 111–113. Association for Computational Linguistics, Morristown (2006)
11. Hoshino, A., Nakagawa, H.: Webexperimenter for multiple-choice question generation. In: Proceedings of HLT/EMNLP on Interactive Demonstrations, pp. 18–19. Association for Computational Linguistics, Morristown (2005)
12. Sumita, E., Sugaya, F., Yamamoto, S.: Measuring non-native speakers proficiency of english by using a test with automatically-generated fill-in-the-blank questions. In: EdAppsNLP 2005: Proceedings of the Second Workshop on Building Educational Applications Using NLP, pp. 61–68. Association for Computational Linguistics, Morristown (2005)
13. Hoshino, A., et al.: A real-time multiple-choice question generation for language testing - a preliminary study. In: Proceedings of the 2nd Workshop on Building Educational Applications Using NLP, pp. 17–20 (2005)
14. Pino, J., Heilman, M., Eskenazi, M.: A selection strategy to improve cloze question quality. In: Proceedings of the Workshop on Intelligent Tutoring Systems for Ill-Defined Domains. 9th International Conference on Intelligent Tutoring Systems (2008)
15. Chen, C.Y., Liou, H.C., Chang, J.S.: Fast: an automatic generation system for grammar tests. In: Proceedings of the COLING/ACL on Interactive Presentation Sessions, pp. 1–4. Association for Computational Linguistics, Morristown (2006)
16. Soubbotin, M.M.: Patterns of potential answer expressions as clues to the right answers. In: TREC (2001)
17. Figueroa, A.G., Neumann, G.: Genetic algorithms for data-driven web question answering. Evol. Comput. 16, 89–125 (2008)
18. Brill, E., Dumais, S., Banko, M.: An analysis of the askmsr question-answering system. In: EMNLP 2002: Proceedings of the ACL 2002 Conference on Empirical Methods in Natural Language Processing, pp. 257–264. Association for Computational Linguistics, Morristown (2002)
19. Hoekstra, A., Hiemstra, D., van der Vet, P., Huibers, T.: Question answering for dutch: Simple does it. In: Proceedings of the 18th BeNeLux Conference on Artificial Intelligence (BNAIC), Maastricht, BNVKI (2006)
20. Keselj, V., Cox, A.: Daltrec 2004: Question answering using regular expression rewriting. In: TREC (2004)

Author Index