Lecture Notes in Computer Science 2553

Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Tokyo

Birger Andersson Maria Bergholtz
Paul Johannesson (Eds.)

Natural Language Processing and Information Systems

6th International Conference on Applications
of Natural Language to Information Systems, NLDB 2002
Stockholm, Sweden, June 27-28, 2002
Revised Papers

Springer

Volume Editors

Birger Andersson
Maria Bergholtz
Paul Johannesson
Royal Institute of Technology
Department of Computer and Systems Sciences
Forum 100, 16440 Kista, Sweden
E-mail: {ba, maria, pajo,}@dsv.su.se

Cataloging-in-Publication Data applied for

A catalog record for this book is available from the Library of Congress

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliographie;
detailed bibliographic data is available in the Internet at <http://dnb.ddb.de>.

CR Subject Classification (1998): H.2, H.3, I.2, F.3-4, H.4, C.2

ISSN 0302-9743
ISBN 3-540-00307-X Springer-Verlag Berlin Heidelberg New York

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

http://www.springer.de

Typesetting: Camera-ready by author, data conversion by Olgun Computergraphik
Printed on acid-free paper SPIN: 10871738 06/3142 5 4 3 2 1 0

Preface

The workshop on Applications of Natural Language to Information Systems (NLDB) has since 1995 provided a forum for academic and industrial researchers and practitioners to discuss the application of natural language to both the development and use of software applications.

The use of natural language in relation to software has contributed to improving the development of software from the viewpoints of both the developers and the users. Developers benefit from improvements in conceptual modeling, software validation, natural language program specifications, and many other areas. Users benefit from increased usability of applications through natural language query interfaces, semantic webs, text summarizations, etc.

The integration of natural language and information systems has been a research objective for a long time now. Today, the goal of good integration seems not so far-fetched. This is due mainly to the rapid progress of research in natural language and to the development of new and powerful technologies. The integration of natural language and information systems has become a convergent point towards which many researchers from several research areas are focussing.

NLDB 2002 was hosted by the Royal Institute of Technology in Stockholm, Sweden during 27–28 June and 42 papers were submitted to the workshop. The papers were distributed to the members of the program committee for reviewing. Each paper was reviewed by a minimum of three referees. The reviews were collected and evaluated resulting in a number of papers that were clearly to be accepted, clearly to be rejected, and some needing additional reviewing. The additional reviewing was done by members of the program committee not previously acquainted with the paper. To support the program committee chair in his final decision some papers were reviewed and discussed by up to five reviewers. Most of this process was done through e-mail.

Due to the large number of high-quality submissions it was decided to accept 17 full papers and 7 short papers for presentation at the conference. Each paper was classified as belonging to one of the following presentation themes:

- Linguistic aspects of modeling,
- Information retrieval,
- Natural language text understanding,
- Knowledge bases,
- Recognition of information in natural language descriptions,
- Natural language conversational systems,
- Short papers.

We wish to thank all members of the program committee for their refereeing work and sharing of valuable insights.

We also would like to thank the conference keynote speaker, Prof. Dieter Fensel, for his speech on the emerging Semantic Web. This is certainly one area where the research results of applications of natural language to information systems are and will continue to be applicable.

Stockholm, August 2002

Birger Andersson
Maria Bergholtz
Paul Johannesson

Organization

Conference Chair

Reind van der Riet	Vrije Universiteit, Amsterdam, The Netherlands

Program Chair

Paul Johannesson	Royal Institute of Technology, Sweden

Program Committee

Alfs T. Berztiss	University of Pittsburgh, USA
Mokrane Bouzegoub	Université de Versailles, France
Fabio Ciravegna	University of Sheffield, UK
Gary Coen	Boeing Company, USA
Isabelle Comyn-Wattiau	Université de Cergy, France
Walling Cyre	Virginia Tech, USA
Günther Fliedl	Universität Klagenfurt, Austria
Norbert Fuchs	University of Zurich, Switzerland
Nicola Guarino	CNR, Italy
Jon Atle Gulla	Norwegian University of Science and Technology, Norway
Zoubida Kedad	Université de Versailles, France
Christian Kop	Universität Klagenfurt, Austria
Marta Lopez Fernandez	Fraunhofer Institute for Experimental Software Engineering (IESE), Germany
Heinrich C. Mayr	Universität Klagenfurt, Austria
Paul McFetridge	Simon Fraser University, Canada
Elisabeth Métais	Conservatoire National des Artes et Métiers de Paris (CNAM), France
Luisa Mich	University of Trento, Italy
Ana M. Moreno	Universidad Politécnica de Madrid, Spain
Veda C. Storey	Georgia State University, USA
Vijayan Sugumaran	Oakland University, USA
Bernhard Thalheim	University of Cottbus, Germany
Babis Theodoulidis	University of Science and Technology, UK
Felisa Verdejo	National Distance Education University, Spain
Roland Wagner	Universität Linz, Austria
Werner Winiwarter	EC3, Austria

Additional Reviewers

Birger Andersson	Maria Bergholtz	Julio Gonzalo
Anselmo Peñas	Luc Schneider	

Local Organizing Committee

Birger Andersson	Royal Institute of Technology, Sweden
Maria Bergholtz	Royal Institute of Technology, Sweden

Table of Contents

Linguistic Aspects of Modelling

An Ontology-Based Framework for Generating and Improving Database Design 1
Vijayan Sugumaran and Veda C. Storey

A Web Information Extraction System to DB Prototyping 13
P. Moreda, R. Muñoz, P. Martínez-Barco, C. Cachero, and M. Palomar

Automatic Help for Building and Maintaining Ontologies 27
Nadira Lammari and Elisabeth Métais

Information Retrieval

Semi-automatic Content Extraction from Specifications 40
Krishnaprasad Thirunarayan, Aaron Berkovich, and Dan Sokol

Integrating Retrieval Functionality in Web Sites Based on Storyboard Design and Word Fields 52
Antje Düsterhöft and Bernhard Thalheim

Vulcain – An Ontology-Based Information Extraction System 64
Amalia Todirascu, Laurent Romary, and Dalila Bekhouche

Natural Language Text Understanding

Automatic Identification of European Languages 76
Anna V. Zhdanova

A Method for Maintaining Document Consistency Based on Similarity Contents 85
Farid Meziane and Yacine Rezgui

Evaluation and Construction of Training Corpuses for Text Classification: A Preliminary Study 97
Shuigeng Zhou and Jihong Guan

Knowledge Bases

Replicating Quantified Noun Phrases in Database Semantics 109
Roland Hausser

Ontological Extraction of Content for Text Querying 123
Troels Andreasen, Per Anker Jensen, Jørgen Fischer Nilsson, Patrizia Paggio, Bolette Sandford Pedersen, and Hanne Erdman Thomsen

Ontology-Based Data Cleaning 137
Zoubida Kedad and Elisabeth Métais

Recognition of Information in Natural Language Descriptions

Retrieving NASA Problem Reports with Natural Language 150
Sebastian van Delden and Fernando Gomez

Access to Multimedia Information through Multisource and Multilanguage Information Extraction 160
Horacio Saggion, Hamish Cunningham, Kalina Bontcheva, Diana Maynard, Cris Ursu, Oana Hamza, and Yorick Wilks

On Semantic Classification of Modifiers 172
Igor A. Bolshakov and Alexander Gelbukh

Natural Language Conversational Systems

Evaluating a Spelling Support in a Search Engine 183
Hercules Dalianis

Opening Statistical Translation Engines to Terminological Resources 191
Philippe Langlais

Short Papers

User-Centred Ontology Learning for Knowledge Management 203
Christopher Brewster, Fabio Ciravegna, and Yorick Wilks

A Multilevel Text Processing Model of Newsgroup Dynamics 208
G. Sampath and Miroslav Martinovic

Best Feature Selection for Maximum Entropy-Based Word Sense Disambiguation 213
Armando Suárez and Manuel Palomar

Linguistics in Large-Scale Web Search 218
Jon Atle Gulla, Per Gunnar Auran, and Knut Magne Risvik

Similarity Model and Term Association for Document Categorization 223
Huaizhong Kou and Georges Gardarin

Omnibase: Uniform Access to Heterogeneous Data for Question Answering 230
Boris Katz, Sue Felshin, Deniz Yuret, Ali Ibrahim, Jimmy Lin, Gregory Marton, Alton Jerome McFarland, and Baris Temelkuran

Automated Question Answering Using Question Templates That Cover the Conceptual Model of the Database 235
Eriks Sneiders

Author Index 241

An Ontology-Based Framework for Generating and Improving Database Design

Vijayan Sugumaran[1] and Veda C. Storey[2]

[1] Department of Decision and Information Sciences
School of Business Administration
Oakland University
Rochester, MI 48309
sugumara@oakland.edu
[2] Computers and Information Systems
J. Mack Robinson College of Business
Georgia State University, Box 4015
Atlanta, GA 30302
vstorey@gsu.edu

Abstract. It is well-accepted that there is a need to incorporate domain knowledge into system development tools of various kinds. Most of these are case tools that have been quite successful in supporting design on a syntactical basis. However, in general, they are not capable of representing and using information about the semantics of an application domain. This research presents a framework for supporting the generation and analysis of conceptual database designs through the use of ontologies. The framework is implemented in a database design assistant prototype that illustrates the research results.

1 Introduction

Much progress has been made in developing methodologies that support conceptual modeling. Database design methodologies that support the distinction of conceptual and logical design have been proposed and research carried out on automating various aspects of the design process [6], [1]. One result of this work is the development of case tools that have sophisticated diagramming capabilities. These tools have proven very useful for consistency checking of the syntactic aspects of the design. They could be even more useful if they had access to knowledge about the application domain. For example, they could use domain specific knowledge to assess whether an entity-relationship model is a complete and consistent representation of a user's application. A tool that incorporates domain knowledge for conceptual modeling would, thus, be useful for database design.

Ontologies have been proposed as an important and natural means of representing real world knowledge [19]. They are thought to provide a natural means for representing information for database management systems because databases have inconsistencies in naming conventions, synonyms, and so forth. Ontologies, then, could

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 1–12, 2002.

provide the domain knowledge needed for design. This domain knowledge could be particularly useful in two respects. First, domain knowledge, as stored in an ontology, can help in creating a database design because it can suggest what terms (e.g., entities) might appear in an application domain and how they are related to other terms, to suggest further entities and their relationships. Furthermore, the constraints that are found in an ontology map to business rules in an application domain which have implications for semantic integrity constraints in database design. Second, given a (partial) design, comparing the constructs in a design with those in an ontology can highlight missing constructs.

The objectives of this research, therefore, are to:

a) develop a framework that augments the conceptual modeling activity of a database designer with application domain specific knowledge embedded within ontologies, and
b) demonstrate the feasibility of the approach by developing a prototype that implements the proposed framework

The framework will model the activities needed for both generating and validating a design.

The contribution of the research is to provide a means for creating better database designs that reflect the semantics of the application while minimizing the amount of work by the designer. This should result in a more comprehensive and consistent conceptual model

2 Related Research

Considerable research is underway for creating ontologies and using them as surrogates for capturing the semantics of an application domain. In information systems development, ontologies are being applied to facilitate component reuse and other development activities [3], [14].

2.1 Ontologies

Ontologies aim at capturing domain knowledge in a generic way to provide a commonly agreed understanding of a domain that may be reused and shared [16]. An ontology refers to a set of concepts or terms that can describe some area of knowledge or build a representation of it [19]. More formally, an ontology is a catalog of types of things that are assumed to exist in the domain of interest, D, from the perspective of a certain language, L, for the purpose of talking about that domain, D [15]. Ontologies aim at capturing domain knowledge in a generic way to provide a commonly agreed upon understanding of a domain that may be reused and shared. There is significant work underway to develop large ontology libraries [10]. For example, the library at http://www.daml.org/ontologies/, contains approximately over 170 ontologies.

Notable work on the development and use of ontologies is found in the CYC project [13] and natural language understanding [5]. In the database area, Embley et al.

[7] have created ontologies for understanding web-based obituaries. Kedad and Metais [12] propose to manage semantic heterogeneity by the use of a linguistic dictionary. Sugumaran and Storey [18] propose that meta-level knowledge is needed to enable ontologies to evolve and subdivide. Bergholtz and Johnannesson [2] classify relationships in conceptual models based upon the assumption that relationships express events and activities that change.

2.2 Ontology Structure

Prior research on ontologies has focused on methodologies for developing ontologies and the structural aspects of ontologies [8]. Much of the development of ontologies, however, has focused on identifying terms and specifying how they are related to each other. Instances of some of the terms may also be included. In our prior research [18], we proposed an extension to the ontology structure that includes not only terms and various kinds of relationships, but also different types of constraints between terms. Four types of constraints are identified: a) pre-requisite, b) mutually-inclusive, c) mutually-exclusive, and d) temporal.

Pre-requisite Constraint. One term/relationship depends upon another. For example, in the online auction domain, a bid requires an item. However, information about the item can exist on its own without requiring a bid. Consider two terms A and B. If A requires B, then, we define the pre-requisite constraint: A $\rightarrow$ B. Term B may occur by itself; however, for term A to occur, term B is required.

Mutually Inclusive Constraint. One term may require another for its existence. For terms A and B, if A requires B and B also requires A, then we have a mutually inclusive relationship between A and B, i.e., these two terms have to occur simultaneously. Then, we define the mutually inclusive constraint as: A$\leftrightarrow$B. For example, bid and bidder require each other. To be a bidder, one must have made a bid. Likewise, a bid cannot be made without a bidder.

Mutual Exclusive Constraint. One term or relationship cannot occur at the same time as another. For example, an individual cannot be the buyer and seller of the same item. This operates on the constraint for availability and the constraint for closed bidding. If A and B are mutually exclusive, then only one of them can occur in a given context. We define the mutually exclusive constraint as: A$\not\leftrightarrow$B.

Temporal Constraint. One term or relationship must occur before another. For terms A and B, if A must precede B, we define the temporal constraint as: A$\mapsto$B. For example, to bid on an item, one must register as a bidder.

3 Ontology-Based Database Design Support

There are two main ways in which ontologies can serve as a knowledge base of domain semantics that can be used to improve a database design. The two primary

functions for which an ontology could be used are design generation and design verification. The first task is to generate a design "from scratch," using the terms and relationships in the ontology as a representative model of the domain. The second involves using the ontology to check for missing terms or inconsistencies in an existing, or partial, design.

Applying ontologies in this way can be useful for several reasons. First, it will reduce the amount of work for any designer, in particular, a novice designer. When a design is incomplete, terms from an ontology can be shown to the user to check for missing terms and identify how they are related to other terms. They can also be used to analyze a design for missing constructs. They can assist in identifying appropriate terms for an application and when comparing heterogeneous databases to create a synthesized database or map a higher-level common database.

3.1 Design Generation

The partial ontology for auction websites in Table 1 is used to illustrate how ontologies can effectively improve database design, and how our methodology facilitates ontology creation. The database design process begins with a statement of the user's requirements. The requirements represent one user view of an application domain. The role of an ontology is to provide a comprehensive set of terms, definitions, relationships, and constraints for the domain. From these, the designer selects the relevant ones to augment the design as illustrated in Figure 1. The term "item" suggests the need for category, customer and shipper. The ontology can also suggest that relationships exist between: 1) item with category, customer, and shipper, 2) item and seller, and 3) item and buyer. The ontology should make further suggestions based on transitive inferences. For example, an item is related to a category. Since a seller is related to an item, it is possible that a seller is related to a category, and corresponding relationships are needed.

Table 1. Partial Auction Domain Ontology.

Term	Synonym	Description	Business Rule	Related to
Item	Product	Bought and sold and bid on		Category, Customer, Shipper
Category		Classification of Product		Item
Customer	Buyer, Seller	Person who buys or sells		Item, Account
Account		Prerequisite for Transaction	Customer Needs to open account for transaction	Customer
Shipper	Vendor	Company that delivers the items to the buyer		Item
Bid	auction price	Current price willing to pay		Item, Customer
Bidder	Buyer Customer	Bids for item or product on sale	Has to be registered to bid	Buyer, Customer, Item
Buyer	Customer	Buys item or product on sale		Item
Buy	Purchase	Buy or acquire and Item	Inverse transaction of Sell	Customer, Buyer, Item
Sell	Liquidate	Selling Items	Inverse of Buy transaction	Buyer, Customer, Item
Seller	Customer	Puts Item up for sale		Item
Shipper	Vendor	Company that delivers items		Item

A designer might need to refine a fragment of an existing entity-relationship model. An ontology could check that the names of the entities are the "best" ones for the application domain. The inclusion of synonyms in the ontology helps to identify the most appropriate label for an entity or relationship. This assumes that the term used in the ontology is the best one for the application domain. In Figure 2, the relationship "Customer has Account" could be refined to Customer owns Account. The ontology should suggest possible missing entities and relationships. The simplest way to do so is to trace through the possible inferences that can be made from one term to another [17]. For example, in Figure 2, the ontology expands the E-R diagram by identifying bidder and seller as subclasses of customer. Relationships between bidder and seller and product are then added to the E-R diagram.

3.2 Design Verification

A designer might have an incomplete design for several reasons. An inexperienced designer might not have been able to extract enough user requirement details from the users. A designer might be using automated design or CASE tools that can deal only with the syntactical nature of a design. Another source of initial designs is the results obtained from reverse engineering of relational databases to obtain a conceptual design [11], [4]. The conceptual design that was reverse engineered may be incomplete because information is lost when translating a conceptual design into a logical one in the first place. For example, when a foreign key is used, the name of the relationship verb is lost and cannot be inferred during the reverse engineering process. If client# were a foreign key in a project relation, it might be because a client sponsors a project. An ontology showing the relationship between client and project could suggest the nature of such a relationship.

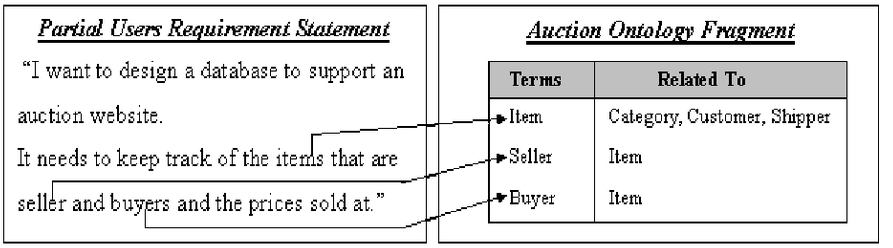


Fig. 1. User requirements mapped to auction ontology.

3.3 Conceptual Model Generation Framework

This section proposes a framework for conceptual model generation. It can be used to generate a conceptual model from scratch or augment an existing one. It is assumed that the domain ontology is expressed using the primitives outlined above.

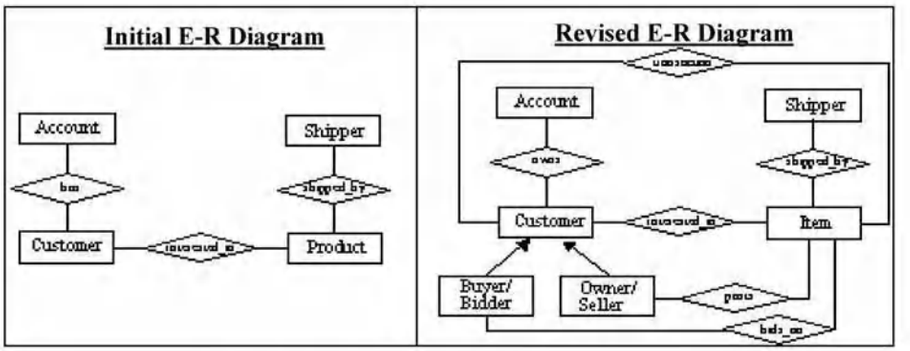


Fig. 2. Revising E-R Diagram Through Inferences.

Creating an ER Model From Scratch. The proposed framework for generating an ER model comprises of the following steps: a) identify initial user terms, b) map the initial terms to domain terms in the ontology and expand the model, c) check for consistency, d) generate the complete ER model, and e) map to a relational model. These steps are described below.

a) Identification of Initial Terms. Identify the key terms and concepts relevant to an application based upon the user's input. The database designer can provide specific application requirements or a textual description of the problem domain and the desired functionalities from which appropriate terms are identified. Simple natural language processing techniques can be used to identify relevant terms from the user descriptions. Alternatively, if the user is familiar with the application domain, he or she can provide appropriate terms directly.

b) Map to domain terms and expand list of terms. Once the initial set of terms are identified, they are compared to those in the ontology for that domain. The "synonym" and "is_a" relationships from the ontology are used to verify the appropriate use of the terms in the design. The initial set of terms is also expanded using the "related-to" relationship. This identifies additional terms the designer may be interested in, but has not explicitly articulated.

c) Consistency checking. After gathering the set of potential terms, additional domain knowledge can be used to check whether the terms and concepts selected are complete and consistent. The four types of constraints: 1). pre-requisite, 2). temporal, 3) mutually-inclusive, and 4) mutually-exclusive, which are used to ensure that all the relevant terms and concepts have been included.

d) Generate ER model. Based on the terms identified in the previous step, entities are created. Relationships between entities are generated by taking into account the application domain semantics incorporated within the ontology. Previously generated ER models and the functionalities supported by these models are stored in a repository.

This repository is also used in identifying commonly occurring relationships within a particular application domain, and this information is also used in generating the relationships between entities. Thus, at the end of this step, an overall ER model is generated and presented to the user for feedback.

e) Map to Relational Model. Once the ER model is created, the corresponding relational data model is generated using the well established rules of mapping from ER to relational [20].

Validation of Existing ER Model. The framework also supports validating an existing ER model by checking it to determine if it includes some of the generic entities that are typically part of similar designs in that application domain. The model can also be checked to assess if the entities and relationships are internally consistent. The ER validation framework consists of the following: a) check appropriateness of terms, b) identify missing terms, and c) generate augmented ER model.

a) Appropriateness of Terms. The database designer provides information related to the ER model. Based on this information, the terms used to name the entities are checked for appropriateness using the domain knowledge contained in the ontology.

b) Identifying Missing Terms. For the terms captured in the ER model, if "is_a" or "related_to" relationships exist, use them to gather related terms. The constraints defined in the ontology for the terms used in the ER model are also enforced to ensure that they are internally consistent.

c) Augmenting the ER Model. For the terms identified in the previous step, appropriate relationships are also suggested based on the knowledge culled from previous ER models stored in the ER model repository. With the designer's feedback, an augmented ER model is generated that meets the requirements specified by the designer.

3.4 Consistency Checking

The structure of the ontology contains various primitives to express different types of relationships and constraints that exist between terms or concepts within an application domain. The constraints feature supported by the ontology allows one to express the application domain semantics and business rules. For example, in the online auction domain, a bid needs (pre-requisite) item, account, and bidder. The temporal constraint for bid indicates that an account must be created before a bid can be placed. The mutually exclusive constraint for the term seller indicates that the seller of an item cannot also be its buyer.

When trying to identify or add new terms (entities) to an ER model, or removing a particular term from an ER model, the constraints are utilized to check for consistency. For example, when adding a new term, if there exists prerequisite, or mutually inclusive terms, those terms must also be included. In the auction domain, the mutually inclusive constraint for bid implies that, whenever bid is included in a design, the associated terms item, buyer, and seller should also be included. Similarly, when

deleting a particular term, if other terms depend upon this term, it should not be deleted. The ontology can be represented as a graph, where the nodes correspond to terms and the arcs to relationships and constraints. While selecting or deleting a term, this graph can be utilized to ensure that the selection or deletion action results in a consistent set of terms. As part of the framework, heuristics perform consistency checking while selecting or removing a term. These heuristics have then been translated into production rules and stored in the knowledge base. They are briefly described below.

4 Database Design Assistant Architecture

The purpose of the Database Design Assistant tool is to facilitate the creation of conceptual models (ER model) or to validate an ER model for consistency with application domain semantics. A novice database designer can use this tool to quickly create an ER model for database design using the application domain knowledge contained in the ontology. This minimizes the cognitive load on the designer and ensures that the model is a reasonably complete and consistent one. The designer can also use the tool to validate an ER model and identify entities and relationships that may be missing. Figure 3 shows the architecture of the system, which follows the traditional three-tier client-server architecture with an HTML client, a web server, and a relational database. The client is a basic web browser for the user to browse or modify existing ontologies, or create new ones. The server contains Java application code; the relational database stores the domain ontologies and information about existing ER models and their functionalities for various domains.

The client serves as the user interface. It provides a web interface for the user to specify requirements for the database application in free form, as well as render the conceptual model generated up to that point. It provides mechanisms for the user to interact with the database design assistant, captures the user's intent, and conveys it to the server. The server translates and processes the client request. It consists of the following five modules, 1) input-output module, 2) visualization module, 3) terms and relationships module, 4) constraints module, and 5) conceptual model generation module. These modules are described briefly in the following paragraphs.

1) Input-output Module: The input-output module deals with capturing the user's specification or requirements for the database application and forwards it to the terms and relationships module. The I/O module is also responsible for displaying the results to the user in an appropriate format.

2) Visualization Module: While creating an ER model, at any point in time, the user can view the ER diagram that has been created up to that point. The visualization module is responsible for rendering the ER diagram.

3) Terms and Relationships Module: The free form specification provided by the user for the database design is analyzed and key words identified. This is accomplished by utilizing simple natural language processing techniques. The terms and relationships module then uses the key words in identifying appropriate terms and relationships by interfacing with the domain ontology repository and the relationship repository.

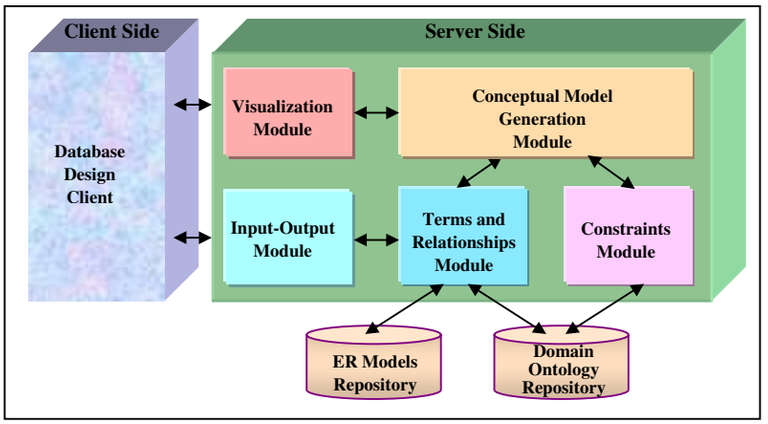


Fig. 3. Database Design Assistant – System Architecture.

4) Constraints Module: This module contains rules for consistency checking while selecting or deleting constructs from the ER model. The constraints module is responsible for checking the consistency and comprehensiveness of the model.

5) Conceptual Model Generation Module: Once the appropriate terms and relationships are identified and checked for consistency, this module generates the ER model and passes it to the visualization module for displaying the final ER model. This module can also create fragments of the overall ER model to be provided to the user for closer examination.

The database design assistant has two knowledge repositories: 1) domain ontology repository, and 2) ER models repository.

1) Domain Ontology Repository: The domain ontology repository consists of ontologies for various application domains. The user can browse these ontologies to learn more about that application domain. These ontologies may created by different domain analysts or stored in a library and may evolve over time.

2) ER Models Repository: The ER models repository contains different ER models that have been created over a period of time for various application domains. For each model, meta information about the functionalities supported by the model is also stored. By examining the ER models in a domain and the corresponding functional requirements, generic relationships that exist between entities can be identified and this information could be used in creating a new ER diagram for an application within that domain with similar entities and functional requirements.

5 Prototype Implementation

A prototype of the system is under development using Jess, a java-based expert system shell [9], Servlets, and JSP (Java Server Pages). The input-output module and the visualization module are implemented using JSP technology. These modules utilize several java servlets for: gathering user input; identifying key terms from the user description of the requirements for the database application, accessing and retrieving

domain ontology information from the ontology database. The terms and relationship module, constraints module and the conceptual model generation module are implemented using Jess. Consistency checking and conceptual model generation are accomplished through rules written in Jess that make use of domain specific knowledge contained in the domain ontology as well as the ER model repository.

Using the database design assistant, the designer can start the process of designing an ER model from scratch or work with an existing ER model. In both cases, the designer can explicitly check the validity of the model in terms of whether the model uses appropriate terms and relationships and that it is not missing any major concepts. When starting with an existing ER model (or a fragment), the designer can input the names of the entities and the relationships that are part of this model. The user inputs this information for every binary relationship that exists within the model, and the system recreates the ER model. In a future version, the designer will be able to input the entire ER model in a predefined file format. Once the ER model information is provided, the system checks the model for missing and superfluous entities and relationships and presents a report to the user.

Figure 4 shows the interface to input the ER model information. For every binary relationship contained in the model, the user types the name of the relationship and the corresponding entities for that relationship. For example, the initial ER fragment shown in Figure 2 contains three binary relationships and information corresponding to these relationships is entered as shown in Figure 4. The designer can enter additional binary relationships by clicking on the "Add More Entities and Relationships" button, or start the validation process by clicking on "Validate The Model" button.

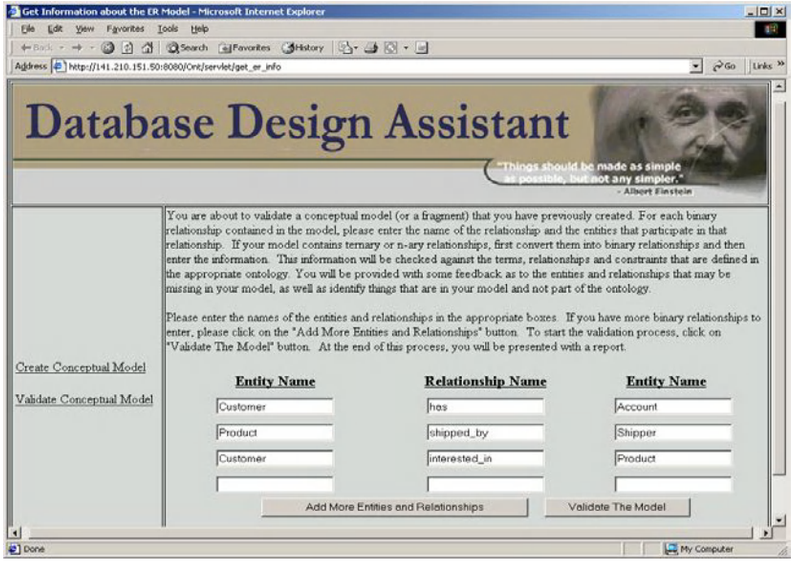


Fig. 4. Entering ER Model Information for Validation.

At the end of the validation process, the system generates a report that contains information about suggested new entities, new relationships, name changes for entities and relationships, etc. As shown in Figure 5, the validation report for the initial ER diagram (Figure 2), suggests adding "Bidder" and "Seller" entities, as well as three other relationships. The designer can print this report and draw the updated ER diagram by clicking on the appropriate buttons. If the system finds terms that are not part of the ontology, this information is captured and sent to the domain analyst for extending the ontology. The prototype provides a limited set of functionalities for importing conceptual models into the ER model repository from a variety of sources. A future version of the system will provide additional mechanisms for generating ER diagrams using different notations, and include additional reporting facilities.

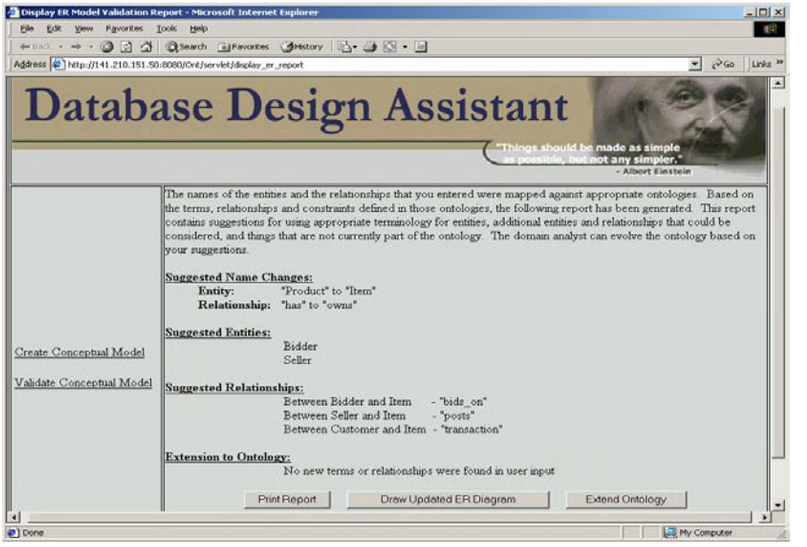


Fig. 5. ER Model Validation Report.

6 Conclusion

This paper has demonstrated how the use of domain knowledge stored in the form of an ontology can be useful to assist in the generation of more complete and consistent database designs, both when the designs are generated from scratch and when a partial design exists. A framework for these two tasks has been presented and implemented. This framework utilizes ontology primitives such as relationships and constraints between terms to generate a consistent design. The prototype incorporates consistency-checking rules that enforce the consistency of terms used while selecting or deleting them from the current conceptual model. Further work is needed to complete the prototype and assess its effectiveness for a variety of design tasks. The initial version of the prototype demonstrates the feasibility of the framework.

References

1. Alter, S. Information Systems: A Management Perspective. Addison-Wesley, 1999.
2. Bergholtz, M., Johannesson, "Classifying the Semantics of Relationships in Conceptual Modeling by Categorization of Roles," Proceedings of the 6th International Workshop on Applications of Natural Language to Information Systems (NLDB'01), Madrid, Spain, June 28-29, 2001, pp. 199 – 203.
3. Braga, R., Mattoso, M., Werner, C. "The Use of Mediation and Ontology Technologies for Software Component Information Retrieval," Proceedings of SSR'01, May 18 – 20, Toronto, Canada, 2001, pp. 19 – 28.
4. Chiang, R.H.L, Barron, T., and Storey, V.C. "Reverse Engineering of Relational Databases: Extraction of an EER Model from a Relational Database," Data and Knowledge Engineering, Vol.12, No.2, March 1994, pp.107-142.
5. Dahlgren, K. "A Linguistic Ontology," International Journal of Human-Computer Studies, Vol.43, 1995, pp.809-818.
6. Dennis, A. and Wixom, B. H. Systems Analysis and Design. John Wiley, 2000.
7. Embley, D., Campbell, D.M., Jiang, Y.S., Ng, Y.K., Smith, R.D., Liddle, S.W. "A conceptual-modeling approach to web data extraction," Data &Knowledge Engineering, 1999.
8. Fensel, D., Harmelen, F.v., Horrocks, I., McGuinness, D.L. and Patel-Schneider, P.F. "OIL: An Ontology Infrastructure for the Semantic Web," IEEE Intelligent Systems (March/April), 2001, pp. 38-45.
9. Friedman-Hill, E. Jess, the Expert System Shell, Sandia National Laboratories, Livermore, CA, 2002. URL: http://herzberg.ca.sandia.gov/jess
10. Hendler, J. "Agents and the Semantic Web," IEEE Intelligent Systems (March/April), 2001, pp. 30-36.
11. Johannesson, P. "A Method for Transforming Relational Schemas into Conceptual Schemas," Proceedings of the Tenth International Conference on Data Engineering, February 1994, pp. 190 – 201.
12. Kedad, Z., and Metais, E., "Dealing with Semantic Heterogeneity During Data Integration," in Akoka, J., Bouzeghoub, M., Comyn-Wattiau, I., and Metais, E. (eds.), Conceptual Modeling – ER'99, 18th Intl. Conference on Conceptual Modeling, Lecture Notes in Computer Science 1728, Paris, France, 15-18 November 1999, pp.325-339.
13. Lenat, D.B. "CYC: A Large-Scale Investment in Knowledge Infrastructure," Communications of the ACM (38:11), 1995, pp. 33-41.
14. Perez, A.G. and Benjamin, V.R. "Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods," Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5), Stockholm, Sweden, 1999,
15. Sowa, J.F., "Ontology, metadata, and semiotics," in B. Ganter & G. W. Mineau, eds., Conceptual Structures: Logical, Linguistic, and Computational Issues, Lecture Notes in AI #1867, Springer-Verlag, Berlin, 2000, pp. 55-81.
16. Stephens, L.M. and Huhns, M.N. "Consensus Ontologies: Reconciling the Semantics of Web Pages and Agents," IEEE Internet Computing (September-October), 2001, pp. 92-95.
17. Storey, V.C., Goldstein, R.C., Chiang, R.H.L., Dey, D., and Sundaresan, S., "Database Design with Common Sense Business Reasoning and Learning," ACM Transactions on Database Systems, Vol.22, No.4, December, 1997, pp.471-512.
18. Sugumaran, V., Storey, V. "Ontologies for conceptual modeling: Their creation, use, and management," Data and Knowledge Engineering, 2002 (forthcoming).
19. Swartout, A.T., "Ontologies," IEEE Intelligent Systems and Their Applications, 1999.
20. Teorey, T.J., Yang, D. and Fry, J.P. "A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model," ACM Computing Surveys (18:2), 1986, pp. 197-222.

A Web Information Extraction System to DB Prototyping*

P. Moreda, R. Muñoz, P. Martínez-Barco, C. Cachero, and Manuel Palomar

Grupo de investigación del Procesamiento del Lenguaje y Sistemas de Información
Departamento de Lenguajes y Sistemas Informáticos. Universidad de Alicante
Alicante, Spain
{moreda,rafael,patricio,ccachero,mpalomar}@dlsi.ua.es

Abstract. Database prototyping is a technique widely used both to validate user requirements and to verify certain application functionality. These tasks usually require the population of the underlying data structures with sampling data that, additionally, may need to stick to certain restrictions. Although some existing approaches have already automated this population task by means of random data generation, the lack of semantic meaning of the resulting structures may interfere both in the user validation and in the designer verification task.
In order to solve this problem and improve the intuitiveness of the resulting prototypes, this paper presents a population system that, departing from the information contained in a UML-compliant Domain Conceptual Model, applies Information Extraction techniques to compile meaningful information sets from texts available through Internet. The system is based on the semantic information extracted from the EWN lexical resource and includes, among other features, a named entity recognition system and an ontology that speed up the prototyping process and improve the quality of the sampling data.

1 Introduction

Nowadays, both practitioners and researchers avow the necessity of conceptual models to design and deploy high-quality non-trivial applications. Following this trend, well known Software Engineering practices, usually based on the *de facto* standard in industry UML [14], are being adopted. However, the yet insufficient number of support for automatic tasks that ease the implementation and testing of the corresponding software artifacts is diminishing the potential benefits of such models. We agree with [1] in that Advanced Software Production Environments that include Model Based Code Generation and Testing Techniques that partially automate the development process are crucial for organizations to maximize the results of applying conceptual modeling concepts.

* This paper has been supported by the Spanish government, projects TIC2000-0664-C02-01/02 and TIC2001-3530-C02-01/02

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 13–26, 2002.

In this sense, efforts made towards such automation have already showed some interesting results. Most current software development processes already include semi-automatic translation of concepts among models, thus promoting the reuse of design specifications and avoiding inconsistencies. Closer to our objective, efforts in the Requirement Analysis field are being made to generate each time more refined conceptual models out of textual descriptions. In fact, we believe that textual analysis is bound to pay an ever increasing role in this automated process, as will be explained in Section 3. This paper presents one of its application possibilities: the population of application prototypes with sampling data that provide meaningful content on which to test and get stakeholder feedback.

2 Class Diagram Definition and Constraints

A Class Diagram is a graphic view of the static structural model of the system [14]. Being a milestone in the conceptual modeling process, it includes domain-related textual information in the form of descriptors (text strings) associated to each of its elements. Due to its nature, the Class Diagram constitutes an invaluable source for domain knowledge: the UML notation helps not only in the conceptual understanding of the domain, but also provides an ontology over the domain vocabulary.

In order to illustrate the whole sampling data extraction process, a small example is going to be employed all along the paper: a *Hotel Reservation System*. The UML class diagram corresponding to such system can be observed in Fig. 1.

In this system, and as a basic explanation (for reasons of brevity) let's assume each hotel has a set of *Rooms (Habitación)* which can be of different *Types (TipoHabitación)*. Rooms can be booked by *Clients (Cliente)*. For each *Booking (Reserva)* the system keeps track of the *Services (Servicio)* provided (laundry, drinks, etc), in order to *Charge (Cargo)* them to the client. On client departure, an *Invoice (Factura)*, which may include more than one *Booking*, is generated and the method of *Payment (MedioPago)* is registered. In this diagram, all classes revolve around a class decorated with a standard *singleton* UML stereotype. UML Stereotypes are a very powerful standard extension mechanism that provide the modeling constructs with additional semantics, in this case the existence of a single instance of the *Hotel* class. This class constitutes the Domain Class, that is, the class that establishes the domain context, whose utility will be shown in section 4.2[1].

2.1 Nomenclature Constraints

In order to facilitate the text analysis, it is necessary to define a set of rules for class, attribute and relationship names that complement the general UML

[1] A whole explanation of Class Diagram design rules is out of the scope of this article. Interested readers are referred to [14].

notation rules. These rules make easy the task of name reconstruction as will be explained in 4.1.

Several features must be considered for each component of the class diagram (classes, attributes and relationships).

Class names. The rules to take into account for the definition of a class name are the following:

1. A simple noun beginning with a capital letter is a class name –i.e. Factura (*Bill*)–.
2. An usual abbreviation of language beginning with a capital letter is a class name –i.e. Tlfn (*Tel*) is the abbreviation of *Telephone number*–.
3. An acronym is a class name –i.e. CP (*Zip*) is the acronym of *Postal Code*–.
4. A complex name made up of any element above described is a class name –i.e. TipoHabitación (*RoomType*)–.

Attribute names. The way of defining attribute names follows rules similar to those applied to class names, its main difference lying in that the attribute name begins with a small letter. These rules are the following:

1. A simple noun beginning with a small letter is an attrbute name –i.e. categoría (*category*)–.
2. An usual abbreviation of language beginning with a small letter is an attribute name –i.e. tlfn (*tel*) is the abbreviation of *telephone number*–.
3. An acronym is an attribute name –i.e. NSS is the acronym of *Social Security Number*–.
4. A complex name made up of any element above described is an attribute name –i.e. numFactura (*billNumber*)–. In order to make easy the name reconstruction task, the attribute names will be split into several words by capital letters.

Relationship names. Last, relationship names must follow the following rules:

1. A verb in its infinitive form beginning with a capital letter is a relationship name –i.e. Reservar (*to book*)–.
2. A verb in its infinitive form beginning with a capital letter plus a class name is a relationship name –i.e. HacerCargo (*to make charge*)–.

3 Prototyping

All the rules presented above simplify the definition of a search activity to find meaningful data that could be part of the system population. Even in automated environments this 'sampling population task' has traditionally been (up to our

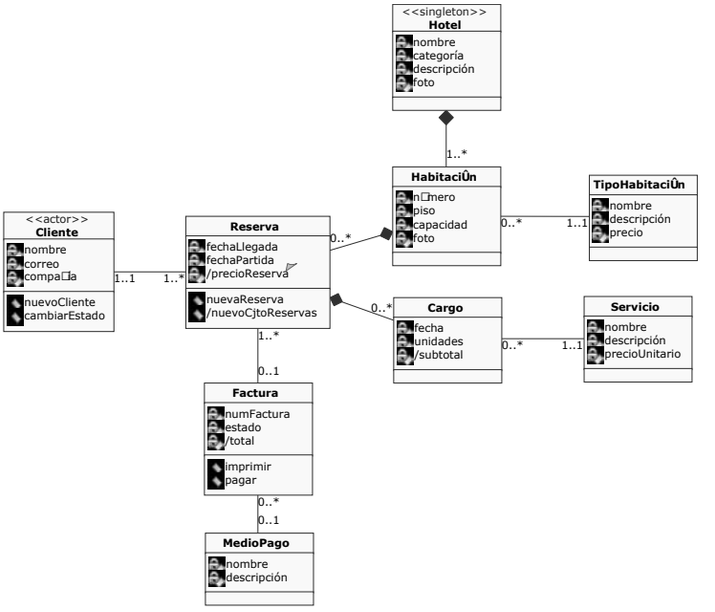


Fig. 1. *Example of a class diagram*

knowledge) left out to the designer, who, either manually or by random generation routines had to create and maintain the necessary 'testing sets'. Although these simple techniques have sufficed for a long time, both the increasingly important role of stakeholders in the software development process and some empirical observations suggest that meaningful sampling data helps to improve not only the stakeholder perception of the application under development, but also facilitates the verification and validation process. Another useful characteristic of the use of meaningful data during the development process is that it may help to refine design aspects. Perhaps one of the most straightforward ones is the evolution of the models to deal with data formats not previously considered. Furthermore, we may be interested in maintaining certain restrictions on the sampling data that permit the verification and/or validation of certain parts of the application. Letting the designer manually assure that these restrictions are met may become cumbersome. All these reasons justify, from our point of view, the use of natural language processing activities such as the ones presented next.

4 Natural Language Processing

As we have stated above, the goal of this paper is to identify some information called relevant in real texts in order to fill the underlying data structures. For this task, Information Extraction (IE) techniques, an application of Natural

Language Processing, are used. Information extraction is the name given to any process that structures and combines data which is found, explicitely stated or implied, in one or more texts. It is important to stress that we do not aim at developing an information extraction system, but at taking advantage of this technique to select data that provides content to the domain models.

In the following subsections we will show the Information Extraction techniques that have been used in this system, as well as the constraints that must be fulfilled in the Class Diagram to help the automation of the sampling data.

4.1 Name Reconstruction

In order to obtain the word sense through WordNet it is necessary to make a construction of class, attribute and relationship names from the Class Diagram. In this construction, a dictionary of acronyms and abbreviations is used. This dictionary returns their full names. Each entry in the dictionary relates the acronym or the abbreviation to its full name. Moreover, this dictionary could be extended with new user-defined entries.

Finally, this dictionary is used together with an appropriate set of heuristic rules (e.g. abbreviation + Capital word → Full word + "of" + word, –*numBill (number of bill)*–) to accomplish the reconstruction of the names. This information will be used in following steps.

Next, an overview of this process is presented.

Class and attribute name reconstruction process

1. If the name is a composed name, then it is split into single words separated by blank spaces building a composed term (e.g. *numBill* → *num bill*).
2. Each single word is looked for in the dictionary of acronyms and abbreviations. If found, the abbreviate word in the composed term is updated with the full word. (e.g. *num* → *number*).
3. The term is looked for in WordNet. If found, the process finish.
4. Blank spaces in the term are replaced with underline chars. (e.g. *number bill* → *number_bill*).
5. The term is looked for in WordNet. If found, the process finish.
6. The preposition 'de' '*of*' is inserted between each pair of consecutive names in the term (with blank spaces). (e.g. *number bill* → *number of bill*).
7. The term is looked for in WordNet. If found, the process finish.
8. Blank spaces in the term are replaced again with underline chars.(e.g. *number of bill* → *number_of_bill*).
9. The term is looked for in WordNet.

Relationship name reconstruction process

1. If the name is a composed name, then it is split into the verb (left part) and the remainder words separated by blank spaces building a composed term. (e.g. *ChangeNumBill* → *change numBill*).

2. The verb is looked for in WordNet.

The remainder words are reconstructed with the class name reconstruction process (e.g.*numBill* $\rightarrow$ *number bill*). This process will be used if it is necessary the disambiguation of the verb.

4.2 Domain Recognition

One of the main tasks that must be accomplished in any kind of web searching is to avoid the problem with word sense ambiguity. In this sense, the part-of-speech of this word is an important clue to detect ambiguities. For instance, the word '*plant*' has a different meaning when it acts as a verb from a noun.

Moreover, the same word with the same part-of-speech could have several senses when it is defined in different domains. For instance, the noun '*mouse*' has a different meaning if it is defined into a computer-science domain or in an animal one.

Furthermore, also the same word with the same part-of-speech inside the same domain could have different senses. In this case, the clue is the belonging to a defined subdomain. This is, for example, the different sense of the word '*charge*' in a hotel domain that could mean the price charged to the custom for the service, or the task that has been assigned to a person in the hotel.

As above shown, our proposal is based on the use of the names of attributes to find possible values to them. Unfortunately, the name of an attribute is not free from a possible ambiguity, as well as the information that is going to be extracted from the World Wide Web. So, ambiguity must be solved in two different fields, on one hand, in the attribute name, and on the other hand, in the information to be extracted from the WWW.

In order to solve the ambiguity in the attribute name, our proposal makes use of the following knowledge:

- **Part-of-speech (POS)**. POS information is known in a class diagram. In this way, attribute or class names are usually nouns. However, the name of a relationship is a verb.
- **Domain definition**. Domain is one of the clues to word sense disambiguation. In order to establish the domain in which the class diagram has been defined, every class diagram is forced to have a DOMAIN CLASS in which information about the general domain of the diagram is user-defined. This domain must be selected from a domain ontology. The definition of this ontology is based on the domain classification used by web searchers as we will further explain.
- **Subdomain recognition**. In those cases in which domain is not enough to word sense disambiguation, a subdomain recognition must be performed. In our proposal a **subdomain recognition module** for class attribute names has been developed. This module is based on heuristic rules that relate attributes with their classes and the relationships between their classes and other classes. For instance, '*name*' is a very usual word as as attribute

name. However, it has no sense in its own. To identify this sense we must know information about the class which it belongs to. In this case, if the name class is '*Person*' the sense of this '*name*' is, without doubt, inferred.

Once all the information to solve the ambiguity is obtained, then an unique word sense is obtained through the **WordNet consulting module**. This module performs a consult in the lexical database WordNet, and filters the result with POS, domain, and subdomain values.

Other problem to solve is the ambiguity in the information that is going to be extracted from WWW texts. Again, some additional information about words must be included in order to define their concrete sense. In this case our system is equipped with some tools providing this kind of information:

- **POS tagger**. This tool provides the information about part-of-speech for every word in the text that is being analyzed.
- **Domain web searcher**. Our system takes advantage of the domain classification developed by any web search engine. By means of this kind of web search engines, the searching scope is reduced to web pages included in this domain. To be more precise, we will reduce the scope to the domain defined in the DOMAIN CLASS of the class diagram. That is the reason why the ontology used to define these DOMAIN CLASSES is based on the web searchers domain classification.
  In this work we have used the Google web search engine (`www.google.com`). Consequently, the ontology used to define the DOMAIN CLASS has been extracted from the one define in Google. The top level of this ontology is shown in Fig.2.

Fig. 2. *Top level of the Google ontology*

- **Word sense disambiguator (WSD)**. Finally, a WSD system is used to disambiguate those cases in which POS and domain are not enough to extract a single sense. In this way, the WSD system developed by [9] is being used. This tools provides a single WordNet label for each word.

Once we have clarified the way of obtaining a single sense for both attribute names and the information that is going to be extracted, then a compatibility feature must be identified between them. This feature determines if the web information fits the attribute name, so it is useful to fill this structure.

However, this feature not only needs to perform a direct comparison between the WordNet senses, but also needs to apply some semantic relationships as defined in WordNet namely the synonym, hyperonym and hyponym relationships between those senses to relax this compatibility feature.

This task provides all needed information to the Named-Entity recognition module that we will explain in the next subsection.

4.3 Named-Entity Recognition

The main technique used to identify real information is the named-entity recognition technique. According to MUC[2] [11], we can distinguish different kinds of categories of named entities: person, organization, location, dates, currency, etc. Named entity recognition involves processing a text and identifying certain occurrences of words or expressions belonging to a category of Named Entity. Different approaches for Named Entity recognition have been developed at MUC competition. Most of them use gazetteers and list of names to help to recognize the entities [13, 10, 4]. Obviously, we will need huge lists to guarantee a high score. However, Cucchiarelli et al. in [2] report that one of the bottlenecks in designing Name Entity recognition systems is the limited availability of large gazetteers. Mikheev et al. in [8] develop a Name Entity recognition system which combines rule-based grammars with statistical models (maximum entropy).

Our Named Entity recognition system can be split in the following steps:

1. **Name identification**. The first problem to be solved is the identification of the boundaries of a named entity. Entities can be complex, consisting of several words. This is specially common when conjunctions, prepositions and definite articles are involved. In Spanish, it is common to find full names with prepositions and definite articles (Joaquín de la Cierva) or conjunction (Santiago Ramón y Cajal). Moreover, in Spanish like in English, the name of companies can be made up of any kind of word (bare nouns, adjectives, numbers, conjunctions, etc.). According to McDonald [7], we can use internal and external evidence to recognize a named entity and to classify it as belonging to the adequate category. The following rules are applied in order to establish the starting and end point of an named entity:

[2] Conferences organized to define the different task accomplished by IE system and to evaluate them using a specific domain.

(a) Looking for a trigger. Internal evidences notify the presence of named entities. These internal evidence can be made up of several specific words called triggers. Every kind of category has specific triggers. The names of companies frequently use corporate designators such as S.A., S.L., Co., Cia., Ltd., Inc., etc. The names of people frequently use person titles, such as Sr., Sra., Mr., Mrs., etc. Also, the names of locations may use words such as Ciudad (City), Calle (Street), Avda. (Ave.). In figure 3, a short sample set of rules is shown.

```
PERSON

Title(Sr., Sra.,) + CapWord                                  Sr. Indurain
Title(D., Dña.,) + CapWord + CapWord                  D. Cristobal Colon
Title(D., Dña.,) + CapWord + CapWord + CapWord        D. José Luis Pérez

ORGANIZATION

Designator(S.A., S.L.,) + {CapWord}                      S.A. El Águila
{CapWord} + Designator (S.A., S.L.,)                        Repsol S.A.

LOCATION

Trigger(C/, Avda., Plza.) + {CapWord}                    Avda. Cataluña
```

Fig. 3. *Example of rules to check the entity triggers*

(b) Grouping all capital words. However, these previous triggers do not always appear with the named entity. In this case, other internal evidence is needed to identify the named entity, such as the use of partial orders of the composing words. Examples of this fact are shown in figure 4.

```
PERSON

ChristianName + CapWord                                    Miguel Indurain
ChristianName + CapLetter + CapWord                      Juan A. Samaranch

LOCATION

{CapWord}+ "," + CountryName                              Alicante, España
```

Fig. 4. *Example of rules to check the entity without triggers*

In addition to internal evidence, external evidence can be used to identify some entities. The external evidence is made up of a set of rules taking advantage of some words that appear next to entities. These spe-

cial words depend on the domain work. For this reason, we use the name classes and attributes from the class diagram to identify some entities together with all the words semantically related to the appropriate sense provided by the WSD. Moreover, we also use EuroWordNet to find semantic names related to the name classes and attributes, as we have previously described. Figure 5 shows an example of this kind of rules in the context of our domain example.

```
PERSON

"director" + ChristianName + CapWord          el director Miguel Indurain

ORGANIZATION

"director" + "of" ChristianName + CapWord            el director of Melia
LOCATION

"located in" + {CapWord}+ "," + CountryName           located in Alicante
```

Fig. 5. *Example of rules using external evidence*

2. **Name entity Disambiguation**. Once the named entity is identified, several problems can be found. This task aims at solving the following problems:
   - Ambiguity capitalized words. Sometimes, the first word of the sentence (capital word) appearing next to the entity can be identified as a part of the entity name ([The Melia] offers... vs. The [Melia] offers...). This problem is solved by matching previous appearances of the entity. If it is the first appearance, both structures are considered, and the removal of one of them is delayed until later steps.
   - Semantic ambiguity. The structure of different named identities (person and location, person and organization, etc.) can be similar. Typical examples are extensively shown in literature, (e.g. John F. Kennedy, Philip Morris). Our named entity recognition system uses the semantic information associated to the verb of the sentence and the top ontology of WordNet in order to solve such problem.
   - Structural ambiguity. Different types of problems can be classified within structural ambiguity. All structural problems are caused by the use of prepositions and conjunctions. This type of words cause the problem of the limitation of the entity.
     - Use of Conjunctions. The appearance of a conjunction between capital words causes a doubt related to whether to add both capital words to the same entity ([Construcciones y Contratas S.A.] construye la autopistas... –[Construcciones y Contratas S.A.] builds the motorway...–) or to split them in two different entities ([Real Madrid y P. Vieira] acuerdan reunirse de nuevo... –[Real Madrid and P. Vieira] agreed to

meet again...–). The use of verb's number (singular, plural) can help to take a decision about to add or to split the words. According to [12], the study of triggers can also to help us to decide the formation of the entity.

- Use of Prepositions. The use of preposition as part of the named entity is mainly used in Organization and Location named entities ([Center for Scientific Researchs]...). This fact can cause some problems when two entities are related using this type of word (notificación del [COI] para [Miguel Indurain]... – notification of [COI] for [Miguel Indurain]). This kind of problem is the most difficult to solve. Our system uses a list of words, such as center, that usually appears next to a preposition and a set of capital words making up an unique entity.

4.4 No Named-Entity Recognition

Frequently, attributes from class diagram requiere values made up of bare nouns instead of proper nouns. For example, an attribute as *Position (Categoría)* from *Employer (Empleado)* can be fill with *Recepcionist (Recepcionista)* and *Waiter (Camarero)*.

The name reconstruction of attributes and classes, the WSD and the external evidence play an important role to recognize them from the text. The external evidence use the name reconstructed to match into the text all occurrences of the name and its semantic related word.

5 NLP Algorithm

Next, a general view of the NLP algorithm that has been used is shown:

1. The class diagram defined according a set of nomenclature constraints is translated into a textual XML structure showing attribute, class, and relationship names.
2. These names are reconstructed in order to obtain their equivalents in natural language.
3. Using the Domain Class, the main domain of the Class Diagram is recognized.
4. For each attribute name, its sub-domain is recognized by means of the class in which it is included. The attribute names together with their domains and sub-domains are used to access WordNet and extract their semantic related words (synonyms, hyperonyms, and hyponyms).
5. Using the Google web search engine we obtain the documents containing any of these attribute and class names, or their semantic related words. To guarantee the correct domain of the document, this web search is performed through the Google directory that includes the Domain Class defined in the Class Diagram.

6. The sentences where the goal words appear are parsed by means of a POS tagger and a WSD to solve the ambiguity, discarding incorrect semantics senses.
7. Once the texts were relevant information can be founded have been selected, then the Named-Entity recognition is applied to recognize proper nouns and bare nouns associate to goal words.
8. Finally, the extracted information is provided to a Filling module using XML tags.

6 Filling the Schema

The last step in the sampling data extraction task is the load of such data in the underlying data structures. Due to the fact that this process differs depending on the underlying storage system, our system provides a middle step in which an XML implementation-independent description of the data found during the information extraction process is provided. Such description does not need to be complete. Some attributes can belong to domains such as *datatime* or *number* and for them random data can be generated. In this random data generation the load module must take into account invariants associated to the Class Diagram constructs. For example, imagine we have associated an OCL expression to the class *TipoHabitación* (see Fig. 1) that establishes that any room price must be between 80 and 200 Euros (*precio¡200 and precio¿80*). Of course, the random numbers generated for such attribute must stick to such restriction to provide sound information.

Once this XML template has been modified with such random data, the actual load process of the system must be performed. In the actual stage of development, we have restricted ourselves to underlying relational structures which are directly derived from the modeled Class Diagram. This load module receives, in addition to the XML data file, another XML document that, by means of rules, gathers general constraints that must be fulfilled by the system population for a given purpose. As a further example, imagine we need at least one unpaid invoice (*Factura.estado=pendiente*, see Fig. 1) in order to be able to verify and validate the functionality associated to such situation. This fact would be modeled by means of the following rule:

¡TSampling¿ ... ¡rule context="Invoice"¿ ... ¡condition name="AtLeastOneInvoice" mandatory="yes" minOcc="1" maxOcc="" value="estado=pendiente"/¿ ... ¡/rule¿ ... ¡/TSampling¿

Inside this rule, we can observe how the mandatory character of the rule determines whether the NLP algorithm can finish or, on the contrary, must continue looking for new sampling data. The data set degree of compliance to the non mandatory rules determines its quality for the system purpose.

7 Conclusions

Prototyping is widely used to validate and verify the functionality of Information System applications. However, nowadays prototyping tools are more focused on

designing than data filling task. In this paper we have presented an automatic system that is able to avoid the hard manual task that developers of prototypes must perform in order to obtain test data. This system looks for real data in web documents to accomplish this goal. Taking advantage of Information Extraction techniques, our system uses a named entity recognizer to obtain the specific information to be related to classes and attributes of the DC. This DC must fulfill a minor set of constraints in order to aid the automation.

As a result, the prototype will be filled with real and understandable information that helps in the verification and evaluation process of the final application.

Despite this system has been successfully tested, several issues must be improved in further versions of our system. These issues are related to the following ideas:

- In our proposal a small ontology based on the Google domains has been used. However, a complete ontology must be defined to allow its use on whatever kind of domain.
- The semantic fine-grained of WordNet produces errors in the WSD module. The WSD module is forced to choose among different senses (very close related) despite all of them belongs to the same domain. To solve this problem, future work will apply the Magnini domain WordNet [6, 5] that clusters close related senses.

The main contributions of this work are:

- Entity recognition system based on UML Class Diagrams
- Ontology
- Data insertion strategy

References

1. R. Bell. Code Generation from Object Models. *Embedded Systems Programming*, 3:1–9, 1998.
2. A. Cucchiarelli, D. Luzy, and P. Velardi. Automatic semantic tagging of unknown proper names. In ACL, editor, *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL'98)*, pages 286–292, Canada, 1998.
3. Morgan Kaufman, editor. *Sixth Message Understanding Conference (MUC-6)*, Los Altos, Ca, November 1995.
4. G. R. Krupka. Description of the SRA system as used for MUC-6. In Kaufman [3], pages 71–86.
5. B. Magnini and G. Cavaglia. Integrating subject field codes into WordNet. In *Proceedings of the LREC-2000*, 2000.
6. B. Magnini and C. Strapparava. Experiments in Word Domain Disambiguation for Parallel Texts. In *Proceedings of the ACL Workshop on Word Senses and Multilinguality*, 2000.
7. D. McDonald. Internal and external evidence in the identification and semantic categorization of proper names, 1996.

8. A. Mikheev, M. Moens, and C. Grover. Named Entity Recognition without Gazetteers. In ACL, editor, *Proceedings of the 11th European Chapter of the Association for Computational Linguistics (EACL)*, pages 1–8, Norway, 1999.
9. A. Montoyo, A. Suárez, and M. Palomar. Combining Supervised-Unsupervised Methods for Word Sense Disambiguation. In Alexander Gelbukh, editor, *Proceedings of 3nd International conference on Intelligent Text Processing and Computational Linguistics (CICLing-2002)*, volume 2276 of *Lecture Notes in Computer Science*, pages 156–164, Mexico City, 2002. Springer-Verlag.
10. R. Morgan, R. Garigliano, P. Callaghan, S. Poria, M. Smith, A. Urbanowicz, R. Collingham, M. Costantino, and C. Cooper. Description of the LOLITA system as used for MUC-6. In Kaufman [3], pages 71–86.
11. MUC-7. Saic:science aplications international corporation. `http://www.itl.nist.gov/iaui/894.02`, 2001.
12. R. Muñoz, A. Montoyo, F. Llopis, and A. Suárez. Reconocimiento de entidades en el sistema EXIT. *Procesamiento del Lenguaje Natural*, 23:47 53, 1998.
13. B. Sundheim. Overview of results of the MUC-6. In Kaufman [3], pages 13–32.
14. OMG Unified Modelling Language Specification. http://www.rational.com/, 2001.

Automatic Help for Building and Maintaining Ontologies

Nadira Lammari and Elisabeth Métais

Laboratoire CEDRIC
CNAM, 292 rue Saint Martin 75141, Paris Cedex 03 - France
{lammari,metais}@cnam.fr

Abstract. "Is_A" links are the core component of all ontologies and are organized into "hierarchies of concepts". In this paper we will first address the problem of an automatic help to build sound hierarchies. Dependencies called "existence constraints" are the foundation for the definition of a "normalized " hierarchy of concepts. In the first part of the paper algorithms are provided to obtain a normalized hierarchy starting either from concepts or from instances using Boolean functions. The second part of the paper is devoted to the hierarchy maintenance: automatically inserting, merging or removing pieces of knowledge. We also provide a way to give synthetic views of the hierarchy.

1 Introduction

Information sharing from multiple heterogeneous sources is a challenging issue. There is now evidence that the adequate tool for dealing with semantically heterogeneous data is ontology. Numerous definitions may be found for "ontology" depending on their fields (databases, mathematics, linguistics, philosophy) [1]. However, one of these definitions is becoming predominant in the field of information systems: "an ontology is a formal conceptualization of a real world, sharing a common understanding of this real world". This definition fits so well to the semantic heterogeneity problem that one of the main application field for ontology is heterogeneous data sharing. Wache & al. present a survey of existing approaches to intelligent information integration after analyzing twenty-five multi-sources information systems based on ontologies [2].

An ontology provides various semantic links between concepts such as synonyms, antonyms, hyponyms/hypernyms (Is-A) and meronyms/holonyms (part-of). An extensive discussion of relation types is presented in [3]. Other links are supported by "canonical graphs" which specify relationships expected among the concepts involved in a given action. According to the domain portrayed by the ontology, particular kind of links may be supplied. For example, in the medical domain, links are required such as "located in", "has an effect on". Tools for information retrieval on the Internet need links relating ideas such as a link between "boat" and "fish".

Although the use of ontology is dramatically increasing for these last years, only a few researches have been undertaken concerning tools for helping in ontology maintenance. Among them, most works concern the integration of several ontologies. A

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 27–39, 2002.

first family of research has been based on Galois lattices, as were some on hierarchy integration [4]. Some practical tools have been proposed, among them we notice ONIONS [5] developed in the context of the GALEN project. The last tendency in research on ontology maintenance is based on description logic that allows the representation of many kind of links (hierarchy of subsumptions of concepts, canonical graphs of the verbs) into a powerful logic reasoning mechanism [6], [7].

Is-a links are the core of all ontologies and are organized into "hierarchies of concepts". In this paper we will first address the problem of an automatic help to build sound hierarchies of concepts. A type of dependencies called "existence constraints" are the foundation for the definition of a "normalized" hierarchy. Algorithms are provided in this paper to obtain a normalized hierarchy starting either from concepts or from instances. The second part of the paper is devoted to the maintenance of a hierarchy: inserting, merging or removing pieces of knowledge. Section 4 provides a mean to give synthetic views of a hierarchy.

2 Functionalities for Ontology Building

In this section we will show the elaboration of the Is-A hierarchy of an ontology starting either from the key-words describing concepts and their existence constraints or from a set of instances. Reverse mechanisms will be also presented with the purpose of building new hierarchies from both new concepts and legacy hierarchies.

2.1 Existence Constraints Definitions

Existence constraints belong to the same family as null values constraints in the relational data base context. In the literature, six types of constraints on null values are mentioned: null-free constraints [8], disjunctive existence constraints [9], existence constraints [10], subset, equality and exclusive constraints [11]. They were initially used to extend the relational theory to relations with null values. Furthermore, they (especially existence, equality and exclusive constraints) have been used to translate conceptual schemas (such as those of the NIAM method) into relational schemas [12], [13]. They have also been used in [14] to express dependencies between objects and to define precisely operations of creation and suppression of objects.

However, all the presented definitions don't take into account the semantic of the null value which can express, according to Codd [15], either the temporary absence of a value for an attribute (the value is temporarily unknown but applicable) or the inapplicability of this attribute to a tuple of a relation (the value will never be known since this attribute is inapplicable for this instance).

Therefore, the concept of existence constraints has been redefined in [16] in order to take into account the semantic of null values. Our definitions are based on the *applicability domain* of an attribute, which is the set of possible instances where this attribute is applicable i. e. the set of instances where this attribute can not have a null value expressing the inapplicability. When the applicability domain of an attribute is equal to all the possible instances of its relation, this attribute is qualified as mandatory.

We consider three kinds of existence constraints: the exclusive existence constraints, mutual existence constraints and the conditioned existence constraints. They express set relations, such as inclusion, equality or disjunction, existing between applicability domains.

• *An exclusive existence constraint* between two attributes x and y of a relation, denoted by $x \nleftrightarrow y$, captures the fact that in each tuple where x is applicable then y is not applicable and vice versa. It means that the applicability domains of x and y have no intersection.

• *A mutual existence constraint* between two attributes x and y of a class, denoted by $x \leftrightarrow y$, captures the fact that in each instance where x is applicable then y is also applicable and vice versa. In other words, the applicability domains of x and y are equal.

• *A conditioned existence constraint* between two attributes x and y, denoted by $x \mapsto y$, captures the fact that in each instance where x is applicable then y is also applicable but the converse is not always true. We also say that x requires y. In other words the applicability domain of x is included in the applicability domain of y.

Existence constraints have also been generalized to attributes sets. The reader interested by these definitions and by the completeness and soundness proofs of the defined inference rules systems can refer to [16]. In this thesis are also studied some aspects related to the inheritance of constraints.

2.2 Building the Is_A Hierarchy from Existence Constraints

The Is-A hierarchy of an ontology describes set of instances with properties called key-words. Its organization into an Is_A hierarchy consist on identifying groups of instances sharing the same key-words. Each group of instances refers to a concept. The identification of concepts can be done by analyzing the existence constraints between the key-words. For example, if we take an ontology O describing instances of an UoD by this set of key-words: {"moves", "transports", "flies", "floats"} and if we want to organize it into an Is_A hierarchy, we have to analyze the relationships between all the key-words. For example, if any instance (represented by O) having the property "moves" have also the property "transports" and vice versa, then our ontology will represent only instances having simultaneously the two properties. This hypothesis is translated by the mutual existence constraint: "moves" $\leftrightarrow$ "transports".

Moreover if we suppose that:

- some instances (represented by O) having the property "flies" have also the property "transports" ("flies" $\mapsto$ "transports"),
- some instances (represented by O) having the property "floats" have also the property "transports" ("floats" $\mapsto$ "transports"),
- and all the instances having the property "floats" don't have the property "flies" and vice versa ("floats" $\nleftrightarrow$ "flies"),

then we reduce the set of instances represented by O to the three following concepts: the concept representing instances having the properties: "moves", "transports", the concept representing instances having the properties: "moves", "transports" and "floats" and finally the concept representing instances having the properties: "moves", "transports" and "flies"

These concepts can respectively be called Vehicle, Plane, and Boat. They can be organized into the Is_A hierarchy of Figure 1.

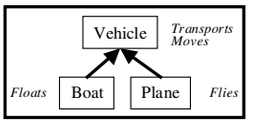


Fig. 1. A first example of an Is_A hierarchy of concepts

This technique allowing to build Is_A hierarchies from a set of key-words and a set of existence constraints between the key-words, has been already used in the context of OO and EER conceptual modeling [17]. It is called a normalization technique. It consists in automatic deducing of valid key-words subsets that are compatible with the existence constraints.

The algorithm used to build an Is_A hierarchy for an ontology is based on this definition. The sets of coexisting key-words are first deduced, by taking into account the mutual existence constraints. Then, by means of exclusive existence constraints, are derived the sets of key-words that may coexist. Finally, sets compatible with the conditioned existence constraints are selected among the sets of key-words that may coexist. These sets describe concepts represented by the ontology. These concepts are then organized into an inclusion graph and by inverting the direction of the graph inclusion arrows, we obtain an Is_A inheritance hierarchy.

Remark 1. Note that this technique of normalization can also derive multiple Is_A links. For instance, if we omit the existence constraint "floats" $\nleftrightarrow$ "flies" we will deduce the Is_A hierarchy of Figure 2a.

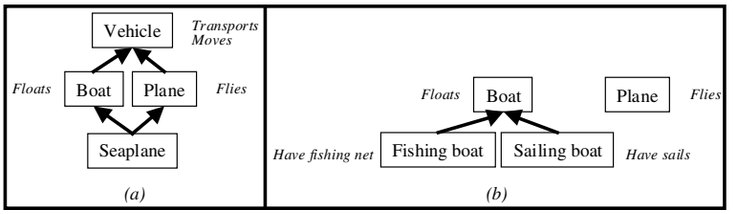


Fig. 2. Other examples of Is_A hierarchies

Remark2. This technique can supply more than one hierarchy. This case will arrive when instances don't share at least one key-word. For instance, if our ontology was defined only by the key-words "flies", "floats", "have sails" and "have fishing net" and if the relationships between the key-words was expressed through the following constraints: {"floats" $\nleftrightarrow$ "flies"; "have sails" $\nleftrightarrow$ "have fishing net"; "have sails" $\mapsto$ "Floats"; "have fishing net" $\mapsto$ "Floats"} then the normalization process would deduce the hierarchies described in Figure 2b. In order to make the ontology usable, all these hierarchies will be gathered under a top (T) node.

2.3 Building an Is_A Hierarchy from UoD Instances

Existence constraints are not always specified in the ontology. We have to derive them by analyzing the ontology instances. For this purpose, we proceed in three steps.

First we determine a Boolean equation describing all the different types of instances encountered in the ontology instances. Then, we transform the left part of this Boolean equation from a disjunctive form to a conjunctive form. Finally, by analyzing the different minterms of the transformed expression we derive the existence constraints. To illustrate this technique let us consider the following example. Suppose that instead of the set of existence constraints (those used to derive the hierarchy of Figure 1), we have the following representative sample of instances for the ontology O:

Instance I1 with the key-words "transports", "moves",
Instance I2 with the key-words "transports", "moves", "Flies",
Instance I3 with the key-words "transports", "moves",
Instance I4 with the key-words "transports", "moves", "Floats",
Instance I5 with the key-words "transports", "moves", Flies",
Instance I6 with the key-words "transports", "moves", "Flies".

This set of instances identifies three types of instances for the ontology O regarding the presence or the absence of properties:

Type 1: instances having only "transports" and "moves" as key-words,
Type 2: instances having only "transports", "moves" and "Flies" as key-words,
Type 2: instances having only "transports", "moves" and "floats" as key-words.

To express the fact that only the types of instances presented above exist in O, we exploit the notion of Boolean expression. In fact, if we associate to each of the key-words "transports", "moves", "flies" and "floats" respectively the Boolean variable X1, X2, X3, X4 that take the value 1 for a type of instances if the associated key-word is valued for this type and 0 otherwise, we can translate the fact that only the type of instances presented above are in the ontology O by the Boolean equation:

$$X1.X2.\overline{X}3.\overline{X}4 + X1.X2.X3.\overline{X}4 + X1.X2.\overline{X}3.X4 = 1$$

To solve this Boolean equation we can transform its left part from this disjunctive form to a conjunctive one and solve a set of equations. For our example, we have to solve the equivalent Boolean equation:

$$X1.X2.(\overline{X}3 + \overline{X}4) = 1$$

that can be solved by the resolution of the following equations:

$$X1 = 1\ ;\ X2 = 1\ ;\ \overline{X}3 + \overline{X}4 = 1$$

Each equation gives a solution that can be materialized by an existence constraint. For our example, the two first equations mean respectively that the key-words "transports" and "moves" are properties of every instances of the ontology. The third equation means that the key-words "flies" and "floats" are related by an exclusive existence constraint. So, every instances of the ontology having the property "flies" don't have the property "floats" and vice versa.

This technique that allows extracting from ontology instances existence constraints between the key-words of this ontology has been used in the context of the relational database reverse engineering. A formal description of this technique can be found in [18].

Once the set of existence constraints extracted, we can, by applying the normalization technique presented in the previous sub-section, deduce the Is_A hierarchy of the ontology described by a representative sample of instances. Each node of the hierarchy will represent a concept. The name associated to this concept is given by the designer of the ontology.

2.4 Extracting Existence Constraints between Key-Words from an Is_A Hierarchy

The normalization technique supplies hierarchies were any key-word of a concept is a property of all the instances represented by this concept. This technique of building hierarchies allows us to propose another reverse engineering technique, which consists in extracting existence constraints relating key-words of an ontology from the Is_A hierarchy of this ontology. We call this reverse engineering technique "the translation technique" because it translates all the semantic represented through the Is_A links into existence constraints. It is based on the following four rules:

Coexistence rule: any two key-words defining the same concept are linked by a mutual existence constraint.

Generalization/specialization rule: If in the hierarchy, a concept B inherits from concept A, then any specific key-word of B requires every key-word of A.

Specialization exclusion rule: if in the hierarchy, two concepts A and B do not share any children concept, then every specific key-word of A is related to every specific key-word of B by an exclusive existence constraint.

Multiple inheritance rule: if in the hierarchy, a concept D is the only direct parent of a concept A and a concept B then every couple composed of a key-word of A and a key-word of B requires every key-word of D.

For example, if we apply this translation technique to the hierarchy of Figure 1 we will obtain all the following constraints:
"moves"$\leftrightarrow$"transports"; "flies"$\mapsto$"transports"; "floats"$\mapsto$"transports"; "floats" $\nleftrightarrow$ "flies"
These constraints have been used in the sub-section 2.2 to derive the hierarchy described in Figure 1.

This presented technique has been already used in the context of modeling in order to optimize OO or EER conceptual schema [19]. It has been formally described in [16].

2.5 A Summary of the Presented Techniques

To conclude this section, we can say that the techniques used for building an Is_A ontology depend on the description of the ontology. We use only the normalization technique if the ontology is described by the set of key-words and the set of existence constraints relating the key-words.

We have also to use the extraction technique if the set of existence constraints is empty. We execute partially this technique if the set of concepts and the description of each concept are given.

We execute all the steps of the extraction technique if the ontology is described by a set of instances.

We can reverse engineer the normalization process (translation technique) and in this case the hierarchy is translated into a set of key-words and a set of existence constraints.

3 Functionalities for Ontology Maintenance

We will present in this section a set of functionalities aiming to automatically update the hierarchy. They allow us to add or remove concepts, hierarchies and instances.

3.1 Merging Hierarchies

Merging hierarchies is a key functionality in hierarchy maintenance. The algorithm associated to this functionality is:

```
Algorithm 1.
Input:  H1, H2, …..,and Hn the Is_A hierarchies to merge.
        W1, W2, …., and Wn respectively the sets of key-words used in
        H1, H2, …., Hn.
Output: H the set of Is_A hierarchies issued from the merge of H1, H2,
       ….. and Hn.
        Φ the Boolean equation associated to H. We note L(Φ) the left
         part of Φ.
Begin
    L(Φ) = 0.
    Solve name conflicts between W1, W2, …., and Wn.
    Assign to every key-word of W1 ∪ W2 ∪ ….. ∪  Wn a Boolean variable.
    Let B be the set of these variables.
    For every hierarchy Hi Do
        Translate Hi into the Boolean equation φi.
        Let L(φi) be the left part of φi.
        For every minterm T of L(φi) Do
            For every variable X of B Do
                If X is not in T and  X̄  is not in T Then T = T.  X̄  EndIf
            EndFor
        EndFor
        L(Φ) = L(Φ) + L(φi).
    EndFor
    Translate Φ into a hierarchy. H is the result of this translation
End.
```

3.2 Adding Concepts

Adding concepts to an ontology affects its Is_A hierarchy. Since a new concept can be considered as a hierarchy of an ontology, adding a new concept can be done by merging the hierarchies. The algorithm associated to this functionality is:

```
Algorithm 2.
Input:  C the set of new concepts
        H the initial hierarchy
output: H' the hierarchy resulting from the merge of the C with H
Begin
    Ω = ∅
    For every concept Ci of C Do
        Determine the hierarchy Hi associated to the concept Ci by
        applying the normalization technique.
        Ω = Ω ∪ Hi
    EndFor
    Merge the hierarchies contained in Ω ∪ H using the functionality
    presented in 3.1.
End
```

3.3 Adding Instances

Adding instances to an ontology affects the Is_A hierarchy only if the instances are not represented by one of the ontology concepts. As a consequence, the algorithm that will add instances to the ontology will encompass these following steps:

```
Algorithm 3.
Input:  I the set of new instances.
        H the initial hierarchy.
OutPut: H' the hierarchy resulting from the merge of the I with H
Begin
   Calculate the subset I' of I such that any instance of I' is not
   represented by a concept of H.
   Reverse engineer the set I' into a set of hierarchies Ω using the
   technique described in 2.3.
   Merge the hierarchies contained in Ω with H using the functionality
   presented in 3.1.
End
```

3.4 Deleting Concepts

We consider two strategies to delete concepts. The first strategy consists in deleting only the concepts. The second one consists not only in deleting the concept but also in deleting all its descendants if they are not descendants of any concept, which is not an ascendant of the concept to delete.

For example, if we decide to delete the concept C3 from the hierarchy Figure 3a, we will obtain the hierarchy of Figure 3b if we apply the first strategy and the hierarchy of Figure 3c if we apply the second one.

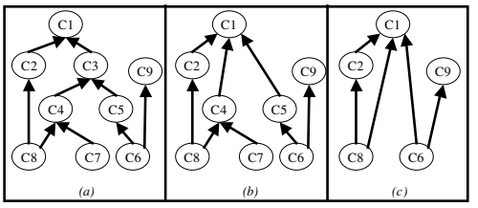


Fig. 3. Resulting hierarchies after applying a chosen deleting strategy

More formally, let:

- C be the set of concepts represented in an hierarchy H of an ontology O,
- C1 be the set of concepts to delete from H. We have $C1 \subset C$,
- D_{C1} be the set of the descendants of C1. $D_{C1} = \bigcup_{x \in C1} D_x$ such that D_x represents all the descendants of the concept x of C1,
- A_{C1} be the set of the ascendants of C1. $A_{C1} = \bigcup_{x \in C1} A_x$ such that A_x represents all the ascendants of the concept x of C1,
- C2 be all the concepts of H that are not in C1 or in D_{C1} or in A_{C1}. $C2 = C-(C1 \cup D_{C1} \cup A_{C1})$,
- D_{C2} be all the descendant of C2.

The application of the strategy 2 implies also the suppression of all the concepts that are in D_{C1} but not in D_{C2}. Therefore, the algorithm that allow us deleting concepts from an hierarchy is sketched bellow:

```
Algorithm 4.
Input:  H the initial hierarchy.
        C1 the set of concepts to delete from H
Output: H' the resulting hierarchy
Begin
   If the designer has chosen the strategy 2 Then
        Determine DC1 the set of the descendants of C1.
        Determine C2 the set of concepts of H that are not in C1 or
        in DC1 .
        Determine DC2 the set of the descendants of C2
        Add to C1 the set DC1 - DC2
   EndIf
   For each concept c of C1 Do
       For each specific key-word w of c Do
           For each direct descendant d of c Do
                 Rename the name of w into w' such that the name of
                 w' = the name of w//1name of d.
                 Add to d the key-word w'
           EndFor
       EndFor
       For each direct descendant d of c Do
           For each direct ascendant a of c Do
                Add an Is_Link from d to a
           EndFor
       EndFor
       Delete c from H
       Replace the name of every renamed key-word by its initial name.
   End For
End
```

3.5 Hierarchy Deletion

This functionality can be executed by re-use of the precedent functionally. In fact, since a hierarchy can be considered as a set of concepts, its removing can be viewed as a deletion of a set of concepts. Each of its concepts is described by its set of specific key-words and those inherited from its parents.

We can also for this functionally allow the designer choosing the strategy of deletion. If the chosen strategy is the strategy 1 then the functionality will delete only the concepts of the hierarchy to delete. Otherwise, it will delete each descendant of each concept of the hierarchy which is not descendant of any concept that is not an ascendant of any concept to delete. For example, suppose that the designer wants to delete from the hierarchy Figure 3a the sub-hierarchy composed by the concepts C3, C4, C5. Then, if the chosen strategy is the strategy 1 the resulting hierarchy is described in Figure 4a. Otherwise, it is described in Figure 4b.

Therefore, the algorithm to delete a sub-hierarchy from a hierarchy is given bellow:

[1] // is the operator of concatenation

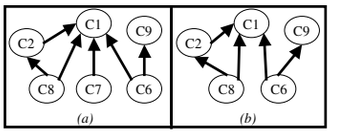


Fig. 4. Resulting hierarchies after deleting a sub-hierarchy according to a chosen strategy

```
Algorithm 5.

Input:  H the initial hierarchy
        H1 the sub-hierarchy to delete from H
Output: H' the resulting hierarchy
Begin
   Determine the set C1 of concepts of the hierarchy H1.
   Remove C1 from H (use the algorithm 4). H' is the resulting hierarchy
End
```

4 Creating Synthetic Views of Ontology

The Is_A hierarchy of an ontology can contain a huge number of concepts and become hard to understand.

This functionality is used to give to provide a synthetic view of the Is_A hierarchy. For example, if the user of the hierarchy of Figure 2b decides to regroup the concepts "Boat", "Fishing boat" and "Sailing boat" in one concept then this functionality will supply it with the hierarchy of Figure 5.

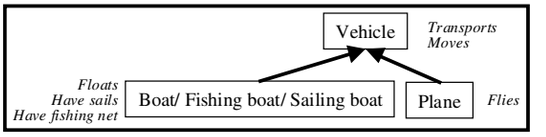


Fig. 5. The resulting hierarchy after merging concepts of Figure 2b

This functionality can merge some concepts, belonging to the same hierarchy, according to a chosen grouping. Its aim is to perform the chosen groupings and replaces the undone links by existence constraints that will give to the user an idea about the possible instances that can be represented by the grouping concept. For the above example, if the user wants to have a description of the concept "Boat/Fishing boat/Sailing boat" which is the result of the grouping of "Boat", "Fishing boat" and "Sailing boat", the functionality will supply it with:

- this set of key-words: "Floats", "Have fishing net" and "Have Sails",
- and this set of existence constraints:

$$\text{"Have fishing net"} \mapsto \text{"Floats"};\ \text{"Have Sails"} \mapsto \text{"Floats"};$$

$$\text{"Have fishing net"} \nleftrightarrow \text{"Have Sails"};$$

that translate the fact that all the instances represented by this concept have the property "Floats" and that some of them have either the property "Have Sails" or the property "Have fishing net".

A proposed grouping of concepts must be *valid*, therefore it must respect the inclusions of semantics expressed through the inheritance links. We consider only two types of valid groupings (see Figure 6). The first one represents all the groupings defined by a set of concepts whose all descendants are part of the grouping. The second type includes all the other cases.

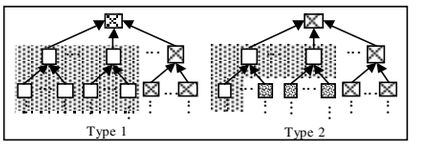


Fig. 6. Examples of valid groupings

A grouping G of concepts is valid if and only if every external ancestor of a concept of G is also an ancestor of each concept of G.

To obtain the final hierarchy, we need first to compute the new concept that replaces the merged concepts. Then this concept is integrated into the initial hierarchy. Its integration depends on the type of grouping (see Figure 7). For the grouping of Type 2, all the descendants of the merged concepts which are not in the grouping will inherit key-words that are not applied for the instances they represent. Therefore, we have to inform the user about this, through an annotation.

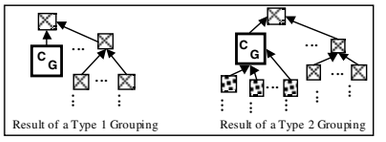


Fig. 7. Result of the groupings according to their type

The algorithm that merges concepts is given bellow:

```
Algorithm 6.

Input:  H the initial hierarchy
        C the set of concepts to merge in H. C = {c1, c2, …, ci, …., cn}
        W the set which each element is the set Wi of key-words of the
        concept ci of C.
Output: H' the resulting hierarchy
Begin
   Compute ROOT = the set of concepts of C that have at least one parent
   (if it exists) not in C.
                 i=n
   Determine W =  ⋃ Wi  .
                 i=1
   Add G to H
   Translate the hierarchy of the concepts of C into existence
   constraints.
   Annotate G by the obtained set of existence constraints.
   Create the inheritance link from G to the direct parent of each
   concept of ROOT.
   Delete from the hierarchy all the concepts that were in C.
   If the grouping of C is of Type 2 Then
```

```
        For each concept B which was not in C Do
             If B is a direct child of any concept A of the C Then
                  Create the inheritance link from B to G
                  S = the keys words of G which are not in B
                  Annotate the concept B: "the key-words of S are not
                  properties of instances represented by this concept.
             EndIf
        EndFor
   EndIf
End
```

If the user requires for a non-valid grouping, an extension to this work should be to propose the closest valid grouping.

5 Conclusion

In this paper we have shown that based on the notion of "existence constraints" we can build sound hierarchies of concepts. After stating their definition, we provided an algorithm for building normalized Is-A hierarchies starting from a set of concepts described by key-words This algorithm uses relationships between key-words expressed through existence constraints. If these constraints are lacking, a method for extracting Is-A hierarchies from instances, by the mean of Boolean functions, is proposed. All the reverse transformations, i.e. from an Is-A hierarchy to Boolean functions and existence constraints are also possible, in order to build new hierarchies from legacy hierarchies. Algorithms are provided for inserting or deleting concepts, instances and pieces of hierarchies. This paper presents also a technique for creating synthetic views of ontology's.

These techniques supply users with functionalities that allow them to maintain their ontologies. They have been implemented in a tool developed in JAVA. To give an idea of the performances, it takes about one minute to derive the normalized schema for a hierarchy of twenty-five classes.

Further works will concern an automatic help to decide the scope of the synthetic views when the user initially requires a non-valid grouping. Other perspectives are related to the other links provided by an ontology (part-of, related-to, etc…).

Acknowledgement

The authors would like to thank their colleagues Jacky Akoka, Isabelle Comyn-Wattiau and François-Yves Villemin for their valuable comments.

References

1. Guarino, N. editor: Formal Ontology in Information Systems. IOS Press, ISBN 90-5199-399-4, 1998.
2. Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H. and Hübner S.: Ontology-Based Integration of Information - A Survey of Existing Approaches. Proceedings of the IJCAI'2001 Workshop on Ontologies and Information Sharing, Editors: Stumme G., Maedche A. and Staab S., Publisher: CEUR-WS, Seattle, USA, Vol. 47, August 4-5 2001.

3. Storey, V. C.: Understanding Semantic Relationships. VLDB Journal, 2, 455-488, 1993.
4. Johannesson, P.: Supporting schema integration by linguistic Instruments. NLDB'95, Versailles (France), June 1995.
5. Gangemi, A., Pisanelli, D. M., Steve, G.: Ontology Integration, Experiences with Medical Terminologies. Guarino N. editor, Formal Ontology in Information Systems, IOS Press, ISBN 90-5199-399-4, 1998.
6. Franconi, E.: Logical form and Knowledge representation: toward a reconciliation. Working Notes on the AAAI Fall Symposium on Knowledge Representation Systems based on Natural Language, Cambridge MA, November 1996.
7. Calvanese, D., Giacomo, G., Lenzerini, M., Nardi N. and Rosati R.: A Principle Approach to Data Integration and Reconciliation. Data Warehousing Workshop on Design and Management of Data Warehouses (DMDW'99), 1999.
8. Atzeni, P. and Morfuni, M.: Functional Dependencies and Constraints on Null Values in Databases. Information and Control 70 (1), 1986, pp. 1-31.
9. Goldstein, B.: Formal Properties of Constraints on Null Values in Relational Databases. Technical report 80-013-REV, University of New York State at Stony Brook, Department of Computer Science. Published in November 1980 and reviewed in July 1981.
10. Atzeni, P. and Morfuni, M.: Functional Dependencies and Existence Constraints in Database Relations with Null Values. ICALP 1984, pp. 69-81.
11. Kornatzky, Y. and Shoval, P.: Conceptual Design of Object-oriented Database Schemas using the Binary-Relationship Model. Data and Knowledge Engineering, volume 14, number 3, (Elsevier, 1995), pp. 265-288.
12. Blaha, M., Premerlani, W. and Shen, H.: Converting OO Models into RDBMS Schema. IEEE Software, IEEE Computer Society Publisher (May 1994) pp. 28-39.
13. Halpin, T.A.: A Fact-oriented Approach to Schema Transformation. Proceedings of MFDBS'91, Rostock, Lecture Notes in Computer Science Vol. 495, Springer-Verlag, 1991, pp. 342-356.
14. Wei, G. and Teorey, T. J.: The Orac Model, an Unified View of Data Abstractions. Proceedings. of ER'91 (10th Entity-Relationships Approach Conference), San Mateo, California, USA, October 1991, pp. 31-58.
15. Codd, E. F.: The Relational Model for Database Management. Version 2. Addison-Wesley Publishing Company, 1990.
16. Lammari, N.: Réorganisation des hiérarchies d'héritages dans un schéma conceptuel objet. Ph.D. thesis, Conservatoire National des Arts et Métiers, Paris, France, October 1996.
17. Lammari, N., Laleau, R., Jouve, M. and Castellani, X.: Deriving Normalized IS-A Hierarchies by Using Applicability Constraints. CAISE'96 (Conference on Advanced Information Systems Engineering), Heraklion, Crete, Greece, Lecture Notes in Computer Science Vol. 1080, Springer-Verlag, 1996, pp. 562-580.
18. Lammari, N.: An Algorithm to Extract IS-A Inheritance Hierarchies from a Relational Database. Proceedings of ER'99, LNCS 1728, Paris, 1999.
19. Lammari, N., Jouve, M., Laleau, R. and Castellani, X.: An Algorithm for IS-A Hierarchy Derivation. Proc. OOIS'94 (International Conference on Object-Oriented Information systems), London , Springer-Verlag, 1994, pp. 469-479.

Semi-automatic Content Extraction from Specifications*

Krishnaprasad Thirunarayan[1], Aaron Berkovich[2], and Dan Sokol[2]

[1] Department of Computer Science and Engineering
Wright State University, 3640, Col. Glenn Hwy, Dayton, Ohio-45435, USA
tkprasad@cs.wright.edu
http://www.cs.wright.edu/~tkprasad
[2] Cohesia Corporation
First National Plaza, Suite 1414, Dayton, Ohio-45402, USA
{aberkovich,dzsokol}@cohesia.com
http://www.cohesia.com

Abstract. Specifications are critical to companies involved in complex manufacturing. The constant reading, reviewing, and analysis of materials and process specifications is extremely labor-intensive, quality-impacting, and time-consuming. A conceptual design for a tool that provides computer-assistance in the interpretation of specification requirements has been created and a strategy for semantic-markup, which is the overlaying of abstract syntax ("the essence") on the text, has been developed. The solution is based on the techniques for Information Extraction and the XML technology, and it captures the specification content within a semantic ontology. The working prototype of the tool being built will serve as the foundation for potential full-scale commercialization.

1 Background

Materials and process specifications are a fact of life for any organization involved in complex manufacturing, particularly for companies in industries such as aerospace, automotive, and materials. Specifications ("specs") are often comprehensive and voluminous covering hundreds of different key characteristics such as the material properties, testing parameters, marking requirements, etc. Specs are basically presentations of numeric data with accompanying text and graphics that explain the quantitative information [1,2].

To illustrate the impact that specs have on the cost of a product, one only needs to consider the on-going activities within a typical materials producing organization. Sales personnel read the specs at the time of design to identify every processing step and component that impacts the price. Design Engineers refer to specs to determine the requirements for creating a design. Manufacturing Engineers analyze spec requirements to define the appropriate operations

* This work was supported in part by NSF SBIR Phases I and II Grants DMI-0078525 (1999-2002). It does not necessarily reflect the opinions of NSF.

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 40–51, 2002.

and detailed data to include in the process plan. Quality Assurance personnel interpret the specs to define the acceptance limits and methods required for inspecting and testing. The use of paper-based specs is extremely labor-intensive, quality-impacting, and time-consuming.

Specs historically have been handled as simple textual documents. In order to do any meaningful analysis, comparison, and integration, one should extract numerical, tabular, or graphical data and operate on that data in the fashion intended by its meaning.

Typically, a spec issued by societies such as SAE, ASTM, AMS, etc is identified by the name of the issuing organization, title, spec number, issue/revision date, etc. The spec describes requirements on the processing of a material (alloy) in the mill, and the capabilities that it should possess eventually. The former includes its composition, melt method used, heat treatment, dimensions, marking and packaging requirements, etc. The latter is ascertained using tests such as tensile test, stress rupture test, macro- and micro-examinations, hardness test, etc. As part of defining an "intelligent" structure for specs, Cohesia created the Specification Definition Representation (SDR) as an ontology to articulate the semantic view of the components that comprise a spec, and capture the user's interpretation of it [3]. SDR introduced constructs such as *Procedures* to indicate boundaries for standards requirements such as chemical composition, tensile test, melt method, etc. Procedures are composed of elemental *Characteristics* that describe the requirements that are essential for performing the associated process (e.g., carbon content, yield strength, minimum temperature range, etc). In fact, a procedure encapsulates collections of characteristic-value pairs glued together using suitable connectives. The SDR technology has been incorporated into a commercial software system called MASS (Management and Application of Specifications and Standards). SDR permits defining a common library of industry terms (domain library) and a consistent representation structure that allows MASS to retrieve, compare, and combine specifications to drive manufacturing process.

Users of MASS capture their interpretation of spec requirements by using an intellectual process of reviewing the hard copy specs and manually converting the data into SDR. The SDR is a simple low-level tree language that is efficient for symbolic manipulation, but is very removed from the spec text. In order to raise the level of abstraction, Specification Definition Language (SDL) was designed and implemented. The Specification Studio is an Integrated Development Environment (IDE) for creating and editing specifications in SDL, and compiling them into an equivalent lower-level SDR for use by MASS. The IDE provides a convenient interface to search and use the Domain Library of controlled terminology. From experience, the analysts have observed an emergence of a pattern of "semantic markup", which is composed into the more rigorous SDL. This process of conversion is still laborious, and is subject to the competence of the individual performing the conversion.

2 Problem Statement

The focus of manual content extraction is to convert paper-based specs (using an appropriate domain library which defines standard vocabulary) into an "equivalent" formalized description in SDL. The extractor relies on his/her experience in interpreting the specs, and for rendering it in the SDL form using domain library vocabulary. Even though there is no rigid requirement either on the technical vocabulary used, or on the format, related specs with common origin share similar structure.

Semi-automatic content extraction involves recognition of phrases in spec that are associated with requirements, and subsequent synthesis of SDL fragments, to assist an extractor. Conceptually, this can be carried out in two steps: (a) Automatically recognize domain library terms present in spec and mark them up appropriately, and (b) Manually organize this information to be able to generate SDL from it. The entire approach is ontology driven, since the ontology is captured by the SDL and the domain library of terms. Semi-automatic approach was explored to improve the quality and efficiency of extractions by minimizing transcription errors, and by automating some of the routine mechanical tasks. A realistic yardstick of success is the extent to which mechanical and routine aspects of extraction can be codified and automated, while simultaneously deferring the more difficult and irregular portions to a human extractor.

This paper describes the approach taken, the progress made, and hurdles encountered in our efforts.

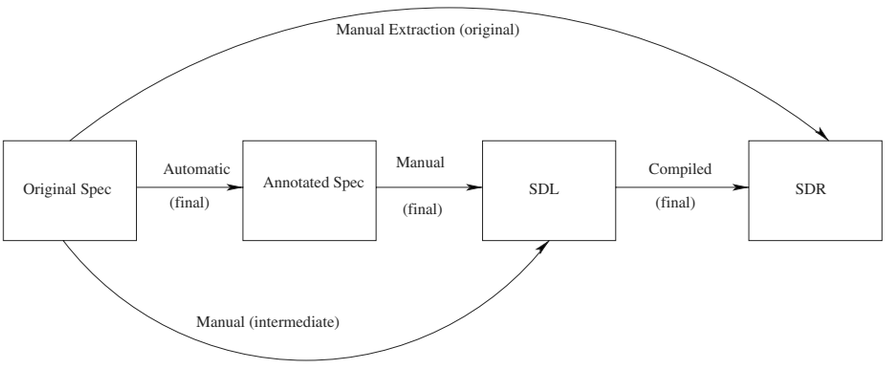


Fig. 1. Semi-automatic Approach to Semantic Markup

3 Related Work in Information Extraction

It is useful to view specs in terms of the "writing styles" [4].

Structured Text: Database Tables, Weather reports found on the web where the fields are delimited with HTML tags, etc.

Free Text: Grammatical texts such as newspaper articles, magazine stories, etc, that contain complete English sentences.

Semi-structured Text: Ungrammatical texts such as Advertisements, Seminar announcements, etc.

We reviewed research on Information Extraction pertinent to content extraction. These technologies included pattern matching systems [4,7,8], sentence analyzers [9], automated dictionary builders [4,10,13,21], W3C related initiatives [20] etc. The DARPA-sponsored Message Understanding Conferences (MUC) provided a major source of insight into the progress in information extraction technology [5]. The MUC tasks consisted of skimming through free-text articles in a limited domain, and filling in well-specified templates.

IE technology has evolved greatly in the last decade and there are also several commercial products and search engines that classify documents based on the subject matter. Unfortunately, these tools could not be used directly because there is significant difference between processing general topic documents and highly technical documents: (1) Specs usually contain much more domain specific vocabulary. (2) Ambiguities in specs are of a different nature than those in general-topic documents. (3) Technical terms are often created by combining existing terms [15].

4 Overall Approach

In order to understand the theoretical limitations of our strategy, and the practical problems to be overcome, the following approach was taken:

- Develop a catalog of ambiguities present in specs that can stymie even a human extractor. This catalog sheds light on situations impossible to deal with automatically, and that can potentially jeopardize the soundness of any extraction algorithm.
- Investigate the nature of association between the original spec and its translation that facilitates semi-automatic extraction and verification.
- Improve domain library searches and make it robust with respect to semantics preserving variations.
- State a practically important variant of the general extraction problem that can be used to demonstrate semi-automatic techniques.
- Study the role and application of XML technology in transforming a spec [14,15,16,17].
- Finally, describe practical problems to be tackled for semi-automatic fine-grained extractions of extant specs into SDL.

5 Details

We attempted to systematically evolve the manual extraction tool (Spec Studio) into a computer-assisted extraction tool with focus on improving extractor

productivity. The approach taken has an "industrial" rather than "academic" flavor because we attempted to deal with real-world specs rather than contrived ones.

5.1 Extraction Ambiguity

Informally, ambiguity means "open to interpretation". Ideally, one should have a unique interpretation for a spec, but in reality, a spec can be incomplete (due to missing information) or inconsistent (due to over-specification or multiple definition), or unclear (due to convoluted wording).

Manual extraction consists of: (1) Identifying and understanding requirements described in a spec, typically written in English. (2) Mapping specs terminology into Domain Library Terms (DLTs). (3) Identifying appropriate SDL connectives to tie the requirements together.

Problems arising in carrying out the above tasks are called extraction ambiguities. The motivation for studying these is to better understand aspects that are outside the realm of automation, to develop a nomenclature for describing ambiguity in the context of extraction from specs, and to build concrete examples to assist an extractor trainee. We skip the details and illustrative examples, for brevity.

5.2 Literal Translation

Conceptually, every piece of information in SDL and SDR translation owes its existence to a phrase in spec (and possibly, in domain library). Thus, for verification (proofing) purposes, it is important to maintain a one-to-one correspondence between fragments of spec and its translation. Such translations are said to be literal.

The existence of a literal translation depends on the target language and the translation scheme. For example, SDR extractions of fine granularity cannot be literal, while SDL extractions can be made literal to a reasonable extent. To see this dependence, consider the sources of non-linearity: (1) The extractions can contain duplicated information that are shared in spec. (2) The extractions can rearrange information present in spec. (3) Tables and footnotes abbreviate information in irregular and complex ways.

Further, a semi-automatic approach to extraction is feasible only if the automatically generated partial translations are intelligible to the human extractor in the context of the original spec, for subsequent modification.

5.3 Domain Library Search Engine

In the past, an extractor mapped a phrase associated with requirements in a spec to a Domain Library Term (DLT) by guessing the corresponding term and verifying it against the domain library. A typical domain library[1] has about

[1] Cohesia creates and maintains domain libraries for in-house use and for use by its clients such as GE, Alcoa, Allvac, etc.

10,000 phrases. The effectiveness of the search can be substantially improved by having the Spec Studio accept an arbitrary phrase and automatically determine the relevant DLT(s) contained in it. To this end, a domain library search engine was developed that improves the hit rate and enhances the flexibility of DL searches, while balancing recall with precision [18,19].

Algorithmic Details: Expression Matcher. The algorithm was designed with the following observations in mind: (1) An input phrase may contain multiple DLTs. (2) The words of a DLT contained in the input phrase may not appear contiguous. (3) Consonants are significant, and "correct" spellings may differ in the vowels. (4) Robustness with respect to misspellings such as transposition of letters or missing vowels is desirable. (5) Ordinary stemmers do not work for the words appearing in the DLTs. Instead, customized tables of prefixes and suffixes were developed to deal with situations that arise in practice. (6) The algorithm was skewed towards recall than precision as a human extractor is overseeing the outcome.

The overall goal of the matching algorithm is to map a phrase to a list of approximately semantically equivalent DLTs. Its main steps are sketched below.

```
List[Phrase] dl, dlwm, inwm;
Phrase ip;     Integer mt;
List[Phrase] dlts;

Step 1: Read list of Domain Library terms (phrases) into dl.
Step 2: Tokenize and construct list of (non-stop) words in the
        domain library terms into dlwm, maintaining links from
        each word back to the DLTs it appears in.
Step 3: Read-in input phrase and match-threshold into ip and mt,
        respectively.
Step 4: Tokenize and construct list of non-stop words into inwm.
Step 5: Determine matching words in dlwn corresponding to each
        word in inwm.
Step 6: Construct list of DLTs that contain these matching words.
        Evaluate each DLT and filter good approximations included
        in ip.
```

A word is matched at four levels: exact, case-insensitive, normalized (based on consonants), and sorted normalized (based on sorted string of consonants). The exact match requires the case and whitespaces to match. Case-insensitive match converts the inputs to lowercase prior to matching. Normalized match deletes all vowels and special characters (such as "-", "?", etc) prior to matching. Sorted normalized match orders the consonant string before matching. If these checks do not yield a match, the strings are then tested for variations in prefixes and suffixes. For this purpose, sorted normalized representations are compared and common letters are eliminated. Of the remaining letters, if they constitute differences accountable using a customized table of viable suffixes or prefixes, then a

match is declared. (Note that the general algorithm also deals with abbreviations by expanding them and aliases by normalizing them to preferred terms.)

5.4 Automatic Tagging

The text of a spec can be annotated to different levels of detail in order to make explicit mechanically processable information. The following table summarizes the various extraction methods (ontologies[2]) used in the industry (and companies such as GE, Alcoa, etc) today, reflecting the different granularity of extractions – from the most abstract to the very fine:

Method	Qualifiers	Requirements	Procedures	References
D	Spec Class Only	In notes	Not used	In notes
C	Spec Class, Product, Alloy	In notes	Not used	In notes
B	Many Qualifiers	In CV pairs and notes	Used	Retrieved
A	Many Qualifiers	In CV pairs, etc	Used	Retrieved

In order to understand the practical problems with semi-automatic extraction, we attempted to automate Method C Approach to Extraction, which involves generating SDL form that contains Qualified Notes labeled with a Procedure name or a Paragraph heading and the note bodies containing paragraphs with information relevant to the label. The qualifiers are the characteristics – product, product type, spec class, and alloy. Later, we wish to attempt Method B Approach to Extraction.

Two implementations of Method C Approach to extraction were pursued: the first involved direct translation of the spec into SDL, and the second involved indirect translation of the spec into SDL via an XML Master. The former implementation enabled convenient integration into the existing Spec Studio, while the latter implementation enabled applying XML Technology to further enrich literal translation strategy by embedding extraction fragments into the original spec. In other words, the spec can be annotated using XML-tags (derived from the ontology and the DLTs) that abstract the requirements, and keep the spec fragment and its abstraction tightly coupled. Subsequently, suitable views of the XML source can be generated using filters and transformations written in XSLT. In the long run, this technique can be used to apply other extraction methods to the (sufficiently annotated) XML Master.

Algorithmic Details: Design of Software Modules. The syntax of nested paragraphs can be expressed as extended regular expressions, and recognized using a lexical analyzer generated by FLEX. The matching algorithm for phrases has been coded in C++. The generated intermediate representations are in XML, with multiple "views" obtained using XSLT.

[2] According to Gruber, an ontology is an explicit specification of a conceptualization, which is an abstract, simplified view of the world that we wish to represent for some purpose.

The XML markup for Method C Approach can be carried out in four steps: (1) Automatically recognize nested paragraph structure of the spec and generate suitable tags. (2) Automatically recognize and tag qualifiers (and procedures) present in the spec. (3) Automatically generate various views of the XML source using transformations described in XSLT. (4) Finally, verify, and possibly amend, the resulting translations manually.

Preprocessing. Save the spec available in MS-WORD format as a plain text with line breaks. Alternatively, scan and OCR paper-based spec to obtain the text in electronic form.

Structure Markup of Spec. Paragraphs (numbered sections) are enclosed within tags `<section level=‘‘...’’ id=‘‘...’’>` and `</section>` to mark the beginning and the end of a (possibly nested) paragraph respectively, where level indicates the nesting level starting at 1, and id is the paragraph number string given as a period-separated sequence of numbers. The paragraph heading is enclosed within `<sectionHeading heading=‘‘...’’>` and `</sectionHeading>`. In addition to preserving the logical organization of the spec, if the original layout information is also desired, then the XML master must also retain indentation information. In practice, the FLEX code has to deal with subtle problems and idiosyncrasies of the saved spec.

Tagging Phrases Corresponding to DLTs for Method C Extraction. Document heading contains information about specification name, revision, products, spec class, and alloy name. The paragraph text can contain references to products, spec class, and alloy name. These are recognized using the appropriate fragment of domain library and tagged as follows:

Products: `<product DLT=‘‘...’’> ... </product>`
Spec class: `<specClass DLT=‘‘...’’> ... </specClass>`
Alloy: `<alloy DLT=‘‘...’’> ... </alloy>`

Method B extractions will need identification of characteristic-value pairs and grouping them under relevant procedures.

Generating Various Views from the XML Master. It is possible to generate various views of the spec from the XML master by applying suitable XSLT stylesheets using Apache's Xalan processor. The views of interest are:

1. The original text of the spec, with the identation preserved as much as possible, for verification.
2. The HTML version of the same for display in a Web-browser.
3. The Method C Extraction that generates syntactically correct SDL with the paragraph text presented as qualified notes (details of which are skipped for brevity).
4. The Method C Extraction that generates syntactically correct SDL with the paragraph presented as qualified notes using paragraph numbers in place of paragraph bodies, to deal with copyright restrictions that prohibit duplication of the spec text verbatim in another document.

5.5 An Example of Method C Extraction

Due to space limitation, we have condensed the example drastically. A fragment of the original specification, GE.txt, is shown below.

```
            SPECIFICATION NO.  B50TF104
                    ISSUE NO.  S4+AM1
                         DATE  January 9, 1996
                    PAGE 1 OF  10
                    CAGE CODE  07482
                   SUPERSEDES  B50TF104-S3
...
ALLOY BAR, FORGINGS, AND RINGS
(INCONEL ALLOY 706)
...
3.2.1  Material Condition.  Material shall be delivered in the following
condition specified on the purchaser order:

(a)     Bar and rod shall be hot finished, heat treated per CLASS A, and
descaled; round bars shall be ground or turned.  Forgings shall be
as ordered.

(b)     Flash welded rings shall not be supplied unless specified or
permitted on Purchaser's part drawing.
...
```

GE.txt is tagged to obtain GE.xml which is then transformed into the following Method C extracted GE.sdl with text suppressed using XSLT stylesheet.

```
document [B50TF104]
{
  title = "ALLOY BAR, FORGINGS, AND RINGS(INCONEL ALLOY 706)";
   org  = "GE Aircraft Engines";
   type = "specification";
  define APM
  {
       [Products] is "Bar";
       [Products] is "Forgings";
       [Products] is "Rings";
  }
  using APM;

  revision [S4+AM1]
  {
...
  if
       ( [Product Type] is "Bar"  or [Product Type] is "Rod"  or [Product Type] is "Bar" or
          [Product Type] is "Forgings"  or [Product Type] is "Rings" ) and
          ( [Spec Class] is "A" )
    then
    {
        note " = Material Condition."
        "Shall be in accordance with paragraph 3.2.1  "
    }
...
  }
}
```

The following script summarizes the actual processing that a WORD document GE.doc, saved as GE.txt, goes through. Specifically, the tagged file GE.xml is processed using the various stylesheets to generate the SDL forms and recover the original text. (Even in this simplified setting, "context-free" analysis causes tagging errors such as misidentifying "section" as a product type.)

```
flex structTagger.flex
gcc lex.yy.c -lfl
a < GE.txt > GE.xml
java org.apache.xalan.xslt.Process -in GE.xml -xsl CSDLStylesheet.xsl -out GE.sdl
java org.apache.xalan.xslt.Process -in GE.xml -xsl CExpSDLStylesheet.xsl -out GE.exp.sdl
java org.apache.xalan.xslt.Process -in GE.xml -xsl OriginalStylesheet.xsl -out GE.org.txt
```

5.6 Practical Problems for Fine Grain Extraction

Here is a list of hard problems to be overcome for full-scale commercialization of this technology:

Expression Matching: Recognition of and Mapping to a DLT. One of the fundamental steps in content extraction is the recognition and mapping of spec phrases that convey requirements, to the corresponding DLTs. A core subproblem is the formalization of DLTs. To appreciate the practical difficulties in recognizing a DLT, consider the following string that illustrates the variety of entities that can be present in a DLT:

```
(Cu + Ag) - 0.2% Al Chemical Composition (Final) (GEAE 42-Delta Phase)
```

Specifically, a DLT can contain a formula, a sequence of words, a parenthesized sequence of words, or a reference to another spec. The formula itself can contain chemical element names, arithmetic operator symbols, other special symbols, etc. Even though the syntax of a formula resembles that of an arithmetic expression in isolation, there are many examples of formula that are difficult to discriminate from other fragments because of the ambiguous use of symbols such as "–" for "minus" and "dash", "/" for "division", "ratio" and "or", and the form of references. Once the syntax of DLTs has been formalized, their concrete manifestation in spec needs to be addressed. All this is further complicated if the phrase that maps to a DLT is not contiguous or several DLTs share words. Subsequent to recognizing DLTs, it is also necessary to recognize their inter-relationships.

Semantic Markup: Procedure Descriptions. Specialized recognizers need to be built for each procedure that can select coherent pieces of text, and organize the information contained in it. Specifically, a list of characteristics and their values relevant to a procedure can be specified and identified using strategy similar to the one employed for recognizing DLTs. However, procedure descriptions also require guessing and inferring information not explicitly provided in spec but needed to express the extraction unambiguously. What is unclear here is the effect of these customized recognizers on each spec, and if their use will decidedly outweigh the tedium of manual extraction for complex specs. A reasonable approach is to build such recognizers for a small set of well-understood procedures

Semantic Markup: Representing and Processing Tables. Tables in spec represent collections of constraints on characteristics and encapsulate procedure descriptions very efficiently. Footnotes modify these tables in interesting ways, often

incrementally and sometimes substantially. The complexity of the table stems from their irregularity. In order to formalize tables, it is important to understand techniques employed for sharing pieces of information in a spec, and propose regular data structures that can hold the same information compactly and be translated into SDL automatically.

Integration: XML Technology. XML Technology is beneficial if it is possible to develop a target language and an extraction discipline that is more or less literal. However, at this time, the semi-automatic approach relies on a human extractor to remedy any inadequacy in the automatic markup by modifying the resulting SDL output. Unless these updates can be reflected back into the XML Master sensibly, the XML intermediary will not be viable. Furthermore, literal translation also requires unconventional use of certain SDL constructs, and unless these nested constructs can be properly handled by the SDR compiler, these will create problems downstream.

6 Conclusions

This project attempted to systematically evolve the manual extraction tool into a computer-assisted extraction tool. To this end, we studied extractions from specs – of different complexities and origin – carried out by highly trained metallurgists, and appreciated the importance of making domain library searches flexible. The recognition of semantically equivalent phrases that map to the same Domain Library Term was attempted using a normalization procedure starting from verbatim searches, to using minor variations on words, to using aliases, to eventually using much richer "patterns of equivalences". This functionality can be incorporated as a separate module, spliced between the IDE and the Domain Library.

For a semi-automatic approach to succeed in practice, the partial results obtained through automation should be intelligible, in relation to the original spec. Otherwise, the extractors will turn-off automation, in the interest of verifiability. This motivates the need to maintain a one-to-one correspondence between the spec and its translation. The XML Technology seems appropriate to extraction to the extent that the XML Master provides a way of tying together the spec and the extracted information, and the use of XSLT stylesheets in transforming this information as desired by the context.

Based on our first experiences with the prototypes we have built, the generated Method C extractions seem reasonable, though not perfect. However, improvements in precision will be needed to scale it to Method B extractions, from human extractor's perspective.

Acknowledgements

The authors would like to thank the reviewers for their feedback.

References

1. Sokol, D.Z.: Concurrent Engineering in the Materials Industry: Case Study in the Application of Information Technology, Fourth Annual Conference on Management of Technology, 1994.
2. Sokol, D.Z., Rowe, J.: Integrating STEP and SGML for Concurrent Engineering, CALS 95 International Expo.
3. Sokol, D.Z.: Concurrent Engineering Design System for High Technology Material Suppliers, NSF Phase II Final Report, 1997.
4. Soderland S. G.: Learning Information Extraction Rules for Semi-structured and Free Text, Machine Learning, Vol. **34**, No. 1-3 (1999) 233-272.
5. Proceedings of the Third to Seventh Message Understanding Conference, Morgan Kaufman (1991) (1992) (1993) (1996) (1997).
6. UNISYS: Description of the UNISYS System used for MUC-3, Procs. of MUC-3, 1991,212-222.
7. Hobbs J., Appelt D., Bear J., Israel D., Kameyama M., Stickel M., and Tyson M.: FASTUS: Extracting Information from Natural-Language Text, 1996. (http://www.ai.sri.com/natural-language/projects/fastus-schabes.html)
8. Grishman R.: The NYU System for MUC-6 or Where's the Syntax?, Procs. of MUC-6 (1995).
9. Lehnert W.G., Cardie C., Fisher D., McCarthy J., Riloff E., and Soderland S.: Evaluating an Information Extraction System, Journal of Integrated Computer-Aided Engineering, 1(6) (1994) 453-472.
10. Riloff, E.: Automatically Constructing a Dictionary for Information Extraction Tasks, Proceedings of the Eleventh Annual Conference on Artificial Intelligence (1994) 811-816.
11. Fujii, A., and Ishikawa, T.: Cross-Language Information Retrieval for Technical Documents (1996).
12. Grishman R.: Information Extraction: Techniques and Challenges, Information Extraction (International Summer School SCIE-97), ed. Maria Teresa Pazienza, Springer-Verlag, 1997.
13. Soderland S.G.: CRYSTAL: Learning Domain-specific Text Analysis Rules, CIIR Technical Report # 43, University of Massachusetts at Amherst.
14. Du Charme B.: XSLT Quickly, Manning Publications Co. (2001).
15. Tidwell D.: XSLT, O'Reilly (2001).
16. Harold E. R.: XML Bible, Hungry Minds Inc. (1999).
17. Dietel H. M.: et al, XML: How to Program, Prentice Hall Inc. (2000).
18. Porter M. F.: An Algorithm for Suffix Stripping, Program, Vol. **14**, No. 3, (1990), 130–137.
19. McCarthy J.: A Trainable Approach to Coreference Resolution for Information Extraction, PhD Thesis. Dept. of Computer Science Technical Report # 78, University of Massachusetts, Amherst.
20. van Harmelen F. and Fensel D.: Practical Knowledge Representation for the Web. Practical Knowledge Representation for the Web. In Proceedings of the Workshop on Intelligent Information Integration (III99), (1999) IJCAI-99.
21. Muslea, I.: Extraction Patterns for Information Extraction Tasks: A Survey, In Proceedings of AAAI-99 Workshop on Machine Learning for Information Extraction, (1999) AAAI-99.

Integrating Retrieval Functionality in Web Sites Based on Storyboard Design and Word Fields

Antje Düsterhöft[1] and Bernhard Thalheim[2]

[1] University of Applied Science Wismar
Department of Electrical Engineering and Computer Science, 23952 Wismar, Germany
duest@et.hs-wismar.de
[2] Brandenburg University of Technology at Cottbus
Computer Science Institute, 03013 Cottbus, Germany
thalheim@informatik.tu-cottbus.de

Abstract. Large information-intensive sites are developed everywhere and need in every case a separate retrieval support. The support comprises functionalities for key word and catalogue search as well as context based retrieval. The search functionality is often added after designing the complete site using common search engines or site map tools. The resulting information of that search facilities is mostly not that specific the user needs. We focus on an integrated development of search facilities within the web site design process. Additionally, we use the linguistic instrument 'word field' for defining abstract search keys during the design process.

1 Introduction

Websites are developed everywhere.

Internet information services have to cope with a broad variety of problems beside presentation, information generation, and graphics, or multimedia design:

- *Services need to be personalized in order to be acceptable for any user.* Since users are treating services in a different form, flexibility of information presentation, of query/answer dialogues and of interpretation of user's intentions become crucial. User abilities in asking a query are very different. User semantics differs. Users are misinterpreting results obtained through summarization and contradiction for instance in multidimensional databases. Summarized data often carry different semantics. Users are used to certain cultural environments.
  Personalization of information, information content and information representation is playing the major role success factor of information services which have to cope with restricted display spaces such as videotext, WAP or phone display and internet services distributed over TV nets based on set-top box technology.
- *Data may very over time.* This approach is partially supported by temporal databases. Data may have a temporal dimension based on a time algebra,

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 52–63, 2002.

time validity for defining, querying or modifying, may have been based on user-defined time and may be computed under restrictions of transaction time. Temporal logics used in services must be based on branching time since we need to take into consideration different schedules of user groups, a variety of user-group specific breakpoints. Branching time need to be handled dynamically by generating service pages on demand based on the current state of service.

- *User act in specific context.* Their context depend on and is limited by technical environment such as available equipment, channel and server capacity currently available, and understanding of their tasks.
  Context is based on the tasks currently under consideration. User are entering sites with certain occasions and intensions in their mind. Well-developed internet sites use this knowledge for modeling scenarios of usage [ScT00]. Services may also vary according to availability of server functionality, communication channel and according to user's preferences. This variation led to the ACE approach of where the adaption of services to heterogeneous and dynamic environments has been investigated. This approach is based on component technology and allows the derivation of the XML content whenever the page content can be specified on the basis of parametric components.

These problems lead to a misunderstanding of services, unacceptable presentations and counter-intensional use of services. Therefore, firstly we need a way to present these services in a consistent and concise way. For that reason we developed the storyboard design concept which is illustrated in [ScT00]. Secondly, we need a user specific retrieval support. The retrieval or search functionality has to depend on the user context. In this paper we will show how content structures can be captured within the storyboard design process.

1.1 Retrieval Support

The challenges. The design process of web sites often does not contain the design of search facilities[1]. These facilities are often added after designing the whole web site. Search tools are used for realising site maps (e.g. [xtreme]) or catalogue structures. But site maps or catalogues are

- often too big especially if we have very large web sites,
- do not have the abstraction level the user needs and
- do not follow semantical guide lines resulting from the application.

Another kind of search facilities are key word based search engines [MetaSearch]. The problems of world wide search engines are well known.

- The result set is too big.

[1] We do not consider aspects of constructing world wide search engines that focus on the analysis of unknown information.

- The ranking algorithm does not fit the user needs. Key words are not specific enough to decide which document is more important than another for a specific user. Natural language used by search engines are very rare and dealing with the problem of ambiguity.
- The search engines do not rate the profundity of information. The user wants to have more common information on a topic which is represented e.g. by information of the top of a website. On the other hand a user wants to have details of a topic represented by the leafs of a website. The most key word search engines do not support the search using different kinds of abstraction levels.

These drawbacks are also relevant if such a search engine used for an own site like MyGoogle. In addition, the provider's meta knowledge about the site is lost.

The potential. The prospective web application is intensively discussed during the process of requirements analysis. The conceptual model developed as a description of the web application is the result of the first phases of the web site design. The storyboard design concept focus on a conceptual description using different abstraction levels. These levels are reflecting in the concepts: story, scenario and scene. The concepts are assigned to specific actors and following content based intensions. Although the more abstract levels (story, scenario) are not directly visible in the implementation of the site we can use them for building search facilities. In most cases natural language is used for the descriptions of the the stories and scenarios. The linguistic instrument 'word field' can help us to identify main actions, main actors and their linguistic representations. Via the accordance of linguistic representations we can identify the abstract contents and using it for search facilities in different natural languages, e.g. in English and German.

Thus, we focus on the integration of developing a story space, an interaction space and a search space. The story space mainly describes the content of the web site as a set of integrated stories . The interaction space defines the possible actions within the web site. Additionaly a *search space* is necessary and modeled resulting from the story- and interaction space (ref. fig. 1). The story space concepts story and scenario are used for defining retrieval functionality.

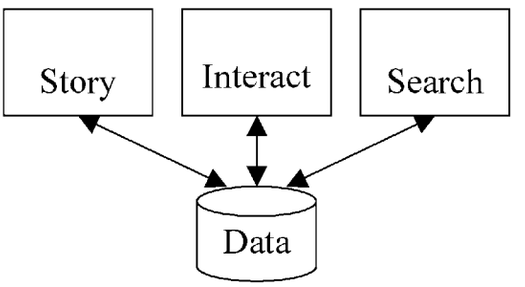


Fig. 1. Integrated design of story-, interaction- and search space

1.2 Overview of the Paper

After illustrating the related work we describe the storyboard design concept (*section 2*). The storyboard design process, the according design components and the technical realisation are discussed. In *section 3* we show how search functionalities can be developed in the storyboard design process. Firstly, we derive semantic structures for constructing a user-centered semantic navigation structure which has different abstraction levels. Secondly, we show how word fields can be used for defining the meaning of scenes. In *section 4* we give a conclusion and future work will be considered.

2 The Storyboard Design Concept

2.1 Components of the Storyboard Design Concept

Site specification might be rather complex. For this reason, we prefer the following structure in order to describe the content of the prospective site in different abstraction levels.
Stories are intended to describe complex action.
Stories are collected into the **story space**. The story space describes the set of all possible intended interactions of actors with the system.
Scenarios are used to describe specific utilization of the website by an actor at a given moment of time. We might have a large variety of scenarios. For this reason, scenarios need to be integrated into stories.
The scenario specification consists of

1. dialogue scenes representing actions to be performed for a specific task and dialogue steps for the representation of an episode,
2. actors with information and interaction requirements and information needs,
3. media object satisfying interaction requirements and information need,
4. description of intension of the dialogue scenes and steps which are consistent with the goals according the mission and outcome of the step and with the occasions depending on the aim,
5. context of the dialogue scenes and steps which are consistent with tasks and the history, with particular and environment according to the occasion.

The following example scenario is used in a website for a membership application:

```
Membership Application extends scenario Purchase with Approval
 Intention: Actor wants to apply for membership
 Roles: Actor as applicant
 Systems: Electronic Commerce Web Server, Certification;
            Authority, ...
 Scenario
   Dialogue Scene: Making Membership Application Available
      Goal: Membership Application Available
      Outcome: Membership Application is available
      Actions: Actor invokes the membership application page
             ....
```

```
Dialogue Scene: Approving Payment
   Occasion: Membership requires payment first
   Goal: Association budget balance increased
   Outcome: Certification authority has approved membership
            fee payment
   Information need: Association account, bank details
   Actions: Membership data are processed
            Web server receives payment information
            Certification authority consulted
            Certification authority approves payment
            Web server notifies notifies actor
   Context: safe connection, existing application
   ...
```

Scenes are macro steps within a scenario. They involve a number of actors in various roles, rights and occasions. They are supported by media objects. Scenes are specified on the basis of the following specification frame:

```
Scene = ( Scene-ID
          DialogueStepExpression
          MediaObject
          Actors  [ActorID,Right ,Tasks ,Assigned Roles ]
          Representation [styles, defaults, emphasis, ...]
          Context [equipment, channel, particular])
```

Media objects describes the 'adornment' of a scene similar to the movie or drama production.

Dialogue steps describe specific actions enabled actors which act in a specific context. Dialogue steps are supporting different scenes. Any scene has a dialogue expression expressing the dialogue workflow.

We use the following frame for dialogue step specification:

```
On event if precondition
    Do action accept on postcond
```

The **interaction space** consists of all possible interactions. The interaction space is broader than the story space and allows arbitrary dialogues and competition among users. For modeling the interactions we use the ASM (abstract state machine) control [ASM].

2.2 Site Specifiction and Implementation

The concepts presented above have been developed in parallel to web site developed of the Cottbus*net* team. The framework does not support *any* website specification. However, it has been proven to be sufficient for specification of more than 25 large websites.

Generic scenes. A scene is supported by media objects. Media objects can be parameterized. Typical parameters are the representation style, the actor frame, and the context frame. Therefore we distinguish between media objects and runtime media objects which are send to the actor in a concrete runtime.

Furthermore, involved actors are specified in dependence on their profiles, tasks assigned to them , their access and manipulation rights, and their roles to be taken during visiting the scene. This specification is similar to profiles of actors in information systems.
It is our aim to specify generic scenes. Thus, we add the representation styles which can be applied to the media object of the scene. Representation depends on the equipment of the actor. In the city site projects, we have experienced with different representation styles: internet display with high-speed channel, internet-display with medium speed display (default style), videotext and WAP display. For instance, for videotext any graphical information is cut out or replaced by textual information.
Finally, the context of access is specified. Access determines the display facilities. Channels can be of high or low speed. The particular of using a scene by an actor depends on the scenario history.
The Onion Approach. Layering of applications is one of the general approaches used in Computer Science. We observe that web site functionality and content may be layered as well. On the outer layer, the presentation facilities may be introduced. Typical functions are style and context functions. Containers are used to ship the information and the functionality provided by the web engine. Containers contain information units which in general are views enriched by functions for operating on the views. Views may be provide information to the dialogue or may be used for updating the database. Further, views may be materialized. If they are materialized then the view handler provides an automatic refreshment support. Thus, we can use the onion system architecture which stepwise via database content, views, units, profiles, and equipment adaptions a web page generates using the database management system, view handler, unit-, container-, and presentation engines.

3 Constructing Retrieval Functionality

Finding information on the Web is a great challenge. The lack of most web sites is that users often don't have an idea of

- which kind of information and
- how many information are available in the web site and
- how the available information is structured.

Analysing the used retrieval functionalities we can obtain the following characteristics which are related to different scientific areas:

- Search patterns depending on information needs and actors, integration with browsing, search iterations;
- Search input: keywords, alternative terms, misspelling, multilingual, natural language searches, text entry support, spelling-reduced searches, fuzzy formulation, modes of searching, clear search options, support for judgement, information retrieval techniques;
- Representation of search results: prioritizing, clustering, navigation support, feedback always or not;
- AI techniques: mining, discovery, concept hierarchies, information structuring, agents, uncertainty, incompleteness, heuristics;

- Search style: search without spelling, scoped searches, expression logics (and, or, NOT), buttons, search capabilities.

That's why the content of a web site must be expressed in different ways for the users. So, we focus on the following kinds of retrieval interfaces: a step-by-step form-based database search, a multi step semantic search and a natural language text search. The step-by-step form search is shown in [Th00] and [DüT01]. In the following parts we show we can develop functionalities 2. and 3. during the storyboard design process.

3.1 Constructing a Semantic Search Structure

Several stories consisting of scenarios and scenes are discussed during the process of developing a website using the storyboard design concept. The different stories explain the different aims users can have. Starting from the aims the according scenarios describe parts of the aims. Then, the scenes give the concrete pages in the web. The interaction space defines the connection between the pages for specific users.

Constructing abstraction levels. The semantic search structure starts with the *most (first) abstract level* – the stories (ref. Figure 2). They give an overview of the web site which is defined by the semantic focuses of the site and the prospective users.

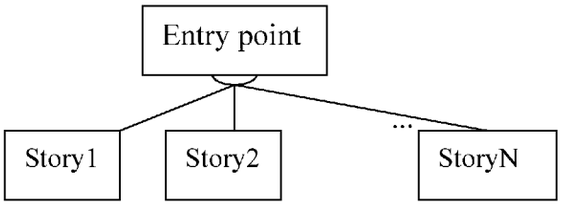


Fig. 2. Story abstraction

The *second abstraction level* are the scenarios of the stories (ref. Figure 3). The parts of the stories are used for specific sub tasks or completed units within the aim of a story. Scenarios can be ordered and have subscenarios itself. Althought stories and scenarios can be overlapped in the web site (i.e. using the same web pages) and by actors they are defined through the process of correct storyboarding and can be used for building the abstraction levels.

The *third abstraction level* is the site structure – the site map. The site map can be generated from the database and the according navigation space.

In contrast to the third abstraction level information of the second and first ones cannot be generated with common site map tools because it is not explicitly integrated into the generated web pages. Otherwise the storyboard structures are not in every case hierarchical. They can be a network of connection if e.g. a scenario belongs to more than one story. Thus, the resulting search structure is more complex than a site map structure.

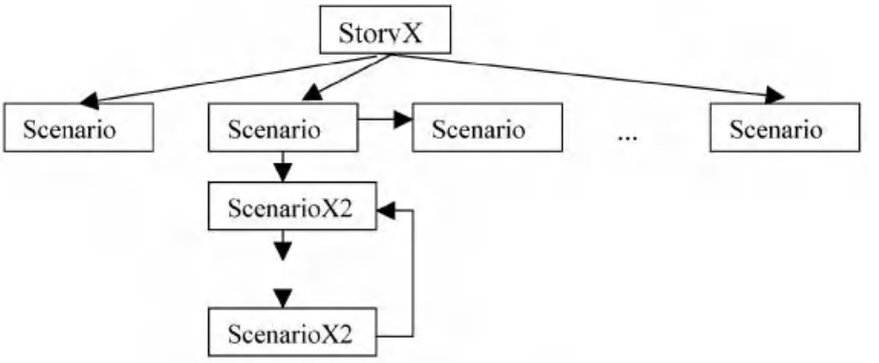


Fig. 3. Scenario abstraction

In a next step the abstraction levels are integrated into a semantic driven search structure (ref. figure 4).

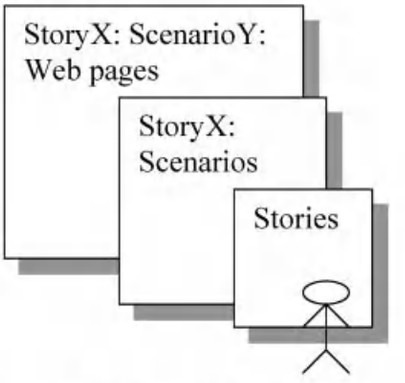


Fig. 4. Integration of abstraction levels

Technical realization. The stories, scenarios, and scenes are defined using the description language SiteLang [DüT01] as shown in section 2. The database design, the created views, contexts and navigations are derived by the storyboard concepts. For building the abstraction levels also the storyboard concepts have to be stored in the database. Additionally, the connection of the storyboard concepts and the page information must be stored in order to retrieve the semantic search structure.
So, in fact the scene representation is extended to expect the story and scenario information in that way that an assignment of scenarios to stories is possible:

```
Scene =
 (Scene-ID
  DialogueStepExpression
  MediaObject
```

```
Actors [ActorID, Right, Tasks, Assigned Roles]
Representation [styles, defaults, emphasis, ...]
Context (equipment, channel, particular)
[Stories]/* list of stories to which belongs this scene */
[Scenarios]     /* list of scenarios of a story belongs
                this scene */  )
```

The scene information stored in the database is used to build the search structure in context of the actual activities of users. So, if the user wants to have a site map coming from a specific web page the different abstraction levels can be shown. For example, a user starts from the entry point of the web site the most abstract level will be presented first. After choosing a specific story the according scenarios and then the web pages will be shown. In another case, the user may be ask the semantic structure coming from a specific web page. He/She gets first the scenes of the according scenario and then he/she can change to another scenario or another story.

The graphical presentation of the story-scenario-scene graphs is implemented using Java tools for directory structures. Other graphical notions and implementations e.g. used for ontologies are actually analysed and will be implemented.

3.2 Constructing a Content Based Key Word Search Facilities

A wide variety of tools and services exist that are useful for key word or text searches in the Internet, but whose efficiency is somewhat limited. The majority of tools used for information searches in the Internet concentrate primarily on so-called syntactical attributes, such as TITLE or DESCRIPTION, without considering the actual meaning of the information `[MetaSearch]`, `[SearchReport]`, `[W3SearchEngines]`.

The bulk of available information in the Internet is provided in natural-language format and supplemented with graphics. Furthermore, user queries are typically formulated using natural language words and phrases. This, despite the fact, that during the past few years the computer linguistic field has developed a wide variety of tools and mechanisms for partially automatic, natural-language processing that could be employed as intelligent search support. There exists some search engine projects dealing with natural language input, e.g. Getess [Getess] or [AskJeeves].

The search engine Getess focuses on an Internet search engine which analyzes information in German and English and provides a natural language access to web pages and documents. The search engine uses a domain-related approach in which all specific appearances of items are seen as representations of that particular item. So these specific words that are important in the domain are mapped onto concepts. For instance 'Master card' is a specific appearance of 'credit card'. The search engine subsequently uses these concepts in order to provide search results. The search engine also maps the natural-language user inquiries onto concepts and tries to match the concepts in the analyzed web documents. Intelligent systems situated between search sites and information will be responsible for condensing extensive and complex data into language-independent, content-weighted summaries (*abstracts*). The abstracts will provide a base and will subsequently be used as

results for queries. Results will then be translated into the user's mother tongue. A fundamental requirement for developing of an Internet search site is an extremely large amount of data. Accordingly, the amount of information allocated to the abstracts will also be substantial. Therefore, databases will be used for efficient storage and quick access to abstracts.
The Getess search engine has two main components: the gatherer and the broker. The Getess gatherer works as an Internet agent that periodically collects data from the Internet. After a data type recognition the Gatherer-controller starts the summarizers and the abstract-generator in order to analyze the documents. The broker realises the natural language user access on the basis of the abstracts.
The bottle neck of the Getess system are the created *abstracts*. Abstracts describe the content of a web page in XML format in a specific domain-related manner. Natural language words and phrases of a web page are used to build abstracts. A search query can only be answered as correct as the abstracts representing the content of a web page. That's why a design process of web sites should include search facilities for natural language text search if that should be used.

Thus, we focus on the extraction and notation of the semantic concepts used for *abstracts* during the design process.

Word fields. Word fields are linguistic systems in which similar words, words with the same basic semen, used in the same context [Kunze02],[SSS90]. In contrast to common synonym dictionary word fields define the possible/necessary actors, the actions and the context. Word fields can be used for verbs, nouns and adjectives. We focus on verb fields and extend the implementations of the WordNet dictionary for verbs.
For each verb field we can at least define the following abstract information:

- logical structure: possible arguments of a verb,
- semantical description: relevant and irrelevant valences (verb frame),
- kernel semantics: description with semen and
- example sentences.

For instance, the inform word field consists of

- the following German representations:
  - inform someone of
    1. 'mitteilen', 'informieren', 'bekanntgeben' (official, factual)
    2. ‚erklären', ‚verkünden', ‚vortragen' (very official)
    3. ‚melden' (factual, short)
    4. 'ankündigen' (focus to future)
    5. 'berichten', 'beschreiben' (factual, detailed)
    6. ‚erzählen', ‚schildern' (detailed, with experiences)
    7. 'bestellen', 'übermitteln' (with the help of another medium or person)
    8. 'hinterbringen' (secret)
  - inform someone of unknown things
    1. 'auseinandersetzen', 'erklären', 'erläutern' (latent facts)
    2. ‚beweisen' (the correctness of facts)
    3. 'verraten' (not allowed)

4. 'bekennen', 'gestehen', 'beichten' (own way of thinking and behave)

- react to information or actions
  1. 'bejahen', 'zusagen' (positive)
  2. ‚verneinen', ‚verbieten' (negative)
- someone give ideas for prospective actions
  1. 'vorschlagen'

- each of the representations has the abstract information; e.g. the abstract information for 'ankündigen' (announce) is
  1. 'inform someone of future things'
  2. verb frame:
     - agent: human or group
     - recipient: human or group
     - content: animinate or inanimate
  3. example 'The stewardess (agent) announce the passegengers(recipient) the landing of the airplane (content).'

Integration into the storyboard design process. Users describe the application with natural language during the design process. In the case of naming the different constructs the user has to define example sentences. These examples are the basis of detecting the right word field, finding the abstract descriptions, and the according actors/arguments (valences). The example sentences will be analysed using natural language parsing techniques in order to grap the kernal semantics. The search information is integrated into a scene description as follows:

```
Scene =
( Scene-ID
  DialogueStepExpression
  MediaObject
  Actors [ActorID, Right, Tasks, Assigned Roles]
  Representation [styles, defaults, emphasis, ...]
  Context [equipment, channel, particular]
  [Stories]  /* list of stories */
  [Scenarios]/* list of scenarios */
  search facilities(word field, representation, kernal
                    semantics, valences, example sentences))
```

After all, natural language search engine can use the search facilities for constructing a search space as a search index. The search space will be constructed in parallel to the generated web site. The specifics of a natural language search engine, e.g. the domain dependend abstracts in the Getess search engine, can be generated on the basis of the (domain independend) semantic description. So, it is possible to realise natural language search facilities for supporting the natural language queries of different search engines.

4 Conclusion

In this paper we have shown how we can use linguistic knowledge for conceptual modeling of web sites. Especially we have illustrated how the storyboard design

strategy works and how the according concepts (story, scenario, scene) can be used for getting a semantic search structure. These structure can be used for creating semantic driven site-network in order to realise a navigation search structure for the user. On the other hand we have shown how word fields can be used to create search facilities for natural language queries of search engines.
We are actually implementing the extensions of the WordNet specifications. Future work will focus on the integration of a word field lexicon into a visual storyboard design editor.

Bibliography

[AskJeeves] http://www.askjeeves.com
[BCF00] Bonifati, A., Ceri, S., Fraternali, & P., Maurino, A. (2000) *Building multi-device, content-centric applications using WebML and the W313 tool suite.* ER Workshops 2000, LNCS 1921, Springer, Berlin, 64-75.
[BGP00] Baresi, L., Garzotto, F., & Paolini, P. (2000) *From web sites to web applications: New issues or conceptual modelling.* ER Workshops 2000, LNCS 1921, Springer, Berlin, 89-100.
[CFP99] Ceri, S., Fraternali, P., & Paraboschi, S. (1999) *Data-driven, one-to-one web site generation for data-intensive applications.* Proceedings VLDB Conference 1999, 615-626.
[DüT01] Düsterhöft, A., Thalheim, B.(2001) *SiteLang: Conceptual Modeling of Internet Sites.* Proc. ER'2001, LNCS 2223, 2001, 179-192
[FOS00] Feyer, T., Odey, K., Schewe, K.-D., & Thalheim, B. (2000) *Design of data-intensive web-based information services.* Proceedings WISE 2000, Hong Kong (China).
[GaT99] Gaedke,M., & Turowski, K. (1999) *Generic web-based federation of business application systems for e-commerce applications.* Proceedings EFIS Conference 1999, 25-42.
[Getess] http://www.getess.de
[GPS93] Garzotto, F., Paolini, P., & Schwabe, D. (1993) *HDM - A model-based approach to hypertext application design.* TOIS 1(1), 1-26.
[GST00] Goldin, D., Srinivasa, S., & Thalheim, B. (2000) *IS = DBS + interaction - towards principles of information systems.* Proceedings ER Conference 2000, LNCS 1920, Springer, 140-153.
[IKK98] Isakowitz, T., Kamis, A., & Koufaris, M. (1998) *The extended RMM methdology for web publishing.* New York University. http://rrr-java.stern.nyu.edu/rmm/
[MetaSearch] http://www.search.com
[MMA99] Mecca, G., Merialdo, P., & Atzeni, P. (1999) *ARANEUS in the Era of XML.* IEEE Data Engineering Bullettin, Special Issue on XML.
[Sch00] Schewe, K.-D. (2000) *UML - a modern dinosaurus.* Proceedings European-Japanese Conference on Information Modelling and Knowledge Bases, Finland.
[ScT00] Schewe, K.-D., & Thalheim, B. (2000) *Conceptual development of internet sites.* Full-day tutorial at ER Conference 2000, Salt Lake City, 2000
[SearchReport] http://searchenginewatch.com
[Tha00'] Thalheim, B. (2000) *Readings in fundamentals of interaction in information systems.* Preprint, BTU-Cottbus, http://www.informatik.tu-cottbus.de/~thalheim.
[Tha01] Thalheim, B. (2001) *ASM specification ofinternet information services.* In. Proceedings Eurocast Conference 2001, Las Palmas, 301-304.
[W3SearchEngines] http://cui.unige.ch/meta-index.html
[xtreme] http://www.xtreme.com

Vulcain – An Ontology-Based Information Extraction System

Amalia Todirascu[1], Laurent Romary[1], and Dalila Bekhouche[1]

INRIA Lorraine
LORIA, Campus scientifique BP 239, 54506 Vandoeuvre-lès-Nancy Cedex, France
{todirasc,romary,bekhouche}@loria.fr

Abstract. This paper describes an information extraction system, Vulcain, dedicated to message filtering for a specific domain. The paper focuses on a method for identifying domain-specific terms and concepts, using syntactic information and an existing domain ontology. We focused on a method for identifying terms by partial syntactic analysis, based on TAG grammars. The domain ontology is represented in description logics, and DL inference mechanisms are used to validate the candidate concepts.

1 Introduction

Information Extraction systems identify relevant entities on texts, on a given domain. They use natural language processing techniques for identifying relevant data, and they use domain-specific knowledge to validate these entities. The main drawback of Information Extraction systems is low portability, due to language-dependent linguistic resources and to domain-specific knowledge (ontology).

To validate data identified in texts, some Information Extraction systems use predefined or fixed ontologies, but these ontologies are often incomplete, contain inconsistencies or redundant data. The costs of adapting them to another domain or application are very important. Generic ontologies (WordNet (Miller and all, 1990), Corelex [5]) are useful for searching information in free texts, but they are not adapted for domain-specific senses, they might not contain some domain-specific senses.

Ontologies extracted semi-automatically (using statistical methods [8], or methods using logical inferences [19]) are used for increasing Information Extraction systems' portability. For each new domain, a domain-specific ontology will be created. In this context, we applied an inference-based method for extending ontologies, to improve our system's portability.

Most Information Extraction systems identify candidate concepts applying shallow Natural Language Processing (NLP) techniques (finite-state methods [7] and simple pattern matching [8] [16]). Preferred candidates are noun phrases or noun-noun collocations [11]. Shallow NLP techniques are adapted to handle large amounts of texts and they are preffered by IE systems. The number of candidates generated by NLP techniques is often very big, and the resulting candidate

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 64–75, 2002.

concepts must be filtered and must be validated by some human experts or by domain-specific patterns [16].

A concept identification method based on a linguistic formalism reduces the number of concept candidates, but most of the Information Extraction tools are highly language-dependent. For this reason, we adapt linguistic tools and resources which are highly modular (a LTAG parser [14]), and we prefer an approach extending domain ontologies based on logical inferences.

The goal of the paper is to describe a method for identifying new concepts on texts, validated by a domain-specific ontology. We focused on the interface between syntax and domain-specific knowledge.

We propose a method which is based on an existing linguistic formalism. It uses some partial syntactic structures provided by a Lexical Tree Adjoining Grammars (LTAG) parser to identify concept candidates, as well as logical inferences available in Description Logics, which is used as knowledge representation formalism. The method is tested on a specific application for filtering electronic messages about computer security. The corpus has some caracteristics: it contains various errors (spelling, syntactic, wrong segmentations), but also entity names and fixed patterns (computer commands). Due to these corpus' properties, we proposed a concept instance identification method which handles these errors.

2 The Architecture

The system includes an LTAG parser adapted for IE, containing a syntax-semantics interface and several modules extracting lexicons or other resources from reference corpora (fig. 1). The domain-specific lexicon, relating words to their syntactic contexts, is extracted from the reference corpus, using the LTAG grammar and lexicon. The corpus is also used by the human expert for manually designing an initial ontology.

The system requires an interface between the syntax and the semantics, in order to build a semantic representation for each candidate concept. The interface contains a lexicon assigning concepts to each word and a module assigning conceptual descriptions to grammar trees. The domain-specific lexicon provides syntactic information for each input word,information which is used by the LTAG parser to identify potential concept instances, among simple noun phrases and verbs. The domain-specific ontology validates the semantic representations, built fromthe syntactic structures (provided by the parser) and from the semantic lexicon.

3 Ontologies

Domain knowledge is a key element in an information extraction application [10], for validating the entities identified in the texts by the linguistic tools. Existing generic ontologies (WordNet [15], Corelex [5]) provide synonymy, hyperonymy and hyponymy relations for free-text information extraction. Domain-specific

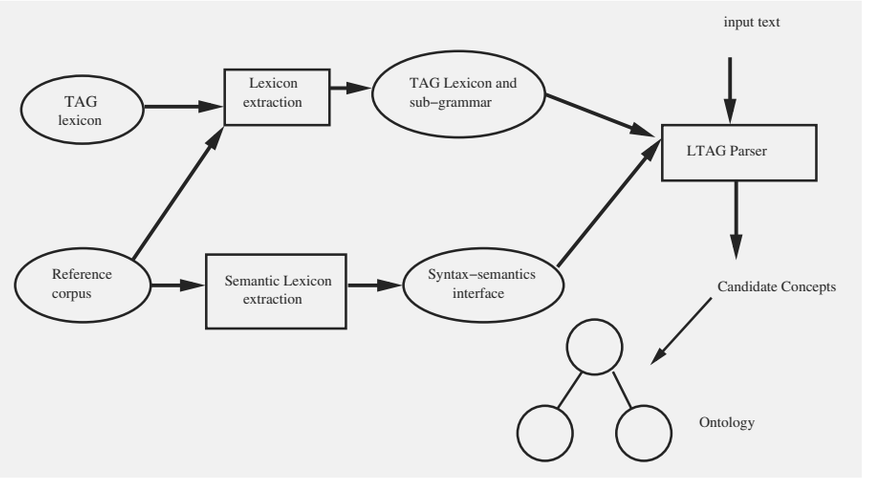


Fig. 1. System architecture

ontologies might contain specific senses that could not be found in a general-purpose ontology. Domain-specific ontologies are highly dependent on the domain of the aplication.

To avoid this drawback, several methods (both statistical-based and inference-based) for automatically extracting ontologies were developed. The methods identify candidate terms in the texts, they create classes of similar terms and they identify relations between term classes.

Statistical methods [1], [8] interpret contexts and groups terms with similar contexts into classes associated with a unique concept. Some methods identify relations between term classes by interpreting syntactic knowledge as subcategorisation [6] or by using statistical information [4]. Statistical approaches require large, stable corpora to extract term classes, and human experts must validate and interpret the results.

Some semi-automatic methods for extracting ontologies propose logical inferences to validate the existing knowledge. Concepts candidates are proposed by inference rules and they are added to the domain hierarchy only if they are coherent with the existing knowledge. The main drawbacks of this approach are concept overgeneration and the low efficiency of the knowledge consistency and coherence checking algorithms. Among knowledge representation formalisms, Description logics try to avoid these drawbacks and we have chosen them to represent and maintain our domain knowledge.

3.1 Description Logics

Description Logics are knowledge representation formalisms derived from semantic nets, but providing well-defined syntax and semantics and also combining features of object-oriented systems, of frame-based systems and of modal logics.

DLs provide a hierarchical organization of the knowledge: a conceptual level (the T-Box), describing abstract classes of objects (concepts) and an assertional level (the A-Box), containing the instances of the classes. The concepts have properties: relations (named roles) with other concepts, and attributes (roles with atomic values).

Syntax and Semantics. DLs provides atomic concept names and roles to define new concepts, as well as DL operators. The DL operators are inspired by first order logics: AND, OR, NOT, SOME, ALL, implication. Using this set of operators, several expressivities are possible: the definition of concepts and roles ALC (using SOME, ALL, AND, OR, NOT operators, concept axioms), the possibility of handling transitive roles (R+), of inverse roles (I), of role hierarchy (H), of attributes (f) or of numeric constraints.

DLs define and maintain knowledge via some commands. A few commands are explained below. CN is a concept name, C is a conceptual description (any combination of AND, SOME, NOT, ALL operators). The DL commands use KRSS syntax ([2]):

1) (define-concept CN C) - defines a new concept as a conceptual description;

2) (instance IN C) - defines an instance of a given concept;

3) (implies C1 C2) - introduces a new concept axiom, defining necessary conditions C1 for the conceptual description C2;

DLs are fragments of first order logics, providing decidable algorithms for coherence and consistency checking. DLs propose logical mechanisms to identify concept subsumption, instance retrieval, role paths relating concepts, classification (partial ordering of the concept hierarchy, due to the subsumption relation).

Description Logics for Information Extraction Systems. Information Extraction systems use domain knowledge to validate the semantic representation of the potential relevant entities identified by NLP techniques [19], [10]. These new concepts could be added to the existing ontology. Shallow NLP techniques propose a lot of candidate entities but most of them do not have a valid semantic interpretation.

Unlike frame-based systems, DLs deal with semi-structured or incomplete data. There are no requirements to explicitly define some values as instances of some concepts, or to define default values. Some role fillers are left unspecified as in the following example:

```
(define-concept OSystem (and Object (some isaOperatingSystem
Computer) (some hasName Name)(some hasCommands Command)(some
hasVersion Version)))
(instance linux (and OSystem (some hasName Linux)))
```

Implicit definitions are possible in DL (**Linux** is not defined explicitly as an instance or a subconcept of the concept **Name**). The role fillers of the roles **isaOperatingSystem, Command, Version** are not provided.

These properties are required for information extraction, due to erroneous input and to incomplete domain knowledge. Hyperonymy or hyponymy are han-

dled by subsumptions between domain concepts. For example, if a candidate concept is identified in the text as

```
(instance y (and Password (some hasUser Root)))
(define-concept Password (and String (some hasAtr secret)(some
hasBelongs User)))
(define-concept Root User)
```

y is an instance of the concept **Password** and it is related to the concept **Root**, which is used by a **User**.

Functionalities. We require some basic functionalites from a DL formalism for representing domain knowledge.

Obviously, we have to define concepts, roles and attributes to describe domain ontology. All implemented DL systems provide this feature, as well as axiom definitions. Transitive roles are required in order to compute path roles between various concepts. Inverse role should be avoided, due to the cyclic definitions, which makes the computing processes undecidable.

Information Extraction systems handle proper nouns (data, person and organization names), represented in Description Logics as instances. We require then instance reasoning. Among the systems proposing an A-Box implementation, we chose RACER [12], which is one of the few classifiers proposing optimised tableaux calculus algorithms and XML-like representation of the ontology, compatible with the standard Ontology Interface Layer (OIL) [9].

4 Concept Identification

We propose a concept identification method based on partial syntactic analysis, using a linguistic formalism. The results (dependecy structures) are validated by the domain ontology. The system is still under development. We developed or adapted several NLP tools and resources, and we built the initial ontology.

4.1 The Resources

The Reference Corpus. The reference corpus (used for generating the lexicon and for creating the ontology) is a set of e-mail messages. It contains about 50,000 occurrences (4039 word forms). From the reference corpus, we obtained the list of the most significant words, used to design the domain concept hierarchy.

The corpus contains several abbreviations, phrases introducing the content of a dialogue (**X wrote:**), as well as organization, system, function names (UNIX functions, constants, variable names), DOS or UNIX commands, which require special preprocessing modules using finite state automata designed for entity recognition.

We applied robust, fault-tolerant, shallow NLP techniques for identifying potential concept instances, because the corpus contains syntax and spelling errors. Sentences are not very well delimited, which is one of the sources of tagging errors.

The Ontology. The ontology is developed by a human expert. The human expert extracts a list of the most frequent words, except the functional words (the determiners, the pronouns, the auxiliaries), from the reference corpus. The nouns and the verbs (considered as main candidates to identify primitive concepts) are evaluated by the expert which assigns to each word a mark (5-very relevant, 1-not very relevant) representing the word relevance to the security domain. From the most relevant words, the expert designed a set of 53 primitive concepts, a set of concept axioms and roles.The expert asks the Racer classifier to test the coherence of the knowledge base with the new concept candidates and they are are added to the ontology if the ontology coherence is not violated.

The human expert designs also concept instances (the terms which are associated to each primitive concepts).

The Domain-Specific Lexicon. We used the same reference corpus for building a lexicon. The initial English LTAG lexicon (511 lexicon entries), was incomplete and it was of little use for our security domain. We built a module creating automatically the lexicon from the reference corpus and from the existing English lexicon. The module is implemented in Java. For this purpose, we used the TreeTagger POS tagger [18], which annotates the words with the lexical category and identifies lemmas. The noun and adjective entries are generated automatically from the existing lexicon. The verbs were added manually. The lexicon is coded using a subset of XML dedicated to represent TAG grammars: TAGML (Tree Adjoining Grammar Markup Language) [3].

The list of POS tags and the tagging errors is given in the table 1:

Table 1. The results of POS tagging and the tagging errors' distribution

Lexical category	Occurences	Errors
Nouns, Proper nouns	2991	10.5 %
Verbs	1162	2.32 %
Adjectifs	796	0.31 %
Adverbes	381	0.07%
Prepositions	105	4.46%

The POS-tagging results are validated by a human expert. Errors as: spelling errors, POS tagging errors due to the bad segmentation, Unix commands (which must be preprocessed by a separate module) are deleted them from the entry list. The resulting lexicon has 3000 entries: 1400 nouns, 787 adjectives, 300 verbs, 105 prepositions, 379 adverbs. The lexicon contains also a set of syntagms ("one of", "kind of", "number of"). We add some entries for the most frequent proper names: BIOS, DOS, lilo etc.

The Grammar. The English LTAG grammar [13] contains 420 elementary trees represented in TAGML format. As our system needs to identify the concept candidates among noun phrases and domain verbs, we use a small local grammar describing simple noun phrases and simple verb phrases.

LTAG grammars have an interesting property, making them suitable for our application: they associate each word its syntactic context (formed by a set of elementary trees). This property is used to avoid some candidate trees and to work only with a subset of grammar rules.

We included in the local grammar noun phrase trees (trees associated with nouns, nouns and adjectives that might modify nouns, past participle verbs playing the role of the modifiers, comparative adverbs and adjectives), proper nouns or prepositions relating two nouns. Verbal phrases are formed by verbs and their arguments (subject, direct object, indirect object). The grammar does not contain the verb trees handling long-distance dependency trees, relative clauses.

The Parser. We adapted Lopez's parser [14] to our application. It is now modular and it supports TAGML-like input and output. We use this parser due to its feature of providing partial parsing results, the possibility of using TAG grammars (for French and for English) available in TAGML format. It identifies some candidate terms, even if a syntactic error occurs. It is implemented in Java.

TAG parsers propose as an output a set of derived trees (representing the syntactic structures identified in the text) and a set of derivation trees (a history of adjoining and substitution operations combining the elementary trees). Derivation trees are usually used to generate a dependency tree (a semantic representation), and for this reason we choose TAG as linguistic formalism. The semantic representation is validated by the domain ontology.

5 Bridging between Syntax and Semantics

In this sectio, we present the interface between the domain semantics (the ontology concepts and roles) to lexicon, to grammars and to parsing output. The interface between syntax and semantics consults a lexicon, it builds a set of conceptual descriptions built for each Elementary Tree and for each derivation tree produced by the parser.

5.1 The Semantic Lexicon

The LTAG lexicon entries contains references to lemmas associated with each word. Each lemma is associated with a set of elementary trees and co-anchors. Each lemma might be associated to several concepts, if the word is ambiguous.

We investigated various information required for each lexical category in the semantic lexicon. Nouns and verbs are key elements used to combine elementary trees. Nouns are used to fill substitutions, while verbs expect substitutions to be filled (as arguments). Nouns are associated with some ontology concepts. Some nouns might be modifiers for other nouns and they have also associated an entry for this case. The semantic description is a DL formula as (some hasMod Concept). The adjectives are also modifiers (excepting the case when they are predicative), so the entry is also a relation between the concepts. The verbs have associated a concept from the ontology and some constraints on the arguments'

type. Using this information, we build a DL conceptual description using the information provided by the derivation trees.

We define a function **Sem** taking as argument a lemma and returning a DL concept as a value:

Sem(lemma)=Concept $\wedge$ Constraint1 $\wedge \ldots \wedge$ ConstraintJ

Example.

- for the noun 'root':

Sem("root")= Root

- for the adjective 'root':

Sem("root")=(some hasMod root)

- for the verb 'delete':

Sem("delete") = delete $\wedge$ (and Substitution (some hasaddress 1)) $\wedge$ (and Substitution (some hasaddress 3))

The constraints on the arguments of **delete** identify the addresses of the two substitutions expected by the verb 'delete'.

Some lexicon entries are highly ambiguous: prepositions might represent several domain-specific relations or general relations (type, possesion). For example, the preposition 'with' could be represented as:

Sem(with)={(some hasProp), (some hasInstrument)}

In the security domain model, the preposition 'with' is interpreted as a property to be specified, or as an instrument.

We manually defined a list of pairs (concept, word). Only a small number of verb entries are in the lexicon (50 verbs). We intend to automatize this task. The concepts will be associated with words using a method similar to [17], starting from a set of seed words and concepts (words extracted from the list of the most relevant words) and from a set of predefined patterns. Constraints imposed on verb arguments should be extracted from a treebank containing subcategorisation information.

5.2 Elementary Trees

The elementary trees are related to the lemmas associated with each word. We define an algorithm building a conceptual representation from the elementary tree and from the concepts associated with the lemma:

Sem(ElementaryTree) = Sem(lemma), if no substitution or adjunction is found in the tree;

Sem(ElementaryTree) = (and Sem(lemma) (some hasSubstitution A)) $\wedge$ (implies (some hasSubstition A) (some argi A)), if a substitution is found in the tree and A is a generic concept.

Sem(ElementaryTree) = (some hasAdjunction Sem(lemma)) if the tree will be adjoined (see 2).

5.3 Derivation Trees

The derivation tree stores all the operations (substitution, adjunction) applied for building it recursively from elementary trees and from other derivation trees.

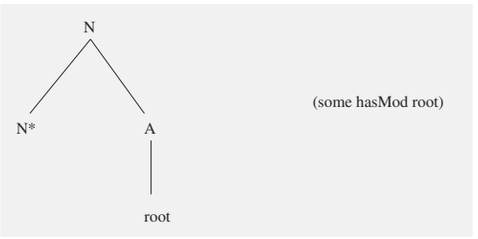


Fig. 2. The elementary tree to be adjoined associated to 'root' and the DL representation

We define an algorithm for extracting complex representations from derivation trees, combining lemmas' semantics and elementery tree's semantics:

Sem(Tree) = (and Sem(A) (Some hasSubstitution Sem(B)) Sem(C)) ∧ (constraints B)

where Sem(A), Sem(B) and Sem(C) are defined recursively, if B or C are derivation trees, and as in the previous subsection if A is an elementary tree. (constraints B) represents the information related to the verb arguments.

Some Examples. The phrase **the user files**, parsed by the LTAG parser, results in two derivation trees (figure 3) The concepts associated to lemmas are:

```
Sem(file) = File
Sem(user) = (some hasMod User)
Sem(user) = User
Sem(the) = (some hasDef definite)
```

The concepts built for the trees are represented in figure 3.

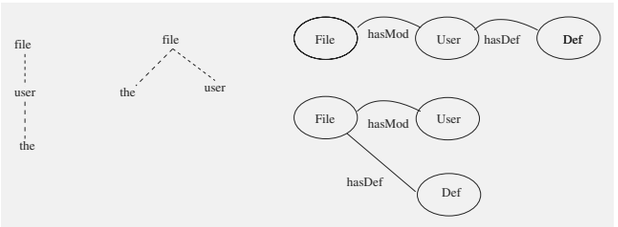


Fig. 3. The derivations tree for "the user files" and their conceptual representations

The phrase **The admin checked the last 10 connections** results into the following derivation trees:

The conceptual descriptions associated to each lemma are:

Sem(check) = check ∧ (implies (and (some hasSubst A)(some hasadr 1))(some arg0 A)) ∧ (implies (and (some hasSubst B)(some hasadr 3)) (some arg1 B))

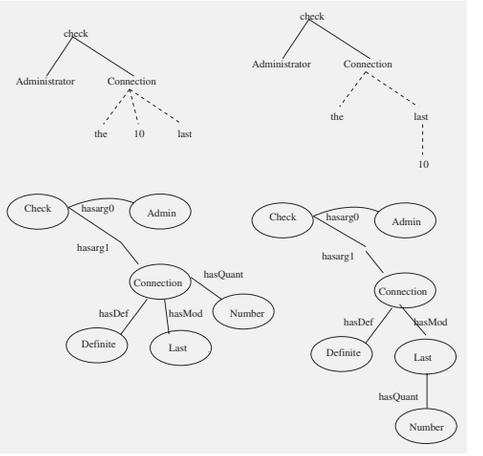


Fig. 4. The derivation tree for "the admin checked the last 10 connections"

Sem(admin) = Administrator
Sem(connection) = Connection
Sem(last) = (some hasMod last)
Sem(the) = (some hasDefine Defined)

The tree representations substituted to the elementary tree anchored by 'check' are:

```
(and Connection (some hasDefine Defined)
(and Administrator (some hasDefine Defined))
```

The representation associated with 'check' contains a set of axioms:

```
(and check (some hasSubst A)(some hasSubst B))
(implies (some hasSubst A)(some arg0 A))
(implies (some hasSubst B)(some arg1 B))
```

In the derivation tree we found the values for A and B:

```
(implies A (and Administrator (some hasDefine Defined)))
(implies B (and Connection (some hasDefine Defined)
(some hasMod last)(some hasNumber 10))
```

The resulting concept is coherent with the existing knowledge:

```
(and Check (some arg0 (and Administrator (some hasDefine Defined)))
(some arg1 (and Connection (some hasDefine Defined)
(some hasNumber 10)(some hasMod last)) ))
```

6 Extending the Ontology

The system uses the partial parsing results to identify a set of relevant entities, which might be added to the existing ontology. The human expert validates

them using the RACER classifier. The first evaluations were done on a small text of 100,000 words containing the most frequent verbs found in the semantic lexicon. 76 % of the concept candidates should be added to the ontology (about 700 various concepts).

Example of some messages:

Trond Hasle Amundsen wrote How can one run a DOS program when lilo and the BIOS are password protected? Maybe some BIOS'es can be reset by a DOS program so that one can still use the machine when one has forgotten the BIOS password?

The entities identified in the text are: run a DOS program, lilo, the BIOS, password protected, some BIOS'es, reset by a DOS program, use the machine, has forgotten the BIOS password .

We obtained the following conceptual descriptions:

```
(and run (some hasarg1 (and program (some hasType DOS))))
(and system (some hasType lilo))
(and entity (some hasName BIOS))
(and password (some hasProp protected))
(and entity (some hasName BIOS)(some hasDef indefinite))
(and reset (some hasarg0 (and program (some hasType DOS))))
(and use (some hasarg1 (and machine (some hasDef definite))))
(and forget (some hasarg1 (and password (some hasDef definite)
(some hasPlace (and entity (some hasName BIOS))))))
```

The human expert check if the concepts are satisfiable in the current ontology. The expert decide to add some new concepts: **run** which might have an argument restricted to the type **program**, **forget**, having an argument restricted to the type **password**, or **reset** linked to a concept **program**.

7 Conclusion and Further Work

The paper focuses on the relation between the syntax and the domain-specific knowledge. We intend to use a meta-grammar for generating automatically the elementary trees as well as conceptual descriptions. We will also develop a method for generating verbal lexicon entries automatically. We have to validate our concept identification method in a real-world application, on a large domain-specific corpora.

References

1. Assadi, H., Bourigault, D.: Analyse syntaxique et statistique pour la construction d'ontologies à partir des textes. In J.Charlet, M.Zacklad, G.Kassel, D.Bourigault (eds.): Ingénierie des connaissances - Evolutions récentes et nouveaux défis, Eyrolles Publishing House(2000), 243–256.
2. Baader, F., Hollunder, B.: A Terminological Knowledge Representation Systems with Complete Inference Algorithms. In Proceedings of the Workshop on Processing Declarative Knowledge (1991).

3. Bonhomme, P. and Lopez, P.: TagML: XML encoding of Resources for Lexicalized Tree Adjoining Grammars. In *Proceedings of LREC2000*, Athens (2000).
4. Bouaud, J., Habert, B., Nazarenko,A., Zweigenbaum, P.: Regroupements issus de dépendances syntaxiques sur un corpus de spécialité: catégorisation et confrontation à deux conceptualisations du domaine. In J.Charlet, M.Zacklad, G.Kassel, D.Bourigault (eds.): Ingénierie des connaissances - Evolutions récentes et nouveaux défis, Eyrolles Publishing House (2000) 275–290.
5. Buitelaar, P.: CORELEX: Systematic Polysemy and Underspecification, Ph.D. thesis, Brandeis University, Department of Computer Science (1998)
6. Capponi, N., Toussaint, Y.: Interprétation de classes de termes par généralisation de structures prédicat-argument. In J.Charlet, M.Zacklad, G.Kassel, D.Bourigault (eds.): Ingénierie des connaissances - Evolutions récentes et nouveaux défis, Eyrolles Publishing House (2000) 337–356.
7. Chanod J.P.: Natural Language Processing and Digital Libraries. In M.T.Pazienza (ed.): Information Extraction, Springer-Verlag, LNAI 1714, (1999) 17–31.
8. Daille, B.: Study and Implementation of Combined Techniques for Automatic Extraction of Terminology. In J.Klavans, P.Resnik (eds.): *The Balancing Act - Combining Symbolic and Statistical Approaches to Language*, MIT Press (1996) 49–66.
9. Fensel D. et al.: OIL in a nutshell. In R. Dieng et al. (eds.): *Knowledge Acquisition, Modeling, and Management, Proceedings of the European Knowledge Acquisition Conference (EKAW-2000)*, Lecture Notes in Artificial Intelligence, LNAI, Springer-Verlag (2000).
10. Guarino, N.: Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration. In M. T. Pazienza (ed.): Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology. Springer Verlag (1997) 139–170.
11. Heid, U.: A linguistic bootstrapping approach to the extraction of term candidates from German text. In *Terminology*, (2000) 161-180.
12. Haarslev V., Muller R.: Description of the RACER System and its Applications. In *Proceedings of the International Workshop on Description Logics (DL-2001)*, Stanford, USA, (2001), 132–141
13. Joshi A.: *An Introduction to Tree Adjoining Grammars*. In *Mathematics of Language*, John Benjamins Publishing, Amsterdam/Philadelphia (1987), 87–115.
14. Lopez, P.: *Robust Parsing with Lexicalized Tree Adjoining Grammars*, Ph.D.Thesis, INRIA, Nancy, France (1999).
15. Miller, G., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.:*Introduction to WordNet: An On-Line Lexical Database*. In *International Journal of Lexicography*, 3(4), (1990), 302–312.
16. Riloff,E., Lorenzen, J.: Extraction-based Text Categorization Generating Domain-Specific Role Relationships Automatically. In T.Strzalkowski (ed.): Natural Language Information Retrieval, Kluwer Academic Publishers, (1999), 167-196.
17. Riloff, E., Shepherd, J.: A Corpus-Based Approach for Building Semantic Lexicons. In Proceedings of the Second Conference on Empirical Methods in Natural Language Processing (1997).
18. Schimd, H.:Probabilistic Part-of-Speech Tagging Using Decision Trees. In Proceedings of the International Conference on New Methods in Language Processing, Manchester, United Kingdom (1994)
19. Vilain, M.: Inferential Information Extraction. In M.Pazienza (ed.): Information Extraction, LNAI 1714, Springer-Verlag, (1999), 95–119.

Automatic Identification of European Languages

Anna V. Zhdanova[1,2]

[1] A.P. Ershov Institute of Informatics Systems, Novosibirsk 630090, Russia
[2] Novosibirsk State University, Novosibirsk 630090, Russia
anna@sib3.ru

Abstract. We describe our word-based implementation of a language identifying system for the text messages written in European languages. Specifically, we use and compare linguistic (based on functional words) and statistic (based on the word frequency) approaches to construction of the identifying vocabularies. Our version of the statistic approach copes with the differences in degrees of word overlap among languages and the problem of the small-size messages. In addition, it allows an user to choose the accuracy of language identification. At present, our system identifies 8 languages (Bulgarian, English, French, German, Italian, Russian, Spanish and Swedish) in various encodings. With the identifying vocabularies of limited size (less than 1500 keys per language), the accuracy of identification attains 99% even for the messages containing only one sentence.

1 Introduction

Automatic language identification (LI) is the process by which the language of a textual message or of a digitized speech utterance is recognized by a computer. The understanding and developing of the principles which are or may be used to perform LI is of high current interest due to the growing internationalization of Internet. For European international companies, for example, LI is one of the primary steps in solving the problems of electronic message classification and auto-responding, because no single language is spoken throughout Europe. Considering the situation of real life where incoming messages written in the same language may arrive differently encoded, encoding identification is also important, e.g., for redirecting a message in an appropriate form to the subsequent message classification system or to a human operator. Historically, LI of textual messages has been accomplished by using such techniques employing dictionaries or other language sources as (i) a bigram probability-multiplication method [1] and the trigram-based methods [2,3], (ii) scoring the words with five characters or less [3], and (iii) scoring the grammatical words and frequent endings [4]. The word-based techniques [(ii) and (iii)] seem to be more convenient, because unlike the letter-based technique (i) they are able to provide LI of multilingual messages.

In addition, the methods for classifying documents by language have been developed. They may use less information about languages than the LI methods

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 76–84, 2002.

and are able to group documents into language similarity clusters. The algorithms for classifying documents by language are based on the vector space model of information retrieval and involve various approaches employing, e.g., linear algebra [5].

At present, there exist online implementations of LI systems such as Euclid [6], Lextek Language Identifier [7] and SILC [8]. All these products seem to employ the letter-based methods or n-gram algorithms, as well as the most mentioned above papers. These methods are good for the typical incoming texts, but they usually fail when a text includes relatively large number of foreign words or transliterated proper and geographical names (e.g., such a message as "Anna Zhdanova will go to Stockholm." may be erroneously identified as not being written in English). Unfortunately, the latter situation is common for e-mail messages and Internet in general. Thus, there is a need of effective methods of word-based vocabularies construction. Although the Lextek Language Identifier operates with up to 260 language and encoding modules and provides rather fast recognition, the processing of messages containing less than 200 characters (about 30 words) is however not allowed there. Thus, the problem of LI of small inputs is still open.

In this paper, we describe and compare two word-based methods of construction of the identifying vocabularies. The first conventional method implies involving experts who know languages [4]. The second new method involves corpora and statistics. In the latter case, we take into account the message size a user expects to have. Thus, the LI system's database is optimized according to the user's needs. This makes it possible to reach the maximum performance efficiency. The resulting set of the identifying vocabularies depends on the languages included in a specific configuration of the LI system. This strategy improves the performance of the LI system. In addition, our system allows a user to choose the needed accuracy of LI.

2 Principles of Word-Based Approaches to LI

2.1 Construction of Identifying Vocabularies

Each LI system usually contains a set of identifying vocabularies. In turn, each identifying vocabulary contains a set of key words corresponding to a given language and encoding. During the parsing of an incoming message, the occurrences of key words are calculated for each vocabulary. When this process is completed, the counts for the vocabularies are compared, and the language and encoding of the vocabulary with the largest count (i.e., the vocabulary containing the key words which occurred most often) are assigned to a message. To prevent misidentification of messages exhibiting only a few key words or written in a language the system does not have, one may establish a threshold value for determining a language. If the ratio of the number of the key words found in the vocabulary with the largest count and the total number of words in a message is lower then the threshold value, no language is identified.

For efficient LI, the identifying vocabularies should be representative. On the other hand, their sizes have to be limited in order to enable the system to work quickly. To satisfy these requirements, we have first constructed the sets of identifying vocabularies based on the linguistic approach, similar to the already known method of grammatical (or functional) words [4]. This method is however not suitable in the case when the processed input is small. For the LI system including four European languages, for example, the grammatical word method becomes inefficient when the number of words in the input is less than eight [4]. With increasing the number of languages, the minimum size of the processed input is expected to be larger.

To provide the possibility to work with relatively small messages, we have used the statistic algorithm. The latter algorithm takes into account the expected number of words in a message and the needed accuracy of LI (these two parameters are set by an user) and compiles the corresponding set of the appropriate identifying vocabularies from the given corpora.

Linguistic Approach to Construction of Identifying Vocabularies. Usually text consists of sentences. The main idea of the linguistic approach is that at least one word from a typical sentence written in some language should be included in the corresponding identifying vocabulary. As a rule, typical sentences contain determiners, prepositions, pronouns, auxiliary verbs, numerals and some words specific for a particular subject domain. In the framework of the linguistic approach, the identifying vocabularies are constructed manually by the experts who are acquainted with the languages and know the appropriate words. The vocabularies compiled in our group make it possible to operate with six languages including English, French, German, Russian, Spanish and Swedish. The sizes of the vocabularies are different due to the difference in the structure of languages (inflections, cases, etc.).

Statistic Approach to Construction of Identifying Vocabularies. The linguistic approach described above requires involving experts in languages. Such experts are not always available. Furthermore, the results obtained depend on experts' educational background, field of expertise, etc. For this reason, there is no guarantee that the corresponding identifying vocabularies are representative. To avoid these complications, we have developed a statistical algorithm for construction of identifying vocabularies. In the framework of this algorithm, the identifying vocabularies for a set of languages are worked out on a set of corresponding corpora. In particular, we have constructed two sets of identifying vocabularies. The first one is for the same six languages that were handled by using the linguistic approach (Sec. 2.1.1). The second one contains six plus two languages, including English (En), French (Fr), German (Ge), Russian (Ru), Spanish (Sp), Swedish (Sw), Bulgarian (Bu) and Italian (It).

The algorithm used depends on the three parameters: n, the expected number of words in a message or a number of words in a single test entry (for testing); a ($a \in [0, 1]$), the needed accuracy of LI; and $\{c_i\}$, a set of corpora (the subscript i

indicates a language). The numbers n and a are expected to be chosen by an user according to his/her needs. He/she should also list the languages which are to be identified. The set of corpora employed now is "standard", i.e., it corresponds to our own choice. In principle, however, the set may be chosen in order to take into account the specifics of the user's field of interests.

To construct the identifying vocabularies corresponding to given values of n and a, it is convenient to introduce the probability P that a word from a typical message written in one of the included languages belongs to the identifying vocabulary of this language. The probability that a word is not found is accordingly equal to $1 - P$. For a typical message with the expected size n, the probability that no words belong to the identifying vocabulary equals $(1 - P)^n$, provided that the correlations in the occurrence of words are not significant. The probability that at least one of the words belongs to the identifying vocabulary is equal to $1 - (1 - P)^n$. The latter probability is identified with the accuracy a, i.e., we use $1 - (1 - P)^n = a$, or

$$(1 - P)^n = 1 - a. \tag{1}$$

By solving this equation, we can calculate P for given values of n and a.

With the specification above, the algorithm of the choice of the key words for the identifying vocabularies is as follows:

1. For each language, a list of words used in the corresponding corpus is constructed.
2. The frequency f_i^j of the occurrence of word j for corpus i is calculated. By definition, f_i^j is the ratio of the number of occurrence of word j to the total number of words in c_i.
3. The words found in more than one corpus are removed.
4. The remaining words are sorted in each list according to their frequencies in a decreasing order. The top words (with the highest frequencies) are included into the corresponding identifying vocabularies as long as the sum of their frequencies is lower than P.

For this algorithm, the sizes of the identifying vocabularies for different languages are different. Every time a new language is added into the LI system, all the identifying vocabularies have to be rebuilt. With increasing n, the sizes of the identifying vocabularies are rapidly decreasing.

Theoretically, due to the inter-lingual homonyms removal [step 3] and possible shortcomings in corpora, this algorithm may lead to an undesired situation when at step 4 all the words remaining in a list are already added to the identifying vocabulary, but the sum of the word frequencies is still less than P. In such a case, the LI system includes all the words from a given list to the corresponding identifying vocabulary and simultaneously warns an user that it works at risk of not obtaining the accuracy a. With our identifying vocabularies, we had no this problem provided that $a \leq 0.99$ and $n \geq 10$.

2.2 Inter-lingual Homonyms and Misrepresentative Words

For the traditional linguistic approach and the statistic approach with step 3 omitted, some of the key words usually belong to several identifying vocabularies. This may result in mistakes in LI, e.g., when a word is an inter-lingual homonym, but is not included into all the identifying vocabularies which might contain this word. For example, the word "se" is a verb with the meaning "to see" in Swedish and a reflexive pronoun with the possible meaning "himself"/"herself" in French. The Swedish identifying vocabulary does not contain the key word "se", while the French one does. If the LI system encounters the word "se" in a message written in Swedish, it will add a point to the count of French. As a result, this will head towards the wrong direction.

To tackle the problem above, it makes sense to prevent the appearance of inter-lingual homonyms in the identifying vocabularies. In the framework of the statistical approach, this was attained by introducing step 3. For the linguistic approach, we have first compiled extensive dictionaries by using large corpora. If a key word from an identifying vocabulary belonged to at least two extensive dictionaries, it was removed from the identifying vocabularies.

An opportunity of choosing various corpora for constructing identifying vocabularies is an advantage and a flaw of the statistical method. While an user is able to tune the LI system to a better performance for a specific domain, the identifying vocabularies lose their universality. For example, the identifying vocabularies, compiled by employing the corpora related to programming, usually contain words typical for this field. But these words are misrepresentative for other domains. This problem can partly be resolved by combining corpora corresponding to different fields.

2.3 Multiple Encodings

In reality, incoming messages written in the same language may be differently encoded. For the languages based on the Roman alphabet (e.g., English, German), the encoding Cp1252 (i.e., Windows Western Europe / Latin-1) is usually sufficient to cover all the letters including the "substandard" ones (e.g., Spanish "ñ" or German "ß") and diacritics. But for Russian, Bulgarian, Ukrainian and other languages based on the Cyrillic alphabet, a few encodings such as Cp1251, Cp866 and koi8-r are widespread. For this reason, the key words from the identifying vocabularies for these languages should be written in all the encodings used in real messages.

3 Results

The system performing LI is written in the Java language in accordance with the principles of the object-oriented programming. The language identifier takes a file or a list of files as an input, makes the parsing by its lexical analyzer, matches the picked out words to the identifying vocabularies, calculates the

score for every possible language and encoding pair, and after all, returns the language and encoding with the highest score.

Our LI system makes it possible to construct and add new identifying vocabularies for different languages and encodings. To study its performance, we have constructed several sets of identifying vocabularies. In addition, we have formed a database of test entries modeling typical incoming messages. The latter database consists of texts representing the banking, insurance, computer semantic domains and fiction literature. Most texts were extracted from the appropriate web pages and used with no preprocessing except converting to the plain text files by removing tags. Some texts were found to contain foreign words and phrases. While the presence of such texts in the database employed to form test messages is a model of real situation, their presence in the corpora created for construction of the identifying vocabularies resulted in the appearance of the "false" inter-lingual homonyms and eventually slightly deteriorated the identifying vocabularies.

In reality, incoming messages usually contain several sentences. Our LI system processes a whole message as an input. To test the system, we used relatively small entries consisting of a single sentence. A sentence was defined to be the characters between the punctuation signs ".", "?", "!" and the symbols signifying the beginning and end of text files.

3.1 Linguistic Approach

In the framework of the linguistic approach, the identifying vocabularies are constructed manually (Sec. 2.1.1). Table 1 shows the sizes of such vocabularies before and after removal of inter-lingual homonyms. The difference between the upper and lower rows is proportional to the degree of overlap among the languages.

Table 1. Sizes (number of words) of the linguistically constructed identifying vocabularies before and after removal of inter-lingual homonyms.

	English	French	German	Russian	Spanish	Swedish
Before removal	581	821	250	1053	181	254
After removal	444	642	210	1053	40	213

To test the identifying vocabularies, the LI system determines a language and encoding for each test entry and compares them to the language and encoding the entry is written in. Table 2 shows the percentage of the correctly identified test entries for the linguistically constructed identifying vocabularies. Spanish and French possessing the highest number of widespread inter- lingual homonyms are seen to have the lowest identification accuracy. This is in agreement with the study [5] of classification of documents by language, where Spanish is mentioned as the most commonly misclassified language.

Table 2. Percentage of correctly identified test entries before and after removal of inter-lingual homonyms for the linguistic approach to construction of the identifying vocabularies.

	English	French	German	Russian	Spanish	Swedish
Before removal	99.0	94.2	98.8	95.7	97.9	98.1
After removal	99.9	90.5	99.7	99.8	70.3	99.9

Table 3. Sizes (number of words) of the identifying vocabularies (for six and eight languages, $a = 0.99$ and $n = 10$) constructed by employing the statistic algorithm with and without step 3.

Number of languages	En	Fr	Ge	Ru	Sp	Sw	Bu	It
6 (without step 3)	24	34	69	97	33	69	-	-
6 (with step 3)	146	527	270	107	433	173	-	-
8 (without step 3)	24	34	69	97	33	69	91	42
8 (with step 3)	146	995	248	599	714	160	742	1427

The results presented in Table 2 indicate that for the linguistic approach the removal of inter-lingual homonyms enlarges the number of unidentified entries. In other words, the removal of homonyms from the manually constructed identifying vocabularies neither improves the performance of the LI system nor solves the problem of small incoming messages.

3.2 Statistic Approach

Using the statistic approach, we have constructed and tested four sets of identifying vocabularies. Table 3 demonstrates their sizes [with and without step 3] in the cases of six and six plus two languages. The corpora employed contained 30000 words for each language. The accuracy of LI and the expected size of entries were $a = 0.99$ and $n = 10$, respectively. For these parameters, Eq. (1) yields $P = 0.3691$. Thus, we have filled the identifying vocabularies in such a way that the sum of the word frequencies within every vocabulary is not lower than 0.3691.

The results of the tests, presented in Table 4, indicate that the expected LI accuracy, $a = 0.99$, is obtained when the complete (with step 3) statistical algorithm is used. Adding Italian is seen to affect slightly the identification of the languages, based on the Roman alphabet, mainly due to the imperfect corpora and misrepresentative words. While adding Bulgarian produces an appreciable increase in the number of misidentifications for Russian, based also on the Cyrillic alphabet.

4 Conclusion

Using the word-based approach, we have worked out the LI system for European languages. Specifically, several alternative sets of identifying vocabularies have

Table 4. Percentage of correctly identified test entries before and after removal of interlingual homonyms for the statistic approach to construction of identifying vocabularies.

Number of languages	En	Fr	Ge	Ru	Sp	Sw	Bu	It
6 (without step 3)	95.3	91.2	99.5	98.9	98.8	98.7	-	-
6 (with step 3)	98.7	99.1	100.0	99.3	99.3	99.8	-	-
8 (without step 3)	94.5	86.6	99.3	49.3	94.4	97.3	96.3	94.5
8 (with step 3)	97.8	98.5	99.2	89.6	98.1	99.9	98.8	98.4

been constructed by employing the linguistic and statistic methods. The latter method is found to be superior, because it allows one to construct the identifying vocabularies taking into account the chosen expected message size and LI accuracy. The sizes of the identifying vocabularies are minimized according to the special formula [Eq. (1)], and this feature provides fast (and accurate) performance of the LI system. For 12 language-encoding pairs recognized by the LI system, the speed of processing is around 2 seconds per megabyte of text and varies from 0.26 to 0.57 milliseconds per sentence depending on the language (for Intel Celeron, 434.31 MHz). Furthermore, the statistic method makes it possible to add a new language to the LI system without any linguistic knowledge and/or employing additional tools. However, this method needs corpora. The identifying vocabularies are optimal when the corpora represent an user field of interest. Thus, the work of our LI system is highly efficient when tuned to the needs of a particular user.

In comparison with the existing LI systems, our LI system with statistically constructed identifying vocabularies has an advantage of tuning its performance. For this reason, it allows one to operate with messages of smaller size or, alternatively, enlarge the speed of LI when the "small-message" feature is not needed. For the messages containing only one sentence, the accuracy of LI approaching to 99% is achieved independently on the degrees of overlap among the languages and the number of languages included into the LI system.

Acknowledgements

The author thanks F. Dinenberg, A. Kleymenov, I. Kononenko, D. Levin, P. Mankevich, D. Petunin, A. Semenov and D. Shishkin for participation in the project and useful discussions.

References

1. Beesley, K.R.: Language Identifier: A Computer Program for Automatic Natural-Language Identification of On-Line Text. In: Languages at Crossroads; Proceedings of the 29-th Annual Conference of the American Translators Association (1988) 47-54.
2. Schmitt, J.C.: Trigram-based Method of Language Identification. US Patent 5,062,143 (1991).

3. Grefenstette, G.: Comparing Two Language Identification Schemes. In: Proceedings of 3-rd International Conference on Statistical Analysis of Textual Data (1995).
4. Giguet, E.: Categorization According to Language: A Step Toward Combining Linguistic Knowledge and Statistic Learning. In: Proceedings of the International Workshop on Parsing Technologies (1995).
5. Mather, L.: A Linear Algebra Approach to Language Identification. In: Proceedings of the 4-th International Workshop on Principles of Digital Document Processing (1998) 92-103.
6. Euclid: Encoding and Language Identification. http://www.basistech.com/products/text-processing/euclid.html (2002).
7. Lextek Language Identifier. http://www.languageidentifier.com (2001).
8. Système d'Identification de la Langue et du Codage. http://www-rali.iro.umontreal.ca/SILC/SILC.en.cgi (2002).

A Method for Maintaining Document Consistency Based on Similarity Contents

Farid Meziane* and Yacine Rezgui**

University of Salford, Salford M5 4WT, UK
{f.meziane,y.rezgui}@salford.ac.uk

Abstract. The advent of the WWW and distributed information systems have made it possible to share documents between different users and organisations. However, this has created many problems related to the security, accessibility, right and most importantly the consistency of documents. It is important that the people involved have access to the most up-to-date version of the documents, retrieve the correct documents and should be able to update the documents repository in such a way that his or her documents are known to others. In this paper we propose a method for organising, storing and retrieving documents based on similarity contents. The method uses techniques based on information retrieval, document summarisation and term extraction and indexing. This methodology is developed for the E-Cognos project which aims at developing tools for the management and sharing of documents in the construction domain.

Keywords: Document Consistency, Document indexation, Ontology

1 Introduction

The main activity of most PC users is about creating, managing, deleting and retrieving electronic documents. Thanks to the existing file management systems, this organisation is performed using hierarchical structures. These structures categorise documents using their properties. For example, we would create a file "Lecture1.ppt" in the subdirectory "Lectures" which is itself a subdirectory of the "Object-Oriented Design" subdirectory. In fact we are associating some semantics to the created file. It is a lecture for the "Object-Oriented Design" module. However, using strict hierarchical filing can make it hard for users to perform the following operations [5]: *File documents:* documents can appear in only one place; *Manage documents:* locations in the hierarchy is used for organisational and management purposes; *Locate documents:* Document may be filed according to one criterion but retrieved according to another; *Share documents:* different structures for different people. The task becomes more complex when dealing with various documents of one or many organisations. The problem becomes even more complex if the WWW is used as the place to exchange and

* School of Sciences, Computer Science
** Information Systems Institute

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 85–96, 2002.

organise these documents. Another major problem faced with shared documents is consistency whereby everybody interested in the document should be aware of any changes made to the document. Some systems have been developed to address some of these issues. The Presto system [4,5] aims at creating placeless documents and attempts to create a more natural and fluid forms of interaction with a document space. DocMan [2] is a document management system which supports cooperative preparation, exchange and distribution of documents. The system particularly stressed on the loss of work done simultaneously on a document and access restrictions. The Zelig System [3] was developed for managing multiple representation documents.

In this paper we present a methodology for maintaining document consistency using similarity content. This methodology is developed for the E-Cognos project which aims at developing tools for the management and sharing of documents in the construction domain. The approach is based on generic principles related to information retrieval and knowledge management. The aim of this project is to exploit these principles to develop an approach that will support consistency across large knowledge repositories maintained in a heterogeneous and distributed collaborative business environment. The approach is based on a solid theoretical foundation, and will be deployed in a real business environment. The remaining of the paper is organised as follows: in section 2 we present the motivation and the background behind the project. Section 3 will define the different types of document handled in this project. In section 4 we present the generic model of the methodology used for poorly structured documents. Sections 5 and 6 presents variants of the methodology for documents with text formatting structure and highly structured documents. We end the paper with a short conclusion.

2 Background and Motivation

Numerous documents of diverse nature are involved in the construction domain. These documents are of two types: drawings and written documents. Drawings are the straightforward media to convey most of the information needed by construction companies and include a lot of information that can be hard to put into a textual format. They are usually more formal and comprehensive than text information. Moreover written documents are complementary to drawings, they are the traditional support of an engineering project description. Some of them such as building codes, examples of technical solutions, computation rules define the legal context of a project. Others like technical specifications documents or bill of quantities are generated by the engineering activities and often have a contractual importance.

The documents generated within the entire life cycle of a construction project, and especially during the design stage, need to be of quality in order to provide a reliable basis for contractors to perform their construction activities. Documents of quality are obtained by ensuring, during their production, a coherent and consistent structuring both on the logical and physical side. This structuring is relevant in the sense that the semantics of a document can be efficiently mastered and thus correctly described.

Moreover, a document has not only to be self consistent but needs also to be consistent with the entire project documentary base as well as the construction standard and regulation base. Furthermore, many practitioners and researchers in the construction domain have recognised the limitations of current approaches to managing the knowledge relating to and arising from a project in a distributive collaborative environment. Among the reasons for these limitations are:

- Much knowledge, of necessity, resides in the minds of the individuals working within the domain.
- The intent behind decisions is often not recorded or documented.
- The knowledge gained during a project is often poorly organised and buried in details. Hence, it becomes difficult to compile and disseminate useful knowledge to other projects.
- People frequently move from one project to another, making it difficult to track those involved in decision making.

Knowledge in the construction domain can be classified into the following three categories:

- *Domain knowledge:* this forms the overall information context. It includes administrative information (e.g. zoning regulations, planning permission), standards, technical rules, product databases, etc. This information is, in principle, available to all companies, and is partly stored in electronic databases.
- *Corporate knowledge:* this is company specific, and is the intellectual capital of the firm. It resides both formally in company records and informally through the skilled processes of the firm. It also comprises knowledge about the personal skills, and project experience of the employees and cross-organisational knowledge. The latter covers knowledge involved in business relationships with other partners, including clients, architects, engineering companies, and contractors.
- *Project knowledge:* this is the potential for usable knowledge and is the source of much of the knowledge identified above. It comprises both knowledge each company has about the project and the knowledge that is created by the interaction between firms. It is not held in a form that promotes reuse (e.g. solutions to technical problems, or in avoiding repeated mistakes), thus companies and partnerships are generally unable to capitalise on this potential for creating knowledge.

This overall context has often resulted in knowledge redundancy and inconsistencies, business process inefficiencies, and change control and regulatory compliance problems. Moreover, the introduction of new national regulations, or amendments made to existing ones, are often not handled effectively within organisations and projects.

3 Documents and Their Logical Representation

A document is a transitional and changing object defined within a precise stage of the Project Life Cycle. Generally, a document is related to many elaborated

documents of the Project Documentary Database. A document has one or many authors. It is described by general attributes such as a Code, an Index, a Designation, a Date of creation and a list of the document's Authors. Ideally, a list of document versions also keeps memory of any amendments made to the document during its lifecycle. An indexing system may be associated to the document. A document is submitted for approval according to a defined circuit of examiners representing diverse technical or legal entities. Each examiner issues a statement that enables the document to be approved, rejected or approved under reservation.

Documents have also been traditionally represented using a set of key words. These key words or indices can either be manually defined by a user with a good knowledge of the semantics of the document, or extracted automatically from the text of the document using proven Information Retrieval (IR) techniques.

A document has a logical and a physical structure, which are both used to convey in the best possible way its internal semantics. The physical structure of a document is described using a properly defined syntax supported by one or several software tools.

Each document should have ideally metadata attached to it. A possible solution for describing metadata is through RDF *(Resource Description Framework - a development based on XML)* that provides with a simple common model for describing metadata on the Web. It consists of a description of nodes and attached attribute/value pairs. Nodes represent any web resource, i.e. Uniform Resource Identifier (URI), which includes URL *(Uniform Resource Locator).* Attributes are properties of nodes and their values are text strings or other nodes.

Documents are classified based on their inherent nature and the structure they exhibit taking into account the specificities of the domain; the construction sector in the E-Cognos project. Three classes of documents have been identified, namely: *poorly structured documents, documents with a clear physical structure,* and *highly structured documents.*

Poorly structured documents are composed of text with no formal structure. These constitute the vast majority of construction documentation. Documents are treated here simply as black boxes. The set of amendments to this category of documents include modifying the content of the document and deleting the document.

Documents with a text formatting structure are documents that are tagged using HTML, or at best XML but without reference to a Document Type Definition (DTD). A physical structure in the form of a hierarchical tree, or hypertext link of nodes can, in theory, be easily generated from this representation. This structure offers a variety of possibilities in terms of text retrieval. These documents include direct references to other documents/document sections. The set of amendments to this category of documents include inserting a new document element (heading, paragraph, section, etc.), deleting/modifying an existing heading in a document.

Highly structured documents are instances of an XML-based meta-language. These documents have a semantic structure that can easily be used as a basis for

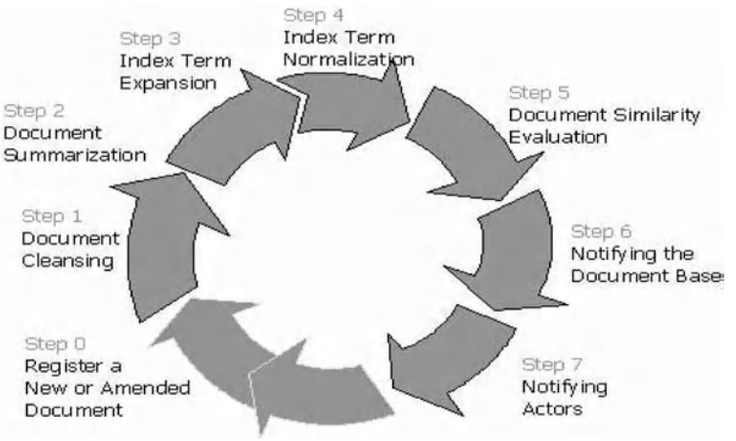


Fig. 1. System Overview for Poorly Structured Documents

text queries and retrieval. Ideally, we can envisage that all the documentation that is used and produced be an instance of a specific XML DTD over which users can exercise control over its internal semantics. These documents include naturally direct references to other documents/document sections. The set of amendments for this category of documents include: adding a new DTD element to the DTD language, instantiating a new DTD element within a document and deleting the instance of a DTD element within a document.

4 System Description

The general framework of the methodology, as shown in Figure 1, is for poorly structured documents. The methodology is composed of 8 steps and these are described in the following subsections. Step 0 is the entry point to the system. It can be the submission of a new document or the re-submission of a modified document. Both instances will go through the same process. A logical document is used for searching and other document related operations. A Physical document is only retrieved on users requests.

4.1 Document Cleansing Module

This step aims at reducing the document to a textual description by eliminating non-discriminating words. The resulting document contains mainly nouns and association of nouns that carry most of the documents semantics. A cleansed document reduces drastically text complexity allowing better performance in document retrieval and processing. This involves the following tasks:

- Lexical analysis of the text in order to treat digits, hyphens, punctuation marks, and the case of letters. This reduces the initial document into a subset of words that are potential candidates for index terms

- Elimination of stopwords to filter out words with very low discrimination values for retrieval purposes.
- Stemming the remaining words with the objective of removing affixes (prefixes and suffixes) and preventing the retrieval of documents containing syntactic variations of query terms, e.g. use, using, used, usage, etc.

4.2 Document Summarization

This step aims at providing a logical view of a document through summarization via a set of semantically relevant keywords. These are referred to, in this stage, as index terms. The purpose is to gradually move from a full text representation of the document to a higher-level representation. This module is composed of the following tasks:

Index Terms Extraction. In order to reduce the complexity of the text, as well as the resulting computational costs, the index terms to be retained are:

- All the nouns from the cleansed text. It is in fact argued that most, if not all, of the semantics of a text document is carried out by nouns as opposed to verbs, adjectives, and adverbs.
- Noun groups (non-elementary index terms) co-occurring with a null syntactic distance (number of words between the two nouns is null).

Extracting the Structure of the Document. This stage will only be possible if the document has been produced using a document formatting language, including RTF, SGML, HTML and XML. Each node of the resulting hierarchical structure will have an identifier that will be used to track nodes, their parents and children. A node might contain other elements including references within or outside the scope of the document, figures, tables, formulas, etc.

Establishing the Inverted File Structure of the Text. The purpose of an inverted file structure is to track the position of each index term occurrence in the text. The positions of index term occurrences can be tracked either on a word or character basis, or on a physical position basis (by pointing to node identifiers for example). The latter technique, referred to as Block Addressing, can only be used where a physical structure of a document is available. It presents the advantage of reducing space storage requirements. Documents with no clearly defined physical structure will make use of word addressing. Two activities are involved in this stage:

- Determining the raw frequency of each index term in the text; This is referred to as intra-clustering similarity in the Vector Model [1,6,7]. This aims at determining the number of times a term is mentioned in a document, as well as the location in the document of the term occurrence.

– Determining the number of documents in which each index term appears. This aims at counting the number of documents of the Document Knowledge Base (Project Knowledge Base and / or Corporate Knowledge base in which the term appears).

Calculating the Index Term Weight for the Document. The purpose here is to quantify the degree of importance in terms of semantics that the index term has over the document. It is proposed that the formula defined for the Vector Model [1,6,7] will be used.

4.3 Index Terms Expansion and Normalization Using a Construction Thesaurus and Ontology

This steps aims at normalizing the index terms obtained from the previous stage by using either direct ontology concept mapping wherever possible, or indirect ontology mapping by using a thesaurus as described in Figure 2. If no direct mapping exist between the initial index term and the list of ontology concepts then the thesaurus is used to provide synonyms for each term. The synonyms are used for indirect mapping. It is important to emphasise that the ontology is the structure that is used to convey semantics and maintain knowledge consistency across the project, corporate and domain layers. As such, the concepts of the ontology are the unique reference for the E-Cognos platform.

Ontology Concept Expansion Based on Concept to Concept Relationship. It is proposed in this methodology that the retained concepts be expanded based on their ontological direct relationships. We distinguish three main types of relationships:

– Generalisation/Specialisation Relationships
– Composition/Aggregation Relationship
– Concept association with varying semantics

Ontology Concepts Weighting. The ontology concepts resulting from a direct document index term mapping, indirect index term mapping, or ontology concept expansion need re-weighting. The following approach is proposed:

1. In case of a direct mapping the weight of the document index term is applied as such to the ontology concept.
2. In case of indirect mapping or concept expansion, it is proposed that a correlation factor be applied to the initial document index term weighting. The correlation factor is obtained by the cosine of the angle between the Index Term Vector and the Ontology Concept Vector. This is based on a technique used in Query Expansion Based on Similarity Thesaurus [8].

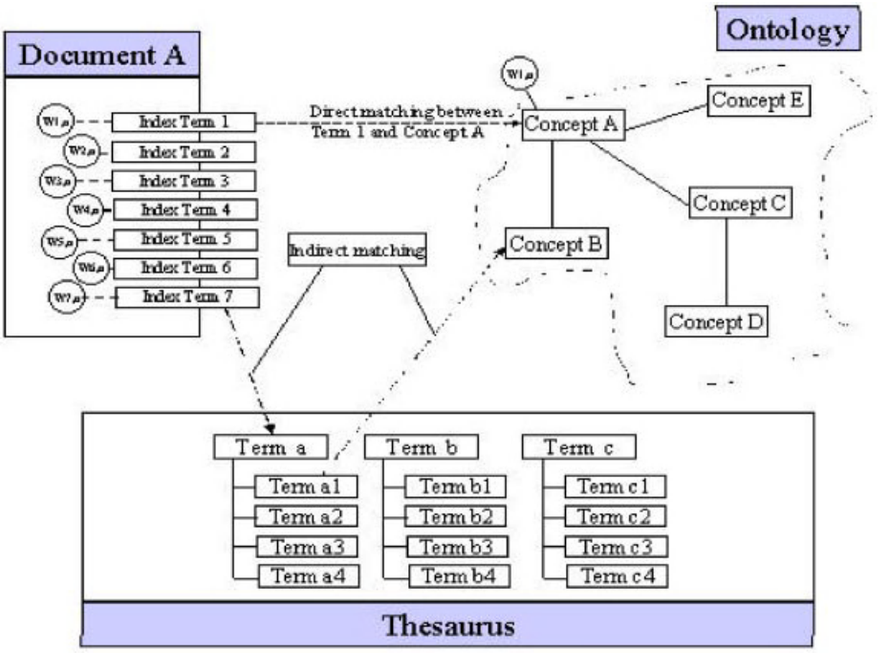


Fig. 2. Index Terms Mapping Against the Ontology

4.4 Document Similarity Evaluation

The purpose of this step is to compare the similarity between a newly uploaded/processed document with the remaining documents (or knowledge items) stored in the various knowledge repositories.

Document Similarity Calculation Against All the Document Set. The purpose here is to provide a function that evaluates the similarity between two documents. We adopt the following approach:

Let t be the number of index terms in the system and k_i a generic index term. $K = \{k_1, k_2, \ldots, k_i\}$ is the set of all index terms. A weight $W_{i,j} > 0$ is associated with each index term k_i of a document d_j . If an index term does not appear in the document text then $W_{i,j} = 0$. Therefore, with a document is associated an index term vector:

$$\overrightarrow{d}_j = (W_{1,j}, W_{2,j}, \ldots, W_{t,j})$$

Based on the document index term vector above, two documents d_i and d_j are represented as t-dimensional vectors. The approach adopted by the Vector Model to evaluate the similarity between a query and a document by measuring the correlation between their index term vectors is used. Furthermore, the similarity between two given documents will be measured by the correlation between

Table 1. An example of a Document Similarity Matrix

	Doc A	Doc B	Doc C	Doc D	Doc E	Doc F	Doc G
Doc A	1	sim(B,A)	sim(C,A)	sim(D,A)	sim(E,A)	sim(F,A)	sim(G,A)
Doc B	sim(A,B)	1	sim(C,B)	sim(D,B)	sim(E,B)	sim(F,B)	sim(G,B)
Doc C	sim(A,C)	sim(B,C)	1	sim(D,C)	sim(E,C)	sim(F,C)	sim(G,C)
Doc D	sim(A,D)	sim(B,D)	sim(C,D)	1	sim(E,D)	sim(F,D)	sim(G,D)
Doc E	sim(A,E)	sim(B,E)	sim(C,E)	sim(D,E)	1	sim(F,E)	sim(G,E)
Doc F	sim(A,F)	sim(B,F)	sim(C,F)	sim(D,F)	sim(E,F)	1	sim(G,F)
Doc G	sim(A,G)	sim(B,G)	sim(C,G)	sim(D,G)	sim(D,G)	sim(E,G)	1

their index term vectors. This correlation can be quantified by the cosine between these two vectors. $sim(d_i, d_j)$ varies between 0 and 1. An example of an illustration of the matrix is given in Table 1.

Establishing Document Clusters Based on the Similarity Table. The purpose here is to propose clusters of documents based on their degree of similarity. These clusters can directly be generated from the Document Similarity Matrix proposed in the previous section.

4.5 Notifying the Constituents of the Document Base

The purpose of this step is to notify relevant documents of the knowledge base, based on the nearest cluster(s), the potential risk of inconsistency that might exist as a result of a new event (upload of a new document, amendment to an existing document, etc.).

4.6 Notifying Relevant Actors

The purpose of this stage is for each potentially inconsistent document to notify actors who have subscribed an interest into the document (including authors) about this last event, and its potential degree of inconsistency. This notification will be materialised by the sending of an XML-based description of the metadata of the newly created or amended document to all the concerned actors.

5 Maintaining Document Consistency Based on Document Explicit Relationships

This case applies to documents that have a clear physical structure and make use of hypertext navigational and cross-referencing links. Hypertext allows non-sequential browsing and editing of text. It can be represented as a network of nodes that are correlated by direct links in a graph structure. Each node is associated with a block of text that can represent a paragraph, chapter, section, or even a web page. Two related nodes are connected one to the other by a

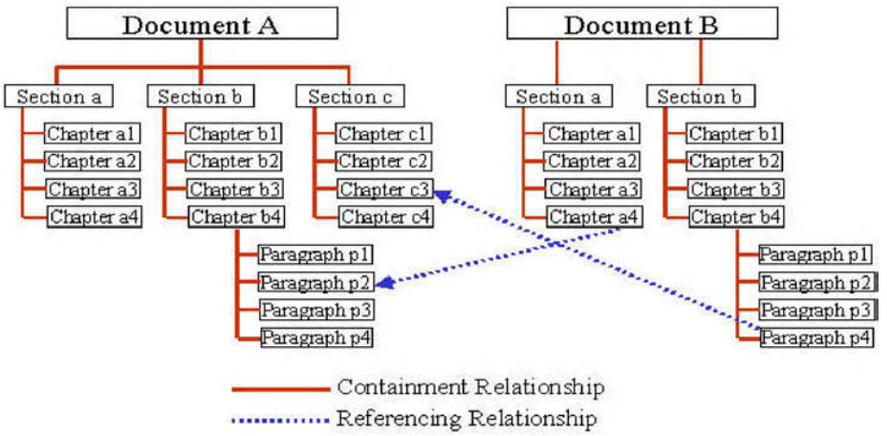


Fig. 3. Relationship Types in a Structured Hypertext Document

direct link, which correlates the text associated with these two nodes. This is explicitly described in the text by a special tag, or a highlighted portion of the text. Figure 3 describes the application of the method for this type of documents. The proposed rules to apply are as follows:

Rule 1: if a node is amended then the node in question as well as the recursive parents should be flagged as potentially inconsistent. For instance, if paragraph P4 of document B in Figure 3 is amended then chapter b4 of document B(Containment Relationship) and chapter C3 of document A (Referencing Relationship) are potentially inconsistent, and should be flagged as such.

Rule 2: if a node is amended then the external nodes that are referencing it might be potentially inconsistent. These external nodes should be flagged as potentially inconsistent as they do reference an amended node. We consider that it is up to the author to look after the consistency of the internal references of the node being modified.

6 Maintaining Consistency of Highly Structured Documents

Highly structured documents are best represented by XML DTD compliant documents. XML documents allow human and machine-readable semantics markup. XML allows users to define new tags and impose data validation on them. This raises the problem of having unified and standardised definitions of tags used across documents. In that respect, it is highly recommendable to use a standardised DTD for authoring XML documents. This is already an area of intense activity (AECXML, bcXML, etc.). It is recommended in this approach that the elements of a given XML DTD be interpreted semantically by indexing them, by the author of the DTD, to the concepts of the ontology. Concepts that

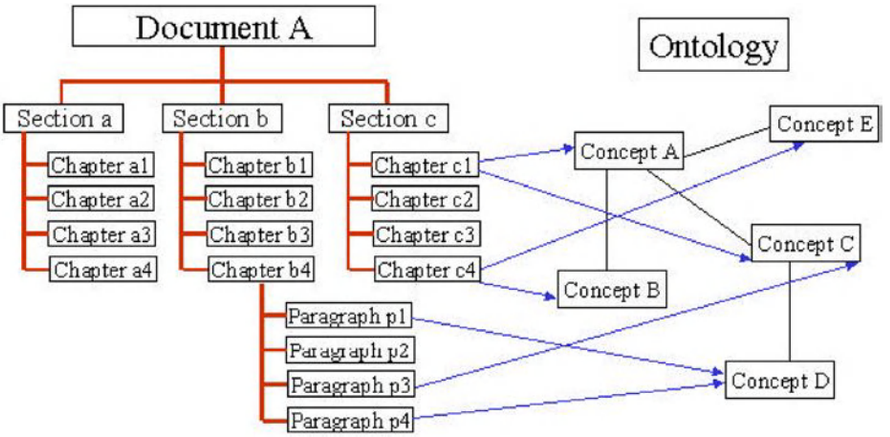


Fig. 4. XML Elements Indexing to the Construction Ontology

highly describe the semantics of the contents of the instance of the DTD element are selected and retained by the DTD author(s) as this is a knowledge intensive activity. Therefore, each DTD element will be associated and indexed to a set of ontology concepts, as described in Figure 4.

In the same way, each ontology concept will be associated with a set of indexing DTD elements. Let us take an example:

DTD_Element_A1 has ontology indexes: *(Ont_Con_1,Ont_Con_3, Ont_Con_4)*
DTD_Element_A2 has ontology indexes: *(Ont_Con_2, Ont_Con_4, Ont_Con_6)*
DTD_Element_A3 has ontology indexes: *(Ont_Con_4, Ont_Con_6, Ont_Con_8)*

If an instance of a *DTD_Element* is amended, then the ontology concepts that index this element will be used to characterize this amendment. A simple but not very effective approach is to flag all documents that index the same ontology concepts of a document/or DTD Element instance that has been amended as potentially inconsistent. Using this approach, instances of *DTD_Element_A2* and *DTD_Element_A3* will be flagged as potentially inconsistent following an amendment to *DTD_Element_A1*.

A further step, which makes use of a more sophisticated approach, will attempt to retain only the instances from the flagged DTD elements that contain or reference the same ontology concept instance. This implies that the E-Cognos platform maintains instances of ontology concepts throughout the system.

7 Conclusion

The work presented in this paper is an initial attempt to specify a methodology for maintaining document consistency across the knowledge repositories of the construction domain. The methodology uses generic principles related to information retrieval and knowledge management that can be incorporated into an

approach that supports consistency across large knowledge repositories maintained in a heterogeneous and distributed collaborative business environment. E-Cognos aims at exploiting those principles to develop such an approach based on a solid theoretical foundation, and to deploy it in a real business environment in the context of the project partners. The methodology will be used in the construction domain. However, the model is generic and should be applicable to any other domain. Few changes might be necessary to take into account the nature of the document of the new domain and the use of another ontology which structure may influence some processes of the proposed model. A web-based implementation will be used for the E-Cognos project and this methodology will be implemented using Java and related technologies. It is also worth mentioning that the proposed methods assume that all documents have been authored using a common natural language. Moreover, the multi-lingual aspect of documents has not been addressed at the moment but will be addressed at a later stage.

Acknowledgements

The authors as well as the E-Cognos Consortium would like to acknowledge the financial support of the European Commission under the IST programme.

References

1. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.
2. A. Bäcker and U. Busbach. DocMan: A document management system for cooperation support. In *Proceedings of the Hawaii International Conference on System Sciences (HICSS-29)*, pages 82–91, 1996.
3. A. Celentano, S. Pozzi, and D. Toppeta. A multiple presentation document management system. In *Proceedings of the 10th Annual Conference on Systems Documentation*, pages 63–71, 1992.
4. P. Dourish, W.K. Edwards, A. Lamarca, and M. Salisbury. Presto: An experimental architecture for fluid interactive document space. *ACM Transactions on Computer-Human Interaction*, 6(2):133–161, 1999.
5. P. Dourish *et al.* Extending document management systems with user-specific active properties. *ACM Transaction on Information Systems*, 18(2):140–170, 2000.
6. G. Salton and M. Lesk. Computer evaluation of indexing and text processing. *Journal of the ACM*, 15(1):8–36, 1968.
7. G. Salton and C. Yang. On the specification of term values in automatic indexing. *Journal of Documentation*, (29):351–372, 1973.
8. Q. Yonggang and H. Frei. Concept based query expansion. In *Proceedings of the 16th Annual International Conference ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 160–169, 1993.

Evaluation and Construction of Training Corpuses for Text Classification: A Preliminary Study*

Shuigeng Zhou[1] and Jihong Guan[1,2]

[1] State Key Lab of Software Engineering, Wuhan University, Wuhan, 430072, China
Zhousg@whu.edu.cn
[2] School of Computer Science, Wuhan University, Wuhan, 430079, China
jhguan@wtusm.edu.cn

Abstract. Text classification is becoming more and more important with the rapid growth of on-line information available. It was observed that the quality of training corpus impacts the performance of the trained classifier. This paper proposes an approach to build high-quality training corpuses for better classification performance by first exploring the properties of training corpuses, and then giving an algorithm for constructing training corpuses semi-automatically. Preliminary experimental results validate our approach: classifiers based on the training corpuses constructed by our approach can achieve good performance while the training corpus' size is significantly reduced. Our approach can be used for building efficient and lightweight classification systems.

Keywords: Text classification, training corpus, evaluation, construction.

1 Introduction

With the rapid growth of online information available, text classification has become one of the key techniques for processing and organizing massive text repositories. Text classification is a supervised learning process, defined as assigning category labels (pre-defined) to new documents based on the likelihood suggested by a training set of labeled documents [1]. This technique has been used to classify news stories [2], to find interesting information on the Web [3], and to guide a user search through hypertext [4], *etc*. In the past decades, text classification has been extremely studied in machine learning, pattern recognition and information retrieval areas; a number of approaches for text classification were proposed [1], which include decision trees [5], regression models [6], *k*NN (*k*-Nearest Neighbor) classification [1,12], Bayesian probabilistic methods [7], inductive rule learning [8], neural networks [9], Support Vector Machines [10] and Boosting methods [11,13], to name a few. Obviously, research in the past had focused on proposing new classification methods for better classification performance (*precision* and *recall*). Once a new classification method was proposed, it was evaluated over some commonly used text corpuses, and its per-

* This work was supported by Hubei Provincial Natural Science Foundation (No. 2001ABB050) and the Natural Science Foundation of China (NSFC) (No. 60173027).

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 97–108, 2002.

formance was compared with that of the existing methods. Certainly, a newly proposed method is usually advantageous over the existing methods somehow or other.

On the other hand, it was observed that even for a specified classification method, classification performances of the classifiers based on different training text corpuses (here we use *training corpus* and *training text corpus* equally) are different; and in some cases such differences are quite substantial [7]. This observation implies that a) classifier performance is relevant to it's training corpus in some degree, and b) good or high quality training corpuses may derive classifiers of good performance. Unfortunately, up to now little research work in the literature has been seen on how to exploit training text corpuses to improve classifier's performance.

With this consideration in mind, in this paper, we study the following two problems. The aim is to explore practical approach to enhance classifier performance from the point of training text corpus' view.

- How to evaluate a training corpus, or how to measure the quality of a training corpus and compare the qualities of two different training corpuses? More simply, what training corpus is good or bad as far as classifier training is concerned? Some measures are necessary for evaluating the quality of training corpuses.
- How to construct training corpuses of high quality? Currently, training corpuses for text classification research were collected and compiled manually [14,15, 16]. But constructing text corpus manually is time-consuming and expensive. What is more, there is no method available to guarantee the quality of the constructed corpuses.

Contributions of this paper are in three aspects: first, three properties of training corpuses are proposed for evaluating training corpuses' quality; then based on the proposed properties, an algorithm is designed for constructing training corpuses of high quality semi-automatically; and finally, experiments are conducted to validate the proposed approach.

Note that in this paper the vector space model (VSM) is used as document representation model and *k*NN is the default classification method. The rest of the paper is organized as follows. Section 2 introduces three properties of training corpuses for evaluating training corpuses' quality; and Section 3 describes an algorithm for constructing training corpuses of high quality semi-automatically; Section 4 presents some experiments to evaluate the proposed approach; and finally Section 5 concludes the paper and outlines some future research directions.

2 Properties of Training Text Corpuses

Before exploring the properties of training corpuses, we give a formal definition of the text classification problem in a general sense.

Definition 1. A *text classification system* (or simply *TCS*) can be described as a 5-ary tuple, that is, $TCS=(CS, TC, CM, PM, CF)$. Here,

1) *CS* indicates the pre-specified class space, it consists of a set of predefined classes, *i.e.*, $CS=\{c_1, c_2, \ldots, c_n\}$. For simplification of analysis, in this paper we assume that *CS* has flat structure (rather than hierarchy), that is, all classes in *CS* are in the same conceptual level and semantically disjointed. Besides explicit classes structure, *CS* also implies the semantic contents of these classes, *i.e.*, all documents possibly included in these classes. We denote *Docs*(*CS*) all documents

possibly implied in *CS,* and specifically, $Docs(c_i)$ all documents implied in class c_i. Naturally, we have $Docs(CS)= Docs(c_1)\cup Docs(c_2) \cup\ldots\cup Docs(c_n)$.

2) *TC* means the training corpus, it consists of training documents that distribute over the classes in *CS*. Here, we denote *docs*(*TC*) all documents in *TC*; and specifically $docs(c_i)$ the subset of documents in *TC* that belong to class c_i. Analogously, $docs(TC)=docs(c_1)\cup docs(c_2) \cup\ldots\cup docs(c_n)$. Obviously, $docs(TC)\subset Docs(CS)$, and $docs(c_i)\subset Docs(c_i)$ (i=1-n). Note we define *docs*(•) and *Docs*(•) as different functions. The former is used for concrete training corpuses; and the latter is for class space.
3) *CM*, *PM* and *CF* stand for classification method, classification performance measurement and the trained classifier respectively.
4) Among these *TCS* elements above, the following relationships exist:

$$CM : (CS, TC) \rightarrow CF ;$$

$$PM : (CF, test\_docs) \rightarrow R^+ .$$

Above, *test_docs* stands for a set of test documents. The first formula reflects the process of classifier training, which maps the training corpus of a certain class space to a certain classification model (or classifier) based on the specified classification method. The second formula represents the process of classifier evaluation. Usually, research on text classification is to find *classification method* with which a classifier is trained so that *PM*(*CF*, *test_docs*) can be maximized as much as possibly. However, in this paper we are to explore how to build *training corpuses* so that the trained classifier can achieve optimal *PM*(*CF*, *test_docs*). That is the major difference between our work and the research in the literature.

According to the vector space model (VSM), given a set of document features (or indexed terms of the training corpus) $\{t_1, t_2, \ldots, t_n\}$, each document in training corpus can be represented by an n-dimensional vector, in which each dimensional component stands for the weight of the corresponding document feature. For document d, we have $\vec{d} = (w_1, w_2, ..., w_n)$, here w_i (i=1-n) is the weight of ith feature t_i. We assume that all document vectors are normalized into unit length, and the inner product of two document vectors are used to measure the similarity coefficient between the corresponding two documents, that is, $sim(d_1, d_2) = \vec{d}_1 \bullet \vec{d}_2$.

From the point of geometry view, every document is equivalent to a unit vector in document space (n-dimensional space, or hyperspace). Basically, documents in the same class are relatively closer with each other in document space compared to those not in the same class, that is, distances (or similarities) among them are relatively smaller (or larger). It can be seen that document vectors of the same class form a relatively dense hyper-cone in document space, and a training corpus corresponds to a number of hyper-cones, each of which corresponds to a class. Certainly, different hyper-cones may overlay with each other. Fig. 1 illustrates an example of training corpus with only two distinctive classes in 3D document space.

We denote $Vol(Docs(c_i))$ the volume of space occupied in document space by all document vectors of class c_i, which roughly corresponds to the space volume of the hyper-cone formed by all document vectors of that class. Analogously, we denote $Vol(docs(c_i))$ the volume of space occupied in document space by the training document vectors of class c_i of a given training corpus.

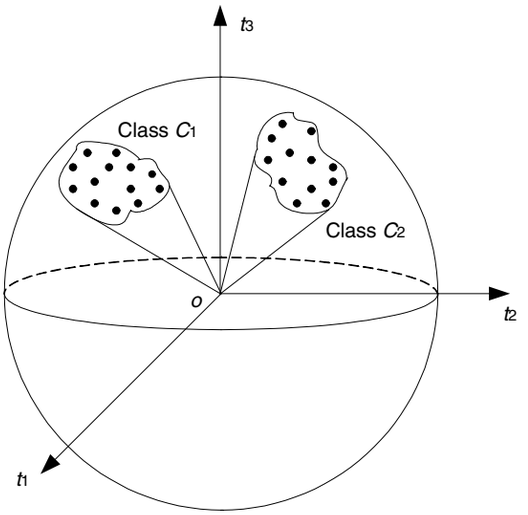


Fig. 1. Training corpus with two distinctive classes in 3D document space

Here, we treat $Vol(Docs(c_i))$ equally to the semantic content of class c_i. The reason lies in the observation: once class space *CS* is given, then semantic content of each class in *CS* as a whole is settled. Considering that in document space, each class corresponds to a relatively dense and bounded hyper-cone area, correspondingly the volume of space occupied in document space by all document vectors of each class is determined, even though the number of all possible documents implied in each class is unknown and unlimited.

Analogously, given a training corpus, the number of documents included in each class is also given; subsequently the volume of space occupied in document space by training document vectors of each class is also decided. However, for an arbitrary class c_i, $Vol(docs(c_i))$ is only part of $Vol(Docs(c_i))$; in other words, the training documents of class c_i represent only partial semantic content of class c_i. Theoretically, $Vol(docs(c_i))$ can infinitely reach $Vol(Docs(c_i))$ as the number of training documents in c_i increases.

Semantically, to describe the partial relation of training corpus with the class space, we propose the *coverage* property for training corpuses as follows. As we have assumed that all classes of a training corpus is semantically disjointed, when exploring the properties of training corpus, we will first focus on a single class, and then advance to the whole training corpus.

Definition 2. For class c_i (i=1~n) of training corpus *TC* with class space $CS=\{c_1, c_2, \ldots, c_n\}$, its *coverage* is the ratio of $Vol(docs(c_i))$ over $Vol(Docs(c_i))$ in document space, *i.e.*, $Vol(docs(c_i)) / Vol(Docs(c_i))$, and we denote it $coverage(c_i)$. To put this definition to the whole training corpus, we have

$$\mathrm{cov}\,erage(TC) = \sum_{i=1}^{n}(Vol(docs(c_i))) \Big/ \sum_{i=1}^{n}(Vol(Docs(c_i))).$$

Generally speaking, this coverage definition is purely qualitative. First, although *Vol*(*Docs*(*c*)) is deterministic in document space, we have no way to definitely know how much it is; second, it is difficult for us to calculate *Vol*(*docs*(c)) even *docs*(c) is given. However, in next section we will give an algorithm for detecting the changing of coverage of a given class after the number of training documents increases or decreases. Fig. 2 illustrates an imaginary training corpus with two classes in 2-D space. The area enclosed by solid line is the whole space of a class, and the shadowed area represents the training instances. *Coverage*(c_1) and *coverage*(c_2) approximate to 0.25 and 0.33 respectively.

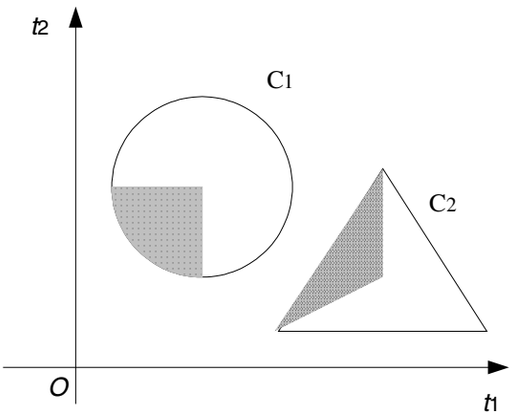


Fig. 2. A training corpus with two classes in 2-D space

Generally, given a training corpus, the larger its coverage is, the richer its semantic content is, and the more discriminative the training corpus is, as far as classification is concerned. However, coverage is not necessary directly related to the number of documents in the training corpus. For example, suppose we have two versions of a training corpus, this first one contains 100 documents, and the second has 200 documents, only the additional 100 documents in the second version are a copy of the first version. In such a case, the first version and the second version have similar coverage even though the number of documents in the second version doubles that in the first version.

For a training corpus, *coverage* reflects the relative covering scope of training documents in document space, it does not shed any light on how the training documents are distributed in document space. For that, we propose another property of training corpuses, *i.e.*, *density*. Intuitively, for documents in a class, if the average distance (or similarity) of documents to their *k* nearest neighbors (documents) in that class is small (or large), then the class is dense; otherwise, it is sparse. The following definition of density is based on averaged similarity.

Definition 3. For class c_i (i=1~n) of training corpus *TC* with class space *CS*={c_1, c_2, …, c_n}, its *density* is represented by a 2-ary tuple $(\vec{s}, \sigma_s)$, and we denote it *density*(c_i)= $(\vec{s}(c_i), \sigma_s(c_i))$. Here,

$$\vec{s}(c_i) = \frac{1}{|c_i|} \sum_{d_j \in c_i \text{ and } j=1}^{|c_i|} \overline{sim(d_j, kNN(d_j))};$$

$$\sigma_s(c_i) = \sqrt{\frac{1}{|c_i|-1} \sum_{d_j \in c_i \text{ and } j=1}^{|c_i|} (\overline{sim(d_j, kNN(d_j))} - \vec{s}(c_i))^2}.$$

Above, $kNN(d_j)$ indicates the k nearest neighbors of d_j in class c_i; and $|c_i|$ is the size of class c_i; $\overline{sim(d_j, kNN(d_j))}$ is the average similarity of d_j to its k nearest neighbors. For the whole training corpus, we have $density(TC) = (\vec{s}(TC), \sigma_s(TC))$, and

$$\vec{s}(TC) = \left(\sum_{i=1}^{n} \sum_{d_{ij} \in c_i \text{ and } j=1}^{|c_i|} \overline{sim(d_{ij}, kNN(d_{ij}))}\right) \Big/ \left(\sum_{i=1}^{n} |c_i|\right);$$

$$\sigma_s(TC) = \sqrt{\sum_{i=1}^{n} \sum_{d_{ij} \in c_i \text{ and } j=1}^{|c_i|} (\overline{sim(d_{ij}, kNN(d_{ij}))} - \vec{s}(TC))^2} \Big/ \left(\sum_{i=1}^{n} |c_i| - 1\right).$$

Obviously, *density* defined in definition 3 measures not only the averaged distribution of training documents in document space, but also the skewness of the distribution. The denser the training corpus is, the closer the k-nearest documents in the same class are to each other averagely. As two documents draw closer and closer in document space, they are more and more semantically similar. When the similarity between two documents reaches a predefined threshold, we may regard them as semantically similar documents, and they are redundant to each another.

Definition 4. Given a similarity threshold S_r ($0 < S_r \leq 1$), two documents d_i and d_j are redundant document to each other if $sim(d_i, d_j) \geq S_r$.

We give an algorithm to calculate the set of redundant documents as follows.

Algorithm 1. Calculate the set of redundant documents in a training corpus *TC*
Input: class space $\{c_1, c_2, \ldots, c_n\}$ and training documents *docs*(*TC*).
Output: redundant documents set *red_docs*;
Process:
1) *red_docs* =Φ;
2) **FOR** each class c_i **DO**
3) S=Φ; take the first document d_1 from *docs*(c_i) and let $S = S \cup \{d_1\}$;
4) **FOR** each document d_i (d_i is not included in S and *red_docs*) in *docs*(c_i) **DO**
5) **IF** there exists a document d_j in S, d_i and d_j are redundant to each other, **THEN**
6) *red_docs* = *red_docs* ∪{ d_i};
7) **ELSE** $S = S \cup \{ d_i\}$;
8) **RETURN** *red_docs*.

As far as classification is concerned, we would not like the size of training corpus to be too large, because large training corpus consumes more storage space and computation resources while carrying out classifier training and evaluating. It occurs to us

that a training corpus should keep useful documents and discard those of redundancy as many as possible. To expose this issue more clearly, let us split hierarchically a concept class c top down to the smallest distinguishable semantic granularity, then we will get a concept hierarchy with class c_i as its root node, and a set of leaf classes $\{c_{11}, c_{12}, \ldots, c_{1m}\}$. Training documents of class c_i should fall into the set of leaf classes of c_i: $\{c_{11}, c_{12}, \ldots, c_{1m}\}$. We do not expect that each leaf class will contain training documents because the number of training documents is limited. However, we do hope that each leaf class contains at most one training document; otherwise, there are redundant documents in the training corpus of class c_i. To avoid redundancy in training corpus, each training document should be representative of a distinct semantic unit of class c_i. With this in mind, we introduce the third property of training corpus: *representative*.

Definition 5. For class c_i (i=1~n) of training corpus TC with class space CS=$\{c_1, c_2, \ldots, c_n\}$, its *representative* indicates how many of its training documents are semantically distinctive in the semantic space of that class, we denote it *representative*(c) , which is evaluated with the following formula:

$$representative(c_i) = \frac{|c_i| - red\_docs(c_i)}{|c_i|} = 1 - red\_docs(c_i)/|c_i|.$$

Here, $red\_docs(c_i)$ indicates the numbers of redundant documents in class c_i. Similarly, advancing this definition to the whole training corpus, we obtain

$$representative(TC) = 1 - \sum_{i=1}^{n} red\_docs(c_i) \Big/ \sum_{i=1}^{n} |c_i|.$$

Suppose there is a training corpus contains 100 documents, in which 10 documents are redundant documents, so its representative is 0.9, which means that in the training corpus only about 90% documents are semantically distinctive.

Up to now, we have proposed three properties of training corpuses for evaluating their quality. *Coverage* reflects the scope that training corpus covers document pace of these classes in question; *density* measures distribution of training documents in document space; and *representative* indicates how many of the training documents are semantically distinctive in the training corpus. Given a training corpus and its coverage, the larger the number of documents in the corpus is, the smaller its representative may be, and most possibly, the higher its density is. A good training corpus should have large coverage, high representative, and even density.

3 An Algorithm for Training Corpus Construction

As mentioned above, a good training corpus should have large coverage, high representative, and even density. Then, how to construct such a corpus? That is the aim of this section. Intrinsically, corpus construction is an incremental process. Our approach is to control the quality of training corpus during the construction process so that coverage keeps growing, representative and density maintain reasonable as new training documents are added into the corpus under construction. The key problem here is to detect whether the coverage of the training corpus under construction is

increasing. Obviously, calculating coverage according to definition 2 is impractical. Here we adopt an indirect way. Given class c and its training documents set $docs(c)$, when a batch of new documents δ-$docs(c)$ are added into c, two cases may arise:

1) Coverage has no change, which means that the newly added documents either are redundant documents of the existing documents or just fill in the sparse space of the corpus in document space, subsequently causing higher density and possibly worse representative;
2) Coverage increases, which indicates that the boundary of the hyper-cone corresponding to the corpus in document space has been extended. So we can detect coverage changes by detecting the changes of the boundary of the corpus in document space.

The boundary of the training corpus of a specified class can be represented roughly by a number of representative documents near the boundary area. To detect the changes in training corpus' boundary, we need only to detect changes in the set of representative boundary documents. Following is an algorithm to implement this idea.

Algorithm 2 is used to select a number of representative boundary documents from a given set of training documents of a certain class. In this algorithm, three functions are called. ***CenterOf***(c) is used to calculate the center vector of class c; ***FurthestFrom***(d_o) is responsible for finding the document in c that is furthest from d_o; and ***FurthestFromBoth*** (d_1, d_2) is dedicated to select the document from c that is furthest from d_1 and d_2.

Algorithm 2. Select boundary documents
Input: the number of boundary documents to be selected: *Bd_Num*;
the set of training documents of a given class c: $docs(c)$;
Output: the set of selected boundary documents: B;
Process:
1) d_o =***CenterOf***(c); d_1 =***FurthestFrom*** (d_o); d_2 =***FurthestFrom*** (d_1);
2) d_3 =***FurthestFromBoth*** (d_1, d_2);
3) B=$\{d_1, d_2, d_3\}$;
4) BD=$\{d_1, d_2\}$; ND=$\{d_3\}$; N=Φ; Finish=False;
5) **DO WHILE**(Finish=False) {
6) **FOR** each document d_i in ND **DO** {
7) **FOR** each document d_j in BD **DO** {
8) d_k =***FurthestFromBoth*** (d_i, d_j);
9) **IF** d_k is not in B **THEN** {
10) B= $B \cup \{ d_k \}$;
11) **IF** $| B | \geq Bd\_Num$ **THEN** {
12) Finish=TRUE; break;}
13) **ELSE** N= $N \cup \{d_k\}$;}
14) }
15) **IF** Finish=TRUE **THEN** break;}
16) BD=$BD \cup ND$; ND=N; N=Φ;}
17) **RETURN** B.

Now we give an approach for constructing training corpus semi-automatically. We adopt an incremental and class-by-class strategy to construct training corpuses. We term it as semi-automatic approach because a raw training corpus must be provided in

advance for constructing the target training-corpus, and the size of the raw corpus should be far larger than that of the target corpus. The reason why our approach can obtain high-quality training corpus is that at each step when new documents are added to the corpus, our approach tries to keep documents contributing to improve coverage, meanwhile discard redundant documents. The major steps of our approach are as follows (only one class is considered).

1) Select a certain number of training documents (without redundant documents) from the *raw training corpus* (*RTC*) as the initial training documents (or seed documents) of the *target training corpus* (*TTC*);
2) Calculate the center vector of the current version of *TTC*;
3) When a batch of new documents are added to the existing *TTC*, take the old center vector as the center of the augmented corpus, select the set of boundary documents from the new corpus by using algorithm 2. At this time, the number of boundary documents selected increases by a number that is in proportion to the number of the added new documents. Note that at this step the new documents are not really included in the *TTC*, even though they join the selection of boundary documents.
4) Among the newly added documents, some are boundary documents of the augmented corpus, we formally adopt these documents to *TTC*; some are redundant documents of the updated *TTC*, we give up these documents; for the rest documents, all are included into *TTC*.
5) Check whether or not *TTC* reaches the desired size or documents of the considered class in *RTC* are run out of. If not, go to 2); otherwise, stop.

Algorithm 3 is an implementation of the steps above. It is used for constructing training corpus for a given class. To construct training corpus of multiple classes, we need only to utilize algorithm 3 repeatedly over the classes in question.

Algorithm 3. Semi-automatically constructing training corpus of a given class
Input: text class: c;
raw training corpus: *RTC*;
the initial size of the *target training corpus* (*TTC*): m;
the number of documents added to training corpus at each time: k;
maximum size of *TTC*: *max_size*;
ratio of the number of boundary documents over the total number of documents in a corpus: r;
Output: target training corpus: *TTC*;
Process:
1) let *TTC* =the set of m training documents (without redundant documents) of class c randomly selected from *RTC*;
2) S=*TTC*;
3) calculate the center vector d_o of S; $N=\Phi$; $NS=\Phi$; $NB=\Phi$;
4) take k new training documents of class c from *RTC*, and put them into N; let $NS=S\cup N$; *RTC*=*RTC*-N;
5) let d_o be the center of NS, select $|NS|*r$ boundary documents from NS , and put them to NB;
6) for each document d in N do
7) if $d\in NB$ then $N=N-\{d\}$;

8) *NS*=*NS*-*N*;
9) for each document *d* in *N* do
10) if *d* is not redundant document of *NS* then *NS*=*NS*+{*d*};
11) *S*=*NS*;
12) if |*S*|<*max_size* and |*RTC*(*c*)|≠0 then goto 3);
13) *TTC*= *S*;
14) return *TTC*.

4 Preliminary Experiments

We use a Chinese text corpus for experiments because we have been doing research on Chinese text classification [12,13]. Statistic information about the text repository *TR1* is listed in Table 1. *TR1* contains totally 1493 documents that belong to 5 distinctive classes. It consists of news documents from the *People's Daily* and the *Xinhua News*. Among the documents in *TR1*, about 80% (selected randomly) are used for training classifiers; and the rest 20% used for test. These two sets are termed *TR1-80* and *TR1-20*. From *TR1-80*, we constructed a training corpus using the proposed approach in this paper, which is termed *TR1-80-constructed*. *TR1-80*, *TR1-20* and *TR1-80-constructed* are listed in the third, fourth and fifth column of Table 1 respectively.

Table 1. Text repository *TR1* and its derived datasets statistics

Classes	Documents number			
	TR1	*TR1-80* (80% of TR1 for training)	*TR1-20* (20% of TR1 for test)	*TR1-80-constructed* (Constructed from TR1-80 by using our approach)
Politics	617	494	123	289
Sports	350	280	70	134
Economy	226	181	45	121
Military	150	120	30	86
Arts	150	120	30	90
Total number	1493	1195	298	720

Three classifiers are evaluated: classifier-1 and classifier-2 are trained with *TR1-80* and *TR1-80-constructed* respectively, while classifier-3 is trained with a dynamically compiled training corpus *TR1-80-random*, which is constructed by randomly selecting training documents from *TR1-80* and requiring each class contains the same number of training documents as that in the corresponding class of *TR1-80-constructed*. In other words, *TR1-80-random* and *TR1-80-constructed* have similar size both for each single class and as a whole, only they are constructed by different approaches.

*k*NN method is used for text classification with k=15, 500 document features are used for each classifier, and information gain method is used for feature selection. The aim of experiments is to compare the three classifiers' performance. We expect that classifier-2 outperforms classifier-3 and is in the same level of performance as classifier-1. Experimental results validate our expectation.

Table 2 provides the experimental results. Two commonly used performance parameters, *precision* and *recall* are used for classification performance evaluation. Note that *TR1-80-random* is constructed by randomly selecting documents from *TR1-80*, we do 10 trials and average performance measures over the 10 trials.

Table 2. Experimetal results

Classifier	Performance comparison	
	Precision	*Recall*
Classifier-1	92%	90%
Classifier-2	90%	89%
Classifier-3	84%	85%

From Table 2, we can see that classifer-2 outperforms classifier-3 even though their training corpuses have similar size. The difference stems from the fact that their training corpuses are different in quality. On the other hand, classier-2 is close to classifier-1 in performance although the former uses a training corpus (*TR1-80-constructed*) that contains only about 60% documents of the latter's training corpus (*TR1-80*). The results show that our approach for training corpus constructing can produce high-quality and condensed training corpuses with which high performance classifiers can be trained. That's really a good result for large training corpuses that will cause high computation and storage cost in both the training phase and the test phase.

5 Conclusion and Future Work

Although research in the pass years had shown that training corpus could impact classification performance, little work was done to explore the underlying causes. This paper tries to propose an approach to build semi-automatically high-quality training corpuses for better classification performance by first exploring the properties of training corpuses, and then giving an algorithm for constructing training corpuses semi-automatically. Experimental results validated our approach: classifiers based on corpus constructed by our approach can achieve excellent performance while the training corpus is significantly condensed, which can lead to lightweight and high efficient classifier systems. Nevertheless, we also realize that this paper is only a starting point of our research in this direction. Some open problems still exist. In the future, we will extend this research in two directions: on one hand, carrying out more experiments to validate our approach, especially experiments over English text benchmarks, such as Reuters datasets [1]; on the other hand, studying further properties of training corpses, especially, the properties between training sets of different classes; and exploring more efficient algorithms for training corpus building.

References

1. Yang Y. and Liu X. A re-examination of text categorization methods. Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99), 1999.
2. B. Masand, G. Linoff, and D. Waltz. Classifying news stories using memory-based reasoning. Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'92), 1992, 59-65.
3. K. Lang. Newsweeder: learning to filter netnews. Proceedings of the twelfth International Conference on Machine Learning (ICML'95), 1995.

4. T. Joachims, D. Freitag, and T. Mitchell. Webwatcher: A tour guide for the World Wide Web. Proceedings of 1997 International Joint Conference on Artificial Intelligence (IJCAI'97), 1997.
5. C. Apte, F. Damerau, and S. Weiss. Text mining with decision rules and decision trees. Proceedings of the Conference on Automated Learning and Discovery, Workshop 6: Learning from Text and the Web, 1998.
6. Y. Yang and C. G. Chute. An example-based mapping method for text categorization and retrieval. ACM Transaction on Information Systems (TOIS), 12(3): 252-277, 1994
7. McCallum and K. Nigam. A comparison of event models for navie bayes text classification. Proceedings of AAAI-98 Workshop on Learning for Text Categorization, 1998.
8. W. W. Cohen. Text categorization and relational learning. Proceedings of the Twelfth International Conference on Machine Learning (ICML'95), Morgan Kaufmann, 1995.
9. E. Wiener, J. O. Pedersen, and A. S. Weigend. A neural network approach to topic spotting. Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval (SDAIR'95), 1995.
10. T. Joachims. Text categorization with support vector machines: learning with many relevant features. Proceedings of 10th European Conference on Machine Learning (ECML'98), 1998, 137-142.
11. R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence- rated predictions. Proceedings of 11th Annual Conference on Computational Learning Theory, 1998, 80-91.
12. S. Zhou, J. Guan. Chinese documents classification based on N-grams. A. Gelbukh (Ed.): Intelligent Text Processing and Computational Linguistics, LNCS, Vol. 2276, Spring-Verlag, 2002, 405-414.
13. S. Zhou, Y. Fan, J. Hu, F. Yu and Y. Hu. Hierarchically Classifying Chinese Web Documents Without Dictionary Support And Segmentation Procedure. H. Lu, and A. Zhou (Eds.), Web-Age Information Management. LNCS, vol.1846, Springer-Verlag, 2000, 215-226.
14. Apte, F. Damerau, and S. Weiss. Towards language independent automated learning of text categorization models. Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94), 1994.
15. D. Lewis and M. Ringuette. Comparison of two learning algorithms for text categorization. Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval (SDAIR'94), 1994.
16. Yang Y. and Pederson J. Feature selection in statistical learning of text categorization. Proceedings of the 14th International Conference on Machine Learning (ICML'97), Morgan Kaufmann, 1997, 412-420.

Replicating Quantified Noun Phrases in Database Semantics

Roland Hausser

Universität Erlangen-Nürnberg
Abteilung Computerlinguistik (CLUE)
rrh@linguistik.uni-erlangen.de

Abstract. Predicate calculus treats determiner-noun sequences like the man, every man, or several men as 'quantified noun phrases.' This analysis in terms of quantifiers, variables, and connectives creates a major structural difference compared to the handling of proper names. The modeling of natural language communication in database semantic (DBS), in contrast, treats the functor-argument structure as primary, regardless of whether an argument is of the sign type symbol (determiner-noun sequence), name, or indexical (pronoun). The meanings carried by different determiners are reanalyzed as controlling the matching between nominal symbols and individuals, or sets of individuals, at the level of context.

1 Introduction

Before turning to quantified noun phrases in predicate calculus, let us review the reconstruction of propositional calculus in DBS. It is based on treating propositional constants as feature structures with the following attributes:

1.1 FEATURE STRUCTURE OF A PROPOSITIONAL CONSTANT

attribute 1: name
attribute 2: proposition number
attribute 3: name of next
attribute 4: connective to next
attribute 5: name of previous
attribute 6: connective to previous

In this way, the relations of a propositional constant to previous and next in the context of a formula are integrated into the constant's feature structure. This is in contradistinction to the graphical characterization of relations in logical notation and has the advantage that the constants of a formula may be stored in accordance with the indexing and retrieval principles of a database, e.g., alphabetically.

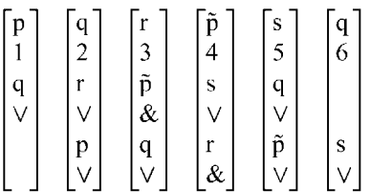
The feature structures are built automatically by parsing formulas of propositional calculus in conjunctive normal form (CNF),[1] using a semantically interpreted LA-gram-

[1] Cf. Hopcroft & Ullman 1979, p. 325. CNF consists of constants with or without negation, connected by '&' (and) and '∨' (or), . The negation of, for example, **p** is written as $\tilde{\mathbf{p}}$, called *external negation*, and paraphrased as *It is not the case that* **p**. CNF formulas are written without parentheses whereby, for example, **p** ∨ **q** ∨ **r** & **s** ∨ **t** ∨ **u** is treated as equivalent to **(p** ∨ **q** ∨ **r) & (s** ∨ **t** ∨ **u)**.

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 109–122, 2002.

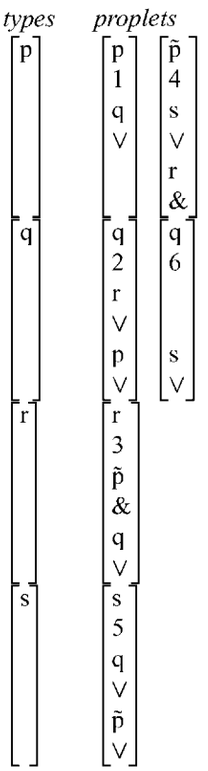
mar called *LA-pci*.[2] For example, the formula **p** ∨ **q** ∨ **r** & **p̃** ∨ **s** ∨ **q** translates equivalently into the following set of feature structures called *proplets*:

1.2 REPRESENTATION OF '**p** ∨ **q** ∨ **r** & **p̃** ∨ **s** ∨ **q**' IN DBS



During storage in a word bank (cf. Hausser 2001), the completed proplets are added at the end of their respective token lines (ordering of a previously unordered set):

1.3 STORING '**p** ∨ **q** ∨ **r** & **p̃** ∨ **s** ∨ **q**' IN A WORD BANK



The purpose of storing proplets in this manner is (i) easy retrieval based on (ii) easy storage plus (iii) easy navigation.

[2] *LA-pci* stands for LA-propositional_calculus_interpretation and is defined in Hausser 2002b.

The truth conditions are coded into the navigation as consistency checking. It is is driven by an LA-grammar called *LA-pccn*.[3] This raises the following questions:

First, how can consistency checking avoid the SAT problem, known to be computationally intractable? By treating propositions as database assertions: unnegated constants, e.g., **p**, are always assigned the value true (1), while negated constants, e.g., $\tilde{\mathbf{q}}$, are regarded as value false (0). This assignment principle allows to express all possible truth-conditional constellations of propositional calculus, yet avoids the need to remember which value was assigned to earlier occurrences of, e.g., **p** in a formula. Consequently, consistency checking is of linear complexity.

Second, how should transitions be handled in which the truth value is undetermined (#) because the remainder of the formula has not yet been read? Consider for example the formula $\mathbf{p} \vee \mathbf{q} \vee \mathbf{r} \,\&\, \tilde{\mathbf{p}} \vee \tilde{\mathbf{q}} \vee \mathbf{r}$. The navigation traverses the formula from left to right. The initial propositions $\mathbf{p} \vee \mathbf{q} \vee \mathbf{r}$ are all 1. Then the navigation reaches $\&\, \tilde{\mathbf{p}}$. At this point, no bivalent truth value can be assigned, for which reason the navigation goes into the state # (undetermined) and continues until the following propositions decide whether the navigation returns from state # to state 1, or terminates in state 0.

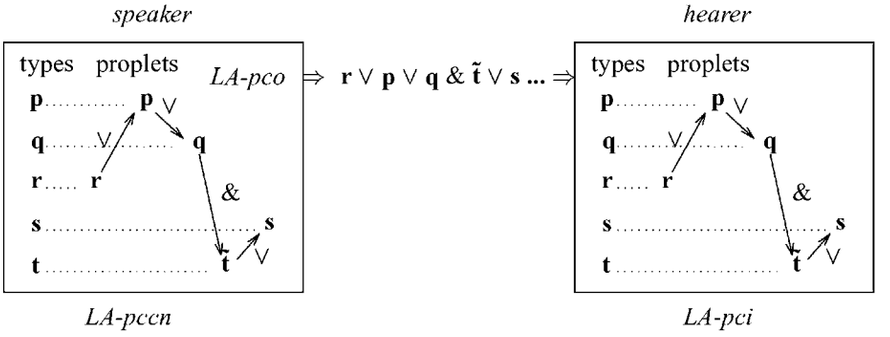
Third, what is the purpose of admitting false propositions, e.g., $\tilde{\mathbf{p}}$, in a database? Here, we rely on the well-known distinction between external and internal negation. Our reconstruction of propositional calculus uses only external negation, whereby **p** stands for a proposition which may or may not be internally negated.

As database assertions, propositions – with or without internal negation – are assumed to be true. However, while some are known to be true by the cognitive agent containing the database, others are less certain. During consistency navigation, an uncertain proposition **p** is temporarily set to $\tilde{\mathbf{p}}$ in order see what the consequences of **p**'s turning out false with respect to external reality would be on the consistency of the database.

Assume, for example, that A forgot a book in B's house. A reasons: either B is at home or the key is under the mat. By externally negating the first, but not the second proposition, a trip back to B's house would be reasonable for A.

This reasoning may be coded into CNF formulas which are used as a simple language for communication. Consider the following schematic summary of our reconstruction:

1.4 PROPOSITIONAL CALCULUS AS A COMMUNICATION LANGUAGE



[3] *LA-pccn*, for LA-propositional_calculus_consistency_navigation, is defined in Hausser 2002b.

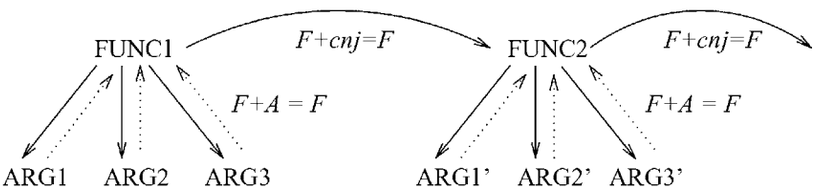
During production, the navigation powered by *LA-pccn* serves as the speaker's conceptualization. The conceptualization is mapped into language by (i) copying the proplets traversed into a sequence of proplets and (ii) mapping the sequence into an equivalent CNF formula. Procedure (ii) is based on the *LA-pco*.[4]

The hearer parses the incoming surfaces using *LA-pci*. The semantic interpretation of *LA-pci* results in a set of proplets which are stored in the hearer's database. The interaction of *LA-pccn*, *LA-pco*, and *LA-pci* ensures that the content mapped by the speaker into CNF formulas is reconstructed by the hearer into an equivalent content, resulting in successful communication.

2 From Propositional Constants to Building Propositions

The first step of upscaling from propositional to predicate calculus consists in (i) disassembling propositional constants into complex structures consisting of a functor and a characteristic number of arguments and (ii) replacing the concatenation between propositional constants by one between the functors. This more differentiated data structure supports the following kind of navigation, illustrated here with three place functors:[5]

2.1 EXTRAPROPOSITIONAL cnj-NAVIGATION



Like the propositional constants illustrated in 1.2, the functors and arguments are defined as feature structures (proplets). Furthermore, proplets for functors like *sleep* (one-place), *see* (two-place), or *give* (three-place), and for arguments like *John* (name) or *man* (noun) are stored alphabetically at the end of their respective token line, similar to 1.3. For example, if **p** equals *John loves Mary* and **q** equals *Mary loves John*, then representations of **p & q** in logical notation and DBS format, respectively, would be as follows:

2.2 REPRESENTATIONS OF *John loves Mary and Mary loves John*

old logic:[6] **love (John, Mary) & love (Mary, John)**

[4] *LA-pco* stands for LA-propositional_calculus_output and is defined in Hausser 2002b.

[5] This navigation is powered by *LA-MOTOR*, defined in Hausser 2001.

[6] Our use of the term 'old logic' is in concord with I. Bochenski 1961, p. 12, C.

DBS:	*types*	*proplets*	
	arg: John mspr: name func: id: prn:	arg1: John mspr: name func: love id: 1 prn: 1	arg2: John mspr: name func: love id: 1 prn: 2
	func: love mspr: 2pl arg1: arg2: ctp: ctn: prn:	func: love mspr: 2pl arg1: John arg2: Mary ctp: ctn: 1 & 2 prn: 1	func: love mspr: 2pl arg1: Mary arg2: John ctp: 1 & 2 ctn: 2 & 3 prn: 2
	arg: Mary mspr: name func: id: prn:	arg2: Mary mspr: name func: love id: 2 prn: 1	arg1: Mary mspr: name func: love id: 2 prn: 2

In this example, logical notation and DBS format have in common that both are based on the principle of functor-argument structure. However, while the notation in old logic is dependent on graphical means (linear order), the equivalent recoding in DBS is not.

The DBS data structure resembles that of propositional calculus (cf. 1.3) in that it consists of feature structures for types and for tokens. The feature structures of the types have only two attributes with non-NIL values. The first is called arg in nominal and func in verbal proplets, and contains the name of the proplet, here John, love, and Mary, respectively. The second attribute is called mspr (morphosyntactic properties)[7] and characterizes properties which are common to all corresponding tokens, such as the sign type[8] in arguments and modifiers, or the number of places in functors.

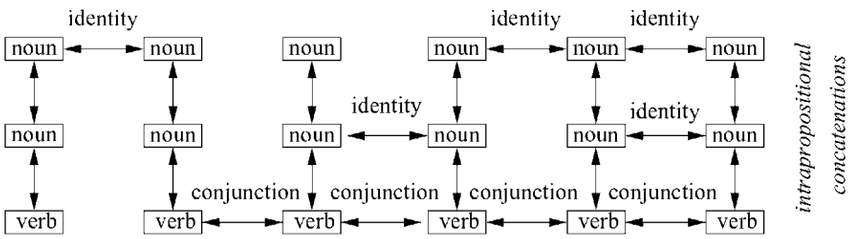
In the feature structures of the tokens (proplets), most attributes have non-NIL values. These values have been copied into the feature structure during syntactic-semantic interpretation. Proplets belonging to the same proposition have a common proposition number (prn), here 1 and 2, respectively. Arguments specify their functor and functors specify their arguments, and similarly for modifiers and modified (cf. 4.1). In addition, there are two kinds of extrapropositional relations: identity between arguments (nouns), represented by the attribute id, and conjunctions between functors (verbs), represented by the attributes ctn (conjunction to next) and ctp (conjunction to previous).

Compared to propositional calculus, the new intrapropositional relations between functors and arguments, and the additional extrapropositional relation of identity between arguments result in a much wider variety of possible continuations. These may be presented schematically as the following 'railroad system,' whereby only elementary nouns and two-place verbs are assumed for the sake of simplicity:

[7] As explained in Hausser 1999/2001, p. 62, propositions may be regarded as part of the world, as part of language, or abstractly in terms of logic. The attribute mspr has its terminological origin in language, but functions equivalently in the other two interpretations of propositions.

[8] The main sign types are symbols, indexicals, and names. See Hausser 1999/2001, Chapter 6.

2.3 SCHEMATIC RAILROAD SYSTEM OF PREDICATE CALCULUS



extrapropositional concatenations

Extrapropositional relations are shown as horizontal double arrows, while intrapropositional relations are shown as vertical double arrows. Each proposition may have up to three extrapropositional relations, one based on verbal conjunction and two based on nominal identity (and accordingly for one- and three-place verbs).

The functors (nouns) and arguments (verbs) forming propositions are distributed all over the data structure. In line with the coding technique of DBS, all intra- and extrapropositional concatenations are established solely in terms of attributes, i.e., proplet name, proposition number, identity, and conjunction.

3 Quantified Noun Phrases versus Proper Names

Old logic uses different syntactic structures depending on whether an argument is a quantified noun phrase or a proper name. This is illustrated below with the comparison of 3.1 and 3.2:

3.1 ALTERNATIVE ANALYSES OF John walks

old logic: **walk(John)**

DBS:

types	*proplets*
arg: John mspr: name func: id: prn:	arg: John mspr: name func: walk id: prn: 23
func: walk mspr: 1pl arg1: ctp: ctn: prn:	func: walk mspr: 1pl arg1: John ctp: ctn: prn: 23

3.2 ALTERNATIVE ANALYSES OF Every man walks

old logic: $\forall x\,[\mathbf{man}(x) \rightarrow \mathbf{walk}(x)]$

DBS:

types	*proplets*
arg: man mspr: noun func: id: prn:	arg: man mspr: noun, every func: walk id: prn: 23
func: walk mspr: 1pl arg1: ctp: ctn: prn:	func: walk mspr: 1pl arg1: man ctp: ctn: prn: 23

In the case of quantified noun phrases, here every man, old logic treats the principle of functor-argument structure as subservient to the structure introduced by the quantifier. The use of two sub-propositions and a connective creates an artifical need to express identity, handled in terms of variables bound by the quantifier. Also, this analysis violates the principle of surface compositionality.

DBS, in contrast, treats the principle of functor-argument structure as primary, regardless of whether an argument is a name or a quantified noun phrase. Also, old logic's syntactic distinction between universal and existential quantification, i.e., $\forall$**x [f(x)** $\rightarrow$ **g(x)]** versus $\exists$**x [f(x) & g(x)]**, reduces in DBS to different values in the mspr attribute of the noun phrase in question, e.g., all or every vs. a(n) or some.

Furthermore, DBS handles coreference uniformly by means of the id attributes. This method is suitable for all kinds of arguments, i.e., names, quantified noun phrases, and pronouns, both within propositions and between propositions. Old logic, in constrast, employs three different methods for expressing coreference: (i) equality of names, (ii) variables bound by a quantifier, and (iii) the connective =.

4 Comparing the Ontologies of Old Logic and DBS

A famous attempt to extend the logical quantifiers to natural language is Russell's (1905) analysis of the in combination with a singular noun. Consider the following example, which is presented with alternative analyses in old logic and DBS:

4.1 Alternative analyses of The present King of France is bald

old logic: $\exists$**x** $\forall$**y [[present_King_of_France(y)** $\leftrightarrow$ **y=x] & bald(x)]**[9]

DBS:

types	*proplets*
func: bald mspr: 1pl arg1: ctp: ctn: prn:	func: bald mspr: 1pl arg1: King_of_France ctp: ctn: prn: 23

arg: King_of_France	arg: noun
mspr: noun	mspr: noun, sg, the
func:	func: bald
modr:	modr: present
id:	id:
prn:	prn: 23
modr: bald	modr: bald
mspr: pos	mspr: pos
modd:	modd: King_of_France
id:	id:
prn:	prn: 23

The complicated structure of old logic's formula has no syntactic counterpart in the linguistic surface. In other words, the difference between the semantic representation of a singular noun combined with the determiner the and that of a corresponding name, e.g., **bald(Charles)**, is even greater than the difference between 3.1 and 3.2. Needless to say, such a major structural difference between the semantic representations of expressions which are syntactically equivalent except for the *kind* of argument used is unjustifiable from the linguistic viewpoint of a surface compositional analysis.

The quantified noun phrases of old logic are also known for frequently (i) creating artificial scope ambiguities and (ii) failing to express representations intuitively required. As an example of the first kind consider

Three boys kissed four girls.

Presupposing a representation of numbers using quantifiers (which is not pretty on any account), this sentence has been claimed to be ambiguous. On one reading there is a large group of boys and a large group of girls, and of all the boys three managed to kiss a total of four girls each. On the other reading a group of three boys is simultaneously engaged in kissing a group of four girls. The number of readings may be multiplied by distinguishing for each pair of boys whether they kissed the same girl or not, depending on the order of the quantifiers in the semantic representation,

The second kind of problem is illustrated by the sentences

Every man who loves a woman is happy.

Every man who loves a woman loses her.

and the following attempt to represent their dominant reading in predicate calculus:

$\forall x$ [[man(x) & $\exists y$ [woman(y) & love (x, y)]] $\rightarrow$ [happy (x)]]

$\forall x$ [[man(x) & $\exists y$ [woman(y) & love (x, y)]] $\rightarrow$ [lose (x, y)]]

The parallel formulas are intended to reproduce the subordinate structure of relative clauses. This does not work for the second formula, however, because the scope of the quantifier $\exists y$ does not reach the y in lose(x,y). An alternative formula with the quantifier $\exists y$ in fronted position (as in prenex normal form) would solve the scope problem, but cannot be derived compositionally by translating the words of the sentence into suitable lambda expressions, as required by Montague grammar.

[9] See also Montague 1974, p. 261, T2.

In the DBS analysis, these problems do not arise. One reason is that the functor-argument structure is treated as primary in DBS, such that the differences between arguments, i.e., between different kinds of quantified noun phrases and proper names, are minimized. The other reason is that old logic and DBS are based on different ontological assumptions.

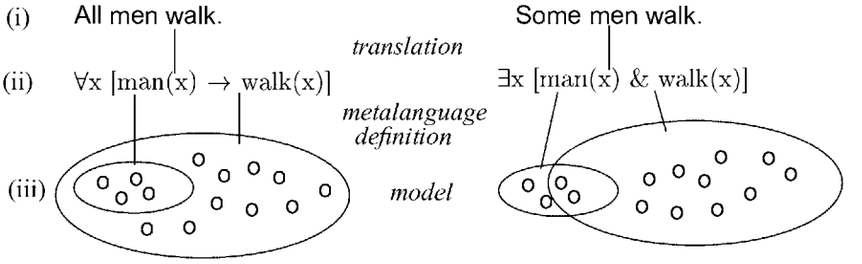
Old logic defines meaning in terms of truth conditions which hold between formulas and set-theoretic models. The definitions are formulated in a metalanguage the words of which are restricted to clearly defined set-theoretic concepts. The truth conditions for the universal quantifier, for example, are defined as follows:

> If 'ϕ' is a sentence, then '$\forall x \phi^{M,g}$' is true relative to a model M and a variable assignment g iff it holds for all alternative variable assignments g' that '$\phi^{M,g'}$' is true relative to M.

Thus, if ϕ equals **[man(x)→walk(x)]**, then $\forall x \phi^{M,g}$ equals $\forall$x**[man(x)→walk(x)]**^{M,g}. The latter is true relative to a model M, iff **[man(x)→walk(x)]** is true for all possible values of x in M.

The truth conditions for the existential quantifier are similar. The only difference is that truth relative to a model must hold for *at least one* variable assignment g, rather than *for all*. This metalanguage-based approach to characterizing the existential and the universal quantifier may be illustrated schematically as follows:

4.2 ONTOLOGICAL BASIS OF OLD LOGIC

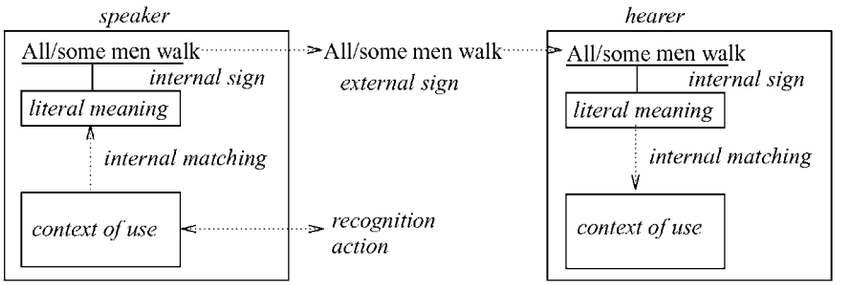


The natural language expressions at level (i) are translated into equivalent logical formulas at level (ii). The logical formulas are interpreted relative to a model (iii) in terms of their truth conditions.

On this approach, the different meanings of the universal and the existential quantifier reside ultimately in no more and no less than the different set-theoretic relations at the level of the model, depicted here in terms of Venn diagrams. Once the relation between the logical operators and their set-theoretic counterparts has been established by metalanguage definitions, additional phenomena are handled by syntactic composition, as shown by the analysis of the in 4.1.

The ontology of DBS, in contrast, may be illustrated schematically as follows:

4.3 ONTOLOGICAL BASIS OF DBS



In contrast to old logic, there is a distinction between the external world containing cognitive agents and the internal structure of the agents. Inside the agents, the levels of language and context are clearly distinguished. At the level of context, the agents interact with the external world in terms of recognition and action, and at the level of language in terms of production and interpretation.

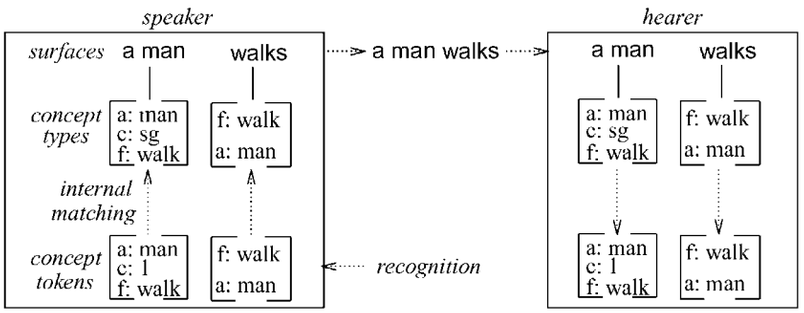
Elementary meanings are not defined in terms of a set-theoretically-based metalanguage, but as concepts defined in terms of perception and action procedures. Language is not interpreted relative to a set-theoretic model, but relative to a context of use defined as the data structure of a word bank. The levels of language and context are related to each other by the principle of internal matching, based on concept tokens at the level of context and concept types at the level of language.

5 Interpreting Different Determiners

In DBS, quantifiers are treated at the level of language as part of feature structures representing nominals (cf. 3.2, 4.1).Their difference of interpretation is focused on the *reference* of determiner-noun sequences (quantified noun phrases) relative to the context of use. Thus the truth conditions of old logic are reinterpreted as pragmatic rules for controlling the procedure of internal matching.

Consider for example a cognitive agent observing a man walking. This content is coded into a language sign (speaker mode). Another cognitive agent interprets the sign by constructing an equivalent content at the level of context (hearer mode):

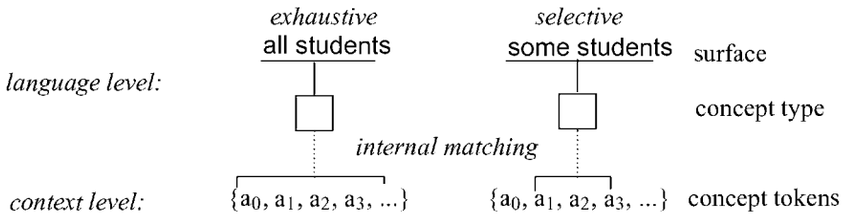
5.1 COMMUNICATING WITH A DETERMINER-NOUN SEQUENCE



The attributes a, c, and f stand for argument, cardinality, and functor, respectively. The sentence a man walks is true if the speaker's observation is correct and the coding into language has been working properly. The communication of this content is successful, if the hearer's decoding results in a data structure which is equivalent to the corresponding part of the speaker's data structure.

The matching between referents at the level of context (concept tokens) and literal meanings at the level of language (concept types) depends on the structures at both levels. At the level of context, referents may be single individuals or sets of individuals. At the level of language, sets of individuals may be referred to either exhaustively by means of all/every, or selectively by means of some/several.

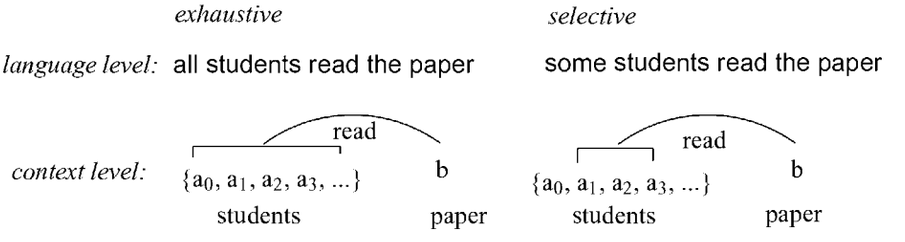
5.2 Exhaustive versus selective reference to sets



For simplicity, the levels of context and language meaning are represented as concept tokens and concept types, respectively. These concepts are assumed to be values in the feature structures as represented in 5.1. Programming the exhaustive vs. selective matching relations in question requires 'vertically' oriented concepts which are quite different from those underlying $\forall$**x f(x)** and $\exists$**x f(x)**.

The distinction between exhaustive and selective reference affects the nature of relations which hold between sets of individuals, or between a set and an individual. Consider the following schematic characterization:

5.3 Relations between a set and an individual



Another distinction at the level of context is the one between definite and indefinite referents. A definite referent is assumed to be known by the partner of discourse, as in the current president, or has been previously introduced. The distinctions between sets of different size, individuals, and known versus unknown referents are reflected in English as follows:

5.4 TYPES OF REFERENTS AND THEIR COUNTERPARTS IN ENGLISH

sets containing three or more elements:	
exhaustive	all students, every student, students
selective	some students, several students
exhaustive, definite	the students, all the students
selective, definite	some of the students, several of the students
sets containing one element:	
	one student
definite	the one student
sets containing two elements:	
	two students
definite	the two students
etc.	
individuals:	
	a student
definite	the student

The above table has been purposely restricted to reference with the sign type *symbol*. The analysis of reference with the sign types *name* and *indexical* (pronouns) is analogous, but requires a separate study (see Hausser 1999/2001, Chapter 6).

6 Replicating Scope Ambiguity

A standard intuition in old logic are scope ambiguities based on alternative orders of existential and universal quantifiers. Consider the following example:

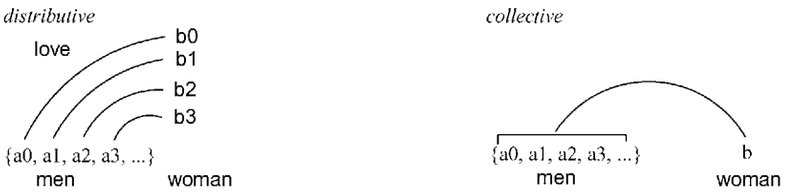
6.1 SCOPE AMBIGUITY OF Every man loves a woman

$$\forall x\, \exists y\, [\mathbf{man}(x)\ \&\ [\mathbf{woman}(y) \rightarrow \mathbf{love}(x, y)]]$$
$$\exists y\, \forall x\, [\mathbf{man}(x)\ \&\ [\mathbf{woman}(y) \rightarrow \mathbf{love}(x, y)]]$$

The standard interpretation of the upper formula is a reading according to which Jack, Bill, and Harry each have their own woman, as in a proper suburb, while the lower formula represents a reading according to which Jack, Bill, and Harry all love one and the same woman, Mary, as in a house of ill repute.

In database semantics, the source of these intuitions may be traced to *distributive* versus *collective* interpretations of relations. At the level of context, this distinction may be presented schematically as follows:

6.2 DISTRIBUTIVE VERSUS COLLECTIVE RELATION INTERPRETATION



The nature of some relations is such that they take an argument which may be singular or plural while being restricted semantically to a distributive interpretation of it. Consider the following examples:

6.3 RELATIONS RESTRICTED TO DISTRIBUTIVE INTERPRETATION

Mary closed her eyes (singular)
The girls closed their eyes (nominative, exhaustive)
Some girls closed their eyes (nominative, selective)
John dealt the cards(oblique, exhaustive)
John sliced some of the apples (oblique, selective)

Other relations require a plural argument and are restricted to a collective interpretation of it. Consider the following examples:

6.4 RELATIONS RESTRICTED TO COLLECTIVE INTERPRETATION

*John mixed the nut (singular)
The girls formed a circle (nominative, exhaustive)
Some girls formed a circle (nominative, selective)
John mixed all the ingredients (oblique, exhaustive)
John mixed some of the ingredients (oblique, selective)

Most relations, however, may interpret their plural arguments either distributively or collectively, as illustrated in 6.2. Furthermore, it seems to be a characteristic property of at least some natural languages, e.g., English, that the distinction between distributive versus collective interpretations of plural arguments has no reflection in the surface.

Consequently, the alleged scope ambiguity arising with relations which may be interpreted either distributively or collectively are handled in DBS as an underspecification concerning the matching conditions between language meanings and contextual referents. Restricted relations like those of 6.3 and 6.4 are best treated by inferences at the level of context instead of scope ambiguities (cf. 6.1).[10]

7 Conclusion

The reconstructions of propositional calculus outlined in the Introduction and of predicate calculus presented in the remainder of this paper are compatible, but different. Propositional calculus has been reconstructed in a 'horizontal' manner, based on the extrapropositional relation of conjunction between propositions (using the connectives $\vee$ and & only). Predicate calculus, in contrast, has been reconstructed here in a 'vertical' manner, concentrating on the matching relation between language meanings and their contextual counterparts.

[10] This analysis will eventually be extended to examples like the 'donkey sentence' (cf. Geach 1969, p. 155, (38), Kamp & Reyle 1993). This requires more work, however, because the examples involve relative clauses and indexicals (pronouns). Cf. Hausser 1999/2001, p. 457 and pp. 110 f., respectively

The treatment of propositional calculus was imported into DBS by coding the extrapropositional relations of conjunction into the functor (Section 2). The reconstruction of predicate calculus maintains the functor-argument structure of propositions regardless of whether an argument is a determiner-noun sequence, a name, or a pronoun. As a consequence, the analysis is strictly surface compositional (Section 3). This is made possible in large part by DBS using an ontology different from logic (Section 4).

The information contributed by a determiner controls the matching relation between a determiner-noun sequence at the language level and an individual or set of individuals at the level of context. The function of quantifiers binding variables in logic is assigned to the attributes asserting identity between arguments (Section 5). The scope ambiguity based on alternate orderings of quantifiers in logic is explained alternatively in terms of a distribute versus collective interpretation of relations (Section 6).

The LA-grammars for parsing simple sentences of natural language into feature structures stored in the word bank (interpretation in the hearer mode), for navigation through the concatenated propositions of the word bank (conceptualization), and for mapping sequences of proplets into corresponding surfaces of natural language (production in the speaker mode) are basically those defined in Hausser 2001. A formal integration of the present analysis into this fragment affects primarily internal matching, i.e. the mapping between structures containing concept types at the level language and corresponding structures containing concept tokens at the level of context. This involves programming the lexical selection of the determiners in the speaker mode, and the construction of set-theoretic concepts and relations based on determiners in the hearer mode. The rules in question are formulated as subclauses of the Fourth Principle of Pragmatics (PoP-4).[11]

References

Bochenski, I. (1961) *A History of Formal Logic*, University of Notre Dame Press.

Davidson, D. & J. Hintikka (eds.) (1969) *Words and Objections, Essays on the Work of W.V.Quine*, Reidel, Dordrecht.

Geach, P.T. (1969) "Quine's Syntactical Insights," in Davidson & Hintikka (eds.).

Hausser, R. (1999/2001) *Foundations of Computational Linguistics, Human-Computer Communication in Natural Language*, Berlin, New York: Springer-Verlag.

Hausser, R. (2001) "Database Semantics for Natural Language," *Artificial Intelligence*, Vol. 130.1:27–74, Elsevier, Dordrecht.

Hausser, R. (2002a) "A Hypothesis on the Origin of the Sign Types," in A. Gelbukh (ed.) *Computational Linguistics and Intelligent Text Processing*, Lecture Notes in Computer Science, Berlin-New York: Springer-Verlag.

Hausser, R. (2002b) "Reconstructing Propositional Calculus in Database Semantics," to appear.

Hopcroft, J.E. & Ullman, J.D. (1979) *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, Mass.

Kamp, J.A.W. & U. Reyle (1993) *From Discourse to Logic*, Kluwer, Dordrecht.

Montague, R. (1974) *Formal Philosophy*, Yale U. Press, New Haven.

Russell, B. (1905) *"On Denoting,"* Mind 14:479-493.

[11] Cf. Hausser 1999/2001, p. 105. For slightly modernized versions see Hausser 2002a.

Ontological Extraction of Content for Text Querying

Troels Andreasen[1], Per Anker Jensen[2],
Jørgen Fischer Nilsson[3], Patrizia Paggio[4],
Bolette Sandford Pedersen[4], and Hanne Erdman Thomsen[5]

[1] Computer Science, Roskilde University, Roskilde, Denmark
troels@ruc.dk
[2] Business Communication and Information Science, University of Southern Denmark
paj@sitkom.sdu.dk
[3] Informatics and Mathematical Modelling, Technical University of Denmark
jfn@it.dtu.dk
[4] Centre for Language Technology, Copenhagen, Denmark
{bolette,patrizia}@cst.dk
[5] Computational Linguistics, Copenhagen Business School, Denmark
het.id@cbs.dk

Abstract. This paper describes a method and a system OntoQuery for content-based querying of texts based on the availability of an ontology for the concepts in the text domain. A key principle in the system is the extraction of conceptual content of noun phrases into descriptors forming an integral part of the ontology.
The retrieval of text passages rests on matching descriptors from the text against descriptors from the noun phrases in the query. The match need not be exact but is mediated by the ontology, invoking in particular taxonomic reasoning with sub- and super concepts. The paper also reports on a prototype implementation of the system.

1 Introduction

This paper describes an approach to querying text sources based on extracting and evaluating semantic content given a formal ontology for the text domain. The method is developed in the OntoQuery project [2,23,5] (short for ontology-based querying) and is being tested with prototypes.

Traditional search engines depend more or less exclusively on recognition of keywords or patterns of keywords in the text material. By contrast, OntoQuery addresses retrieval of pertinent text segments based on the conceptual content of the text. Queries take the form of natural language expressions and the system is primarily intended to retrieve text segments whose semantic content matches the content of noun phrases in the query phrase.

This means first of all that lexical synonyms and morphological variants must be recognized. However, paraphrases in a broad sense including conceptual generalisations and specialisations must be recognised; this calls for partial syntactic and semantic analysis of the phrases.

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 123–136, 2002.

The semantic analysis is based on a domain-specific ontology for the target domain of the text set up prior to the text analysis. The presence of an ontology in particular permits identification and exploitation of conceptual specialisations and generalisations which are crucial to the conceptual structuring of the target domain. More generally, the ontology makes it possible to introduce the notion of conceptual distance between the conceptual content of a pair of noun phrases, thereby ranking the phrases matching the query.

Like OntoQuery, the project OntoSeek [10] addresses content-based search by incorporating an ontology into the system. In OntoSeek, the ontology is used to help users interactively construct precise and unambiguous descriptions of resource texts and formulate unambiguous queries which may subsequently be generalised or specialised. By contrast, in OntoQuery we aim at fully automatic generation of descriptors of text and queries formulated in natural language and generalisation / specialisation is used for ranking matches rather than reformulating queries.

Interesting results have been reported recently in the new domain combining Natural Language Processing and indexing for Information Retrieval. The current OntoQuery prototype especially has similarities to the term extraction principles introduced in [11] and [12]. However, our approach differs on two points: the introduction of the ontology and the key role this ontology plays in extraction and seeking.

Expansion of concepts in queries and texts by exploiting semantic similarity in wordnets has also been experimented with in projects dealing with English, applying among other sources Wordnet [32], [33], [30] as well as EuroWordNet [9].

The rest of the paper is organised as follows: Section 2 and 3 introduces the descriptors and their embedding in the ontology by means of a notion of ontological grammar. Section 4 presents the lexicon used in the system. Section 5 describes disambiguation of noun phrases, whereas section 6 considers semantic relations as part of descriptors. Finally section 7 and 8 describe the query-matching process and the prototype implementation, respectively.

2 Mapping Noun Phrases into Descriptors

The aim of the linguistic and conceptual analyses of the text is to identify the possibly complex concepts corresponding to the noun phrases occurring in the text and to ensure that noun phrases with (nearly) identical conceptual content with respect to the ontology receive the same description. This goal clearly goes beyond synonym recognition and identification of morphological variants: It calls for a more comprehensive linguistic and ontological analysis of the text material.

The choice of focussing on noun phrases is motivated by the fact that these have clear conceptual content that can be captured in an ontology-based framework. Verb phrases, on the other hand, are not taken into account since they do not fit easily into ontological structures shaped by conceptual inclusion rela-

tionships[1]. For the same reason, our analysis is further restricted to those parts of the noun phrase that contribute to its conceptual content, i.e. the head noun and the modification provided by adjectives, genitive phrases and prepositional phrases. For the moment we exclude articles, numerals, and relative clauses.

The OntoQuery approach consists of translating noun phrases into ontological descriptors formally shaped as frames (feature structures) and representing compound concepts. For instance, lack [WRT: vitaminD] represents the ontological semantics of phrases such as "lack of vitamin D", "deficiency with respect to vitamin D", "vitamin D deficiency", and possibly others. As an extra complication in Danish (and many other languages) N-N compounds are spelt as one word, raising the problem of splitting compounds into their proper lexical elements.

Nested descriptors are also supported, e.g. disorder [CBY: lack [WRT: vitaminD]] for "disorders caused by/due to lack of vitaminD".

Ontological reasoning mechanisms must recognise, say lack [WRT: vitaminD] as a conceptual specialisation of lack [WRT: vitamin] assuming that vitaminD is given as a specialisation of vitamin in the ontology.

In noun phrases the focus is on the conceptual contributions from nouns (including decomposed compounds), adjectives, prepositional phrases, and genitive constructions.

3 Situating Descriptors in Ontologies

Our ontologies are basically structured as taxonomies, that is, using the conceptual inclusion relationship (isa). This ontology (termed the "skeleton ontology") reflects the repertoire of nouns (including proper nouns and deverbal nouns) in the applied lexicon.

A salient feature of our approach is that the taxonomic ontology is enriched with descriptors attached to the nodes in the ontology [19]. Thus noun phrases are each mapped onto a node in the so called *generative ontology* whose potentially infinite structure is generated by means of a set of rules constituting an ontological grammar [21,20].

The underlying formal theoretical basis for this representation is an extended relation-algebraic logic OntoLog (akin to description logic) outlined in [18]. In this logico-algebraic framework the skeleton taxonomy becomes a distributive lattice, which is able to cope with multiple classifications.

The general form of descriptors φ are multi-entry feature structures

$$c \begin{bmatrix} a_1 : \varphi_1 \\ \ldots \\ a_m : \varphi_m \end{bmatrix}$$

where the attribute names are semantic roles.

[1] We recognise, however, that a strategy for including at least domain-relevant verbs in the descriptions may improve the search results.

In the logico-algebraic conception the attribute names are binary relations imposed on the lattice and the descriptors are composed by means of the lattice meet operation, which set-theoretically expresses intersection. This implies that the entries act as specialisations of the head concept c.

In OntoQuery, the SIMPLE semantic lexicon described next is adopted as a pragmatic basis in which static table entries replace the logico-algebraic generative ontology framework outlined above.

4 The Semantic Lexicon

The lexical backbone in the project, which contributes to the descriptors, is constituted by the Danish SIMPLE lexicon enhanced with domain-specific extensions.

The EU-project SIMPLE (Semantic Information for Multifunctional Plurilingual Lexicons) aims at providing harmonised semantic lexicons for Natural Language Processing for 12 European languages [16]. The language specific encodings in SIMPLE are performed on the basis of a unified, ontology-based semantic model - the so-called SIMPLE model - representing an extended qualia structure based partly on Pustejovsky [29], partly on experience in previous lexical projects such as Genelex, WordNet [32,33], and EuroWordNet [9]. A general design model is provided allowing for the encoding of a large amount of semantic information such as ontological typing, domain information, semantic relations, argument structure, event structure and selectional restrictions.

In SIMPLE, the possible value types of the qualia roles have been extended to encompass more fine-grained distinctions between a large set of semantic types. For each qualia role a set of optional semantic relations and features have been established and so-called templates have been elaborated for each ontological type in order to suggest which type-defining relations and features this particular type should include. For abstract concepts like time concepts, this extended qualia structure facilitates a fairly rich semantic description as seen in the lexical entry for vinter (winter) in Figure 1, where the constitutive role in particular is rich in information[2] (for more information on the Danish SIMPLE lexicon cf. [26] and [27]).

The most important part of the encoding with regard to application in the current OntoQuery system, is constituted by the ontological typing and the formal role (is_a). Thus, for the concept vinter (winter) the ontological type Time as well as its closest supertype årstid (season of the year) are the features currently exploited by the system in analysis and search.

5 Disambiguating Noun Phrases

The disambiguation of noun phrases is handled by the ontological grammar, which defines ontologically admissible combinations of concepts and semantic

[2] Please observe that in this section we focus on the Danish lexicon of concepts and therefore refer to the relevant concepts in Danish. Whenever we refer to the SIMPLE ontology, however, or when we refer to concepts in more general terms in the rest of the paper, we apply English concept names.

Semantic unit	vinter (winter)
Definition:	*den koldeste og mørkeste årstid, som kommer efter efteråret og før foråret, og hvor der kan falde sne* (the coldest and darkest season of the year where there ... ?
Corpus example :	*Den usikre politiske situation sidste vinter ødelagde alle forventninger om en stigning i salget* (The uncertain political situation last winter spoiled all the expectations of an increase in the sale)
Ontological Type:	Time
Supertype:	Abstract Entity
Domain	General
Formal quale:	is_a = årstid (season of the year)
Agentive quale:	Nil
Telic quale:	Nil
Constitutive quale:	Nil Is_a_part_of = år (year) Has_as_part_of = vintermåneder (winter months) Iterative = yes Successor_of = efterår (autumn) Punctual = underspecified
Complex:	Nil
Synonymy:	Nil

Fig. 1. Lexical entry for vinter (winter)

roles. For instance, the noun phrase *lack of vitamin D in winter* is syntactically ambiguous between a "situation of lacking vitamin D obtaining at winter time" versus an interpretation combining the temporal specification "at winter time" with vitamin D. The latter interpretation is obviously ontologically inadmissible since temporal qualifications cannot be ascribed to substances [14,21]. Hence, the appropriate descriptor generated by the ontological grammar has to be

$$\mathsf{lack} \begin{bmatrix} \text{WRT: } \mathsf{vitaminD} \\ \text{TMP: } \mathsf{winter} \end{bmatrix}$$

rather than the corresponding nested descriptor

$$\mathsf{lack} \left[\text{WRT: } \mathsf{vitaminD} \left[\text{TMP: } \mathsf{winter} \right] \right]$$

The specific formulation and implementation of the ontological grammar are still under discussion. Among the formalisms taken into considerations are Context Free Grammars, Extended Relation Algebras, E/R-diagrams, Description Logic, and typed feature structures. Currently, the project is exploring the suitability of a typed feature-structure formalism for the task of implementing a conceptual grammar. Typed feature structures, as implemented for instance in the LKB system [16], support the definition of concept ontologies as well as the expression of selectional restrictions associated with the concepts.

Fig. 2. Semantic relations

6 Semantic Roles as Semantic Relations

The semantic roles appearing as attributes in descriptors in this paper are part of a list of abstract, general-purpose roles originating in Fillmore's case roles [8]. In OntoQuery the semantic roles are conceived of as binary semantic relations, and the set of relations to be used is subject to discussion. The inventories of relations used in various approaches of lexical semantics (a. o. SIMPLE) and terminology (e.g. [22]) are taken into consideration. This work constitutes an attempt to arrive at a universal set of relations, including an account of their logical properties, for use both in ontological semantics for natural language and in ontological reasoning and search. So far, a hierarchical ordering of relations (an ontology of relations) has been proposed in [17], see figure 2.

The hierarchy needs further elaboration, for instance the TMP-relation between *lack of vitamin D* and *winter* in the above example is not covered by the proposed temporal relation types. Furthermore, not all the terms used are adequate, e.g. the term "activity" should probably be altered to "event" in order to capture the WRT-relation between lack and vitamin D as an event-patient relation rather than an activity-patient relation.

The hierarchy of relations may be used as a refinement of the conceptual distance measure to rank two concepts otherwise equally distant from a third in the ontology. A text containing the noun phrases *vitamin D produced by the liver* would get the descriptor vitaminD [AGENT-RESULT: liver], while *vitamin D in the liver* would get vitaminD [ENTITY-STATIC-LOCATION: liver].

In the ontology both of these concepts are immediate daughters of vitaminD and will thus be equally distant from another sister concept, e.g. vitaminD [SOURCE: liver] (corresponding to the noun phrase *vitamin D in liver*). In order to rank the first two descriptors with respect to their distance from the third, it is necessary to look at the distance between the relations in the relation hierarchy. Experiments will have to be carried out to evaluate the usefulness of this relational distance measure.

7 Querying

In the present approach ontology-based querying relies on comparison of a description of the query with descriptions of texts from the database. Queries and texts are mapped onto descriptors organized in structures called "descriptions". Descriptions are sets of sets of descriptors of the form:

$$D = \{D_1, \ldots, D_n\} = \{\{D_{11}, \ldots, D_{1m_1}\}, \ldots, \{D_{n1}, \ldots, D_{nm_n}\}\}$$

where the D_is are a sets of descriptors $D_{ij}, j = 1, \ldots, m_i$. Descriptions are derived at the level of sentences. Each D_i corresponds to an NP in the text described, and, as it appears, it may embed one or more descriptors. As explained above, the aim is to map an NP into a single descriptor. However, the approach is to firstly identify chunks of words representing NP's and secondly to derive descriptors from these chunks.

If a set of words maps to a point in the ontology which corresponds to a compound descriptor, this descriptor may replace the set of words in the description. If not, the set of words may still contribute to the description. Words are simply considered special cases of descriptors, so, the structure of the descriptions allows an incremental conceptual refinement of these, as explained in more detail below.

The comparison of descriptions is not merely syntactic. Rather, their resemblance is measured in terms of similarity derived from concept relations in the ontology. Similarity between the descriptors, as situated in the ontology, is thus obtained from engaging ontological reasoning capabilities.

Initially, in the processing of a query, a description is generated. Then this query description is compared, in principle, to every description of every sentence in every document appearing in the database. Finally, sentences in the documents in the database are ranked by the degree to which their description resembles the description of the query. The query answer is a ranking of the sentences that are most similar to the query.

Descriptions are not unique and may vary by level of detail and combinability. Among the possible descriptions for the phrase: "Behandling med diæt og sygdom forårsaget af D-vitaminmangel" (Dietary treatment and disorder due to lack of vitamin D) are the following increasingly accurate descriptions:

{{dietary},{treatment},{disorder},{lack},{vitaminD}}
{{dietary, treatment},{disorder},{lack, vitaminD}}
{{treatment [CHR: dietary]}, {disorder [CBY: lack [WRT: vitaminD]]}}

The principle applied for comparing descriptions is a fuzzy nested aggregation combining descriptor similarity into similarity of full description structures. Descriptor similarity is derived from concept relations in the ontology, for instance based on distances measured over the shortest paths of reasoning. This principle is described in more detail in [3].

Descriptions for each sentence in the database are compiled and stored in the database and, furthermore, all compound descriptors in database descriptions are, in principle, expanded to also cover subsuming concepts. For instance, the element

{disorder [CBY: lack [WRT: vitaminD]]} *

may be expanded to the set

{disorder, disorder [CBY: lack], disorder [CBY: lack [WRT: vitaminD]]}

thereby including subsuming concepts. Assuming a disjunctive interpretation, this means that the database object described by * is considered a good candidate answer object for a query on e.g. disorder. A specific answer to a general query is useful, while the opposite is typically not the case.

Finally, the evaluation is initiated with an expansion of the query to include similar descriptors as alternatives. Thus, if the ontology includes "vitaminD ISA vitamin", then the term "vitamin" may be replaced by e.g. the fuzzy set of terms "1/vitamin + 0.9/vitaminD", indicating that "vitamin" matches fully while "vitaminD" matches to some degree.

8 The Prototype

A prototype that implements simplified variants of the description mapping and the query principle described above has been developed. Apart from the primary data, that is, the base of texts that are to be queried, we distinguish in the prototype, a knowledge base mainly comprising the ontology and lexicon specific for the domain of the texts. The descriptions of the texts are compiled and are also stored in the database, so the texts of the database are tied to the ontology in the knowledge base through these descriptions, as indicated in figure 3.

Compound descriptors (by semantic relations) are not derived in the current prototype. However, based on a simplified analysis, a bracketing of noun phrases in sentences is performed and used in forming description structures as introduced above.

For additional details on the motivation behind the architecture of the prototype we refer to [2]. Below we will introduce the current prototype system. We describe briefly the knowledge base (ontology and lexicon), the description generator and the query tool.

8.1 The Knowledge Base

The prototype ontology and ontology-based lexicon is built on the SIMPLE framework as referred to in Section 4. The SIMPLE ontology – currently consisting of 10,000 concepts – is being extended with a domain-specific part developed

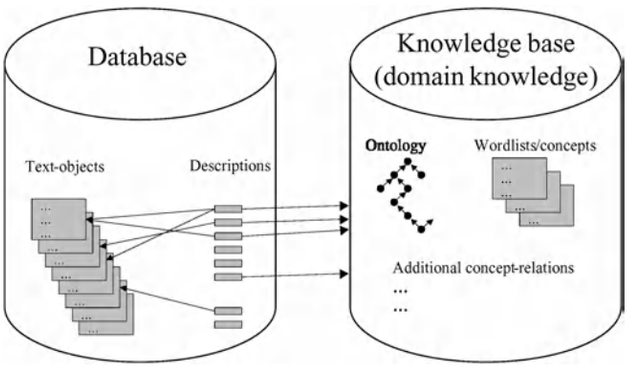


Fig. 3. System resources - a database containing text objects (documents) and a knowledge base comprising knowledge about the domain of the texts connected through descriptions

empirically on the basis of texts on nutrition from the Danish Encyclopedia. These texts also make up the coverage of the domain specific lexicon, currently consisting of approximately 1000 nutrition concepts. The English concepts at the top of the OntoQuery ontology correspond to the ontological types common to all languages in SIMPLE, while the Danish concepts with their lexical entries are situated lower in the ontology

The prototype ontology is built around the one major relation; concept inclusion (ISA). Figure 4 shows a concept "mangelsygdom" as situated in the ontology[3].

8.2 Description Generator

Descriptions are derived by means of an analysis involving a tagger, an NP recogniser, and a subcomponent that builds descriptions in the description language as illustrated in Figure 5.

Tagging is performed by an implementation of Eric Brill's tagger [6] customised for interaction with a text database and trained for Danish on the Danish PAROLE corpus [15], consisting of 250,000 tokens. The results obtained with a tagset of 43 tags, are comparable with those reported by Brill, namely an accuracy of 96.5 %.

NP recognition is performed by the chunk parser "Cass" [1]. The grammar has been developed manually on the basis of the occurrence of various NP types in the PAROLE corpus and covers NP chunks extending from the beginning of the constituent to its head and including postmodifying prepositional phrases.

Descriptions are generated from the result of the NP recogniser through morphological processing, part-of-speech filtering and grouping into the description-structure – sets of sets of words. The grouping simply corresponds to the NP's recognised; thus only words belonging to NP's are included in descriptions.

[3] The navigator is an auxiliary tool available at www.ontoquery.dk/prototype/ontonav.

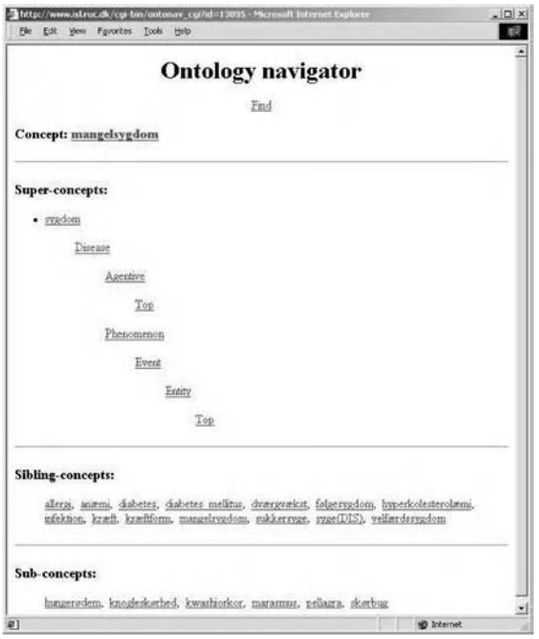


Fig. 4. The ontology navigator (www.ontoquery.dk/prototype/ontonav); showing that **mangelsygdom** (deficiency disease) has for instance **sygdom** (disease) as super- and **pellagra** as sub-concept and further that e.g. **allergi** (allergy) is a sibling and that the only two paths to the top of the ontology are **mangelsygdom** → **sygdom** → **disease** → **agentive** → **top** and **mangelsygdom** → **sygdom** → **disease** → **phenomenon** → **event** → **entity** → **top**

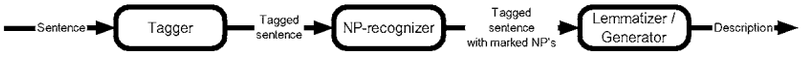


Fig. 5. Prototype architecture - the process of generating descriptions

Compared with the descriptors in Section 2, a set of words representation can be seen as an approximation where the relations between concepts contributed mainly by prepositions are left unspecified.

A set of words is thus considered as an abstraction of a concept in the way it is applied in the prototype. This implies that apart from the concept inclusion through the ISA relation of the ontology, we have a concept inclusion as derived from set inclusion.

8.3 The Query Tool (Description Comparison)

Query evaluation is, as already mentioned, reduced to comparison of a description of the query with descriptions of text objects in the database. As explained

Fig. 6. The querying tool of the OntoQuery prototype (www.ontoquery.dk /prototype/query) showing the answer to a query. Notice that the first object in the answer matches the query but not fully since **sygdom** is a second level super-concept to **pellagra** (see Figure 4) and **(ensidig,mangelfuld,kost** is considered a sub-concept of **(ensidig,kost)**

above in section 7 this mainly involves an aggregation of measures of correspondance between elements of the descriptions.

The query evaluation part of the current prototype[4] is based on a simplified aggregation approach as compared with the principle introduced in section 7. The simplified aggregation is a two-level order-weighted aggregation with the two levels arising from the set of sets structure of descriptions. The method is explained in more detail in [4] and it is a special simplified case of the hierarchical aggregation principle introduced in [3].

As for the similarities to be aggregated, we use a naive approach based on distance (shortest path length) in the ontology. This is also explained in [4] but as an example take the part of the ontology shown in figure 4 and the query shown in figure 6. A concept X of the query matches a sub-concept Y by $\frac{1-d(X,Y)}{1}$, where $d(X,Y)$ is the distance, thus X matches X to 1.0 and matches an immediate sub-concept to 0.9, a second-level sub-concept to 0.8, etc. Therefore **sygdom** matches **pellagra** to 0.8. Further, the degree to which **(ensidig,mangelfuld,kost)** matches **(ensidig,kost)** is aggregated over the query description part by simple average. Thus the grade becomes 0.67 since **(mangelfuld)** is missing.

The grading of the first object in the answer in Figure 6 is derived from an aggregation of similarities (0.8, 0.67, 1.0), where the latter corresponds to

[4] The prototype is available at www.ontoquery.dk/prototype/query

tryptofan, which describes both the query and the text object. The (outer) aggregation is in this case also a simple average of the similarities – thus the grading becomes 0.82.

Text objects in the prototype are sentences from the articles in the database. Thus a sentence level description is assumed. In the tool, sentences from the answer appear as hyperlinks leading from a sentence to the article it appears in. The tool includes a separate article viewing part, not shown above.

9 Conclusion

We have described principles for ontology-based querying of texts resting on extraction of the conceptual content of noun phrases into formal feature representations termed descriptors. These descriptors are integrated into the domain ontology by means of the notion of generative ontology.

As such the project faces the following methodological challenges:

- Integration of a comprehensive object database structured semantic lexicon with the logical grammar-structured generative ontology framework.
- Establishment of an "ontological semantics" for noun phrases in which the generative ontology provides the semantic domains.

Currently experiments with a more structured representation of the conceptual content of NPs as described in Section 2 are being carried out (see [25]).

Acknowledgements

The research in the OntoQuery project is supported by the Danish National Science boards with a grant 1999-2004.

References

1. Abney, S.: Partial parsing via finite-state cascades. *Proceedings of the ESSLLI'96 Robust Parsing Workshop*, 1996. Available from: http://www.sfs.nphil.uni-tuebingen.de/abney/.
2. Andreasen, T., Nilsson, J. Fischer & Thomsen, H. Erdman: Ontology-based Querying, in Larsen, H.L. *et al.* (eds.) Flexible Query Answering Systems, *Flexible Query Answering Systems, Recent Advances*, Physica-Verlag, Springer, 2000. pp. 15-26.
3. Andreasen, T.: Query evaluation based on domain-specific ontologies. In *NAFIPS'2001, 20th IFSA / NAFIPS International Conference Fuzziness and Soft Computing*, pp. 1844-1849, Vancouver, Canada, 2001.
4. Andreasen, T.: On knowledge-guided fuzzy aggregation. In *IPMU'2002, 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, 1-5 July 2002, Annecy, France
5. Andreasen, T., Jensen, P. Anker, Nilsson, J. Fischer, Paggio, P., Pedersen, B. Sandford & Thomsen, H. Erdman: *OntoQuery:* Ontology-based Querying of Texts, to appear at AAAI 2002 Spring Symposium, Stanford, California, 2002.

6. Brill, E.: Transformation-based error-driven learning amd natural language processing: a case study in part-of-speech tagging. *Computational Linguistics*, 21(4), pp.5543–565, 1995.
7. Copestake, A.: The (new) LKB system – version 5.2. CSLI, Stanford, 1999.
8. Fillmore, C.: The Case for Case. In Bach, E. & R. Harms (eds.): *Universals in Linguistic Theory*, New York: Holt, Rinehart & Winston, 1968.
9. Gonzales J., Verdejo, F., PETERS, C., Calzolari, N. Applying EuroWordNet to Cross-lingual Text Retrieval, in *Computers and the Humanities* Vol. 32: 185-207, 1998. Kluwer Academic Publishers, The Netherlands.
10. Guarino, N., Masolo, C., & Vetere, G.: OntoSeek: Content-Based Access to the Web. *IEEE Intelligent Systems*, 14(3) (1999) 70-80.
11. Jacquemin, C., and Tzoukermann, E.: NLP for Term Variant Extraction: A Synergy of Morphology, Lexicon and Syntax. In T. Strzalkowski (ed.), *Natural Language Information Retrieval*, pp. 25-74, Kluwer, Boston, MA, 1999.
12. Jacquemin, C., and Bourigault, D.: Term Extraction and Automatic Indexing. In R. Mitkov (ed.), *Handbook of Computational Linguistics*, Oxford University Press, Oxford, 2001.
13. Jensen, P. Anker & Skadhauge, P. (eds.): Proceedings of the First International OntoQuery Workshop *Ontology-based interpretation of NP's*, Department of Business Communication and Information Science, University of Southern Denmark, Kolding, 2001, to be republished at www.ontoquery.dk.
14. Jensen, P. Anker, Nilsson, J. Fischer & Vikner C.: Towards an Ontology-based Interpretation of Noun Phrases, In: P. A. Jensen & P. R. Skadhauge (eds.): Ontology-based Interpretation of Noun Phrases, in [13].
15. Keson, B.: Morfosyntaktisk tagging af danske tekster, in: P. Widell & M. Kunøe (eds.) 8. møde om Udforskning af Dansk Sprog, Århus Universitet, 1998.
16. Lenci, A., Bel, N., Busa, F.,Calzolari, N., Gola, E., Monacini, M., Ogonowski, A., Peters, I., Peters, W., Ruimy, N., Villegas, M., Zampolli, A.: SIMPLE - A General Framework for the Development of Multilingual Lexicons, in: T. Fontenelle (ed.) *International Journal of Lexicography* Vol 13. pp. 249-263. 2000. Oxford University Press.
17. Madsen, B. Nistrup, Pedersen B. Sandford & Thomsen, H. Erdman: "Semantic Relations in Content-based Querying Systems: a Research Presentation from the OntoQuery Project". In: K. Simov and A. Kiryakov (Eds.): Proceedings of OntoLex'2000: Ontologies and Lexical Knowledge Bases. OntoText Lab.,Sofia 2001. Forthcoming.
18. Nilsson, J. Fischer: A Logico-algebraic Framework for Ontologies ONTOLOG, in [13].
19. Nilsson, J. Fischer: Concept Descriptions for Text Search, in Proceedings from the 11th European-Japanese Conference on Information Modelling and Knowledge Bases, Maribor, Slovenia, 2001, to be republished in the series: Information Modelling and Knowledge Bases, IOS press.
20. Nilsson, J. Fischer: Are there Conceptual Grammars?, panel contribution at the 11th European-Japanese Conference on Information Modelling and Knowledge Bases, Maribor, Slovenia, 2001, forthcoming in the series: Information Modelling and Knowledge Bases, IOS press, 2002.
21. Nilsson, J. Fischer: Generative Ontologies, Ontological Types and Conceptual Grammars, in [31].
22. Nuopponen, A.: Concept Systems for terminological analysis. Acta Wasaensia, No.38. Universitas Wasaensis, Wasa, 1994.

23. OntoQuery project net site: www.ontoquery.dk
24. Paggio, P., Pedersen B. S. & Haltrup, D:: Applying Language Technology to Content-based Querying – The OntoQuery Project, forthcoming in Proceedings of Nordiske Datalingvistikdage, NoDaLiDa 2001, Uppsala, Sweden.
25. Paggio, P.: Parsing in ontoquery – experiments with LKB, in [13].
26. Pedersen, B.S & Keson, B.: SIMPLE Semantic Information for Multifunctional Plurilingual Lexicons: Some Examples of Danish Concrete Nouns, in: SIGLEX 99: Standardisaing Lexical Resources pp.46-51, ACL Workshop, 1999. University of Maryland, USA.
27. Pedersen, B.S. & Nimb, S.: Semantic Encoding of Danish Verbs in SIMPLE Adapting a verb-framed model to a satellite-framed language¡É in Proceedings from 2nd Internal Conference on Language Resources and Evaluation, pp. 1405-1412. Athens, Greece.
28. Pedersen, B.S. & Paggio, P.: A Danish Semantic Lexicon and its Application in Content-based Querying, submitted for review to Bouillon & Viegas (eds.) Semantic Lexicons in Natural Language Processing, Special Issue of T.A.L., Hermes, France.
29. Pustejovsky, J.: *The Generative Lexicon*, MIT press, 1995.
30. Smeaton, A. & Quigley, A: Experiments on Using Semantic Distances between Words in IMage Caption Retrieval, in *Proceedings of the 19th International Conference on Research Development in IR*, 1996.
31. Thomsen, Hanne Erdman (ed.): Ontologies and Search – 2nd OntoQuery Workshop January 2000, LAMBDA, nr.28, HHK (Copenhagen Business School), Frederiksberg, 2001.
32. Voorhees, E. M.: Using WordNet to disambiguate word senses for text retrieval, in Korfhage, R., Rasmussen, E., and Willett P. eds., *Proceedings of the 16th Annual ACM SIGIR Conference on Research and Development in Information Retrieval*, Pittsburgh, 1993, pp. 171 - 180.
33. Voorhees, E. M.: Query expansion using lexical-semantic relations. In Croft, W. Bruce and C. J. van Rijsbergen, eds., *Proceedings of the 17th Annual ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994, pp. 61 - 69.

Ontology-Based Data Cleaning*

Zoubida Kedad[1] and Elisabeth Métais[2]

[1] Laboratoire PRiSM, Université de Versailles 45, avenue des Etats-Unis
78035 Versailles Cedex, France
Zoubida.Kedad@prism.uvsq.fr
[2] Laboratoire Cedric, CNAM
192 rue Saint Martin, 75141 Paris cedex 3, France
elsa@cnam.fr

Abstract. Multi-source information systems, such as data warehouses, are composed of a set of heterogeneous and distributed data sources. The relevant information is extracted from these sources, cleaned, transformed and then integrated. The confrontation of two different data sources may reveal different kinds of heterogeneities: at the intensional level, the conflicts are related to the structure of the data. At the extensional level, the conflicts are related to the instances of the data. The process of detecting and solving the conflicts at the extensional level is known as data cleaning. In this paper, we will focus on the problem of differences in terminologies and we propose a solution based on linguistic knowledge provided by a domain ontology. This approach is well suited for application domains with intensive classification of data such as medicine or pharmacology. The main idea is to automatically generate some correspondence assertions between instances of objects. The user can parametrize this generation process by defining a level of accuracy expressed using the domain ontology.

Keywords: Multi-Source Information Systems, Data Cleaning, Ontology.

1 Introduction

A Multi-Source Information System (MSIS) is composed of a set of heterogeneous and distributed data sources, and a set of views (or queries) defining user requirements over these sources. Examples of MSIS are web systems and data warehouse systems. The main difference between an MSIS and a classical information system resides in its definition and its feeding with data. While the database schema of a classical information system is defined as an integrated data structure, the database structure of an MSIS is defined as a set of (possibly independent) views, either virtual or materialized. Moreover, while the feeding of a traditional database is done by the users through their applications, the feeding of the database in a MSIS is automatically done by the system from the data sources. MSIS applications aim only at the use of this data without any direct update.

* This work has been partly founded by the French Government in the framework of the REANIMATIC project

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 137–149, 2002.

MSIS have introduced new design activities such as selection of relevant data sources, data extraction, selection of views to materialize, update propagation of source changes, definition and generation of mediators, definition of computing expressions for each view in the system and data integration and cleaning.

One key problem in designing MSIS is the reconciliation of the data contained in the data sources, which is called the data cleaning problem. When merging the values contained in the sources, several problems may occur. A conflict may arise when the values of two semantically equivalent attributes in two distinct data sources are expressed using different units, different data format, or different scales. A conflict may also arise when two semantically equivalent attributes in two distinct data sources have sets of values which can be syntactically different but semantically close. In a data warehousing context, cleaning the data contained in the different sources is a critical issue, and performing this complex task is time consuming.

In this paper, we will focus on the problem of differences in terminologies an we propose a solution based on linguistic knowledge provided by a domain ontology. This approach is well suited for application domains with intensive classification of data such as medicine or pharmacology. The main idea is to automatically generate some correspondence assertions between instances of objects using this metadata. The user can parametrize this generation process by defining a level of accuracy expressed using the domain ontology.

The remaining of the paper is organized as follows. In section 2, we will present some data cleaning problems and existing solutions; section 3 will present the general approach for solving terminological conflicts. In section 4, we will describe the knowledge used for data cleaning and show how this knowledge stored in a domain ontology is used to deal with terminological conflicts. Some concluding remarks are given in section 5.

2 Data Cleaning: Problems and Existing Solutions

The problems that may arise during data integration can be classified according to two distinct levels: the extensional level and the intensional level. At the intensional level, the goal of the designer is to state the correspondences between the schemas of different data sources, ie. to find the semantically equivalent object types. At the extensional level, the goal of the designer once the correspondences between objects types are stated, is to find the correspondences between their instances, ie. to find the semantically equivalent instances of object types.

At the intensional level, the conflicts are related to the structure of the sources; examples of such conflicts are (i) differences in data models, which require some schema transformation techniques; (ii) differences in constructs, when two data sources use the same data model but some concept is modeled using different constructs: for example, the address of a person may be represented as an attribute in one source, and as a table in another; (iii) differences in the terminology, for example when two object types have the same name but correspond to different real world entities.

The conflicts related to the intensional level have been widely addressed by schema integration approaches [2], [16], [11], [23]. Once all the problems related to this level are solved, the designer is faced with another kind of conflicts, the ones related to the extensional level; these conflicts may be one of the following:

- Differences in types: such conflict occurs when the same real world property is represented using different data types in two distinct data sources; for example, the same attribute may be a number in one source and a string in another source.
- Differences in data format, for example when the instances of two equivalent attributes have different lengths in two distinct sources.
- Differences in scale, when the instances of two equivalent attributes are expressed using different units, such as Euro and USD for financial data.
- Differences in the encoding of attributes, for example the attributes *gender* in two different data sources can be encoded using the set of instances {1, 2} and {F, M} respectively.
- Differences in the terminology, for example, the attributes *medicine* may have in two distinct sources the two values *antibiotic* and *penicillin* for naming the same medicine.
- Differences in granularity, for example, two attributes *sales* may refer to monthly sales in one source and annual sales in another one.
- Conflicts related to the identifiers: two possible situations may arise; (i) two instances may have the same identifier value in two distinct sources although representing two distinct real world objects, or (ii) two instances may have two different identifier values in two distinct sources although representing the same real world object.

This enumeration of the problems related to data cleaning is not exhaustive, and a number of other problems may occur. In this paper, we focus on the problems related to the extensional level. Broadly speaking, a data cleaning process is composed of a detection step in which the existing conflicts are identified, and a resolution step in which the inconsistencies are solved. Several approaches have addressed these problems. A classification of the data quality problems is given in [19], and a distinction is made between single-source and multi-source problems, and between schema related (intensional level) and instance related problems (extensional level).

Some data cleaning approaches have proposed a set of data transformation operators, such as [8], where a general framework for data cleaning is provided, along with a set of cleaning operations. The Potter's Wheel interactive system is described in [20], integrating transformation and discrepancy detection. The system allows to define customized domains, and detects a discrepancy when the values of an attribute do not conform this domain. A set of data value transforms is proposed to remove the discrepancies. Some cleaning primitives allowing to detect integrity constraints violation are also proposed in the ARKTOS ETL tool [26], which allows to model and execute practical ETL scenarios. The conflicts existing among numerical data values are addressed in [5], and a methodology for mining data conversion rules is proposed, based on statistical techniques such as correlation analysis and regression analysis.

Another trend of approaches have focused on the problem of instance identification, that is, the determination of equivalent instances in different sources. A probabilistic technique is proposed in [3] to match the tuples of two distinct relations; the comparison is done considering not only the identifiers but also all the non-key attributes common to the two relations. Another solution to this problem is the sorted neighborhood method [10] where two considered relations are merged and sorted, then compared within a fixed size window so as to limit the number of comparisons. These comparisons are done using distance functions, such as edit, phonetic or type-

writer distances. A knowledge based approach is described in [14] for duplicate elimination; the considered records are fed into an expert system together with a set of rules forming the knowledge base. An algorithm for general textual matching is described in [18]; it consists of two passes: the first pass considers each record as a long string and sorts the records lexicographically reading from left to right; the second pass does the same reading from right to left. After the sort, the records are compared using the edit distance to find the duplicates. The Active Atlas system [25] compares the common attributes of two objects to determine matching ones. The system learns to tailor mapping rules to a specific application domain and proposes some functions related to text formatting inconsistencies.

In all existing approaches, the values of textual attributes are compared using distance functions which are based on the syntax of the values, such as the edit distance used in [8], [10] and [18]. The operators or algorithms using such distances are well-suited for solving some discrepancies, such as typographical mistakes during the insertion of the data, but they cannot solve the terminological differences. In such case, the values should be compared not only lexicographically, but also considering their semantic. In this paper, we are interested in the differences of terminology existing in different sources among data instances; we propose an approach to deal with these differences using linguistic knowledge provided by a domain ontology. Linguistic knowledge has already been used for solving some schema integration problems [11][16], and particularly during schema comparison to detect the redundant or equivalent objects in two schemas ; we propose to use this knowledge during the process of data cleaning to detect the equivalent data values which may be syntactically different.

3 The General Approach

Several conflicts may occur when integrating data from different sources; we are interested in the resolution of the terminological conflicts. To illustrate this kind of discrepancy between the data values, let us consider two source relations to be integrated $R_1(\#K_1,A_1,A_2,A_3)$ and $R_2(\#K_2,B_1,B_2,B_3)$ where K_1 and K_2 are the keys, A_i and B_i the attributes of R_1 and R_2 respectively. Let us suppose that the problem of determining the correspondences between the instances of R_1 and R_2 and the correspondences between their respective attributes have been stated. In this situation, the conflicts between the values of the non-key attributes remain to be solved and extra-knowledge related to the instance level is needed. If the attribute A_1 of R_1 and the attribute B_1 of R_2 are semantically equivalent, a terminological conflict may occur when a given value for the attribute A_1 is syntactically different from a value of the attribute B_1 although being semantically equivalent. To take this equivalence into account, some assertion is needed to state the semantic link between the two values of A_1 and B_1.

The example of figure 1 enlightens the problem we aim to solve. Let us suppose that we want to perform a union operation between two relations describing cars in two distinct data sources. We can see that the result of this union is hard to interpret and that it contains inconsistencies.

The two tuples having the number 1 in the resulting table should be merged and considered as the same since *azure* is a kind of *blue*; and there is an inconsistency in the resulting relation since the two tuples having the number 2 have conflicting values

for the attribute color, since *blue* and *vermilion* are not semantically close. These conflicts could have been automatically detected if some knowledge related to the semantics of the different colors were available. Using such domain knowledge, the result of the union would have been as described in figure 2. The tuple having the number 1 is present only once in the result, with the value *azure* for the attribute color, since this value is a kind of blue, and a table containing the different inconsistencies is produced as a second result. This table contains in our example the tuple having the number 2 because two conflicting values have been found for the attribute color.

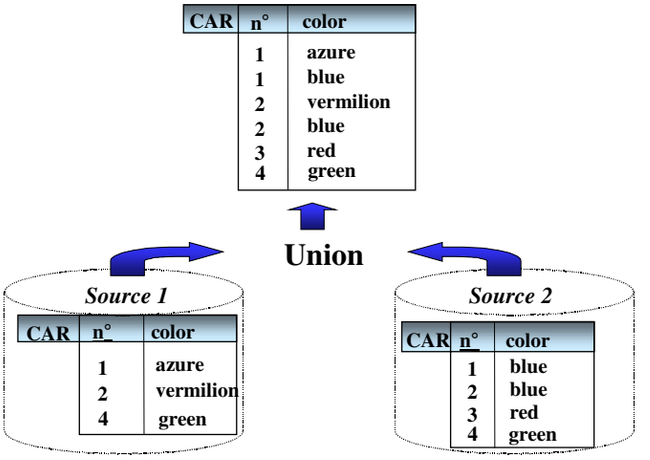


Fig. 1. Integration without linguistic knowledge

To integrate the two relations of figure 1, correspondence assertions between data values are needed for the attribute *color*. In general, considering the thousands of tuples from the tens sources, it is not realistic to store these assertions, like schema assertions are. Furthermore, these assertions are partially user dependant. For example, for the same query "search for cars having the same color than their cover-seats", some users expect blue cars and covers, whatever the kind of blue it is; while other users expect only cars and covers both *light-blue*, or cars and covers both *deep-blue*.

Some data cleaning problems can be solved by a preliminary cleaning of the data. For example, if the coding used for the attribute *gender* in two data sources is different, we can pre-process the corresponding relations in order to conform the different values. In the case of a terminological conflict, a preliminary cleaning of the data is inadequate because the values are not exactly equivalent. In figures 1 and 2, replacing the *azure* value by *blue* will lead to a lack of semantic; and cleaning in the second relation *blue* in *azure* is impossible without having the knowledge that the value in the other relation is *azure* and not, for example, *navy*.

Natural language techniques could be useful for solving the differences in the terminology. These techniques have been used for schema integration. [6] proposes the use of fuzzy and incomplete terminological knowledge to determine the correspondence assertions between the compared objects. In [11], correspondence assertions between the schemas to integrate are determined using case grammar and speech act

theory. [16] suggests the use of semantic dictionaries storing hierarchies of concepts and canonical graphs to retrieve the semantics of the schemas. [17] uses a fuzzy thesaurus built using linguistic tools to compute a fuzzy semantic nearness coefficient between objects. In the Carnot project [22], which provides a framework for heterogeneous database integration, a common ontology is used to semantically relate different schemas with each other and thereby enable the interoperation of the underlying databases. The common ontology is expressed using CYC, a knowledge base providing the "commonsense" knowledge of an application domain.

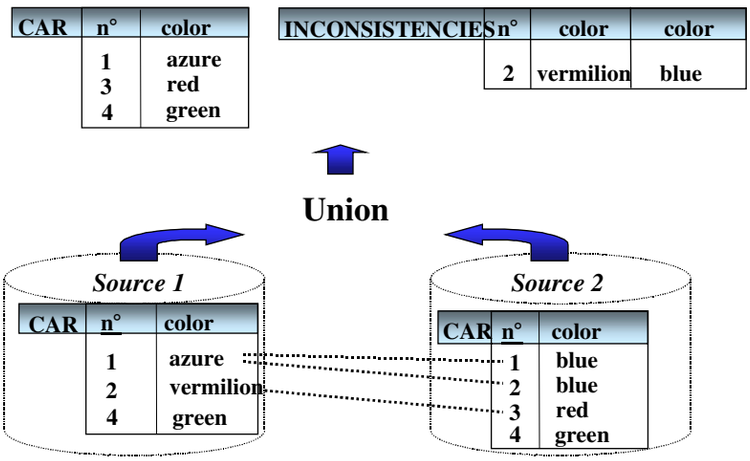


Fig. 2. Semantic integration using linguistic knowledge

Our approach proposes a linguistic matching to deal with terminological conflicts during data integration. This matching uses some knowledge specific to the application domain stored in a domain ontology and it allows the user some flexibility in the comparison of the values. A traditional challenge when introducing some flexibility is to define its scope. Numerous works [23], [21] have proposed semantic distances for integration or retrieval purposes. However, although this distance could be computed using a dictionary, we didn't want the user to specify it because this is not a natural way of thinking. Instead of pre-defining a distance, the user automatically defines classes of values by the mean of a level of accuracy specified over the domain ontology. Two values are therefore considered semantically close if they belong to the same class, and semantically far if they do not.

4 Solving Semantic Conflicts during Data Instances Confrontation

In this section, we will discuss how a domain specific ontology can help the process of data cleaning and especially for solving terminological conflicts; we will first present the knowledge used in our approach and introduce the notion of level of accuracy, a parameter specified by the user and used to determine the semantic closeness of the

concepts; then we will present the generation of data cleaning assertions for terminological conflicts and we will finally show how the meta-data provided by the ontology and the level of accuracy specified by the user are used during data integration.

4.1 Linguistic Knowledge for Data Cleaning

Numerous definitions may be found for "ontology" depending on their fields (databases, mathematics, linguistics, philosophy) [9]. However, one of these definitions is becoming predominant in the field of information systems: "an ontology is a formal conceptualization of a real world, sharing a common understanding of this real world". This definition fits so well to the semantic heterogeneity problem that this latter is nowadays the main client for ontologies, and the main application of ontologies is heterogeneous data sharing.

Numerous efforts have been performed to elaborate general ontologies, mainly in the NL community, and several large-scale projects aiming at building electronic linguistic dictionaries have been proposed: the CYC project [12], [13] which also provides some "common sense" knowledge, the EDR project [4], WordNet [7] which has the particularity to be currently available and free in the public domain, and EuroWordNet, the adaptation of WordNet to European languages [27]. Beyond syntactical and morphological information, these dictionaries provide semantic links between concepts such as synonym, antonym, hyponym/hypernym (is-a) and meronym/holonym (part-of). An extensive discussion of relation types is presented in [24]. Other links are supported by "canonical graphs" which specify relationships expected among the concepts involved in a given action. The "is-a" links are particularly useful and allow to organize concepts into a "hierarchy of concepts".

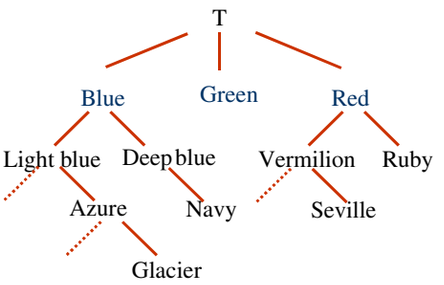


Fig. 3. Example of a hierarchy of concepts describing color's names

These general linguistic dictionaries are very helpful for retrieving the semantic of the nouns in schemas or data for their integration. They allow to supply with all the semantic lost during the modeling process. In previous works [15], [16] we have used WordNet to integrate database schemas. For example, one contribution of such dictionaries is the detection of generalization links between objects of different views (using the hierarchy of concepts). Another contribution is the deduction of the roles played by each entity in a relationship (using the canonical graphs).

Dealing with naming heterogeneity in the data cleaning process differs from dealing with naming heterogeneity in schema integration because we are faced with

instances (attributes values, e.g. *Seville*, *ticarcilline*) instead of type values (relation or attributes names, e.g. *color*, *medicine*). This entails the use of domain ontologies rather than general ontologies. An example of hierarchy of concepts elaborated for an ontology of color's names in the car industry is presented on figure 3.

4.2 The Level of Accuracy

The hierarchy of concepts shows the nature of the link between two compared concepts; in the example given in figure 3, we can see that the concept *navy* is a subtype of the concept *blue*, and that these concepts are semantically close since an inclusion relation exists between them; if we consider the concepts *ruby* and *vermilion*, we can also infer that they are semantically close because both of them are directly included in the same concept which is *red*. But the hierarchy of concepts alone is not sufficient to determine the semantic closeness of concepts; if we state that two concepts C_1 and C_2 are semantically close if there is a concept C in the hierarchy such that C_1 and C_2 are both included in C, then all the concepts of the hierarchy will be considered as close in meaning, since two concepts are always related to the top of the hierarchy denoted "T". Thus, this definition is too general and does not allow to evaluate the semantic closeness between concepts.

The semantic closeness between two given concepts often depends on a given user, or even on a particular need of this user; for example, the user might be interested in the red cars existing in different data sources, no matter what the kind of red it is; in another query, the same user might be interested in a specific kind of red cars. The notion of level of accuracy is a parameter specified by the user over the nodes of the hierarchy of concepts to capture the accuracy of his query or need by delimiting the scope of semantic closeness of values. It consists of several nodes selected by the user to specify classes of equivalent values in the hierarchy. The class induced by a node in the level of accuracy is represented by the subtree having the considered node as its top (i.e. the class of a node contains the set of values that it subsumes). The level of accuracy is denoted *LA* and is defined as $LA = \{C_i, i=1,n\}$ where each C_i is a concept of the hierarchy. Each class of values corresponding to the subtree having C_i as its top is denoted C_{Ci}.

The choice of the level of accuracy is arbitrary and may change from one query to another. The user may change the nodes as parameters. This level is a very easy and natural manner for the user to specify his classes of semantically close values, without manipulating any distance. However, it is important to notice that in practice, this choice is quite stable.

4.3 Generating Cleaning Assertions

Given the hierarchy of concepts and the level of accuracy set by the user, we can derive a set of extensional correspondence assertions that can be used during instance reconciliation. If we consider a level of accuracy defined as $LA = \{C_i, i=1,n\}$ and the corresponding classes of values C_{C1}, C_{C2},…., C_{Cn}, we can derive some correspondence assertions between data values. Two concepts C_i and C_j can be related in two different ways according to a given level of accuracy:

- If one of the two concepts is a subtype of the other, there is a subsumption relation between C_i and C_j denoted $C_i \subseteq C_j$.
- If the two concepts both belong to one of the classes C_{C1}, C_{C2},…., C_{Cn}, there is a similarity relation between C_i and C_j denoted $C_i \approx C_j$.

The notion of similarity between two concepts doesn't mean that they are always considered as equivalent; this similarity will be interpreted depending on the context, and particularly on the query issued by the user as we will see in section 4.4.

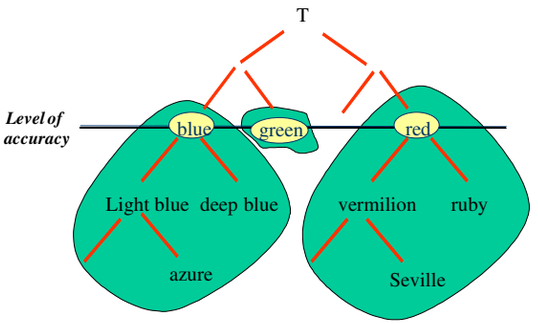


Fig. 4. Setting the level of accuracy in the hierarchy

In the example of figure 4, the level of accuracy set by the user is such that:

LA = {blue, green, red}

The partial extensions of the corresponding classes of equivalent values are as follows:

C_{blue}={blue, light blue, deep blue, azure,…..}

C_{red}={red, vermilion, seville, ruby,…..}

C_{green}={green, …..}

According to the level of accuracy set by the user, these are examples of cleaning assertions that can be derived from the domain ontology:

- azure ≈ deep blue, vermilion ≈ ruby
- azure ⊆ light blue, vermilion ⊆ red, ruby ⊆ red

The user may change the level of accuracy. For example in figure 4 the user wanted to consider that all the different blue colors were similar. In another query, he might want to distinguish between different kinds of each color, and he will set a more precise level of accuracy as it is shown in figure 5. This new level corresponds to a refined definition of similarity between concepts.

According to this level, some of the previous cleaning assertions become not valid; for example:

- the two assertions (azure ≈ deep blue) and (vermilion ≈ ruby) are not valid anymore.
- the subsumption assertions remain valid: azure ⊆ light blue, vermilion ⊆ red, ruby ⊆ red.

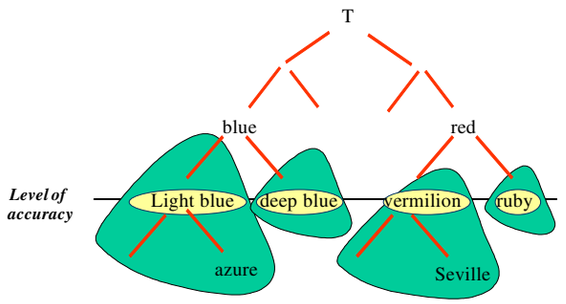


Fig. 5. Refining the notion of similarity

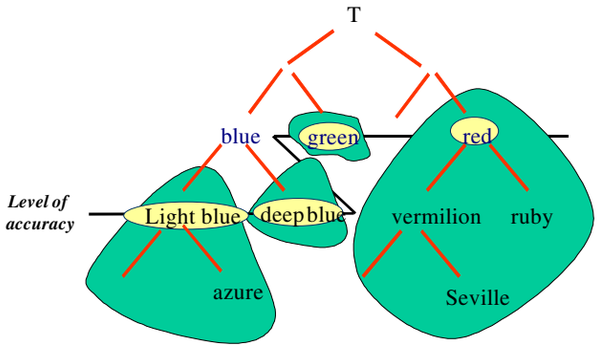


Fig. 6. Level of accuracy at different depths

The different concepts composing the level of accuracy don't need to be at the same depth in all the paths of the hierarchy. As it is shown in Figure 6, they can be at various levels. In such case, the user has obviously chosen to consider the blue colors more important than the others.

4.4 Using Cleaning Assertions for Answering a User Query

Once the level of accuracy is stated, the cleaning assertions can be used during the evaluation of a given user query. We will illustrate this process through two different relational operators, the restriction and the join.

4.4.1 Case of the Join Operator

A join operator is used for queries such as "Do people buy cover-seats and cars having the same color ?". The names of the colors for cars and cover-seats may be very different, since they have different manufacturers, each one having its own names. For this query, the user is not interested in an exact match of the color names. He just expects these colors to be the same with respect to the level of accuracy specified over the domain ontology.

Let us consider two relations $R_1(\#K_1,A_1,...A_i,...A_n)$ and $R_2(\#K_2, B_1,...,B_j,...,B_m)$; suppose that we want to perform a join operation on these two relations with the join

predicate $R_1.A_i = R_2.B_j$. The value of the attribute A_i corresponding to a tuple t_1 is denoted $t_1(A_i)$. Provided a domain ontology representing the values of the attributes A_i and B_j and a level of accuracy set by the user, the matching between two tuples t_1 and t_2 from R_1 and R_2 respectively is described by the following procedure:

```
Matching[t1(Ai), t2(Bj)]
   If t1(Ai) ⊆ t2(Bj) or t1(Bj) ⊆ t2(Ai)
      then matching succeeds
      else If t1(Ai) ≈ t2(Bj)
              then matching succeeds
              else matching fails
End;
```

If we suppose that the join attribute is the attribute color, and that the level of accuracy is as described in figure 4, then the pairs of values (*ruby*,*vermilion*) and (*seville*,*red*) will match, the first one because of a similarity assertion and the second one because of a subsumption assertion.

The matching relationship is reflexive, symmetric and not transitive; these properties are the three characteristics of a "similarity" relationship (versus an "equality" relationship which is reflexive, symmetric and transitive).

4.4.2 Case of the Restriction Operator

A restriction operator is used for queries such as "Do red cars have more accidents than others ?". In such cases, the matching between the attribute color of a table should succeed even if the value of the attribute is not exactly *red* but a specific kind of it, such as *vermilion*. For this query, the user is not interested in an exact match of the color names. Unlike the join operator, he doesn't expect the colors to be the same with respect to the level of accuracy, he expects the values to be included in the one given as a predicate for the restriction.

Let us consider a relation $R(\#K,A_1,...A_i,...A_n)$. Suppose that we want to perform a restriction on R with the predicate $R.A_i = val$, where *val* is a constant value in the domain of the A_i attribute. The values of A_i are represented by a domain ontology. Each value of the attribute A_i corresponding to a tuple t is denoted $t(A_i)$. The matching between *val* and $t(A_i)$ is described by the following procedure:

```
Matching[t(Ai), val]
   If t(Ai) ⊆ val
      then matching succeeds
      else matching fails
End;
```

If we suppose that the restriction predicate is "color = *red*" where the values of the attribute color are described in the hierarchy given figure 4, then the values *ruby*, *vermilion* and *seville* will match the criteria because of a subsumption assertion between each value and the concept *red*. If the restriction predicate is "color = *vermilion*" in another query, then the value *seville* will match because it is subsumed by *vermilion*, but the value *ruby* won't match in spite of a similarity assertion between *ruby* and *vermilion* according to the level of accuracy.

The cleaning assertions can be used for other relational operators. In the UNION operation, duplicate values could be avoided, for example if a same car's color has two different names in the two sources. In the DIFFERENCE operation if all cars except the blue ones are asked for, all kinds of blue will be considered. An ORDER BY statement can sort the values of an attribute *color* by categories. For all these operations, a matching using the cleaning assertions can be defined.

5 Conclusion

In this paper, we have proposed a method for dealing with one particular kind of semantic heterogeneity during the process of data cleaning, which is the difference of terminologies in distinct data sources. The proposed approach uses some linguistic knowledge stored in a domain ontology in order to generate some correspondence assertions between data instances. These assertions are used during the integration of the data. We have illustrated this process through two relational operators, the join and the restriction. The generation of the assertions is done according to the needs of the user, by the mean of a level of accuracy specified over the domain ontology. This level consists of a set of concepts in the domain ontology which are used as parameters by the user to define the scope of the semantic closeness in a very natural and flexible manner, without specifying any distance.

The proposed method for cleaning data has been implemented in the REANIMATIC project [1]. The objective of this project is the elaboration of a prediction tool based on a data warehouse populated by data from Intensive Care Units. Problems in this context are due to the heterogeneity in the naming of diseases and medicines. Correlations between cares and diseases couldn't be found with syntactical joins (e.g. finding antibiotics means finding all the different medicines considered as antibiotics, whatever their names are). This application validated most results, and showed one important problem we have to deal with in future works. It concerns the introduction of time in the domain ontology. Indeed, in all scientific domains, knowledge is frequently updated. Without an indication of its validity time, the knowledge stored in the domain hierarchy may become inconsistent.

Acknowledgements

We would like to thank Professor Mokrane Bouzeghoub for his stimulating discussions and his valuable comments.

References

1. Agarwal S., Keller A.M., Wiederhold G., Krichna S. "Flexible relation: an approach for integrating data from multiple, possibly inconsistent databases." Eleventh International Conference on Data Engineering, IEEE (1995).
2. Batini C., Lenzerini M., Navathe S.B. "A Comparative Analysis of Methodologies for Database Schema Integration", ACM Computing Surveys, 15 (4), Dec. 1986.

[1] http://www.outcomerea.org/acceuil.html

3. Chatterjee A., Segev A. "Data Manipulation in Heterogeneous Databases", Sigmod Record, Vol. 20, N°4, December 91.
4. "EDR Electronic Dictionary Technical Guide", Japan Electronic Dictionary Research Institute, Ltd. Mita-Kokusai-Bldg. Annex, Mita 1-4-28, Minato-Ku, Tokyo 108, Japan. August 1993.
5. Fan W., Lu H., Madnick S.E., Chueng D. "Discovering and reconciling value conflicts for numerical data integration", Information Systems Journal, Vol 26, N°8, dec. 01.
6. Fankhauser, P., Kracker, M., Neuhold, E.J. "Semantic vs. Structural Resemblance of Classes" Sigmod Record, 20 (4) October (1991).
7. Fellbaum C. , "WordNet, an Electronic Lexical Database", The MIT Press, ISBN 0-262-06197-X, 1998.
8. Galhardas H., Florescu D., Shasha D., Simon E., Saita C. "Declarative Data Cleaning: Language, Model and Algorithms", INRIA report n°4149, March 2001.
9. Guarino N. editor, "Formal Ontology in Information Systems", IOS Press, ISBN 90-5199-399-4, 1998.
10. Hernandez M.A., Stolf S.J., "The Merge/Purge Problem for Large Databases", SIGMOD'95.
11. Johannesson P. "Using conceptual graph theory to support schema integration" Proc. of the 12th ER Conf. (1993).
12. Lenat, D.B. " CYC: A Large-Scale Investment in Knowledge Infrastructure" in CACM 38 (11): 32-38 (1995).
13. Lenat D.B., Millar G.A., Yokoi T., "CYC, WordNet, and EDR: Critiques and Responses" in CACM 38 (11): 45-48 (1995).
14. Low W.L., Lee M.L., Ling T.W. "A knowledge based approach for duplicate elimination in data cleaning", Information Systems Journal, Vol 26, N°8, december 01.
15. Métais E., Meunier J.-N., Levreau G., "Database Schema Design: A perspective from natural Language techniques to Validation and View Integration", 12th International Conference on the Entity/Relationship Approach, Dallas(Texas), Dec. 1993.
16. Métais E., Kedad Z., Comyn-Wattiau I., Bouzeghoub M. "Using Linguistic Knowledge in View Integration: toward a third generation of tools", DKE 97.
17. Mirbel I. "Semantic integration of conceptuel schemes" First International Workshop on Application of Natural Language to Data Bases (1995).
18. Monge A.E. "Matching Algorithms within a Duplicate Detection System". IEEE Data Engineering Bulletin 23(4) (2000)
19. Rahm E., Do H.H. "Data Cleaning: Problems and Current Approaches" Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 1999.
20. Raman V., Hellerstein J.M., "Potter's Wheel: An Interactive Data Cleaning System" Proceedings of the 27th VLDB Conference, Roma, Italy, 2001.
21. Resnik P. "Using Information Content to Evaluate Semantic Similarity in a Taxonomy", IJCAI'95 (1995).
22. Singh M.P., Cannata P.E., Huhns M.N., Jacobs, N., Ksiezyk T, Ong K, Sheth, A.P., Tomlinson C., Woelk D. "The Carnot Heterogeneous Database Project: Implemented Applications. In Distributed and Parallel Databases" Journal, vol. 5, n° 2, pages 207-225, April (1997).
23. Song W.W., Johannesson P., Bubenko, J.A. "Semantic similarity relations and computation in schema integration" in the review Data and Knowledge Engineering", 19(1996).
24. Storey V.C., "Understanding Semantic Relationships", VLDB Journal, 2, 455-488, 1993.
25. Tejada S., Knoblock C.A., Minton S. "Learning object identification rules for information integration" Information Systems Journal, Vol 26, N°8, December 01.
26. Vassiliadis P., Vagena Z., Skiadopoulos S., Karayannidis N., Sellis T. "ARKTOS: towards the modeling, design, control and execution of ETL processes" Information Systems, Vol 26, N°8, December 01.
27. Vossen P. "EuroWordNet - A Multilingual Database with Lexical Semantic Networks", Kluwer Academic Publishers, 1998.

Retrieving NASA Problem Reports with Natural Language*

Sebastian van Delden and Fernando Gomez

Department of Computer Science
University of Central Florida
Orlando, Florida 32816
{sdelden,gomez}@cs.ucf.edu

Abstract. A system that retrieves problem reports from a NASA database is described. The database is queried with natural language questions. Part-of-speech tags are first assigned to each word in the question using a rule-based tagger. A partial parse of the question is then produced with independent sets of deterministic finite state automata. Using partial parse information, a look up strategy searches the database for problem reports relevant to the question. A bigram stemmer and irregular verb conjugates have been incorporated into the system to improve accuracy. The system is evaluated by a set of fifty five questions posed by NASA engineers. A discussion of future research is also presented.

1 Introduction

At the National Aeronautics & Space Administration(NASA), reports documenting technical problems that have been encountered over the years are stored in a continually growing database. These Problem Reports(PRs) can be divided into specific areas such as Mechanical, Fuel Cell, Orbiter Structure, Orbiter Electrical... They are written by NASA engineers at remote terminals and range in size from as few as fifty words to almost two thousand words. This research implements a system which searches this database and returns relevant problem reports to questions which are asked in natural language by NASA engineers.

The system described here uses Brill's tagger[2] to first assign part-of-speech tags to each word in the question. The tagger was not trained on the database of NASA problem reports. To avoid simple errors that occur due to unknown words in the lexicon, a new type of transformation was incorporated into the tagger.

Once the text has been tagged, a partial parse of the question is produced with a set of deterministic Finite State Automata(FSAs). Once a partial parse of the question has been produced, each report in the database is scanned for the noun phrases in the question. A *score* is given to each report based on how many constituents in the noun phrases are matched. If too many reports are given a high score, verb phrase information is used to identify the most relevant reports.

* This research has been supported by NASA grant 16-40-201.

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 150–159, 2002.

Matching is improved using two techniques 1) a bigram stemmer, and 2) irregular verb conjugates. The primary role of the bigram stemmer is to return reports containing noun phrases with typos, which would otherwise be overlooked. Since the PRs are entered by NASA engineers at remote terminals in an informal manner, many PRs contain typos which can prevent relevant reports from being returned. Irregular verb conjugates are also used to generate more matches in both the initial noun phrase search as well as constrict the number of relevant matches in the secondary verb phrase search. Because of the possibility of nominalization of irregular verbs, noun phrase searches also benefit by incorporating the conjugate forms of irregular verbs.

The remainder of this paper is as follows: Section 2 describes the partial parsing of the question. Section 3 describes the look up strategy. Section 4 shows results of evaluating the system and Section 5 concludes the paper with a discussion of future research.

2 Interpreting the Question

Part-of-speech(POS) tags[1] are first assigned to each word in the question by Brill's tagger[2]. A set of Finite State Automata(FSAs) then identify the syntactic relations in the question. Fig. 1 illustrates this process. The rules of each component in this process are read in from separate text files.

Rules are in the form of lexical and contextual transformation rule lists for the part-of-speech tagger and FSAs for the other components. Nothing is hardwired in software, the system is entirely declarative.

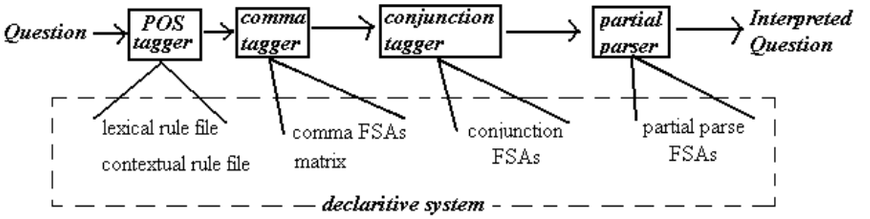


Fig. 1. The partial parsing system which is comprised of independent, declarative components.

2.1 The POS Tagger

Because of the time intensive nature of training Brill's tagger, the tagger was used as is. Brill's tagger which has been trained on the Penn Treebank Corpus[10] is available for download. The tagset consists of 36 *morphosyntactic* tags. The tags are called morphosyntactic because they contain more than just basic part-of-speech information. For example, they indicate whether or not a noun is singular or plural, proper or not proper.

[1] Refer to [13] for a thorough description of the Penn Treebank Tagset which is used by Brill's tagger.

Since the Penn Treebank consists of such a large collection of English text(about 4.5 million words in the Penn Treebank version 1), using the tagger without training still yields good tagging results. However, some simple errors that would not arise had the tagger been properly trained[3] still occurred. The main problem is the manner in which contextual rules are applied by Brill's tagger. Contextual transformations alter the tagging of a word from X to Y iff either:

1. *The word was not seen in the training corpus OR*
2. *The word was seen tagged Y at least once in the training corpus*

Many words in the problem reports need a part-of-speech tag that did not occur in the Penn Treebank corpus. For example, the word *pitted* only occurs as a past participle(VBN) and past tense(VBD) verb in the Penn Treebank. So even when it was used as *the pitted bearing*, it was tagged as a past participle verb when it should have been an adjective(JJ). None of Brill's contextual transformations could prevent this, since neither 1. or 2. are satisfied. To remedy the problem a new type of transformation was incorporated into the system:

$$X\ Y\ FPREVTAG\ Z \tag{1}$$

which changes the tag of $word_a$ from *X* to *Y* if it is preceded by $word_b$ which is tagged *Z,* regardless of whether or not $word_a$ includes the *Y* tag in the lexicon. To correctly tag *the pitted bearing*, the transformation was instantiated as *VBN JJ FPREVTAG DT* – where DT is the tag for determiner. Another instantiation was included to change a word that is tagged VBN to JJ when it is preceded by a preposition or subordinating conjunction: *VBN JJ FPREVTAG IN*, as in *with pitted bearings*. Adding these transformations corrected many simple errors without producing any of their own.

2.2 The Finite State Approach

Our approach to partial parsing is opposite to Abney's *easy-first parsing*[1]. We identify larger relations first and then focus on the smaller constituents comprising them. Consider the example*: John went to the black board after the teacher threatened to expel him.* In the easy-first approach, simple FSAs are first used to capture smaller relations. Noun phrases are first identified, followed by prepositional phrases, subordinate clauses, etc… This would result in *after the teacher* incorrectly being identified as a prepositional phrase.

In our *larger-first parsing* approach, we first use simple FSAs to identify the larger syntactic relations and then identify the smaller constituents inside these relations. In the example above, the subordinate clause *after the teacher threatened to expel him* would be identified first, followed by the smaller constituents inside of it. Since the Penn Treebank Tagset does not distinguish between a subordinating conjunction and preposition, this error will occur frequently.

A shortcoming of the part-of-speech tagger is that no attempt is made to categorize commas or coordinating conjunctions. Two intermediate components have been

added to the system which accomplishes this by assigning descriptive tags to these elements. Following our larger-first philosophy, the comma tagger[4] is the next component after tagging. Larger relations delimited by commas such as relative clauses, subordinate clauses, and appositions are identified first, as well as commas coordinating a series or pair of syntactic relations. A total of sixteen types of descriptive comma tags have been defined.

Fig. 2 and 3 show the Finite State Automata which identify relative clauses and appositions. The first tags commas which enclose reduced relative clauses. So, for example, in the sentence: *John, urged on by his classmates, walked up to the board.* The term NP represents *any* tag that can occur in a noun phrase, and PP represents the set of preposition tags.

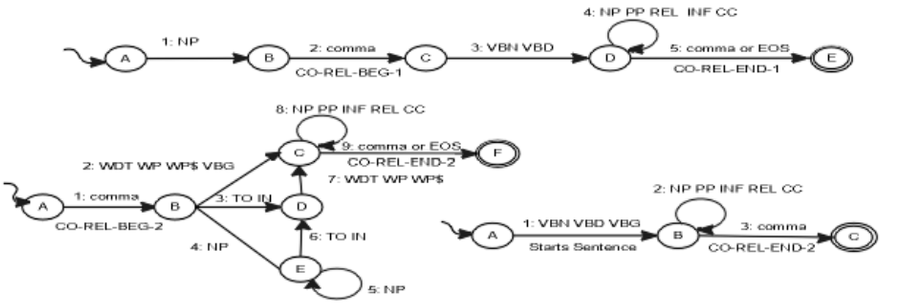


Fig. 2. The FSAs which identify commas enclosing relative clauses.

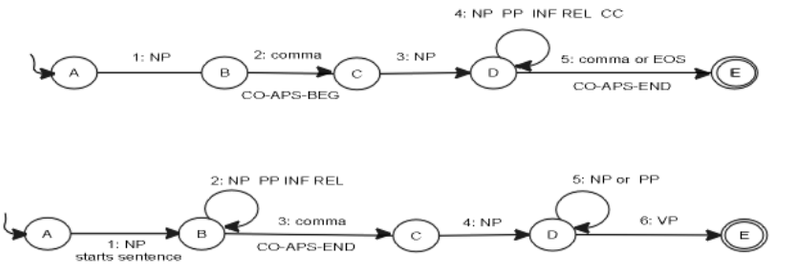


Fig. 3. The two FSAs which tag commas enclosing appositions.

The second FSA in Figure 3. (bottom left) tags commas enclosing relative clauses which are introduced by a Wh-Determiner or Wh-Pronoun, a preposition followed by a Wh-Determiner or Wh-Pronoun, or a noun phrase which is then followed by a preposition and a Wh-Determiner or Wh-Pronoun. The following three sentences are examples of these cases, respectively: *Mary wanted to study with John, who was currently out of town. Denmark has many busy seaports, of which Copenhagen is the most important. Frank read five books this year, the first of which he like the most.* The last FSA(bottom right) tags the comma which concludes a relative clause starting a sentence. For example: *Urged on by his students, John approached the black board*.

The first apposition FSA in Fig. 3 tags the comma which introduces an apposition and the comma that concludes the apposition if one is present. For example in the sentence: *She wanted to study with John , the best student in the class*. The second apposition FSA is for appositions starting a sentence and appositives the noun phrase which directly follows it. For example in the sentence: *A good student, John showed up to class everyday on time.* On arc 6, VP represents any tag in a verb phrase.

An important aspect of the comma is that it can delimit numerous syntactic relations simultaneously. For example, *In the Fall of 1992, a great year for sports, my favorite team won the World Series*. Here the first comma concludes a prepositional phrase, but also introduces an apposition. Every comma FSA is considered for every comma in the sentence, and a temporary set of comma tags are assigned to each comma. A co-occurrence matrix identifies which comma tags can occur with each other, removing incorrect co-occurrences. This is different to the conjunction and partial parsing application of FSAs since conjunctions and syntactic relations require only one descriptive tag. van Delden and Gomez [4] also show that this co-occurrence matrix can be automatically acquired using a greedy learning alogorithm which is similar to transformation-based learning techniques([3], [12]).

Comma tag information is used by the conjunction tagger as well as the partial parser. Since many coordinating conjunctions occur in close proximity to commas, we found that over 30% of conjunctions could automatically acquire their tag from comma tag information. These results were obtained by an inspection of Section 23 of the Wall Street Journal Penn Treebank 3. Since the comma tagger achieves 95% accuracy on correctly tagged text, this proves to be a good bases for the conjunction tagger.

Relying on part-of-speech, comma, and conjunction tags, the partial parser FSAs produces the final partial parse of the sentence. The syntactic relations identified here will be used in the searching strategy.

3 Search Strategy

First the noun phrases which have been identified by the partial parser are searched for in the problem reports. Each report that matches at least one constituent of a noun phrase is given a score and recorded. Pronouns and determiners are removed from the noun phrases. The searching strategy is as follows:

```
report-score = 0
FOR EACH noun phrase in question
    np-score = 0
    FOR EACH sentence in report
        t-score = # of constituents in common
        IF(np-score < t-score)
            np-score = t-score
    report-score = report-score + np-score
```

Each noun phrase is compared with each sentence in the report. The sentence which matches the most constituents in the noun phrase determines the score of that particular noun phrase. The constituents of the noun phrase are matched in the context of the entire sentence disregarding the order of the constituents. All noun phrases in the question generate scores for a particular report. These are combined to form a total *report score*. For example, the following question and example text illustrates this idea:

Is there a large white house in New York for sale?

Mr. York purchased a boat.
A house which is large and white is for sale.
The house is in New York.

Since pronouns and determiners are disregarded, the three noun phrases that will be searched for are: *large white house*, *New York,* and *sale. Large white house* will generate a score of 3 for the second sentence. This will remain its score since the third sentence will only generate a score of 1. *New York* will first be given a score of 1 by the first sentence, but the last sentence will change this score to 2. Finally, *sale* generates a score of 1 from the second sentence. The total score would be 6 for this sample text, given the question asked.

Identifying noun phrases and searching for them only in the context of a single sentence reduces the number of irrelevant reports returned. For example, consider the question: *Is there a large white boat for sale in New York?* If a mere match is performed between the noun phrase constituents across any sentence and in any order, this sample text above will also generate a score of 6 for this question. This is undesired since the sample text is clearly more relevant to the first question than the second one. Using our method, the sample text will only generate a score of 5 for the second question, making the sample text less relevant for this question.

Once all reports have been scored. The report with the best score is chosen first and returned to the user. Any remaining reports with a score that at least 75% of the best score are also returned. If there is a tie for the best score, one report is randomly chosen and returned first. If there are more than five reports with a score of at least 75% of the best score, verb phrase information is used to narrow in on the most relevant reports – see Section 3.2.

3.1 Enhancing Noun Phrase Matching

Two enhancements were made to increase the number of relevant reports returned: 1) a bigram stemmer, and 2) irregular verb conjugates.

Instead of only including morphological variations of the constituents in the noun phrase searches, the bigram stemmer[5] returned more relevant reports because of the nature of the problem reports. Since the problem reports were inputted by engineers at remote terminals in an informal manner, they contained not only several typos but

abbreviations that would not be recognized by morphology. For example, trying to match *the broken bearing* with *the brokn bearing*. Neither morphology or irregular verb conjugates with help to match *broken* with *brokn*. This could have been a typo by the NASA engineer or probably just an impromptu abbreviation of the word while entering the report. The bigram stemmer calculates a similarity measure between the two words, in our case Dice's coefficient was used:

$$S = 2C /(A + B) \tag{2}$$

where A is the number of unique bigrams in the first word, B is the number of unique bigrams in the second word, and C is the number of bigrams shared by A and B. Applying this to our previous example:

```
broken => br ro ok ke en => A is 5
brokn => br ro ok kn => B is 4
C is 3 – br, ro and ok
```

This yields a similarity measure of:

```
S = 2*3/(5+4) = 6/9 or .67
```

Using the recommended cutoff similarity measure of .6[5] would produce a match on these words even though there is a misspelling in the word.

Even though a bigram stemmer can be used to match many morphological terms, it cannot find matches between irregular verb conjugates. For example, a bigram stemmer will find a match between *reports* and *reported*, however, it would fail at matching *buy* and *bought*. To remedy this problem a list of irregular verbs was extracted from the Collaborative International Dictionary of English [6].

Over four hundred irregular verbs were automatically extracted from this resource. This list proved to be useful in matching noun phrases as well as verb phrases, because of the nominalization of irregular verbs. For example, consider the following question that was posed by a NASA engineer: *What causes excessive wear on the ET door latch paddles?* A problem occurs with the noun phrase *excessive wear*. *Wear* is a nominalization of the irregular verb *to wear*. In the most relevant report to this question, however, only the conjugate form *worn* occurs: *…forward and aft et door latchs are worn where roller hits latch lock indicator.…* Including the conjugate form *worn* produced a noun phrase match, increasing the relevance of this problem report.

3.2 Verb Phrase Matching

Verb phrase information is used when too many reports have been produced by the noun phrase matcher. All reports with a score that is at least 75% of the best score are used in the search. The algorithm is similar to that of the noun phrase matcher described earlier in Section 3. The verb score of the report is added to the report score

that was generated by the noun phrase matcher. Once all reports have been evaluated, the report with the new best score is chosen and returned along with any other reports which have a score of at least 75% of the best score.

If there are still many reports with scores over 75% of the best score, only the top five are returned. In cases such as these, the user is recommended to try re-phrasing the question or making it more specific.

Similar to the noun phrase matcher, both the bigram stemmer and irregular verb conjugates have been incorporated into the verb phrase matcher to improve performance. Auxiliary and modal verbs are removed from verb phrase searches.

4 Evaluation

The system was evaluated with a set of fifty five questions created by NASA engineers. The fifty five questions were asked to the Mechanical database which contains over five hundred problem reports ranging in size from fifty words to almost two thousand words. The results are shown in Table 1. The most relevant report was always found within the top three reports returned, and was the first report found for 82% of the questions.

Table 1. Evaluation results from a set of fifty five test questions created by NASA engineers.

Most Relevant Report Returned	
First	82%
In Top Two	95%
In Top Three	100%

In the case of a tie of the best report scores, the best reports are returned in random order. For example, when the most relevant report was returned second, it sometimes had a score equal to the first report returned.

Similarly, when the most relevant report was returned first, the second report sometimes had an equally good score. The aim of this research was not to always return the most relevant report first, but within the top five. The strategy described here exceeded our expectations for the set of test questions by returning the most relevant report within the top three. Since there were about five hundred reports, this can be considered a filtering of irrelevant reports with an accuracy of 99.4%.

5 Future Research

This system so far solves an *information retrieval* problem by incorporating natural language processing. A small set of relevant reports are filtered out of a larger database. Currently, the database has about five hundred reports and the system returns the most relevant report within the top three. A NASA engineer can use this system to find previous dispositions to problems that have already been encountered.

Future research on this project is two fold: 1) Enhancing information retrieval, and 2) Extracting knowledge from the most relevant reports. As more problem reports are added to the database, the information retrieval techniques current employed maybe not be sufficient to always return the most relevant problem reports. Since our partial parsing system assigns syntactic roles to the commas in the question, the search strategy could be enhanced to incorporate this information. For example, consider the test question: *Has pitting increased on MWA inner bearing races, which were flown with known pitting?* Using comma information, *which were flown with known pitting* is identified as a relative clause. When searching for the noun phrase *MWA inner bearing races*, a higher score can be given to the report in which *MWA inner bearing races* is being modified by *flown with known pitting* or a variation there of. Currently, even if these two relations occur in separate sentences, they will receive the same score as when they are found in the same sentence. Similarly, the conjunction tags may be incorporated in the search strategy to score reports differently.

WordNet[11] might also enhance the search strategy. Incorporating WordNet synsets has been shown to improve recall[14] because more relevant documents, that do not include the specific query terms, are matched. However, there have been few successful applications using WordNet to improve precision. Gonzalo et al.[9], however, does describe how incorporating WordNet synsets in the indexing space improve performance. Including some of these techniques may also improve the accuracy of our system.

The second focus of future research is to extract knowledge from the most relevant reports that have been filtered out of the database. After a partial parse of both the question and the most relevant problem reports are created, a more sophisticated analysis can be performed to directly answer the question. For example, return a *yes* or *no* for a yes/no question, including an explanation which supports your answer.

The first step in this process would be a semantic analysis of the sentence. Gomez ([7], [8]) explains an algorithm which uses an enhanced WordNet to determine the meaning of the verb as well as its thematic roles, adjuncts and prepositional phrase attachment. The algorithm is based on predicates that have been defined by Gomez for WordNet verb classes. The thematic roles in the predicates contain information about the grammatical relations and ontological categories that realize them. The ontological categories belong to WordNet noun ontology, which has undergone reorganization and modification as dictated by the semantic interpretation algorithm. The use of the semantic interpreter will allow not only to obtain more precise answers, but also to mine the texts in search of semantic patterns across a large number of reports.

References

1. Steven Abney. *Partial Parsing via Finite-State Cascades*. In Proceedings of the ESSLLI'96 Robust Parsing Workshop (1996)
2. Eric Brill. *Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging* Computational Linguistics (1995) 21(4):543-565.
3. *Eric Brill.* A Report of Recent Progress in Transformation-Based Error-Driven Learning. *In the* Proc. ARPA Human Language Technology Workshop (1994)

4. Sebastian van Delden and Fernando Gomez. *Combining Finite State Automata and Transformation-based Learning to Determine the Syntactic Roles of Commas*. University of Central Florida. Technical Report TR-CS-02-01 (2002)
5. W. Frakes. *Stemming algorithms*. Information Retrieval: Data Structures & Algorithms, Prentice Hall, (1992) 131-160
6. GCIDE. *The Collaborative International Dictionary of English – CIDE.* Version 0.41. (1999)
7. Fernando Gomez. *An Algorithm for Aspects of Semantic Interpretation Using an Enhanced WordNet*. In Proceedings of the North American Chapter of the Association of Computational Linguistics June 2-7 (2001), 87-94
8. Fernando Gomez. *Linking WordNet Verb Classes to Semantic Interpretation.* Proceedings of the COLING-ACL Workshop on the Usage of WordNet on NLP Systems. Universite de Montreal, Quebec, Canada. (1998) 58-64
9. Gonzalo, J., F. Verdejo, I. Chugur and J. Cigarrán. *Indexing with WordNet synsets can improve text retrieval,* in ACL/COLING Workshop on Usage of WordNet for Natural Language Processing (1998)
10. M. Marcus, B. Santorini, and M. Marcinkiewiecz. *Building a large annotated corpus for English.* Computational Linguistics (1993) 19(2):313-330
11. George Miller. *Introduction to WordNet: An On-line Lexical Database*, Princeton, CSL Report 43, (1993)
12. G. Ngai and R. Florian. Transformation-Based Learning in the Fast Lane. North American Chapter of the Association for Computational Linguistics (2001)
13. Beatrice Santorini. *Part-of-speech Tagging Guidelines for the Penn Treebank Project*, 3rd Revision, 2nd Printing (1995)
14. A. Smeaton, F. Kelly, and R O'Donnell. *TREC-4 Experiments at Dublin City University: Thresholding posting lists, query expension with WordNet and POS tagging of Spanish*. In Proceedings of TREC-4 (1995)

Access to Multimedia Information through Multisource and Multilanguage Information Extraction

Horacio Saggion, Hamish Cunningham, Kalina Bontcheva, Diana Maynard, Cris Ursu, Oana Hamza, and Yorick Wilks

Department of Computer Science - University of Sheffield
211 Portobello Street - Sheffield, England, UK, S1 4DP
{saggion,hamish,kalina,diana,cursu,oana,yorick}@dcs.shef.ac.uk
Tel: +44-114-222-1947, Fax: +44-114-222-1810

Abstract. We describe our work on information extraction from multiple sources for the Multimedia Indexing and Searching Environment, a project aiming at developing technology to produce formal annotations about essential events in multimedia programme material. The creation of a composite index from multiple and multi-lingual sources is a unique aspect of this project. The domain chosen for tuning the software components and testing is football. Our information extraction system is based on the use of finite state machinery pipelined with full semantic analysis and discourse interpretation.

1 Introduction

The vast amount of multimedia information available and the need to access its essential content accurately to satisfy users' demands encourages the development of techniques for multimedia indexing and searching. It is well known that there are no effective methods for automatic indexing and retrieving of image and video fragments on the basis of analysis of their visual features. Some projects, like Pop-Eye and OLIVE [10], exploit collateral linguistic media as a means for the automatic identification and indexing of relevant multimedia material. Pop-Eye uses subtitles to index video streams offering time-stamped text to satisfy user queries. OLIVE uses ASR to generate transcriptions of news reports indexed in ways similar to Pop-Eye. Semantic Video Indexing [7] is a proposal emphasizing the use of domain knowledge to index and retrieve video material, but has not been implemented. The Informedia Projects I and II (http://www.informedia.cs.cmu.edu) combine visual features and shallow language analysis (i.e., keyword spotting) to characterize multimedia content.

The Multimedia Indexing and Searching Environment (MUMIS) Project[1] is the first multimedia indexing project which carries out indexing by applying

[1] MUMIS is an on-going EU-funded project within the Information Society Program (IST) of the European Union, section Human Language Technology (HLT). Project participants are: University of Twente/CTIT, University of Sheffield, University of Nijmegen, Deutsches Forschungszentrum für Künstliche Intelligenz, Max-Planck-Institut für Psycholinguistik, ESTEAM AB, and VDA.

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 160–171, 2002.

information extraction to multimedia and multilingual information sources in Dutch, English, and German, merging information from many sources to improve indexing quality, and combining database queries with direct access to multimedia fragments on the multimedia programme.

In this paper, we present our approach to Information Extraction from English texts that is based on the use of finite state machinery pipelined with full semantic analysis and discourse interpretation. The rest of the paper is organized as follows: in Section 2 we give an overview of the MUMIS project, then in Section 3 we introduce the English Information Extraction System. In Section 4, we describe our approach to syntactic parsing and semantic interpretation. Discourse interpretation and coreference resolution is presented in Section 5, and finally, in Section 6 we present our conclusions.

2 Project Overview

In MUMIS various software components operate off-line to generate formal annotations from multisource linguistic data in Dutch, English, and German to produce a composite index of the events on the multimedia programme. The domain chosen for tuning the software components and for testing is football, and in particular the Euro 2000 Championships. This subject was chosen because of the huge amount of information available on the domain, as well as for the economic and public interest. An analysis of the domain has led us to propose 31 types of event for a football match (kick-off, substitution, goal, foul, red card, yellow card, etc.) that need to be identified in the sources in order to produce a semantic index. The elements to be extracted that are associated with these events are: players, teams, times, scores, and locations on the pitch.

A corpus of collected textual data in the three languages was used to build a multilingual lexicon and shared ontology of the football domain. Based on this shared model, three different off-line Information Extraction components, one per language, were developed. They are being used to extract the key events and participants from football reports and to produce XML output. A merging component or cross-document coreference mechanism has been developed to merge the information produced by the three IE systems. Audio material is being analysed in order to obtain transcriptions of the football commentaries (spontaneous speech). Keyframes extraction from MPEG streams around a set of pre-defined time marks - result of the information extraction component - is being carried out to populate the database.

The on-line part of MUMIS consists of a state of the art user interface allowing the user to query the multimedia database (e.g., "The fouls committed by Beckham"). The user is first presented with selected video keyframes as thumbnails that can be played obtaining the corresponding video and audio fragments.

3 Overview of the English Information Extraction System

Our system is conceptualised as a *Java front-end system* based on finite state transduction followed by a *Prolog back-end system* for inference over a classi-

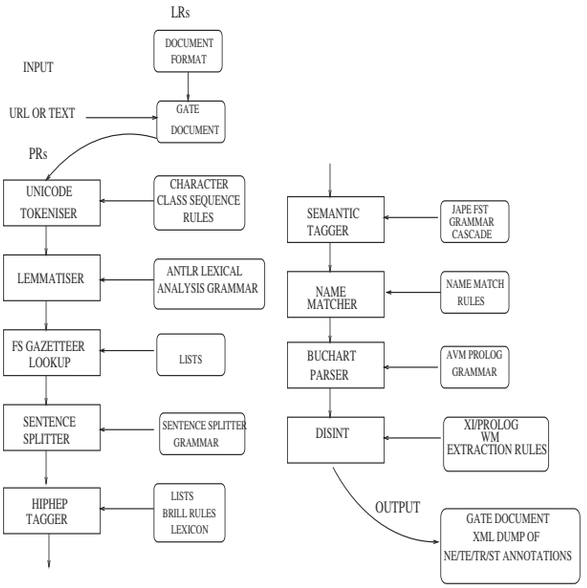


Fig. 1. English Information Extraction System Architecture

fication hierarchy implemented in SICStus Prolog. The system architecture is shown in Figure 1.

The finite state machinery is based on ANNIE, a free IE system available as part of GATE, a General Architecture for Text Engineering [3] (see `http://gate.ac.uk/`). The input to the process is a document that is automatically transformed into a GATE document: a structure containing the "text" of the original input and a number of annotation sets. Each component in the pipeline adds new information to the document in the form of annotations. An annotation has a type, a pair of offsets, and a set of feature-values that allow encoding orthographic, lexical, syntactic, and semantic information. The finite state components of the system are:

- Unicode Tokeniser that splits the text into very simple tokens such as numbers, punctuation and words of different types;
- Gazetteer Lookup process that identifies and classifies key words related to particular entity types in a particular domain. For MUMIS, we have collected information in order to identify players, trainers, referees, time markers, as well as stadiums, and sites. Features such as the affiliation and the position of each player in the Championships are particularly important for discourse interpretation (e.g., the team affiliation and position helps during entity coreference); and
- Semantic Tagging: it identifies and classifies more complex sequences of tokens in the source document. We use JAPE (Java Annotation Pattern Engine) [3], a pattern-matching engine implemented in Java, to identify and

annotate regular expressions over annotations. JAPE grammars are sets of rules which act on annotations assigned in earlier phases, in order to produce annotated entities. The rules are separated into two parts: the left hand side (LHS) of the rule performs pattern-matching; the right hand side (RHS) of the rule describes the annotation to be assigned. On the LHS, the pattern is described in terms of the annotations already assigned while the RHS of the rule contains information about the annotation to be produced, including attributes and their corresponding values. Distinctive characteristics of JAPE grammars are the possibility of specifying context for the pattern, control strategies and priority for triggering of the rules, and Java code that is executed whenever the LHS of the rule matches the annotations, allowing for a deeper level of analysis. JAPE uses a compiler that translates grammar rules into Java objects that target the GATE API (and a regular expression library). JAPE grammars were also used to develop the rule-based sentence splitter used in the system.

These processes are essential to identify as early as possible typical expressions and domain jargon that can be semantically interpreted without relying on full parsing. For example, an expression like "Fiore (Totti, 82)" indicates substitution, and can be identified with gazetteer lookup and a regular grammar. Additional processing using Java code in JAPE is done to extract semantic features (the player that goes in "Totti", the player that goes out "Fiore", and the time of the event "83").

Regular grammars and semantic tagging components were developed through careful corpus analysis. We evaluated the performance of our semantic tagger in the named entity recognition task using GATE's automated precision and recall evaluation tool AnnotationDiff [3]. As our training corpus refers to matches where 'England', 'Germany' or 'Holland' participated, we are relying on unseen texts reporting other events to evaluate our named entity recognition component. We have manually annotated with category 'Person' the set of all BBC reports about matches of the Group B of the Euro2000 championships: these constitute our gold standard for evaluation. In group B none of the above mentioned countries participated. Our semantic tagging component for the 'Person' (e.g., players) category was measured at 91% precision, 76% recall, and a combined F-measure of 82% (when precision and recall are equally weighted). This compares favourably with results obtained using a 'default' named entity recognition system that also uses gazetteer and grammar rules. The baseline system obtained 91% precision, 30% recall, and 45% F-measure.

The finite state mechanism is complemented with an orthographic name matcher process that looks for association between named entities by verifying a set of rules (e.g., "D. Beckham" and "David Beckham"). The information produced by this module is used during discourse interpretation.

4 Parsing and Semantic Interpretation

While the software components being developed are adaptable to any kind of football report, in this paper we focus on the analysis of tickers. Ticker reports

are in essence dynamic texts: a verbal account of events over time stamps, and this fact is taken into account during analysis: scores change as well as players leaving or entering the game. These texts also have a specific text structure that we take into account when parsing. We have developed a simple pre-processing step that identifies 'ticker header', where information about lists of players and the result of the game is usually stated, and 'ticker sections' grouping together sentences describing events under single time stamps. These sections are different from traditional paragraphs because they can either span multiple paragraphs or even a paragraph can contain multiple ticker sections. Other textual sources (e.g., match reports and comments) do not provide such rich temporal information, nevertheless they contain complete event descriptions ("David Beckham - a muted force in attack - was shown a yellow card for a late challenge on Kirsten"). Match reports and comments have paragraph structure that is identified by the GATE document structure analyser.

Part of speech tagging is done with an implementation of the "independence and commitment" learning approach to POS tagging [8]. For MUMIS, we have enriched the default lexicon produced by the learning step with our own vocabulary. The lexicon of the domain was obtained from the corpus and appropriate part of speech tags were produced with the help of the CELEX database [1]. We have also implemented a rule-based lemmatiser that produces an affix and root for each noun and verb in the input text. The external lemmatiser program is implemented as a set of regular expressions specified in flex and translated into C code.

We are using an implementation of the Bottom-up chart parsing described in [6], enriched with semantic rules that construct a naive semantic of each sentence in first order logical form. The parser is complete in the sense that every analysis licensed by the grammar is produced, though there is a mechanism to control this. On completion a "best parse" algorithm is run to select a single analysis of the sentence, which may be partial if no tree spanning the whole sentence can be constructed. The parser uses two grammars: the first is a domain dependent grammar used to produce logical forms for the entities of the football domain; the second is a context-free phrasal grammar of English enriched with features and values, that has been used in many Information Extraction projects [2]. It consists of a sequence of subgrammars for: noun phrases (NP), verb phrases (VP), prepositional phrases (PP), relative phrases (R) and sentences (S). The semantic rules produce unary predicates for entities and events (e.g., *player(e1)*, *save(e2)*) and binary predicates for properties (e.g. *lsubj(e1,e2)*). Constants (e.g., $e1$, $e2$) are used to represent entity and event identifiers. First order predicate names are: (i) the citation forms obtained during lemmatisation; (ii) forms used to code syntactic information (e.g. $lsubj$ for the logical subject of a given verb); or (iii) specific predicate names being used for domain modelling (e.g., $player\_of$). Instantiated values of properties attached to events are used as the slot fillers in the template representation the system is producing (e.g., the name of the scorer in a goal event). As an illustration, the partial semantic representation

for the sentence "41 mins: Beckham is shown a yellow card for retaliating on Ulf Kirsten seconds after he is denied a free-kick" is shown below:

```
time_stamp(e416), time_count(e416,'41'), player(e419), name(e419,
'Beckham'), player_team_name(e419,'England'), midfielder(e419),
card(e422), adj(e422,yellow), retaliate(e423), time(e423,none),
aspect(e423,simple), voice(e423,active), player(e425), name(e425,
'Ulf Kirsten'), player_team_name(e425,'Germany') ...
```

In Section 5.3, we will show how the semantics is used by the discourse interpreter. The parser is written in SICStus Prolog (version 3.6.8). In order to call the parser and to obtain the output back we rely on Jasper, a bi-directional interface between Java and SICStus [12].

5 Domain Modelling and Discourse Interpretation

The discourse interpreter is based on a World Model representing the ontological (or hierarchical) knowledge about a particular domain. The interpreter works by mapping the information produced by the parsing and semantic interpretation into an evolving Discourse Model of the input text. The World Model contains rules allowing the deduction of new knowledge from the "explicit" information found in the text, and also the connection between new and old instances mentioned in the input text (coreference).

$$
\begin{aligned}
&\mathit{entity}(X) \Longrightarrow \mathit{object}(X) \vee \mathit{event}(X) \vee \mathit{attribute}(X)\\
&\mathit{object}(X) \Longrightarrow \mathit{time}(X) \vee \mathit{person}(X) \vee \mathit{soccer\_artifact}(X)\\
&\mathit{person}(X) \Longrightarrow \mathit{player}(X) \vee \mathit{official}(X) \vee \mathit{player\_collective}(X)\\
&\mathit{player}(X) \Longrightarrow \mathit{goalkeeper}(X) \vee \mathit{midfielder}(X) \vee ...\\
&\mathit{soccer\_artifact}(X) \Longrightarrow \mathit{ball}(X) \vee \mathit{goalmouth}(X) \vee \mathit{goalpost}(X) \vee \mathit{crossbar}(X)\\
&\mathit{event}(X) \Longrightarrow \mathit{substitution\_event}(X) \vee \mathit{goal\_event}(X) \vee \mathit{save\_event}(X) \vee ...\\
&\mathit{attribute}(X) \Longrightarrow \mathit{functional}(X) \vee \mathit{relational}(X)\\
&\mathit{player\_of} \longleftarrow \mathit{functional}(X)\\
&\mathit{trainer\_of} \longleftarrow \mathit{functional}(X)
\end{aligned}
$$

Fig. 2. Partial Ontology for the Football Domain

Football reports make reference to unique concepts and events: our corpus of football reports and a user study helped in their identification (we used WordSmith [11] as the main tool for corpus analysis). Based on the corpus study we have specified a world model of the football domain. We have adopted the XI Knowledge Representation Language [5], a formalism that allows the user to code and operate with symbolic knowledge. XI is compiled into Prolog, making it possible to mix procedural knowledge with the basic declarative formalism's constructs. We have chosen XI because it has been proved successful in other domains for information extraction [9].

5.1 Knowledge Coding

XI provides basic language constructs to specify hierarchical relations. Individuals, classes of individuals, inclusion relations between classes of individuals and multiple inheritance hierarchies can be defined and attribute-values may be associated with classes or with individuals. In XI, classes are represented as unary predicates and individuals as atoms. An attribute or property is a binary predicate, the first argument of which is the class/individual the attribute is assigned to and the second being the value of this attribute. The value can be a fixed term or a term that becomes instantiated in appropriate situations when new knowledge is deduced during processing. Figure 2 gives an overview of our ontology of the football domain. Clauses $A \Longrightarrow B$ are used to specify 'inclusion' relations (all B is an A). Clauses $I \longleftarrow C$ are used to specify 'is a' relations between individuals and classes (I is a C). Operators $\vee$ and & indicate disjunction and conjunction respectively. Properties of individuals and classes are specified through the *props* predicate in the form:

```
props({Class|Individual},[Properties])
```

where *Properties* can be unconditional or conditional on the current state of the world. For example, to specify that "Paul Ince" plays for England we write:

```
props(e1,[player_of(e1,e2)])
```

where $e1$ is the constant for "Paul Ince" and $e2$ is the constant for "England"; this is an unconditional property. In addition to the declarative operators, a number of constructs can be used during deduction: $A \Rightarrow B$ is used to verify class inclusion (every B is a A), $A \leftarrow B$ is used to verify if A 'is a' B, and $hasprop(I, P)$ is used to verify if I 'has property' P (also, $nodeprop(I, P)$ can be used to verify properties but only at the instance level). Properties can be attached to individuals or classes conditionally on the actual state of the world. Conditional properties are specified with the "if" operator ($: -$). We have created our world model by defining the three types of properties required by the discourse interpreter: (i) rules that prevent entity coreference (e.g., a player who is playing cannot corefer with a player who is no longer playing); these are specified using the reserved predicate *distinct*, (ii) rules that allow the modification of the discourse model with presuppositions of events and entities (e.g., a substitution event presupposes a player that is playing and a player that is on the bench); these are compiled using the reserved predicate name *presupposition*, (iii) rules allowing the modification of the discourse model with consequences for events and entities (e.g., a goal event has as consequence a change in the state of the game); these are defined using the reserved predicate *consequence*. These rules also allow for completing the partial parse that the parsing process might have produced. More than 300 rules for discourse interpretation have been manually created for the MUMIS project.

5.2 Knowledge Processing

Knowledge processing is done in the following way:

1. The naive semantics of each sentence produced during parsing is mapped into an evolving discourse model of the text. Each element in the semantics is mapped into a node in the hierarchy by consulting a conceptual dictionary. Ambiguous predicates (e.g. goal for the event "to score a goal" and goal for the object on the pitch "the two goal-posts and the crossbar") are dealt with by discourse interpretation rules that take into account the context of the predicate. Predicates not found in the dictionary are mapped into objects or events depending on their syntactic category (noun or verb). Attributes are associated to instances using *props* constructs.
2. Presuppositions are added to the model. This causes the creation of new hypothesised instances.
3. Object coreference is applied to solve instances produced by the parser or by presupposition rules. The coreference mechanism will try to solve a hypothesis using the current sentence, any unresolved hypothesis in this step will be removed from the model.
4. Consequences are added to the model, any hypotheses will remain active because they might be solved later as new information is processed.
5. Event coreference is applied mainly to merge partially instantiated events.

We model the dynamic nature of tickers by maintaining the "dynamic state of the game", a logical structure that records among other elements: the teams playing, the time stamp under consideration, the players participating in the game, the final score, and the score at particular time stamps. This structure is the basis for many of the deductions the system has to make: for example when the system needs to find the time of a particular event or when it needs to decide which team scored the goal. The dynamic state of the game is updated as the text is processed sentence by sentence. Note that in tickers, expressions like "England 2 - 1 Germany" clearly indicate the state of the game, nevertheless, out of context they do not provide enough information about the team that scored the goal, actually the previous state of the game could be "England 1 - 1 Germany" or "England 2 - 0 Germany". The "dynamic state of the game" provides the information or context to disambiguate.

5.3 Coreference Resolution and Deduction

Event and entity coreference is the basic mechanism that allows the system to fill in properties for each event in the domain. Coreference is a unification-based process that is based on: (i) the notion of distance between nodes in the ontological hierarchy; (ii) a measure of similarity between entities and events computed using the values of associated properties (for example two players can be made coreferent if they are found similar by the name matcher process); and (iii) rules that constrain coreference. We use the basic coreference algorithm described in

[4]. It is a general algorithm that requires careful coding of the semantic properties of events and entities of the domain. We approach coreference resolution in a symbolic way by specifying a number of rules that constrain coreference. Coreference rules are mainly negative in the sense that they specify when entities should not corefer. A typical case is when two players participate in the same event, the two players should be different. Another typical case would be an event whose semantics indicate two players of the same team so the coreference mechanism should ensure that player from different teams cannot participate in that event. The rule below states that *PLAYER1* and *PLAYER2* are different if they participate in the same *GOAL* event.

```
props(player(PLAYER1),[(distinct(PLAYER1,PLAYER2):-
GOAL <-- goal_event(_),hasprop(GOAL,player_1(GOAL,PLAYER1)),
hasprop(GOAL,player_2(GOAL,PLAYER2)))]).
```

Presuppositions allow hypothesis of entities essential to complete the semantics of a particular event. A typical case is when the parser is unable to find the logical subject or logical object of a domain verb, in which case they need to be presupposed and passed to the discourse interpreter to complete the meaning of the domain verb. The following rule, for example, deals with the addition of the logical subject to the verb "save" in case the parser is unable to find the subject with the grammar.

```
props(save(SAVE),[(presupposition(SAVE,[lsubj(SAVE,PLAYER)]):-
hasprop(SAVE,voice(SAVE,passive)),hasprop(SAVE,by(SAVE,PLAYER)))])
```

Presuppositions are hypotheses local to the sentence under analysis. The rule below says that if the concept "substitution" is found, then presuppose a "substitution" event and hypothesise two players where the first player is introduced by preposition 'for.'

```
props(substitution(SUB),[(presupposition(SUB,
[substitution_event(E),player(P2),player_1(E,P1),
player_2(E,P2),player_of(P2,TEAM)]):-
current_instances(I), member(P1,I),P1<--player(_),
nodeprop(P1,for(_,P1)),nodeprop(P1,player_of(P1,TEAM))))]).
```

Consequences also allow hypothesis of entities and events but in a more general way, because they can be instantiated with entities coming from remote parts of the document or not yet processed. The rule below indicates that *PLAYER* plays for a given *TEAM* with name *NAME* (where *NAME* is knowledge coming from the gazetteer lookup step on the finite state machinery).

```
props(player(PLAYER), [(consequence(PLAYER,[team(TEAM),
player_of(PLAYER,TEAM),name(TEAM,NAME)]):-
hasprop(PLAYER,team_name(PLAYER,NAME)))])
```

The time of the event under consideration is instantiated by consulting the property *current_time* of the "dynamic state of the game" using the following property:

```
props(event(EVENT),[(presupposition(EVENT,
[event_time(EVENT,TIME)]):-SOG <-- dynamic_sog(_),
nodeprop(SOG,current_time(SOG,TIME)))])
```

The treatment of the scores is similar to treatment of time stamps, the dynamic state of the game records the current score and so it can be retrieved as each event is deduced.

The treatment of locations within this project is particularly challenging because the IE system's goal is to map location expressions into a 3x3 grid representing the pitch. Locations on the pitch are different from locations in traditional IE systems where they are usually identified by name. In football reports, expressions like "the box" or "the left" indicate locations relative to a particular team or player and are highly ambiguous. In our approach, we define a number of conceptual locations associated with the teams (a team has "a box", "a left", and "right"). When the teams are mentioned for the first time in the ticker (England vs Germany) the coordinates of the grid are set for each team. In order to obtain the correct coordinates for a linguistic expression like "the English box" we rely on the identity of the team that qualifies the box (note that if a player's box is mentioned instead, his team will be used). Once the identity of the team is obtained by explicit mention or by coreference, the coordinate can be obtained with the following discourse rule:

```
props(the_box(X),[(coordinate(X,Y):-hasprop(X,team_of(TEAM,X)),
hasprop(TEAM,box_coordinate(TEAM,Y)))])
```

where *team_of* is a property used to associate the team with the location and *box_coordinate* is used to associate the location with its coordinate. Other less ambiguous expressions like "the centre of the field" are mapped directly into their fixed coordinates.

5.4 Event Entailment and Event Coreference

Event entailment is particularly useful in dealing with tickers: some events in the game are a clear consequence of one another. For example, if a player receives a "yellow card", a "foul" committed by that player can be hypothesised, as well as the fouled player who might be unknown. This situation can be modelled using the following piece of code:

```
props(trigger_foul_event(FOUL),[(presupposition(FOUL,
[foul_event(E),player(PLAYER2),player_2(E,PLAYER2),
event_time(E,TIME),event_type(E,'foul'),player_1(E,PLAYER1),
team_player_1(E,TEAM)]):-SOG <-- dynamic_sog(_),
nodeprop(SOG,current_time(SOG,TIME)),
```

```
CARD <-- yellow_card_event(_),
nodeprop(CARD,event_time(CARD,TIME)),
nodeprop(CARD,player_1(CARD,PLAYER1)),
nodeprop(CARD,team_player_1(CARD,TEAM)))])
```

where *trigger_foul_event* is a special class used to map lexical items like "retaliate" or "challenge" that indicate fouling. Additional knowledge is used to constrain coreference between the two players involved in the foul.

Event coreference is carried out to merge partially instantiated events. Consider for example the text "Goal! England 1 - 0 Germany": Both expressions "Goal!" and "England 1 - 0 Germany", are used to indicate a goal event; if two goal events are deduced from the text, the event coreference mechanism should merge them using constraints such as the time stamp of the events and the participating players and teams. This is carried out with specific Prolog code. A template extraction and writing algorithm is used to read the events deduced by the system and to produce XML output.

6 Conclusion and Future Work

MUMIS is the first multimedia indexing project which carries out indexing by applying information extraction to multimedia and multilingual information sources, merging information from many sources to improve the quality of the annotation database, and combining database queries with direct access to multimedia fragments.

In this paper, we have presented our approach to IE. We have been successful in the integration of two complementary paradigms: finite state machinery implemented in Java with full syntactic analysis and discourse interpretation implemented in Prolog. The system is complete and operative and we are in the process of producing complete XML annotations for the full corpus.

Our work in progress involves formal evaluation of the information extraction task, which is being carried out using gold standards produced by annotators. Future work will address information extraction from transcribed spoken commentaries.

References

1. G. Burnage. *CELEX: A Guide for users*. Centre for Lexical Information, Nijmegen, 1990.
2. H. Cunningham, R.G. Gaizauskas, K. Humphreys, and Y. Wilks. Experience with a Language Engineering Architecture: Three Years of GATE. In *Proceedings of the AISB'99 Workshop on Reference Architectures and Data Standards for NLP*, Edinburgh, April 1999. The Society for the Study of Artificial Intelligence and Simulation of Behaviour.
3. H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, and C. Ursu. *The GATE User Guide*. `http://gate.ac.uk/`, 2002.

4. R. Gaizauskas and K. Humphreys. Quantitive Evaluation of Coreference Algorithms in an Information Extraction System. In *DAARC96 - Discourse Anaphora and Anaphor Resolution Colloquium*. Lancaster University, 1996.
5. R. Gaizauskas and K. Humphreys. XI: A Simple Prolog-based Language for Cross-Classification and Inhetotance. In *Proceedings of the 7th International Conference in Artificial Intelligence: Methodology, Systems, Applications*, pages 86–95, Sozopol, Bulgaria, 1996.
6. G. Gazdar and C. Mellish. *Natural Language Processing in Prolog: An Introduction to Computational Linguistics*. Addison-Wesley Publishing Company, 1989.
7. A. Hampapur. Semantic video indexing: approach and issues. *SIGMOD-Record*, 28(1):32–9, March 1999.
8. Mark Hepple. Independence and commitment: Assumptions for rapid training and execution of rule-based POS taggers. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*, Hong Kong, October 2000.
9. K. Humphreys, R. Gaizauskas, S. Azzam, C. Huyck, B. Mitchell, H. Cunningham, and Y. Wilks. Description of the LaSIE system as used for MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*. `http://www.itl.nist.gov/iaui/894.02/related_projects/muc/index.html`, 1998.
10. K. Netter. Pop-eye and olive. human language as the medium for cross-lingual multimedia information retrieval. Technical report, Language Technology Lab. DFKI GmbH., 1998.
11. M. Scott. *WordSmith Tools Manual*. Oxford University Press, 1996.
12. SICSTUS. Intelligent Systems Laboratory. Swedish Institute of Computer Science, PO Box 1263. SE-164 29 Kista, Sweden. *SICStus Prolog User's Manual*, May 2000.

On Semantic Classification of Modifiers*

Igor A. Bolshakov and Alexander Gelbukh

Center for Computing Research (CIC)
National Polytechnic Institute (IPN)
Mexico City, Mexico
{igor,gelbukh}@cic.ipn.mx,
www.gelbukh.com

Abstract. To search a large dictionary for a collocation expressing a desired meaning, the human reader needs some kind of hierarchical structure that would facilitate such search. In this paper, fragments of semantic classification of modifiers are elaborated based on several highly modifier-productive nouns, namely, common nouns *person, action, look, corporation,* and *price,* as well the terms *coating, medium,* and *check*. By modifiers meant are adjectives, participles, or preposition phrases syntactically dependent on the nouns. The classification rubrics proved to be heavily dependent on the modified headword noun and are representative fragments of a Roget-like thesaurus. It is shown that the modifiers under consideration are rather selective in their use, similarly to standard lexical functions (LFs) by Mel'čuk, while for many nouns LFs can be absent. The obtained classification rubrics can be used for other English nouns and for other languages. Some deficiencies of the proposed rubrics are discussed.

1 Introduction

Let us consider modifiers of nouns in European languages. They are mainly adjectives (*beautiful flower*) or participles (*obliging woman*). However, these languages use, for the same semantic purposes, prepositional phrases (*person of importance*) and other nouns in attributive function: in preposition (***stone*** *wall*) or postposition (Spanish *palabras* ***clave*** 'keywords'). We will refer to all such modifiers as ***adjectivals***.

The most developed electronic lexical databases—WordNet [4] and EuroWordNet [6]—include ca. 20,000 different adjectivals for each language. Standard grammars divide them semantically into two large classes: descriptive and relational adjectivals. Some small classes of descriptive adjectivals are singled out, e.g., colors. As to relational adjectivals, no attempt to classify them semantically has been made in the frame of the mentioned projects.

I. Mel'čuk [3, 7] had introduced syntactical and semantic relationships between words called lexical functions (LFs). Among standard LFs there were proposed those for descriptive adjectivals: **Magn** 'large, intensive', **Bon** 'good', and **Ver** 'as it should

* Work done under partial support of the Mexican Government (CONACyT), CGEPI-IPN, and SNI, Mexico.

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 172–182, 2002.

be.' For example, *high* = **Magn**(*temperature*), *beautiful* = **Bon**(*flower*), *authentic* = **Ver**(*signature*). Combining them with the negation **Anti**, the antonymous functions **AntiMagn**, **AntiBon**, **AntiVer** can be obtained: *bogus* = **AntiVer**(*marriage*). The specific value of a LF depends on its argument, so a given LF determines the meaning and the use of the corresponding adjectival. In other words, a noun to be modified imposes restrictions on selection of its adjectivals in texts.

This suggests that modified nouns could help to develop semantic classification of adjectival modifiers. To put it otherwise, the collocations 'noun—its adjectival modifier' are relevant for this purpose. Some references could be given on compiling collocations of this type [1, 2], but there have been no attempts to classify these collocations further.

If the majority of modificative collocations combine nouns with adjectivals in a free manner based only on their semantics, any classification would be unnecessary in linguistic applications. However, looking through large dictionaries—especially bilingual ones—one can notice that lexicographers understand well that numerous language-dependent modifiers should be given for each modifier-productive noun, to prevent uncommon combinations in both comprehension and composition of texts. To facilitate the selection of relevant combinations from this multiplicity, they should be classified.

In this paper, we develop a few small subtrees of semantic classification based on several modifier-productive English nouns. We intend to answer the following questions:

- Do the classification rubrics constructed for rather arbitrarily selected nouns depend on the selection of these nouns?
- Are these rubrics valid for other nouns and languages? Can they form an integrated classification system?
- Are lexical functions frequent among adjectival modifiers?
- What are advantages and disadvantages of the proposed classification rubrics?

We answer these questions by compiling and structuring several rather large modifier sets. For this, we analyze numerous examples and distribute them by the proposed rubrics. To reduce the total amount of examples given in the text without loosing information on their available number, we preface each list of examples with an (*N*+) sign denoting that this partial collection includes not less than *N* modifiers we have found in general and terminological dictionaries. Any rubric may include both positive and negative estimates of the modificative feature, if this feature is scalable. Note that we do not specify the required word order within the collocations: e.g., the modifiers *average* and *of distinction* at the noun *person* correspond to the collocations *average person* and *person of distinction*.

2 Modifier Set for *Person*

One of the most modifier-productive English nouns, *person*, is the argument of various linguistic predicates, mainly evaluative. At the highest classification level, these modifiers are divided into the rubrics corresponding to *social*, *behavior*, *moral*, *intel-*

lectual, and *outward* (*physical*) features. Each rubric is then subdivided into more fine-grained rubrics, possibly overlapping.

Social features:

- ***Significance*** (33+): *average, beloved, dear, eminent, favorite, futile, great, highly descended, humble, important, in high position, influential, insignificant, inconspicuous...*
- ***Juridical status*** (21+): *accused, aggrieved, artificial, convicted, displaced, exterritorial, free...*
- ***Reputation*** (14+): *disreputable, esteemed, honorable, of excellent qualities, perfect, reputable...*
- ***Publicity*** (12+): *anonymous, familiar, famous, infamous, intimate, known, mysterious, new...*
- ***Exceptionality*** (19+): *average, common, curious, distinctive, exceptional, extraneous, normal...*
- ***Prosperity*** (12+): *impecunious, indigent, poor, rich, prosperous, poverty-stricken, superrich...*
- ***Family status*** (5+): *divorced, family, single, unmarried, widowed...*
- ***Social class*** (8+): *army, civil, country, navy, middle-class, rural, urban, working-class...*

Behavior features:

- ***Sociability*** (37+): *adaptable, affable, agreeable, bashful, boring, candid, cheerful, devious, difficult, direct, disagreeable, dreary, easy, easy-going, frank, forbidding...*
- ***Breeding*** (30+)***:*** *amiable, arrogant, attentive, audacious, backward, boorish, coarse, considerable, courteous, crude, cultural, daring, haughty, ignorant, ill-bred, impolite...*
- ***Enterprise*** (12+): *active, constantly occupied, creative, enterprising, inactive, inventive, sly, versatile...*
- ***Practicality*** (26+): *business-like, careless, economical, efficient, experienced, extravagant, helpless, idle, ill-advised, impractical, miserly, niggardly, practical, self-destructive, sober...*
- ***Consistency*** (16+): *accurate, careless, conscientious, executive, inaccurate, obliging, regular...*
- ***Temper*** (51+): *active, animated, ardent, capricious, energetic, enthusiastic, fearless, fervent, firm, grave, gushing, high-strung, highly strung, hot, hot-headed, hot-tempered, hysterical...*
- ***Character*** (32+): *arrogant, commanding, consequential, credulous, democratic, despotic, dreamy, easy, frivolous, genuine, haughty, imperious, inoffensive, intrepid, iron, lazy, lukewarm...*

Moral features:

- ***Kindness*** (26+): *aggressive, amicable, benevolent, brutal, cruel, friendly, full of kindness, heartless, hard, good, good-natured, kind, malicious, merciful, merciless, mild, nice, ruthless...*

- ***Moral level*** (29+): *corrupted, crafty, dirty, dishonorable, deceitful, disinterested, ethical, false, high-principled, ignoble, immoral, impartial, innocent, insidious, magnanimous...*

Intellectual features:

- ***Education*** (12+)*: advanced, aware, backward, unenlightened, educated, competent, illiterate...*
- ***Endowments*** (11+)*: able, capable, colorless, genial, gifted, talented, of no talent...*
- ***Luck*** (9+): *fortunate, happy, ill fated, ineffectual, lucky, successful, unlucky...*
- ***Intellect in general*** (31+): *banal, brilliant, clever, common, commonplace, cunning, immature, inquisitive, intellectual, intelligent, judicious, mature, meditative, observant, witty...*
- ***Skills*** (7+): *handy, skillful, skilled, unskillful...*

Outward features:

- ***Age*** (11+): *adult, aged, elderly, along in age, middle-aged, of a certain age, old, young...*
- ***Health*** (9+): *ailing, gouty, healthy, insane, rheumatic, sick, sickly, ulcerous, unhealthy...*
- ***Hairs*** (11+): *bald, bearded, curly, curly-headed, dark-haired, fair-haired, hairy...*
- ***Eyes*** (6+): *big-eyed, blue-eyed, brown-eyed, dark-eyed, grey-eyed, narrow-eyed...*
- ***Skin*** (14+): *black, colored, dark-complexioned, dark-skinned, of color, pale, tanned, white...*
- ***Mood*** (19+): *angry, blue, contented, discontented, displeased, dissatisfied, glad, merry, sad...*
- ***Clothing*** (14+): *dowdy, dressed up, frumpy, ragged, shabby, thread-bare, sleek...*
- ***Neatness*** (5+): *neat, sloppy, slovenly, tidy, untidy...*
- ***Outward attractiveness*** (19+): *attractive, beautiful, charming, disgusting, nice, ridiculous, ugly...*
- ***Size*** (6+): *big, broad-shouldered, large, narrow-shouldered, small, tiny...*
- ***Height*** (9+): *high, of medium height, stocky, short, squat, strapping, tall, thick-set...*
- ***Strength*** (8+)*: decrepit, mighty, muscular, powerful, silky, sound, strong, weak...*
- ***Build*** (20+): *anorexic, athletic, delicate, emaciated, frail, gangly, leggy, long-legged, long-limbed, pear-shaped, spindle-legged, of fine physique, rotund, shapely, solidly built...*
- ***Motion*** (4+): *adroit, agile, clumsy, dexterous...*
- ***Nutritional state*** (16+): *bony, burly, corpulent, heavy, fat, lean, plump, obese, well fed...*
- ***Physical deficiency*** (12+): *blind, crippled, cross-eyed, disabled, dumb, handicapped...*

The abovementioned rubrics cover the vast majority of modifiers for *person*. As to the rest of them, the following should be taken into account:

- The closed set of ***quantifying, determining,*** and ***demonstrative*** modifiers (*all, some, many, few, both, another, certain, each*, *every, following, individual, next, other, particular, this, same, single, specific, mentioned*, *former, last, latter*...) is important to specify nouns in communication. It seems impossible to invent a common-language title for this set (maybe ***which***). Note that such modifiers exist for every noun.
- Modifiers can form idiomatic expressions like *nether person* 'the lower part of human body,' whose meaning cannot be deduced from that of the components. For other nouns, idioms can be more numerous.

It is also unclear how to classify, even while replenishing the classification scheme, the modifiers like *dead*, *possible, special*, *catholic-minded*, etc. Assigning a separate rubric to each small group of modifiers makes the classification subtree too unbalanced. Otherwise, we have to assign the remaining items to the rubric ***Miscellanies***, introducing thereby a "dump" for the "nonstandard."

3 Modifier Sets for *Action* and *Look*

The noun ***action*** is a predicate with an abstract meaning, having no arguments except for the subject (who takes an action?). As to the modifiers, this predicate is an argument of at least two evaluative predicates. One of them reflects the correspondence of the action to the norms of the human community and reasonable behavior. The other reflects the method by which action was taken. Within the norm-oriented modifiers, there is an additional subdivision by the types of the norms under consideration:

- Correspondence ***to the*** norms ***of:***
  - ***Moral and law*** (45+): *adequate, abominable, amoral, bad, callous, crafty, cruel, deserved, dishonorable, disgraceful, disgusting, disinterested, evil, fine, fitting, foul, good, heartless...*
  - ***Common way of conduct*** (11+): *boyish, diplomatic, gentlemanly, inexplicable, natural...*
  - ***Reasonable conduct*** (13+): *deliberate, foolish, idiotic, justified, logical, stupid...*
- ***Method, means*** or ***objective*** *of* ***taking*** the action: *belated, collective, covert, double, explosive, impressive, impulsive, joint, mental, overt, physical, preventive...*

The modifiers of the noun ***look*** are classified based on the estimate of the effect produced by the look, as well as on the emotional and physiological state of the look-taking person. All these estimates are made by an outer observer.

- ***Outward effect*** (14+): *bashful, comical, common, gentle, hangdog, idiotic, kingly*, *mischievous...*
- ***Emotional state*** (21+): *brave, cloudy, concerned, contemplative, contemptuous, contented, cowed, disappointed, downcast, ferocious, gentle, gloomy, guilty, injured, sulky...*
- ***Physiological state*** (5+): *haggard, healthy, robust, sickly, unhealthy...*

Outward effect and emotional state could be classified further based on the rubrics introduced for *person*.

4 Modifier Sets for *Corporation* and *Price*

The modifiers for the noun *corporation* are classified in the following way:

- Type of ***business*** (14+): *business, charitable, civil, commercial, industrial, insurance, political...*
- Type of ***ownership*** (19+): *aggregate, closed, government, joint-stock, municipal, private, public, publicly-owned, sole, sponsored, state-owned, stock, trustee...*
- ***Size*** and ***disposition*** (6+): *big, foreign, multinational, small, top, transnational...*
- ***Interconnection*** with other organizations (3+): *affiliated, main, subsidiary...*
- ***Availability*** and ***efficacy*** (6+): *advanced, backward, offending, insolvent, shell, thin...*

The modifiers for the noun *price* are classified rather specifically:

- ***Level*** of the price for:
  - ***buyer*** (22+): *attractive, bargain, dear, exorbitant, attractive, fabulous, fair, fancy, heavy, outrageous, outside, prohibitive, ransom, reasonable, smart, soaring...*
  - ***seller*** (13+): *asked, bed-rock, best, bottom, competitive, fair, honest, nominal, premium...*
  - ***uninvolved observer*** (10+): *buying, discriminative, dump, great, high, low, moderate, pegged...*

These three subsets overlap broadly. Nevertheless, it seems unreasonable to join them. Indeed, only an uninvolved observer can call a price *dump*, while the buyer qualifies it as *low* or *reasonable*, and the seller, as *fair*, *honest*, or *premium*.

- ***Scope*** of the price (51+): *administered, agreed, all-in, all-inclusive, asking, base, blanket, buying, carry-over, cash, ceiling, close, closing, consumer, contract, cost, current, going, export, import, list...*
- ***Variability*** of the price (10+): *determined, dropping, growing, fixed, flat, inflated, oscillating, pegged, reduced, standard...*

5 Modifier Sets for *Coating, Medium,* and *Check*

The noun *coating* is the value of the lexical function S_{res} (= result) of the predicate 'to cover' with four arguments: subject, object, means, and instrument. The corresponding collocations are numerous, since this is a term and a term-generating nucleus broadly used in technology. Among its modifiers, the arguments of *coating* occur in a specific way: the subject is not reflected, but additional circumstantial arguments are added: the goal of the coating and the main features of the coated object. Thus, the modifiers were classified as follows:

- ***Object*** (what is covered?) (4+): *airfield, deck, electrode, roadway...*

- ***Contents*** (with what is something covered?) (20+): *asphalt, brass, copper, enamel, lead, metal, oxide, paint, pigment, rubber, silicon, vitreous, zinc...*
- ***Method*** (by what way is it covered?) (13+): *cathodic, chemical-conversion, dip, sprayed, surfaced...*
- ***Goal*** (what for is it covered?) (19+): *anticorrosive, antifouling, antifreezing, antiradar, antisonar, antistatic, antirust, corrosion-resistant, decorative, diffusion...*
- ***Main property*** reached (10+): *bright, flexible, nonrigid, nonskid, protective...*

For the noun *medium*, we considered only sci-tech modifiers applicable to inanimate objects:

- ***Contents*** (7+): *agar, aqueous, gaseous, dielectric, fluid, liquid, nutrient...*
- ***Main property*** (46+): *absorbing, absorptive, arc-extinguishing, active, aggressive, bacteriological, enhanced, communication, conducting, cooling, corrosive, defined, dense...*
- ***Structure*** (12+): *amorphic, anisotropic, continuous, homogeneous, isotropic, solid...*
- ***Scope*** (5+): *extended, external, infinite, internal, unbounded...*

The noun *check* is a linguistic predicate with the following arguments: the checking subject, the object under check, and the object's parameter to be checked. In technology, the set of its arguments is broader:

- ***Subject*** (who or what is checking?) (2+): *author's, designer's...*
- ***Object*** (what parameter is checked?) (30+): *accuracy, consistency, credit, copy, grammar, health, identity, validity...* A peculiar subgroup is *dope, antidope, drug, antidrug*, with the same meaning.
- ***Method*** by which something is checked (11+): *automatic, comparative, competitive, graphical, logical, marginal, program, programmed, residue, statistical, summation...*
- ***Quality*** with which something is checked (9+): *accurate, all-round, attentive, careful, close, detailed, exhaustive, meticulous, permanent...*
- ***Scope*** of the check (9+): *built-in, current, external, internal, periodic, run-time, spot, static...*

6 Can Proposed Rubrics be Used for Other Words?

Though we have considered in detail only few nouns, they are rather productive in English, giving in total not less than 2,000 modificative collocations. Let us illustrate now that the same rubrics can be used for some other nouns belonging to the modifier-productive elite.

- The nouns *human*, *man, woman, child, boy, lad, chap, fellow, guy, girl, lass* differ from *person* only by sex, age and/or literary style. All rubrics for *person* prove to be directly applicable to these nouns as well. However, this does not mean that specific modifiers are the same: for the same meaning, specific modifiers expressing this meaning can be quite different for different nouns.

- For the noun *glance,* similarly to *look*, modifiers reflect an emotion expressed (*admiring, amused, conspiratorial, cool, disapproving, furious*...) and the manner the glance is cast (*backward, casual, cursory, fleeting, hasty, passing, penetrating, probing, quizzical, searching, sidelong*...).
- For the nouns *agency, organization, enterprise, firm, venture, house* all rubrics of *corporation* are valid.
- For the terminological nouns *index* and *indicator*, the some semantic arguments of *coating* are applicable: a parameter under evaluation (*aerodynamic, acoustical, viscosity*...), an estimate (*low, unprecedented, high*...), and a method (*absolute, aggregative, analytical, basic, main, integral*...).

A tentative analysis of modifiers for Spanish nouns *persona, acción, mirada, compañía, precio, capa, medio*, and *inspección,* approximately corresponding to the considered English nouns, has shown that the English-oriented rubrics are totally applicable to these Spanish analogues. This proves that the rubrics can be used across languages.

7 Can the Proposed Rubrics Form an Integrated System?

The rubrics introduced above form only several slightly overlapping disjoint classification sub-trees, while overall dimensions of the total classification can be clarified on the basis of a more massive analysis. However, we can consider these rubrics as representative fragments of an integrated classification system.

At the upper level, nouns in any language are divided into two large semantic classes: those for living beings (in their vast majority, humans) and for inanimate entities, which can be the names of predicates (actions, processes, properties, etc.) or objects (natural of artificial).

Living beings are characterized in social, behavioral, moral, intellectual, and physical aspects. For them, morals and laws are introduced as usual way of life and behavior. They have an emotional and physical state, opinion, etc.

For inanimate objects, the rubrics reflect at least the following aspects:

- Active semantic valencies of a given predicative noun, namely: subject (agent), object (patient), objectives and way of functioning, materials and tools to be used, structure, temporal and spatial scope of functioning, etc. Some of these roles can be played by the entities considered by traditional grammars as circumstantial.
- Passive semantic valencies, namely: size, efficacy in reaching the objectives and readiness to function (for organizations), important consumption features (for products), etc.
- Passive co-valencies, which can be illustrated by the example of the triple *price*, *buyer*, and *seller*. These notions are co-subordinated to the predicate *selling*, meanwhile the set of modifiers for *price* depends on the two main participants of the selling situation and an indirect participant (uninvolved observer).

Qualitative modifiers characterize the scalable parameters of nouns. The scalable parameters with various or continuous values or only two opposite values can be considered at an axis (scale). For example, nearly all modifiers characterizing *pros-*

perity of a person can be roughly ordered from *poverty-stricken* to *superrich.* Many entities have their own temporal and spatial scope, and then modifiers can characterize the variability of these values within this scope.

The scalable parameters sometimes correspond to the standard lexical functions **Magn**, **Bon**, and **Ver**. However, the considered nouns are so versatile entities that it turns to be difficult, if possible at all, to unambiguously specify these LFs for them. For example, it is quite unclear what properties of *person* (a common word) or of *coat* (a technical term) should be taken as **Magn**, **Bon**, or **Ver.** The properties usually described by LFs are lost in the multiplicity of modifiers. However, the use of these modifiers is rather selective, similarly to the standard LFs. According to Mel'čuk [3], some highly restrictive modifiers can be considered as nonstandard lexical functions, though without further research the use of this term seems a mere slogan.

Obviously, the partial rubrics introduced above do not fit to any sci-tech thesaurus. Indeed, the words appearing in such thesauri are mostly names of artifacts (technical products), with the 'genus–species' relation between them as the basic one, while the relations discussed above are much richer.

On the other hand, many of these rubrics, after corresponding changes, can be identified with those in the most developed natural language thesauri like Roget [5]. In spite of that the latter is already 150 years old, it remains the most popular thesaurus for English, seemingly because of its recurrent modernizations. The basis structure or such thesaurus is also a hierarchy, though it includes the set of abstract notions that characterize such semantic roles as subject, object, goal, method, etc.

The titles of our rubrics have been selected understandable for the potential users of any dictionary or collocation database. For purely practical reasons, common words and word combinations have been taken for them. However, in this way we sometimes can obtain poorly defined, partially synonymous, or even ambiguous titles. For example, while selecting the title *Moral features* within the classification for *person*, some vacillations are inevitable among synonymous variants, say, *moral aspects* or *moral characteristics*. This means that a synonymous group (synset) with the dominant titled *moral features* should be formed (this is easy if the dictionary is implemented as a computer-based system rather than a paper book), to facilitate retrieval of this rubric with a query containing any synonymous option coming first to the user's mind.

Note that if we use totally disambiguated scientific constructs for these titles (they sometimes occur in the Roget thesaurus), the users without any linguistic background would not recognize such artificially constructed terms. Such users might not even understand which of the given terms is broader in meaning. Hence, we suggest small hierarchies forming a classification system of synsets, to facilitate end-user's navigation through the whole title hierarchy.

8 Deficiencies of the Proposed Classification Scheme

Our analysis has shown that splitting the set of modifiers into subgroups with subtitles motivated by their common semantic elements is quite possible. However, the implementation of the idea reveals some its deficiencies:

- The development of a rather complete set of rubrics for all relevant nouns is not easier than compilation of a new Roget-like thesaurus, for any given language. Such a task seems affordable only for several hundred of the most modifier-productive nouns.
- One cannot directly use the classification principles of existing thesauri, since the classification should be carried out on different grounds for different headwords. For example, the rubrics can be related to semantic valencies for some nouns and to classes of properties for the other ones.
- It is usually impossible to select non-intersecting co-subordinated rubrics. A typical example is ***character*** versus ***temper*** for *person*. It is unclear if these notions are overlapping synonyms or two different sub-rubrics of a single rubric (as we have given it in our classification), or sub-rubrics of two different rubrics, ***intellectual features*** and ***physical features***.
- Even when two given rubrics seem non-intersecting, the corresponding groups of modifiers can intersect, since the same versatile modifier could appear in two or more rubrics at the same time. For example, the modifiers *high* and *low* at *prices* are equally used by the buyer, seller, and uninvolved observer.
- Only few nouns permit one-level (flat) classification. Usually, a classification subtree for a noun contains several levels. The variability of levels seems to be inevitable.
- Not for all rubrics an easily understandable title can be proposed. For example, we failed to invent a good "layman" title for the set of quantifying and determining adjectives.
- Sometimes, some modifiers do not fit in any rubric among those already introduced. In such cases, an individual rubric can be allotted to each of them, or they all can be included into a special rubric ***Miscellanies***. The latter strategy poses an obstacle to information retrieval.
- As it was already mentioned, easy search within the joint hierarchy of rubrics requires synonymous titles, since to remember or construct the "correct" title for each rubric is practically impossible for the user.

9 Conclusions

A method of classification of modifier sets based on the semantics of modified nouns is proposed. On this stage, we cannot propose how to automate the classification process; however, even after solving the problem for several hundred of the most modifier-productive nouns, the obtained classification scheme will be of both practical and theoretical value.

From a practical viewpoint, the rubrics of modifiers in dictionaries or collocational databases speed up the search of the desired modifiers, both for comprehension and composition of texts: otherwise the user would have to look through large alphabetically ordered lists of all possible options.

In theoretical perspective, the proposed classification is a contribution to lexical portrayal method. The more numerous the modifiers for the given noun, the more precise the lexicographic portrait based on their classification.

References

1. Benson, M., et al. *The BBI Combinatory Dictionary of English*. John Benjamin, Amsterdam / Philadelphia, 1989.
2. Bolshakov I. A., A. F. Gelbukh. *A Very Large Database of Collocations and Semantic Links.* In: Mokrane Bouzeghoub *et al*. (eds.) *Natural Language Processing and Information Systems.* 5th International Conference on Applications NLDB-2000, Versailles, France, June 2000. Lecture Notes in Computer Science No. 1959, Springer, 2001, p. 103-114.
3. Mel'čuk, I., A. Zholkovsky. *The explanatory combinatorial dictionary*. In: M. Evens (ed.) *Relational models of lexicon*. Cambridge University Press. Cambridge. England, 1988, p. 41-74.
4. Miller, K. J. *Modifiers in WordNet*. In: C. Fellbaum (ed.) *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge/London, 1998, p. 47-67.
5. *Roget's International Thesaurus*. Fifth edition. HarperCollins Publ., 1992.
6. Vossen, P. (ed.). *EuroWordNet General Document*. Vers. 3 final. 2000, www.hum.uva.nl/~ewn.
7. Leo Wanner (ed.) *Lexical Functions in Lexicography and Natural Language Processing*. Studies in Language Companion Series ser.31. John Benjamin Publ., Amsterdam, Philadelphia 1996.

Evaluating a Spelling Support in a Search Engine

Hercules Dalianis

NADA-KTH
Royal Institute of Technology
100 44 Stockholm, Sweden
hercules@nada.kth.se

Abstract. The information in a database is usually accessed using SQL or some other query language, but if one uses a free text retrieval system the retrieval of text based information becomes much easier and user friendly, since one can use natural languages techniques such as automatic spell checking and stemming. The free text retrieval system needs first to index the database but then it is just to search the database.
Normally a search engine does not give any answers to queries when the search words does not exist in the index, therefore we connected a spell checker module into a search engine and evaluated it.
The domain used was the web site of the Swedish National Tax Board (Riksskatteverket, RSV), where the search engine was used between April and Sept 2001. One million queries were made by the public. Of these queries 10 percent were "misspelled" or erroneous and our spell checker corrected around 90 percent of these.

1 Introduction

A search engine is a device that reads through a document collection and indexes each word in each text and then builds an inverted index. The inverted index contains all the words in the text collection and each word points at its corresponding document/s. A search engine then searches the index to find the relevant documents. There is a number of different ranking models, e.g., Boolean ranking, term weight frequency and the vector space models, (van Rijsbergen, 1979).

Using a search engine entering key words does sometime not give any answer either because the word is not in the index or because the user misspelled the word, or because the user did not know the right inflection of the word as written in the index.

The information in a database is usually found using SQL or some other query language, but if one uses a free text retrieval system the retrieval of text based information becomes much easier and user friendly. The free text retrieval system needs first to index the database but then it is just to search the database.

2 Previous Research

One method to increase recall in information retrieval is to use truncation or manual word expansion but the most correct and efficient method is automatic inflection or stemming.

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 183–190, 2002.

An automatic stemmer for Swedish is described in (Carlberger et al 2000), where it is shown that for Swedish, stemming improves the precision and recall by 15 percent and 18 percent respectively[1]. One of the first attempts to use something between stemming and spell checking to increase the recall was described by Pearce & Nicholas (1993). They called it similarity score. One description of a spell checker for English in a search engine can be found in Hodge & Austin (2001), where recall is calculated to 97.5 percent. Both search engines Google (www.google.com) and AltaVista (www.altavista.com) have spell checking systems connected to the search engines though no evaluation is available. (We asked them and they said that their spellchecker was good, but not how good in figures).

Misspelling can be carried out by three reasons either because the user does not know the spelling or because of a typing error or because the user is not completely sure about the spelling

Sometimes the mistake can be different spelling as: *Kyrkskatt / Kyrkoskatt* (Church tax). This is similar to British or American spelling or that the concept can be close but not really right (See Appendix).

According to Knutsson (Knutsson 2001) there are around 2.4 percent spelling errors in texts written by second language users of Swedish.

In an investigation made by Hultman & Westman, (1977) on handwritten essays made by Swedish high school students they had a range from 0.4 percent spelling errors for the best students to 1.2 percent spelling errors for the worst students. According to Kukich (1992) there are around 0.2 percent to 3 percent spelling errors in English texts.

3 Issues

- Swedish word are very often compounds. What happens if the spell checker splits the word in two or more elements, do the user obtain more answers? Does recall increase?
- What is a misspelled word in the document collection?
- If the word occurs at least twice in a small document collection (< 6000 document) is it then a correct word?
- How do we treat “misspelled” words in the document collection?
- Should misspelled words be indexed?
- Are the proposed corrections valid or valuable?
- Should the search engine propose not "correct" spelled word that can be in the retrieved text?
- Can bad suggestions be filtered away?
- One other very important issue is how much a spelling correcting algorithm for Swedish (or any other language) connected to a search engine would improve the precision and recall?

Search scenarios

1) *Search and find*
   This is the normal *successful* case

[1] Precision = number of found relevant documents / total number of found documents
Recall = number of found relevant documents / total number of relevant documents

2) Search and not find and no proposal of other spelling
   This is the other normal *not successful* case in a "normal" search engine.
3) *Search and find and propose other spellings simultaneously.*
   This is possible the best scenario, the user obtains answer and gets also some possible alternative spellings.
4) *Search and not find and propose other spellings.*
   This is the RSV case where one gets some spelling proposals if no answer is found.
5) *The proposed spelling can be spelling mistakes performed in the text collections by an author.*
   This scenario can give wrong impression to the seeker, how should we treat these examples? Maybe scenario 3) can solve this problem. One more possible option is to tell the user the number of hits with this particular spelling so that s/he can get a picture of the frequency of the spellings.
6) *The spell checker splits compounds in two or more words.*
   This is important since Swedes are very often influenced by English compound splitting of words in their Swedish writing.
7) *Spell check of a search word performing dictionary lookup*
   Not a good idea since the proposed spelling might not be at all in the index and using the index as a will also make it possible to make spelling corrections in many languages not only Swedish.

4 Building a Spelling Correction Algorithm

According to Kukich (1992), the four error types, insertion, deletion, substitution and transposition encompasses 80 percent of all spelling errors, therefore we think that connecting a spell checker with a search engine will assist the searcher a lot.

Our spell checking algorithm stems from Stava and Granska (Domeij et al 1994, Carlberger & Kann 1999, 2000, Knutsson 2001) and makes specifically use of the Edit-distance techniques described in (Kukich 1992). Normally spell checkers use string matching techniques to a specific dictionary. In a search engine the spell checker uses the index as lexicon, otherwise it might propose words which are not in the index.

5 Experiment

From April to September 2001 the Swedish National Tax Board (Riksskatteverket, RSV) used Euroseek's search engine Euroseek Remote Indexing with Eurolings (KTH) built-in stemmer and dynamic spell checker.

The RSV site had almost 6000 documents mostly in Swedish containing almost 11 million words. There is a very large discrepancy in the size of each document ranging from very small documents around a half a page to several megabytes.

During the testing period from April to September 2001 we obtained 1 031 700 queries to the search engine of these queries 101 446 (9.8 %) where errors (spelling errors, similar spelling or words not in the index) to which our algorithms proposed

alternative words, of these 9 percent where bad alternatives and to 36 253 (3.5%) of the total queries the system could not give any alternative at all (not in the Appendix).

Of the top 100 spelling errors the system gave 92 percent good suggestions and 40 percent of these contained split compound words, 22 percent was spelling errors and 30 percent was alternative spellings.

Some examples: (word => good suggestion)

40 percent of the good suggestions were compund splitting, see examples below.

utrikestraktamente	=>	traktamente utrikes (allowance abroad)
bilavgifter	=>	avgifter bilar
expertskatt	=>	expert skatt
skattejämkningsblankett	=>	jämkningsblankett skattejämkning

22 percent of the good suggestions were plain spelling errors, see examples below.

engångskatt	=>	engångsskatt
giftemål	=>	giftermål (marriage)
jämnkning	=>	jämkning
skillsmässa	=>	skilsmässa (divorce)
skiljsmässa	=>	skilsmässa
skattejämnkning	=>	skattejämkning

30 percent of the good suggestions were alternative spellings or stemming, see examples below.

engångskatt	=>	engångsskatt
kyrkskatt	=>	kyrkoskatt (church tax)
hempc	=>	hem-pc
rotavdrag	=>	rot-avdrag
arvsskifte	=>	arvsskiftet
pharmasia	=>	pharmacia
skattåterbäring	=>	skatteåterbäring

Regarding compound splitting there is good idea to distinguish two cases, either there is *traktamente* OR *utrikes* OR there is *traktamente* NEAR *utrikes*, the latter example means that the words go together and have some relation.

In the Appendix we can see parts of the logs from RSV, we can also see that from the logs one can build or hard code a synonym list to directly help the user. For example if a common spelling error does not give any result then one can hard code the correct result.

Yes, one should propose "misspelled" words when found in index. The automatic indexing system does not know if a word is misspelled or not, since the system does not have a lexicon to compare with. In the RSV case there is the word *fastighetsskatt* meaning *tax on real estate* which is misspelled to *fastighetskatt*, (with one "s"), meaning *cat on real estate*. That misspelled document contained important information on *tax on real estate* that of course should be found by the search engine and presented.

6 Conclusion and Future Improvements

We found that 10 percent of all queries to a search engine were in some sense erroneous or where not present in the index. Using our spelling correcting algorithm corrected 92 percent of these errors, 40 percent of all suggestion included compound splitting to obtain hits in the document collection, 22 percent were spelling errors and 30 percent were similar spelling or stemming.

To reveal the precision and the recall of the spell checker we would propose a new experiment similar to the stemming experiment performed in Carlberger et al (2001), where we compared precision and recall with and without a stemmer. The new experiment would be to compare precision and recall with and without a spell checker.

The experiment though need to be performed by giving the user implicit written queries so s/he can not see the spelling or give them correct keywords orally.
This paper presents a novel approach and needs to be further evaluated.

Acknowledgement

I would like to thank Johan Carlberger at Euroling AB for the implementation and integration of the spelling correction algorithm with the search engine. I would also like to thank Ola Knutsson, Richard Domeij and Martin Hassel at NADA-KTH for valuable comments on the paper and pointers to the literature.

References

1. Carlberger, J., H. Dalianis, M.Hassel and O. Knutsson. Improving Precision in Information Retrieval for Swedish using Stemming. In the Proceedings of NODALIDA '01 - 13th Nordic Conference on Computational Linguistics, May 21-22, 2001, Uppsala, Sweden, 2001.
2. Carlberger, J. and V. Kann. 2000. Some applications of a statistical tagger for Swedish. Proc. 4:th conference of the International Quantitative Linguistics Association (Qualico-2000), pp. 51-52, August 2000
3. Carlberger, J. and V. Kann. Implementing an efficient part-of-speech tagger. Software Practice and Experience, 29, pp. 815-832, 1999.
4. Domeij, R., J.. Hollman, and Viggo Kann. Detection of spelling errors in Swedish not using a word list en clair. Journal of Quantitative Linguistics 1:195-201, 1994.
5. Hodge, V and J. Austin. A Comparison of a Novel Neural Spell Checker and Standard Spell Checking Algorithms, accepted for Pattern Recognition, Elsevier Publications, 2001
6. Hultman, T. and M, Westman,. Gymnasistsvenska, LiberLäromedel, Lund, 1977.
7. Knutsson, O. Automatisk språkgranskning av svensk text (in Swedish), (Automatic Proofreading of Swedish text), Licentiate Thesis. IPLAB-NADA, Royal Institute of Technology, KTH, Stockholm, 2001.
8. Kukich, K. Techniques for automatically correcting words in text, ACM Computing Surveys. Vol.24, No. 4 (Dec. 1992), pp. 377-439, 1992.
9. Pearce, C and C. Nicholas. Generating a Dynamic Hypertext environment with N-gram Analysis. Proceedings of the second International Conference on Information and knowledge Management, Washington DC, USA, pp. 148-153, 1993.
10. van Rijsbergen, C. J. Information retrieval. Butterworths, 1979

Appendix: 100 Most Common Spelling Errors

100	Most common spelling errors			Spelling suggestion No 1		No 2		No 3		No 4	
No	of spelling errors			Objective ranking		Objective ranking		Objective ranking		Objective ranking	
1204	taxeringskalender		1	taxeringsr	0	taxeringsnämnder	1	taxering	1	taxeringsärendet	1
829	akassa		1	a-kassa	1	kassa	1				
826	fackavgift		1	avgift	1						
338	hempe		1	hem-pe	1	hempå	0				
313	taxeringskalender		1	taxeringsr	0	taxering1	0	taxerings	1	taxeringsärendet	1
311	jämnkning	1		jämkning	1						
290	allmänhetenstermina l		1	allmänheten	1						
243	rsv2252			rsv260	1	rsv2341	1	rsv297	0	rsv2	0
195	taxeringskalendern		1	taxeringsr	0	taxeringsärenden	1	taxerings	0	taxering1	0
182	taxeringskalender			taxeringsärendet	1	taxeringsnämnder	1	Taxeringsärende n	1	taxeringsärende	1
174	personbeskattningsc nheten	1	1	personbeskattning	1	persona	0	persone	0	persons	1
173	rsv379			rsv297	1	rsv354	1	rsv360	1	rsv409	1
167	vägskatt		1	skatt	1	värnskatt	1	vägskäl	0	vägrat	0
159	skillsmässa	1		skilsmässa	1						
150	rotavdrag		1	rot-avdrag	1						
141	skattejämnkning	1		skattejämkning	1						
140	fackavgifter		1	fallavgifter	0	avgifter	1				
137	trygghansa			trygg-hansa	1						
134	komunalskatt	1		kommunalskatt	1						
132	personaldator			persondator	1	personas	0				
131	giftemål	1		giftermål	1						
130	engångsskattetabell		1	engångsskatter	1	engångsskatt	1	skattetabell	1		
126	boupptäckning	1		bouppteckning	1						
121	rsv418			rsv408	1						
121	hinderprövning	1		hindersprövning	1						
121	epsilon		0	epiros	0						
115	personaldatorer		1	persondatorer	1	personal	1				
110	rsv285			rsv297	1	rsv260	1	rsv2	1	rsv08a	1
110	kyrkskatt			kyrkoskatt	1						
109	pergo		0	perso	0						
108	transaktionskod		1	transaktion	1	transaktionskostnad	1	transaktionslag	1	transaktionskedja	1
107	a-kasseavdrag		1	a-kasseavgift	1	a-skatteavdrag	1	a-kassa	1	avdrag	1
105	engångskatt	1		engångsskatt	1						
101	skattetabel	1		skattetabell	1						
101	födelsebevis		1	födelsen	1	födelser	1	födelse	1	föder	1
99	rsv4620			rsv4820	1						
98	skiljsmässa	1		skilsmässa	1						
98	altavista		0	avista	0	altan	0	alaista	0	altamira	0
96	passansökan		1	ansökan	1	passar	0	passad	0	passas	0
95	studsvik		0	studie	0						
94	förmånsskatt		1	förmånsrätt	1	förmåns-rätt	1	förmåns	1	förmånstaket	1
93	pharmasia			pharmacia	1						
93	akasscavgift		1	a-kasscavgift	1						
89	vårdförbundet		1	vårdförmåner	1	förbundet	1	vårda	0	vårds	0
86	rsv4350			rsv4314	1	rsv4820	1	rsv2310	1	rsv330	1
86	dilligentia	1		diligentia	1						
84	skattebevis		1	skattek	0	skattei	0	skattem	0	skatten	1
84	rsv4720			rsv4820	1						
84	assi-domän			assidomän	1						
83	rsv426			rsv408	1	rsv409	1	rsv4820	1	rsfs26	0
83	astrazenica			astrazeneca	1						
81	expertskatt		1	expert	1	skatt	1				
81	akassan			a-kassan	1	kassan	1				

80	sif		0	sig	0	sid	0	sir	0	siv	0
79	utrikestraktamente		1	traktamente	1	utrikes	1				
77	utlandsflytt		1	utlandsf	0	utland	1				
77	rsv4301			rsv4801	1	rsv4314	1	rsv2310	1		
76	firmabil			firmav	0	firman	1				
75	traktamentesbelopp	1		traktamentsbelop p	1						
75	rsv426			rsv325	1	rsv408	1	rsv409	1	rsv323	1
75	rsv425			rsv408	1	rsv409	1	rsv4820	1	rsv2	1
75	arvsskifte			arvsskiftet	1	arvskifte	1				
73	pensionskod			pensionskost	0	pensionsfond	1	pensionsskydd	0		
72	resersättning	1		reseersättning	1						
71	diligenta	1		diligentia	1						
70	rsv379			rsv309	1	rsv297	1				
69	addressändring	1		adressändring	1						
68	körersättning		1	ersättning	1	körer					
67	utlandstrakt		0	utlandsf	0	utlandstrafik	0				
67	statligskatt		1	statliga	1	statligt	1	statlig		stata	0
67	bilförmånsuträkning			bilförmånsberäkn ing	1						
65	rsv308			rsv309	1	rsv408	1				
64	telcca	1		telcsa	1						
64	ocrnummer			ocr-nummer	1						
64	lunarstorm		1	storm	1						
64	bilkalkyl			kalkyl	1	bilkö	0				
63	bilavgifter		1	avgifter	1	bilutgifter	1	bilar	1	bilregister	1
62	taxeringskalendern		1	taxeringsr	0	taxerings	1	taxeringsärend en	1	taxering1	0
62	enskildfirma		1	enskilda	1	enskilde	1				
62	avdrag		1	fackavgift	1	avgift	1				
61	traktemente	1		traktamente	1						
61	skatte-och		1	avgiftsanmälan	1	skatt1	0	skatte	1	skatt8	0
61	rsv306			rsv360	1						
61	fskattsedel		1	f-skattsedel	1	skattsedel	1				
61	bilförmånberäkning	1		bilförmånsberäkn ing	1						
59	bilavg			biland	0	bilagd	0	bilage	0	bilars	1
58	evreka		0	everöd	0	evernia	0	every	0	evert	0
57	stadslåneränta	1		statslåneränta	1						
57	skattejämkningsblan kett		1	jämkningsblanket t	1	skattejämkning	1	skattei	0	skattek	0
57	lärarförbundet			läkarförbund	1	förbundet	1	lärarv	0	lärare	1
57	arbetsgivareavgift	1	1	arbetsgivaravgift	1	arbetsgivar-avgift	1				
56	k4blankett		1	k4-blankett	1	k-blankett	1				
56	bokmoms		0	bokförs	0	bokens	0	bomhus	0	bokform	0
55	överskjutandeskatt		1	överskjutande	1	överskjutan	0	översko	0	överst	0
55	måltidsavdrag		1	måltid	1	avdrag	1				
54	rsv349			rsv354	1	rsv409	1	rsv330	1	rsv360	1
53	ludvikamoms		1	ludvika	1	ludvig	0				
52	förmånsvärdesberäk ning		1	förmånsvärden	1	förmånsvärdet	1	förmånsvärde	1	beräkning	1
51	skattåterbäring	1		skatteåterbäring	1						
51	rsv355			rsv354	1						
12577	Top 100 spelling errors	22	48	Good suggestions	82	Good suggestions	43	Good suggestions	21	Good suggestions	19
			8	Bad suggestions	18	Bad suggestions	16	Bad suggestions	16	Bad suggestions	15

	40	No of compound splittings	6835 (of total top 100)
	22	No of spelling correctings	2325 (of total top 100)
	30	No of similar spellings, similar words, other inflections, etc	
	8	No of bad suggestions	
	100	Of the top spelling errors	

Total Good suggestions	165	61%	Percentage Good suggestions when at least one god suggestion
Total Bad suggestions	65		
Total queries		1031700	
Total spelling errors		101446	
God suggestions		92%	(calculated on top 100 spelling errors)

Opening Statistical Translation Engines to Terminological Resources

Philippe Langlais

Université de Montréal
Département d'Informatique et de Recherche Opérationnelle (DIRO)
C.P. 6128, succursale Centre-ville, Montréal (Québec), Canada
felipe@iro.umontreal.ca
http://www.iro.umontreal.ca/~felipe

Abstract. The past decade has witnessed exciting work in the field of Statistical Machine Translation (SMT). However, accurate evaluation of its potential in real-life contexts is still a questionable issue.
In this study, we investigate the behavior of a SMT engine faced with a corpus far different from the one it has been trained on. We show that terminological databases are obvious resources that should be used to boost the performance of a statistical engine. We propose and evaluate a way of integrating terminology into a SMT engine which yields a significant reduction in word error rate.

1 Introduction

SMT mainly became known to the linguistic community as a result of the seminal work of Brown et al. [3]. Since then, many researchers have invested effort into designing better models than the ones proposed in the aforementioned article and several new exciting ways have been suggested to attack the problem.

For instance, Vogel et al. [19] proposed to overcome the independence assumption made by IBM models by introducing order-1 Hidden Markov alignment models. In the same spirit, Toutanova et al. [17] introduced an HMM alignment model which uses part-of-speech tags and which compares well to an IBM-4 model. Yamada and Knight [21] suggested an interesting model in which the noisy-channel takes as input a parsed sentence rather than simple words. Marcu et Wong [12] proposed a joint probability model where both phrase- and word- pairs are learned at the same time. Och and Ney [16] also described a flexible maximum entropy model which can accomodate any selected feature in a pair of source-target training sentences.

Other efforts have been made to take advantage of SMT in different applications. For instance, SMT has been embedded in an interactive translation prototype (called TransType) whose task is to predict in real time what a translator (translating a given source sentence) will type next [6]. SMT has also been used in the context of information retrieval (IR): first as a way to perform crosslingual IR [13]; second, as methodology in itself to perform monolingual IR [2], where SMT is used to model how a document can be "translated" into a query.

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 191–202, 2002.

Although there exists a vast amount of publications on MT evaluation, it is not obvious to tell how well statistical translation can do on a given task. We know that on a specific task of spoken language translation, Wang [20] provided evidence that SMT compared favorably to a symbolic translation system; but as mentioned by the author, the comparison was not totally fair. In any case, it is well known that evaluating a translation system (statistical or not) is a very difficult problem [1]. It is worth mentioning that the NIST'2002 machine translation evaluation will be of a great help in providing a clear picture of the state-of-the-art of machine translation, whatever the approach advocated[1].

Evaluation campaigns such as the aforementioned NIST one are not necessarily warrant of the adequacy of (S)MT in a real translation environment. We prefer not to commit ourselves to defining what a real translation task is; instead, we adopt the conservative point of view that a viable general-purpose translation engine (statistical or not) is one that copes with texts that may be very different in nature from those used at training time. This fairly general definition suggests that adaptativity is a cornerstone of a successful general-purpose SMT engine. Curiously enough, we are not aware of much work on adaptative SMT, despite the tremendous amount of work done on adaptative statistical language modeling.

We must stress at this point that the scope of this study is limited to general-purpose translation engines, despite the fact that the most sucessful systems in the history of machine translation are restricted-domain MT systems; among them, the METEO system [5] developed at the Université de Montréal and which is used to translate Canadian weather reports since 1977.

In this paper, we propose to evaluate how a statistical engine behaves when translating a very domain specific text, that is far different from the corpus used to trained both our translation and language models. We first describe in section 2 our translation engine. In section 3, we quantify and analyse the performance drop of an SMT engine trained on a broad-based corpus (the Hansard) when translating a domain specific text (in this study, a manual for military snipers). We then propose in section 4 a simple but natural way of improving a broad SMT engine; that is, opening the engine to available terminological resources. In section 5, we report on the improvement we observed by implementing our proposed approach. Finally, in section 6 we discuss other approaches we feel can lead to more robust translation.

2 Our Statistical Engine

2.1 Statistical Machine Translation in Brief

In this study, we built an SMT engine designed to translate from French to English, following the noisy-channel paradigm first described by [3], and which

[1] For more see: `www.nist.gov/speech/tests/mt/doc/2002-MT-EvalPlan-v1.3.pdf`. The results of this evaluation campaign will be discussed at a dedicated meeting in Santa Monica, July 2002.

relies on equation 1, where $e_1^{\hat{I}}$ stands for the sequence of $\hat{I}$ target words (here English words) to be found, given the source sentence (here a French one) of J words f_1^J:

$$\hat{e}_1^{\hat{I}} = \underset{I, e_1^I}{\text{argmax}} \underbrace{P(e_1^I)}_{\text{language}} \cdot \underbrace{P(f_1^J | e_1^I)}_{\text{translation}} \tag{1}$$

Developping a SMT engine in this paradigm encompass two major steps:

- a modelling stage, which consists of designing models that (hopefully) capture enough structural dependencies of the translation process and whose parameters can be estimated automatically from a bilingual corpus;
- a decoding stage, which consists in performing as efficiently as possible the argmax operation of equation 1.

2.2 The Statistical Models

To train our statistical models, we assembled a bitext composed of 1 639 250 pairs of sentences that have been automatically aligned at the sentence level. In this experiment, every token has been converted into lowercase before training.

The Language Model. The language model we used is an interpolated trigram [7] we trained on the English sentences of our bitext. The perplexity of the resulting model is fairly low – 65 –, which actually reflects the fact that this corpus contains many fixed expressions (*e.g* `pursuant to standing order`). Formally, the interpolated trigram model takes the form of a linear combination of a trigram, a bigram, a unigram and a 0-gram models:

$$\begin{aligned} p(w|h) = {} & \lambda_3(w'', w') p_3(w|w'', w') + \lambda_2(w'', w') p_2(w|w') + \\ & \lambda_1(w'', w') p_1(w) + \lambda_0(w'', w') p_0(w), \end{aligned} \tag{2}$$

where $p_i()$ is the empirical (relative frequency) distribution over i-grams (for $i > 0$) and $p_0(w)$ is a uniform distribution over all words in the vocabulary. λ_i is the weigth associated to $p_i()$ when w'', w' are the last two words in h, under the constraint that $\sum_{i=0}^{3} \lambda_i(w'', w') = 1$. A standard pratice is to consider the weigthing coefficients to depend only on the frequency of the conditioning bigram. Values for the weigths are estimated using the EM algorithm on a heldout corpus.

The Translation Model. The inverted translation model we used in equation 1 is an IBM2-like model [3] which states an independance over the French tokens:

$$p(f_1^J | e_1^I) = \underbrace{p(J|I)}_{\text{length model}} \prod_{j=1}^{J} p(f_j | e_1^I) \tag{3}$$

The probability that a French word f_j appears in the translation in position j is calculated by the weighted contribution of each English word (following the standard practice [3], an English word is added at position 0 to account for French words without clear association in the English sentence):

$$p(f_j|e_1^I) = \sum_{i=0}^{I} \underbrace{p(i|j,J)}_{\text{alignment}} \underbrace{p(f_j|e_i)}_{\text{transfert}} \tag{4}$$

We assumed a normal distribution for the length model of equation 3. Ten iterations of IBM1-training[2] were run first (reducing the perplexity of the training corpus from 7776 to 90), followed by ten iterations of IBM2-training (yielding a final perplexity of 54).

2.3 The Search Algorithm

We extended to a trigram language model the decoder described by Niessen et al. [14]. The basic idea of this search algorithm is to expand hypotheses along the positions of the target string while progressively covering the source one. The search may be organized by the following recursion where $Q_I(c,i,j,e)$ is the probability of the best partial path ending at position i in e_1^i and j in f_1^j, where the last word e_i is e and where exactly c source positions have been covered:

$$\begin{array}{ll}
\hat{e}_1^{\hat{I}} & = \max_I\{p(J|I).\max_{j,e} Q_I(J,I,j,e)\} \\
Q_I(c,i,j,e) & = \max\{Q_I^S(c,i,j,e), Q_I^n(c,i,j,e)\} \\
Q_I^n(c,i,j,e) & = \max_{l>0} \prod_{\tilde{j}=j-l+1}^{j}\{p(i|\tilde{j},J).p(f_{\tilde{j}}|e_i)\}\times \\
 & \qquad \max_{e'}\{p(e|e'back(e')). \\
 & \qquad\qquad \max_{j'}[Q_I(c-l,i-1,j',e').v(j,l)]\} \\
Q_I^S(c,i,j,e) & = \max_{e'}\{p(e|e'back(e')).Q_I(c,i-1,j',e')\}
\end{array} \tag{5}$$

where $back(e)$ is an operator that retrieves the best target word that resulted from the hypothesis being extended[3], $v(j,l)$ is a binary control function that guarantees that the source positions within $[j, j+l[$ are free[4]; and $p(J|I)$ is the so-called length model. We assume here that the length (counted in words) of French sentences which are translations of an English sentence of a given length are normally distributed.

Though, a partial hypothesis is fully determined by four parameters: the source and target positions, the coverage and the target word found at the target position. Therefore, the search space can be represented as a 4-dimension table, each item in this table containing backtracking information and the hypothesis score.

[2] An IBM1 model may be seen as a special case of an IBM2 model, where the alignment probabilities are fixed at $p(i|j,I) = 1/I + 1$.

[3] For instance if we extend the hypothesis *the prime* by the word *e=minister*, then e' will be *prime* and $back(e')$ will be *the*.

[4] In which case, v returns 1.

Table 1. Active vocabulary $\mathcal{A}$ for the source sentence *la loi anti-tabac*. Words are listed in alphabetical order. Words in bold are those used to built the best translation found: *the anti-smoking legislation*.

a	$\lvert\mathcal{A}\rvert$	$\mathcal{A}$
5	15	act **anti-smoking** anti-tobacco bill bills house in law **legislation** no-smoking non-smoking of **the** to tobacco
7	18	act **anti-smoking** anti-tobacco bill bills clause house in it law **legislation** no-smoking non-smoking of on **the** to tobacco
10	24	act **anti-smoking** anti-tobacco bill bills clause house in it law **legislation** no-smoking non-smoking of on **the** to tobacco
20	40	's . a act **anti-smoking** anti-tobacco at be bill bills by c-21 c-22 clause for health house in is it justice law **legislation** legislative madam most motion no-smoking non-smoking of on passed piece position situation statute **the** this to tobacco

We know that better alignment models have been proposed and extensively compared [15]. We must however point out that the performance we observed on the HANSARD corpus (see Section 3 for the description of this corpus and performance figures) with our engine is comparable to the rates published elsewhere on the same kind of corpus. In any case, our goal in this study is to compare the behavior of a SMT engine in both friendly and adverse situations. In our view, the present SMT engine is suitable for such a comparative study.

2.4 Tuning the Decoder

Several tunings have been made on the decoder in order to reduce its computations without detrimentally affecting the quality of its output. The first thing we do when the decoder receives a sentence is to compute what we call an *active vocabulary*; that is, a collection of words which will likely occur in the translation. This is done by ranking for each source word the target words according to the joint probability $\text{argmax}_e\{p(e).p(f|e)\}$, where $p(e)$ is given by a unigram target language model, and $p(f|e)$ is given by the transfer probabilities of our inverted translation model) and keeping for each source word at most a target words.

Increasing a raises the coverage of the active vocabulary, but also slows down the translation process and augments the risk of admitting a word that has nothing to do with the translation (see Table 1). We have conducted experiments with various a-values, and found that an a-value of 10 offers a good compromise.

Last, we also prune the space to make the search tractable. This is done with relative filtering as well as absolute thresholding. The details of all the filtering strategies we implemented are however not relevant to the present study.

3 Performances of Our SMT Engine

3.1 Test Corpora

In this section we provide a comparison of the translation performances we measured on two corpora. The first one (namely, the HANSARD) is a collection of

Table 2. Main characteristics of our test corpora and global performance of our statistical translator without any adjustments. $|length|$ reports the average length (counted in words) of the source sentences and the standard deviation; *nbs* is the number of sentences in the corpus.

corpus	*nbs*	$\|length\|$	SER	WER
HANSARD	1038	$\langle 16.2, 7.8 \rangle$	95.6	59.6
SNIPER	203	$\langle 20.8, 6.8 \rangle$	100	74.6

sentences extracted from a part of the Hansard corpus we did not use for training. In particular, we did not use any specific strategy to select these sentences so that they would be closely related to the one we used for training.

Our second corpus we gathered (here called SNIPER) is an excerpt of an army manual on sniper training and deployment that was used in an other study [10]. This corpus is highly specific to the military domain and would certainly prove difficult to any translation engine not specifically tuned to such material.

3.2 Overall Performance

In this section, we evaluate the performance of our engine in terms of sentence- and word- error rates according to an oracle translation[5]. The first rate is the percentage of sentences for which the decoder found the exact translation (that is, the one of our oracle), and the word error rate is computed by a Levenstein distance (counting the same penalty for both *insertion*, *deletion* and *substitution* edition operations). We realize that these measures alone are not sufficient for a serious evaluation, but we were reluctant in this experiment to resort to manual judgments, following for instance the protocol described in [20]. Actually a quick look at the degradation in performance we observed on SNIPER is so clear that we feel these two rates are informative enough !

Table 2 summarizes the performance rates we measured. The WER is close to 60% for the HANSARD corpus and close to 74% on SNIPER; source sentences in the latter corpus being slightly longer on average (21 words). Note any single sentence was found identical to the gold standard translation on the SNIPER corpus.

These rates serve as a very rough evaluation criterion, for it is not really obvious what a 60 % word error rate means precisely. Table 3 helps to understand these figures[6].

The first example illustrates a typical bias of a translation evaluated against a single reference: the system output is arguably as good as the oracle one (actually it is even closer to the original sentence than the oracle translation is), but was credited a WER of 11% due to the insertion of the word `emerging`. The other

[5] Both corpora have been published in both French and English, and we took the English part as the gold standard.

[6] The full output of our translation sessions is available at the internet address: `www.iro.ummontreal.ca/~felipe/ResearchOutput/NLDB2002`

examples illustrate the fact that a severe WER penalty may not fully warranted in case of a translation that is only partially erroneous (this fact would have to be confirmed by human judgments). In any case, we believe that an average WER figure computed over a corpus allows us to capture general tendencies (a definite decrease in WER is certainly indicative of better translations).

Table 3. Examples of translations made on HANSARD at various WER. SRC is the sentence to be translated, REF is the oracle translation and CAN is the candidate translation produced by the translation engine.

SRC	cependant , il y a ici deux problèmes qui apparaissent .	
REF	however , there are two problems here .	
CAN	however , there are two problems emerging here .	11%
SRC	nous sommes fiers de ces habitants de london et d' autres canadiens qui consacrent leur temps et leur énergie à bâtir un monde meilleur .	
REF	we are proud of these londoners and of other canadians who devote their time and energies to improving our world .	
CAN	we are proud of these people of london and other people spend their time and energy to build a better world	50%
SRC	le mois de la nutrition	
REF	nutrition month	
CAN	in the month of nutrition	80 %

3.3 Analyzing the Performance Drop

The poor performance observed on the SNIPER text is mainly due to two reasons: the presence of out of vocabulary (OOV) words and the incorrect translations of terminological units.

In the SNIPER corpus, 3.5% of the source tokens and 6.5% of the target ones are unknown to the statistical models. This means rougly that the OOV rate is 10 times larger in the SNIPER corpus. 44% of the source sentences and 77% of the target sentences contain at least one unknown word. In the HANSARD text, the OOV rates are much lower: around 0.5% of the source and target tokens are unknown and close to 5% of the source and target sentences contain at least one OOV words. These OOV rates have a clear impact on the coverage of our active vocabulary. On the SNIPER text, 72% of the oracle tokens are in the active vocabulary (only 0.5% of the target sentences are fully covered); whilst on HANSARD, 86% of the oracle's tokens are covered (24% of the target sentences are fully covered).

It is more difficult to quantify the impact of the presence of terminological units on the quality of the translation. This would normally require identifying all the terms and their translations in our text. Furthermore, many words within terminological units are not even known by the statistical models. An indirect measurement of this impact can be seen in section 5 where we demonstrate that introducing terminological entries improves the performances. Table 4 shows some examples of mistranslated terminological units.

Table 4. Examples of mistranslated terminological entries of the SNIPER corpus.

source term	oracle	translation
âme	bore	heart
huile polyvalente	general purpose oil	oil polyvalente
chambre	chamber	house of common
tireur d' élite	sniper	issuer of elite
la longueur de la crosse	butt length	the length of the crosse

4 Integrating Non-probabilistic Terminological Resources

There are several ways to improve the situation. First, it may be the case that a domain-specific corpus is available allowing to train specialized models that we can combine with the general ones. Second, we could try to design an adaptative translation engine. For instance, a cache-based language model could be used instead of our static trigram model. The design of an adaptative translation model is however a more speculative enterprise. It would at least require a fairly precise location of errors in previously translated sentences; and we know from the ARCADE campaign on bilingual alignments, that accurate word alignments are difficult to obtain [18].

There is a third option, which involves taking advantage of existing terminological resources, like *Termium*[7], for instance. After all, one of a translator's first tasks is often terminological research; and many translation companies employ specialized terminologists. Therefore it makes sense from the user's point of view to open a SMT engine to existing terminological lexicons.

Using terminological resources to improve the quality of an automatic translation engine is not at all a new idea. However, we know of very few studies that actually investigated this avenue in the field of statistical machine translation. Among them, [4] have proposed a way to exploit bilingual dictionnaries at training time. However, this does not solve the problem if the trained engine is to be used to translate corpora that contain a specific terminology not seen at training time. Therefore, we feel that an online approach to the problem is still worth the effort.

One problem with terminological lexicons is that we can not expect them to come with attached probabilities, and therefore, their online integration into the decoder (see equation 5) is hopeless. We therefore propose to view any entry in such a lexicon as a constraint that may be employed to reduce the search space. For instance, knowing that `sniper` is a sanctioned translation of *tireur d'élite*, we may require (or reinforce the fact) that current hypotheses should associate the target word `sniper` with the three French words. Only the position of the target translation is unknown.

We had to slightly modify our implementation of equation 5 in order to: 1) forbid a given word e_i from being associated with a word belonging to a source terminological unit, if it is not sanctioned by the lexicon; and 2) add at any target

[7] See `http://www.termium.com/site/`.

position an hypothesis linking a target lexicon entry to its source counterpart. Whether these hypotheses will survive intact will depend on constraints imposed by the maximum operation over the full translation. The score associated with a target entry $e_i^{i'}$ when linked to its source counterpart $f_j^{j'}$ in the latter case is given by:

$$\sum_{k\in[i,i']} \log p(e_k|e_{k-2}e_{k-1}) + \max_{l\in[j,j']} \log(p(k|l,J))$$

The rationale behind this equation is that both the language and the alignment models have information that can help to decide the appropriateness of an extension: the former knows how likely it is that a word (known or not) will follow the current history[8]; and the latter knows to some extent where the target unit should be. We hope in absence of a better mechanism (a cache-model should be worth a try) that it will be sufficient to determine the final position of the target unit.

5 Results

We considered three terminological lexicons whose characteristics are summarized in Table 6; they essentially differ in terms of number of entries and therefore coverage of the text to translate.

The first lexicon (namely `sniper-1`) contains the 33 entries used in the study of terminological consistency checking described in [10]. The second and third lexicons (namely `sniper-2` and `sniper-3`) contain those entries plus other ones added manually after an incremental inspection of the SNIPER corpus.

As can be observed from Table 6, introducing terminological lexicons into the translation engine does improve performance, measured in terms of WER, and this even with lexicons that cover only a small part of the text to translate. With the narrow coverage lexicon, we observe an absolute reduction of 7%, and a reduction of 10% with the broader lexicon `sniper-3`. Table 5 provides two examples of translation outputs, with and without the help of terminological units (TU). However, a systematic inspection of the outputs produced with TU reveals that the translations are usually less faithful to the source text than the translations produced for the HANSARD text. OOV words are still a problem.

6 Discussion

In this study, we have shown that translating texts in specific domains with a general purpose statistical engine is difficult. This suggests implementing an adaptative strategy. Among the possible scenarios, we have shown that opening the engine to terminological resources is a natural way of softening the decoder.

In the same vein, Marcu [11] investigated how to combine Example Based Machine Translation (EBMT) and SMT approaches. The author automatically

[8] Our trigram model has been trained to provide parameters such as $p(UNK|ab)$.

Table 5. Two examples of translation WITH and WITHOUT a terminological lexicon; TU appear in bold.

SRC	le **tireur d' élite** voit simultanément les **fils croisés** et l' image (l' objectif) .
REF	the **sniper** sees the **crosshairs** and the image - target - at the same time .
WITHOUT	*the gunman being same son sit and picture of the hon. members : agreed .*
WITH	*the **sniper** simultaneously see the **crosshairs** and the image (objective .)*
SRC	contrôle de la **détente** .
REF	exercising **trigger** control .
WITHOUT	*the control of détente .*
WITH	*control of the **trigger** .*

Table 6. Translation performance with different terminological lexicons. *nb* is the number of entries in the lexicon and *coverage* reports the number of different source entries from the lexicon belonging to the text to translate and the total number of their occurrences.

lexicon	nb	coverage	SER	WER
`sniper-1`	33	20/247	99	67.4
`sniper-2`	59	47/299	98	66.2
`sniper-3`	146	132/456	98	64.3

derived from the Hansard corpus what he calls a translation memory: actually a collection of pairs of source and target word sequences that are in a translation relation according to the viterbi alignment run with an IBM4 model that was also trained on the Hansard corpus. This collection of phrases was then merged with a greedy statistical decoder to improve the overall performance of the system.

What this study suggests is that translation memories collected from a given corpus can improve the performance of a statistical engine trained on the same corpus, which in itself is an interesting result. A very similar study is described in [8], in the framework of TRANSTYPE, but with much more weaker results. Besides the different metrics the authors used, the difference of performance in these two studies may be explained by the nature of the test corpora used. The test corpus in the latter study was more representative of a real translation task, while the test corpus taken in the former work was a set of around 500 French sentences of no more than 10 words.

Our present study is close in spirit to these two last, except that we do not attack the problem of automatically acquiring bilingual lexicons, but instead consider it a part of the translator's task to provide such lexicons. Actually, we feel this may be one of the only ways a user has of retaining some control over the engine's output, a fact that professional translators seem to appreciate [9].

As a final remark, we want to stress that we see the present study as a first step toward a unification between EBMT and SMT, and in this respect we agree with [11]. Of course, EBMT can offer much more than just a simple list of equivalences, like the ones we used in this study. The basic approach we describe here still holds, however, as long as we can extend the notion of *constraint* used in this study to include non-consecutive sequences of words. This is a problem we we plan to investigate in future research.

Acknowledgment

We would like to thank Elliott Macklovitch and George Foster for the fruitful comments they made on this work. The models we used in this study have been trained by a package written by George Foster.

References

1. Doug Arnold, Louisa Sadler, and R. Lee Humphreys. Evaluation: An assessment. *Machine Translation*, 8:1-24 (1993)
2. Adam Berger and John Lafferty. Information Retrieval as Statistical Translation In *proceedings of the 22nd Conference on Research and Development in Information Retrieval, (SIGIR)*, Berkeley, (1999) 222–229.
3. Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2) (1993) 263–311.
4. Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, Meredith J. Goldsmith, Jan Hajic, Robert L. Mercer, and Surya Mohanty. But Dictionaries are Data too In *Proceedings of Human Language Technology (HLT)*, Princeton, NJ, March (1993) 202–205
5. M. Chevalier, J. Dansereau, and G. Poulin. Taum-meteo: description du système. Technical report, TAUM, Université de Montréal (1978)
6. George Foster, and Pierre Isabelle, and Pierre Plamondon. Target-Text Mediated Interactive Machine Translation In *Machine Translation*, vol 12 (1997) 175–194
7. F. Jelinek, and R.L. Mercer Interpolated estimation of Markov source parameters from sparse data. In E.S. Gelsema and L.N. Kanal, editors, *Pattern Recognition in Practice*, North-Holland, Amsterdam (1980)
8. Philippe Langlais, George Foster, and Guy Lapalme. Unit completion for a computer-aided translation typing system. In *Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP)*, Seattle, Washington, May (2000) 135–141
9. Philippe Langlais, George Foster, and Guy Lapalme. Integrating bilingual lexicons in a probabilistic translation assistant. In *Proceedings of the 8th Machine Translation Summit*, Santiago de Compostela, Galicia, Spain, September (2001) 197–202
10. Elliott Macklovitch. Can terminological consistency be validated automatically ? Technical report, CITI/RALI, Montréal, Canada (1995)
11. Daniel Marcu. Towards a unified approach to memory- and statistical-based machine translation. In *Proceedings of the 39th Annual Meeting of the ACL*, Toulouse, France (2001) 378–385
12. Daniel Marcu and William Wong. A Phrase-Based, Joint Probability Model for Statistical Machine Translation In *proceedings of the 7th Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 6-7 july, Philadelphia, PS (2002) 133–139
13. J.Y. Nie, P. Isabelle, P. Plamondon and G. Foster. Using a probabilistic translation model for cross-language information retrieval In *COLING-ACL, Sixth Workshop on Very Large Corpora*, Montreal, August (1998) 18–27

14. Sonja Niessen, Stephan Vogel, Hermann Ney, and Christoph Tillmann. A dp based search algorithm for statistical machine translation. In *Proceedings of the 36th Annual Meeting of the ACL and the 17th COLING*, Montréal, Canada, August (1998) 960–966
15. Franz Joseph Och and Hermann Ney. A comparison of alignement models for statistical machine translation. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, Saarbrucken, Luxembourg, Nancy, August (2000) 1086–1090
16. Franz Josef Och and Hermann Ney Discriminative Training and Maximum Emtropy Models for Statistical Machine Translation In *Proceedings of the 40th Annual Meeting of the ACL*, 7-12 july, Philadelphia, PA (2002), 295–302
17. Kristina Toutanov, H. Toga Ilhan, and Christopher D. Manning Extensions to HMM-based Statistical Word Alignment Models In *EMNLP'2002*, Philadelphia, 6-7 july, PA (2002), 87–94
18. Jean Véronis and Philippe Langlais. *Evaluation of parallel text alignment systems: The ARCADE project* Parallel Text Processing, Kluwer, volume 13, chapter 19 (2000) 369–388.
19. Stephan Vogel, Hermann Ney, and Christoph Tillmann. Hmm-based word alignement in statistical translation. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, Copenhagen, Denmark, August (1996) 836–841,
20. Ye-Yi Wang. *Grammar Inference and Statistical Machine Translation.* Ph.D. thesis, CMU-LTI, Carnegie Mellon University (1998)
21. Kenji Yamada and Kevin Knight: A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the ACL* Toulouse, France (2001) 531–538

User-Centred Ontology Learning for Knowledge Management

Christopher Brewster, Fabio Ciravegna, and Yorick Wilks

Department of Computer Science, University of Sheffield
Regent Court, 211 Portobello Street, Sheffield, UK
{C.Brewster,F.Ciravegna,Y.Wilks}@dcs.shef.ac.uk

Abstract. Automatic ontology building is a vital issue in many fields where they are currently built manually. This paper presents a user-centred methodology for ontology construction based on the use of Machine Learning and Natural Language Processing. In our approach, the user selects a corpus of texts and sketches a preliminary ontology (or selects an existing one) for a domain with a preliminary vocabulary associated to the elements in the ontology (lexicalisations). Examples of sentences involving such lexicalisation (e.g. ISA relation) in the corpus are automatically retrieved by the system. Retrieved examples are validated by the user and used by an adaptive Information Extraction system to generate patterns that discover other lexicalisations of the same objects in the ontology, possibly identifying new concepts or relations. New instances are added to the existing ontology or used to tune it. This process is repeated until a satisfactory ontology is obtained. The methodology largely automates the ontology construction process and the output is an ontology with an associated trained leaner to be used for further ontology modifications.

1 Introduction

The importance of ontologies is widely accepted in a number of domains including the Semantic Web, Knowledge Management and electronic commerce [1][2]. They provide a means to structure and model the concepts shared by a group of people concerning a specific domain. While a great deal of effort is going into planning the use of ontologies, much less has been achieved in automating their construction: in making feasible a computational process of knowledge capture.

Ontologies traditionally are built entirely by hand and the source of information for these knowledge structures is usually introspection or protocol analysis [3]. In this context, the automation of the process of knowledge capture is still in its infancy. The process of knowledge capture or ontology construction can be analyzed as involving three major steps: first, the construction of a concept hierarchy; secondly, the labelling of relations between concepts, and thirdly, the association of content with each node in the ontology [4]. The dynamic nature of human knowledge makes an automatic system that can be trained on real data (i.e. texts) an imperative.

In the past, a number of researchers have proposed methods for creating conceptual hierarchies or taxonomies of terms from processing texts by applying methods from Information Retrieval (term distributions in documents) and Information Theory (mu-

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 203–207, 2002.

tual information) [2]. It is relatively easy to show that two terms are associated in some manner or to some degree of strength [5][6]). It is possible also to group terms into hierarchical structures of varying degree of coherence [7][8]. However, the most significant challenge, which has not been resolved, is to be able to label the nature of the relationship between the terms [9]. Only if relations are explicit can an ontology be used with problem solving methods (PSMs) [10] i.e. for some form of logical inference.

A set of "lexico-syntactic patterns" which would identify specific ontological relations was proposed by Hearst [11] and later implemented by Morin [12] but with repeated user intervention. Building on this work, and in order to build ontologies for real world applications in Knowledge Management (KM), we propose a methodology based on a co-operative model of user and system interaction. The model is based on the integration of Natural Language Processing (NLP) techniques (especially Information Extraction from text - IE) with user input, so as to limit the user's effort and yet obtain the most accurate possible ontology. Our objective is to make as effective as possible the user's input to the system without expecting any understanding of the nature of 'lexico-syntactic patterns'. The rest of this paper is organized as follows: Section 2 describes the characteristics of the user and the system. We present the major steps in the learning process in Section 3 and an overview of the system interface and learning engine in Section 4. The paper finishes with a description of future work and a conclusion.

2 Building Ontologies for Knowledge Management

We need to have a greater understanding of the qualities and characteristics of the user, who wishes to build an ontology, and the system and its potential capabilities.

User Characteristics. The system we are proposing is developed for the specific context and needs of KM and this implies users with specific characteristics. They are assumed not to have any specialised knowledge but we do assume that they are able to a) draft an ontology, or select or reuse an existing one, and provide this as input to the system; b) validate sentences which are exemplars of a particular relation between two terms; c) name/label a relation exemplified in a particular sentence, and to recognise when they encounter further instances of such a relation. Such characteristics are not specific of KM only, but of a set of users in different fields: for example Semantic Web users tend to have the same profile.

The Characteristics of the System. These are to some extent the characteristics of computer systems in general, but here we focus on those of a combined NLP/IE system. In general they are able to a) analyse large quantities of texts at speeds which often approximate real time; b) find regularities and identify all occurrences of a given regularity; c) cluster words and other patterns into groups; d) establish that a relationship exists between any given term *x* and another term *y*.

These characteristics have already revolutionised lexicography and should have a similar effect on ontology construction and knowledge capture. The ability to find regularities is particularly significant in view of the large quantities of data involved.

3 User - Centred Pattern Learning

The learning process is divided in two stages: the system first attempts to learn about the ISA/hyponymy relations between concepts, and once these have been established (via the steps below) the skeletal ontology is presented to the user who may select further relations to learn. Each of the two stages consists of three steps: bootstrapping, pattern learning and user validation, and cleanup.

Bootstrapping. The bootstrapping process involves the user specifying a corpus of texts, and a seed ontology. The draft ontology must be associated with a small thesaurus of words, i.e. the user must indicate at least one term that lexicalises each concept in the hierarchy.

Pattern Learning & User Validation. Words in the thesaurus are used by the system to retrieve a first set of examples of the lexicalisation of the relations among concepts in the corpus. These are then presented to the user for validation. The learner then uses the positive examples to induce generic patterns able to discriminate between them and the negative ones. Pattern are generalised in order to find new (positive) examples of the same relation in the corpus. These are presented to the user for validation, and user feedback is used to refine the patterns or to derive additional ones. The process terminates when the user feels that the system has learned to spot the target relations correctly. The final patterns are then applied on the whole corpus and the ontology is presented to the user for cleanup.

Cleanup. This step helps the user make the ontology developed by the system coherent. First, users can visualise the results and edit the ontologies directly. They may want to collapse nodes, establish that two nodes are not separate concepts but synonyms, split nodes or move the hierarchical positioning of nodes with respect to each other. Also, the user may wish to 1) add further relations to a specific node; 2) ask the learner to find all relations between two given nodes; 3) refine/label relations discovered in the between given nodes. Corrections are returned back to the IE system for retraining.

This methodology focuses the expensive user activity on sketching the initial ontology, validating textual examples and the final ontology, while the system performs the tedious activity of searching a large corpus for knowledge discovery. Moreover, the output of the process is not only an ontology, but also a system trained to rebuild and eventually retune the ontology, as the learner adapts by means of the user feedback. This simplifies ontology maintenance, a major problem in ontology-based methodologies.

4 Adaptiva

Adaptiva is a system implementing the methodology above that has been developed as part of the AKT project [15]. The ontology learning process starts with the definition of the draft ontology, which is imported into the system's internal format by using a converter. **Adaptiva** is based on GATE [14] which provides facilities for corpus management. Lexicalisation of concepts and relations in the ontology are used to retrieve the first set of examples in the corpus. Such examples are presented to the user

for validation by using a simple interface shown in Figure 1, thus specifying whether the sentence presented is a positive, a negative or an irrelevant example.

The actual complete interface consists of three panes which present i) the examples still to be classified, ii) the examples classified as positive, and iii) those classified as negative. As each example is validated, the user checks one of the two check boxes or leaves the example alone (e.g. because it is too difficult or thought to be irrelevant). According to which box is checked the example moves to the positive or negative pane, thereby allowing the user to revise their decision.

Name of Relation	*Exemplar sentence*	*Positive Example*	*Negative example*
ISA	…countries such as England, France and Italy….	☑	☐

Fig. 1. The interface for user validation

The outcome of the validation process is used by a pattern learner, which in our case is Amilcare (cf. below). Once the learning process is completed, the induced patterns are applied to unseen corpus and new examples are returned for further validation by the user. This iterative process may continue until the user is satisfied that a high proportion of exemplars is correctly classified automatically by the system.

Using a learning algorithm. The methodology described above is generic in that it is not tied to one specific Machine Learning algorithm or approach. The precise methodology is irrelevant from the user's perspective. In **Adaptiva**, we have integrated Amilcare [13], a tool for adaptive Information Extraction from text (IE) designed for supporting active annotation of documents for the Semantic Web.

Using Amilcare, positive and negative examples are transformed into a training corpus where XML annotations are used to identify the occurrence of relations in positive examples. The learner is then launched and patterns are induced and generalised. After testing, the best, most generic, patterns are retained and are then applied to the unseen corpus to retrieve other examples. From Amilcare's point of view the task of ontology learning is transformed into a task of text annotation: the examples are transformed into annotations and annotations are used to learn how to reproduce such annotations.

5 Conclusions and Future Work

We have presented a novel method of user-system interaction for the purposes of ontology building, specifically in the context of knowledge management. This work implements to a larger degree the ideas first proposed by Hearst and built on by Morin. The advantage of our methodology with respect to the previous works is that it does not require any ability to define lexico-semantic patterns. The only knowledge needed is the ability to sketch an ontology and to validate examples, characteristics that are common to users in many application domains. We believe that this is a new direction for user-centred ontology building that could have considerably impact on the way in ontologies are built for real world applications. Future work will concern the evaluation of qualitative and ergonomic aspects so as to establish what the benefits are, and to what degree and how the system can be further improved for the user. It is difficult

to benchmark complex systems such the one presented above, but we are developing criteria to help determine how the system can be improved [2].

Acknowledgements
This work has been carried on in the framework of the Advanced Knowledge Technologies (AKT) project [19], an Interdisciplinary Research Collaboration (IRC) sponsored by the UK Engineering and Physical Sciences Research Council (grant GR/N15764/01). AKT involves the Universities of Aberdeen, Edinburgh, Sheffield, Southampton and the Open University. Its objectives are to develop technologies to cope with the six main challenges of knowledge management: acquisition, modelling, retrieval/extraction, reuse, publication and maintenance. Amilcare is based on GATE [18]. Thanks to the GATE group at the University of Sheffield for providing the Annie system and for help in integrating Amilcare and Annie.

References

1. Fensel, D., F. van Harmelen, I. Horrocks, D.L. McGuiness, P.F. Patel-Schneider, (2001) OIL: An Ontology Infrastructure for the Semantic Web, *IEEE Intelligent Systems* (16)
2. Brewster, C. (2002) Techniques for Automated Taxonomy Building: Towards Ontologies for Knowledge Management, in *Proceeding of the 5th Annual CLUK Research Colloquium*, Leeds
3. Ericsson, K. A. & H. A. Simon, (1984) *Protocol Analysis: verbal reports as data.* Cambridge, Mass.: MIT Press
4. Brewster, C., F. Ciravegna, and Y. Wilks, (2001) Knowledge Acquisition for Knowledge Management: Position Paper, in *Proceeding of the IJCAI-2001 Workshop on Ontology Learning* held in conjuction with the 17th International Conference on Artificial Intelligence (IJCAI-01), Seattle, August, 2001
5. Grefenstette, Gregory, (1994), *Explorations in Automatic Thesaurus Discovery*, Amsterdam: Kluwer.
6. Scott, Michael, (1998) Focusing on the Text and Its Key Words, *TALC 98 Proceedings*, Oxford, Humanities Computing Unit, Oxford University.
7. Brown, Peter F.,Vincent J. Della Pietra, Petere V. DeSouza, Jenefer C. Lai, Robert L. Mercer, (1992) Class-based n-gram models of natural language, *Comp. Linguistics*, (18)
8. Sanderson, Mark, and Bruce Croft, (1999) Deriving concept hierarchies from text, in *Proceedings of the 22nd ACM SIGIR Conference*
9. Wilks, Y. (2000) Artificial Intelligence and Information Retrieval, in *Proceedings of the IEE Symposium on Information Retrieval and Artificial Intelligence*, Glasgow**.**
10. Gomez-Perez, A., (1999) Evaluation of Taxonomic Knowledge in Ontologies and Knowledge Bases, in *Proceedings of the 12th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Alberta, Canada
11. Hearst, M.A., (1992) Automatic Acquisition of Hyponyms from Large Text Corpora, in *Proceedings of COLING 92*, Nantes
12. Morin, Emmanuel, (1999b), Using Lexico-Syntactic patterns to Extract Semantic Relations between Terms from Technical Corpus, in *Proceedings of TKE 99*, Innsbruck, Austria,
13. Ciravegna, Fabio, Alexiei Dingli, Daniela Petrelli, (2002) Document Annotation via Adaptive Information Extraction, *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* August 11-15, 2002, Tampere, Finland.
14. GATE, http://www.gate.ac.uk/
15. Advanced Knowledge Technologies, http://www.aktors.org

A Multilevel Text Processing Model of Newsgroup Dynamics

G. Sampath and Miroslav Martinovic

Department of Computer Science, The College of New Jersey, Ewing, NJ 08628
{sampath,mmmartin}@tcnj.edu

Abstract. We present a multilevel model of discussions in USENET newsgroups that includes the use of statistical and linguistic methods to obtain lexical, semantic and discourse characteristics of the text. We expose constraints that make information extraction and summarization more amenable to analysis at different levels. Our model makes use of posting structure, times of posting, time spans, and length and depth of a thread in order to extract higher-level information on subject matter, interest level, topicality, and discussion trends.

1 Introduction

Database schemas allow mining algorithms to perform various well defined operations such as association and classification [11] on them with ease. With numeric or quasi-numeric fields statistical methods are often used effectively to extract patterns, and various metrics can be used to reason about the data [11]. Text mining in general is considerably more complex because of the absence of a structure or any meta level descriptors. In web mining, since web documents are labeled, linked, and tagged using an underlying tag language, it has been possible to successfully devise methods to extract and summarize information from them [6]. These methods are based on a combination of graph theoretic models [2], shallow natural language processing [8], and statistical techniques [7], and have resulted in a number of models of the web that have been shown to be useful in designing search engines and mining algorithms. Another area in which text dominates the information, but is also accompanied by a form of meta data, is that of discussions on USENET. A survey of the data mining literature shows only limited focus on mining of newsgroups, while a survey of text extraction and summarization from USENET shows that it is mostly used as a data provider and a testing ground for general extraction and summarization methods [1, 3, 4, 9].

The present work is aimed at developing a model of newsgroup dynamics in which the content and activity in one or more related newsgroups can be processed to extract meaningful information about discussions, such as topicality, depth of interest, and discussion focus. It is based on our belief that USENET discussions possess specific constraints that can be conveniently exploited to achieve a useful information extraction and summarization.

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 208–212, 2002.

2 Structural Properties of Usenet Text

At the organizational level, newsgroups have a tree structure with node names that are grouped by suffix (e.g., alt.culture.us.1980s). One advantage of this is that it gives locality to a discussion by confining a topic to a subtree.

At the activity level of a single newsgroup, postings are generally a part of a thread that defines a subtree of activity. The threaded structure of discussions introduces a pathway for algorithms for efficient recognition of patterns. Postings have explicit references or structural relationships to other postings in the same or different thread and/or newsgroup. The topic and posting times are a part of the header, so that subject information and patterns of temporal activity can be easily and directly extracted.

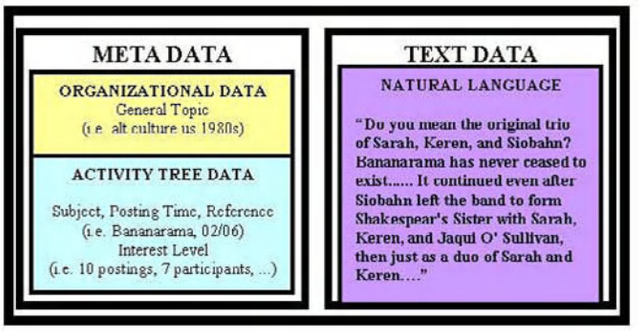


Fig. 1. USENET Discussion Groups Information Levels

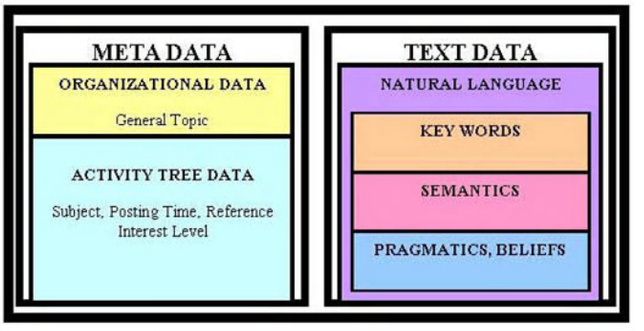


Fig. 2. USENET Discussion Groups Information Model

The shape of a discussion tree further provides information on the following attributes related to the activity level: number of participants, time span of discussion, and lengths of postings. This information can be easily elicited by a rather simple processing of the discussion's tree structure.

The rest of the available information is in the form of natural language text. Any further processing would require some kind of natural language processing (NLP) of the running text and would produce deeper and more sophisticated characteristics.

These would include a more narrowly identified discussion topic (e.g., most popular movies in U.S. in 1980s), discussion polarity (number of sides or viewpoints to an exchange), and the like. Additionally, USENET archives are managed by GOOGLE, a search engine capable of furnishing information on text patterns based on occurrences of word groups, timespan of occurrences, authorship, and posting group.

Once, the USENET information layers have been identified, the corresponding multi-layered model is quite naturally and directly derived. We propose the organization of the text data into: (i) key word characterization of the text, (ii) characterization including semantic information, and (iii) characterization that includes pragmatic and belief aspects.

3 Processing Information from USENET

Information that is embedded in Usenet discussions includes currency of topic, volume of discussion, polarity of discussion, and extent of interest. The currency of a topic is indicated by referents in a discussion that connect the posting time to time references in the thread. The volume of discussion generated is measured by the depth of its tree and the number of postings. The polarity of discussion is the number of viewpoints. The extent of interest can be characterized by the number of postings and their time span.

3.1 Processing Meta Data

The general topic (recorded in organizational data), discussion subject, posting time and references (recorded in activity tree data) are obtainable by a rather straightforward extraction. The interest level can be characterized by the number of postings on the subject, number of participants, as well as time span and size of postings. This information can be learned from the information available in activity tree data utilizing simple methods like counting, adding and subtracting of posting times, and the like. A change in the subject can be handled by a simple bidirectional crosslinking between the parent thread subtree and the spawned child subtree.

3.2 Processing Text Data

Here, we discuss the implementation level for the model presented in the previous section. Processing of the text data will utilize various NLP as well as data mining techniques for handling of the running text and will produce a much deeper and more sophisticated characterizations of it. These would include a more narrowly identified discussion topic (e.g., most popular movies in U.S. in 1980s), polarity of discussion (number of sides to it), and the like.

A three-layer model is proposed that includes: (i) *Keyword Level*: implemented through lexical processing on the keyword level (utilizing a variety of shallow statistical techniques), (ii) *Semantic Level*: implemented through processing resulting in a semantic representation of the postings (utilizing statistical techniques for semantic parsing [9] and techniques like those used in [10] for characterizing polarity), (iii) *Context, Discourse and Beliefs Level*: implemented through processing of the context, discourse and belief systems of participants (using different linguistic theories on semantics, context, discourse and belief system).

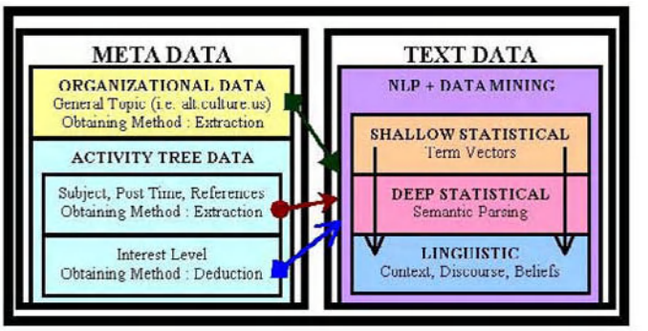


Fig. 3. USENET Discussion Groups Processing Model –Implementation

All three levels are assumed to have access to the meta data. Obviously, all NLP and mining techniques being employed can and are expected to take advantage of the present meta data to define and drive their own processing. This seems especially beneficial for the discourse module which automatically gains information on discussion participants, statements they make, and all of it in a fully defined temporal context. Taking into consideration that most of these facts have to be inferred by any general purpose discourse systems, it is obvious that with USENET discussion groups, what ordinary systems have to work hard to get is already a given.

While the first proposed level clearly can be realized at a minimal computational expense, the information it provides may not be a characterization of the required quality and accuracy. The second level includes more computational power and an increased accuracy level but may still miss the point when the context or beliefs are heavily involved in the discussion. These two levels are envisioned as statistical while the third level which must handle the context-, discourse-, and belief-dependent cases has to rely on more sophisticated and computationally expensive linguistic theories.

4 Future Plans and Directions

While the proposal presented here is of a theoretical nature, our ultimate goal is testing and evaluating it in a real life implementation of our model and integrating it into our question-answering system QASTIIR.

Acknowledgements

This paper is based upon work supported by the NSF, Grant EIA 0130798.

References

1. Agrawal, D., Dolin, R., El Abbadi A. 1999. Scalable Collection Summarization and Selection. In *Edward A Fox and Neil Rowe, editors, ACM DL'99, 49--58,* August 1999.
2. Broder, A., Kumar, R., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., Wiener, J. 2000. Graph Structure in the Web. *Procs. 9th WWW Conference*, Amsterdam, May, 2000.

3. Cardie, C. 1997. Empirical Methods in Info Extraction. In *AI Magazine*, 18:4, 65-79 1997.
4. Dolin, R., Agrawal, D., El Abbadi, A., Pearlman, J. 1998. Using Automated Classification for Summarizing and Selecting Heterogeneous Information Sources. In D-Lib Magazine, ISSN 1082-9873, January 1998.
5. Jiang, M-F., Tseng, S-S., Tsai, C-J. 1999. Discovering Structure from Document Databases. In N. Zhong and L. Zhou, editors, Methodologies for Knowledge Discovery and Data Mining. Lecture Notes in Artificial Intelligence 1574. Springer-Verlag, New York, 1999.
6. Kumar, R., Raghavan, P., Rajagopalan, S., Tomkins, A. 1999. Trawling the Web for Emerging Cyber-communities. *Procs. 8th WWW Conference*, Toronto, May, 1999.
7. Manning, C., Schutze, C. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge (Mass.), 1999.
8. Melnik, S., Raghavan, S., Yang, B., Garcia-Molina, H. 2001. Building a Distributed Full-Text Index for the Web. *Procs. 10th World-Wide Web Conference*, Hong Kong, May, 2001.
9. Ng, H.T., Zelle J. 1997. Corpus-Based Approaches to Semantic Interpretation in Natural Languge Processing. In *AI Magazine Winter 1987*, 18:4, 45-64 1997.
10. Staab, S. 1999. *Grading Knowledge: Extracting Degree Information from Texts. Lecture Notes in Artificial Intelligence **1744***. Springer-Verlag, New York, 1999.
11. Witten, I. H., Frank, E. 1999. *Data Mining*. Morgan Kaufman, San Francisco, 1999.

Best Feature Selection for Maximum Entropy-Based Word Sense Disambiguation*

Armando Suárez and Manuel Palomar

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante
Alicante, Spain
{armando,mpalomar}@dlsi.ua.es

Abstract. In this paper, a supervised learning method of word sense disambiguation based on *maximum entropy conditional probability models* is presented. This system acquires the linguistic knowledge from an annotated corpus and this knowledge is represented in the form of features. Several types of features has been analyzed for a few words selected from the DSO corpus. The main contribution of this paper consists of the selection of the best sets of features for each word from the training data in order to build the classifiers. Our experimentation shows that our method reaches a good accuracy when it is compared with, for example, the systems at SENSEVAL-2.

1 Introduction

Word sense disambiguation (WSD) is an open research field in natural language processing (NLP). The task of WSD consists in assigning the correct sense to words using an electronic dictionary as the source of word definitions. This is a hard problem that is receiving a great deal of attention from the research community.

Currently, there are two main methodological approaches in this research area: *knowledge-based* methods and *corpus-based* methods. The former approach relies on previously acquired linguistic knowledge, and the latter uses techniques from statistics and machine learning to induce models of language usage from large samples of text [1]. This last method can perform supervised or unsupervised learning. With supervised learning, the actual status (here, sense label) for each piece of data in the training example is known, whereas with unsupervised learning the classification of the data in the training example is not known[2]. At SENSEVAL-2 [3], researchers showed the latest contributions to WSD.

This paper presents a system that implements a corpus-based method of WSD. The method used to perform the learning over a set of sense-disambiguated examples is that of maximum entropy (ME) probability models. Linguistic information is represented in the form of feature vectors, which identify the oc-

* This paper has been partially supported by the Spanish Government (CICYT) under project number TIC2000-0664-C02-02.

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 213–217, 2002.

currence of certain attributes that appear in contexts containing linguistic ambiguities. The context is the text surrounding an ambiguity that is relevant to the disambiguation process. The features used may be of a distinct nature: word collocations, part-of-speech labels, keywords, topic and domain information, grammatical relationships, and so on. Instead of training with the same kind of information for all words of the same part of speech, which underestimates which information is more relevant to each word, our research shows that each word is more effectively learned using a different set of features.

2 The Maximum Entropy Framework

ME modeling provides a framework for integrating information for classification from many heterogeneous information sources [2]. ME probability models have been successfully applied to some NLP tasks, such as part-of-speech (POS) tagging or sentence boundary detection [4].

$$f(x,c) = \begin{cases} 1 \text{ if } c' = c \text{ and } cp(x) = true \\ 0 \text{ otherwise} \end{cases} \tag{1}$$

$$p(c|x) = \frac{1}{Z(x)} \prod_{i=1}^{K} \alpha_i^{f_i(x,c)} \tag{2}$$

The WSD method used in this paper is based on conditional ME probability models. It has been implemented using a supervised learning method that consists of building word-sense classifiers using a semantically tagged corpus. A classifier obtained by means of an ME technique consists of a set of parameters α_i which are estimated using an optimization procedure. Each parameter is associated with one feature (see equation 1) observed in the training data. The main purpose is to obtain the probability distribution (see equation 2) that maximizes the entropy, that is, maximum ignorance is assumed and nothing apart from the training data is considered. The ME framework allows a virtually unrestricted ability to represent problem-specific knowledge in the form of features [4].

The implementation of the ME-based WSD system described here was done in C++ and included the learning module, the classification module, the evaluation module, and the corpus translation module. The first two modules comprise the main components.

The learning module produces the classifiers for each word using a corpus that is syntactically and semantically annotated. The classification module carries out the disambiguation of new contexts using the previously stored classification functions. The evaluation module has been constructed in order to check the accuracy of the method. Finally, the corpus translation module is an auxiliary tool that changes the data in the corpus to a predefined input format. If necessary, each corpus is preprocessed with a parser in order to add part-of-speech labels to words as well as all syntactic information.

- *Non-relaxed functions*
  - ***0* features**: ambiguous-word shape
  - s **features**: words in positions ± 1, ± 2, ± 3
  - p **features**: POS-tags of words in positions ± 1, ± 2, ± 3
  - km **features**: lemmas of nouns at any position in context, occurring at least $m\%$ times with a sense
  - r **features**: grammatical relation of the ambiguous word
  - d **features**: the word that the ambiguous word depends on
  - m **features**: the ambiguous word belongs to a multi-word, as
- *Relaxed functions*
  - L **features**: lemmas of content-words in positions ± 1, ± 2, ± 3
  - W **features**: content-words in positions ± 1, ± 2, ± 3
  - S **features**: words in positions ± 1, ± 2, ± 3
  - B **features**: lemmas of collocations in positions $(-2,-1)$, $(-1,+1)$, $(+1,+2)$
  - C **features**: collocations in positions $(-2,-1)$, $(-1,+1)$, $(+1,+2)$
  - P **features**: POS-tags of words in positions ± 1, ± 2, ± 3
  - D **features**: the word that the ambiguous word depends on
  - M **features**: the ambiguous word belongs to a multi-word, as identified by the parser

Fig. 1. List of types of features

3 Feature Implementation

The set of features defined for the training of the system is described in Figure 1 and is based on the feature selection made in [5] and [6]. Moreover, new features have also been defined, using other grammatical properties.

An important issue in the implementation of this ME framework is the form of the functions that calculate each feature. These functions are defined in the training phase and depend upon the data in the corpus. For example, let us assume these three contexts as the learning corpus in which "interest" is the word to be learned, and that all necessary syntactic information is available:

... considering the widespread **interest#1** in the election ...
... to the best **interest#5** of both governments ...
... anonymous persons expressing **interest#1** in the trial ...

Then, six "non-relaxed" features could be defined with the word previous to the ambiguous word and its POS-tag (s_{-1} and p_{-1} features). For example: *is "widespread" previous to interest#1?* Moreover, other types of features, "relaxed features", could be defined: *is "widespread" or "expressing" previous to interest#1?*. These types of features, in this example, reduce the number of functions to four.

4 Evaluation

Table 1 shows the best results obtained using a 10-fold cross-validation evaluation method. Some polysemous nouns and verbs have been selected and eval-

Table 1. 10-fold cross-validation best results on DSO files

	Senses	Examples	Features	Functions	Accur	MFS
age.N	3	491	0CsprDMk5	1587	73.8	62.4
art.N	4	393	0sprdm	1594	65.2	48.0
car.N	2	1363	s	3036	97.1	96.3
child.N	2	1057	sp	2731	90.5	81.8
church.N	3	367	0rDMCk3	228	67.9	62.0
cost.N	2	1456	0WrDM	62	90.0	89.6
head.N	7	844	sprdm	2911	80.8	41.6
interest.N	6	1479	0sprDM	4059	70.1	45.9
line.N	22	1320	0LSBCrdm	1542	54.7	22.7
work.N	6	1419	0sprdm	4784	53.2	32.8
fall.V	6	1341	LSBCrdm	503	84.9	70.1
know.V	6	1425	0rDMCk10	230	47.9	34.9
set.V	11	1246	BsprDMk5	4569	57.3	36.9
speak.V	5	510	0sp	1667	74.5	69.1
take.V	19	794	LWBCsrDMk10	3706	43.0	35.6
Averages	**7**	**1034**		**2214**	**70.1**	**55.3**
Nouns	6	1019		2253	74.3	58.3
Verbs	9	1063		2135	61.5	49.3

Senses is the number of distinct senses in the corpus
Features the feature selection with the best result
Functions the number of functions generated from features
Accur (for "accuracy") the number of correctly classified contexts divided by the total number of contexts. Column
MFS is the accuracy when the most-frequent-sense is selected[6]

uated using the DSO sense-tagged English corpus [5]. Several feature combinations have been tested in order to find the best set for each selected word. The main goal was to achieve the most relevant information from the corpus for each feature rather than applying the same combination of features to all words.

Our ME system were also compared with the systems at SENSEVAL-2 for the Spanish lexical sample task (Table 2). ME obtained an accuracy rate of 67.7% using the feature selection that obtains the better accuracy for each POS: $LWSBCk5$, $sbcprdk3$, and $0spdk5$ for nouns, verbs and adjectives, respectively (obtained with a 3-fold cross-validation over the training data).

5 Conclusions

A WSD system based on maximum entropy conditional probability models has been presented. It is a supervised learning method that needs a corpus previously annotated with sense labels.

For a few words selected from the DSO corpus, several combinations of features were analyzed in order to identify which were the best. In addition, relaxed definitions of the functions that calculate these features were also presented. This is an important issue since it increases WSD efficiency and allows for the inclusion of more attributes so as to enrich the classifiers. The evaluation results of the system were compared with the Spanish task ones at SENSEVAL-2.

Table 2. ME *vs.* Spanish task systems at SENSEVAL-2

System	All	Nouns	Verbs	Adjectives
JHU(R)	0.71	0.70	0.64	0.80
JHU	0.68	0.68	0.61	0.76
ME	**0.68**	**0.68**	**0.58**	**0.77**
CSS244	0.67	0.65	0.59	0.77
UMCP	0.63	0.60	0.58	0.70
DULUTH 8	0.62	0.62	0.51	0.73
DULUTH 10	0.61	0.61	0.52	0.71
DULUTH Z	0.59	0.61	0.49	0.69
DULUTH 7	0.59	0.59	0.50	0.71
DULUTH 6	0.58	0.59	0.47	0.69
DULUTH X	0.58	0.59	0.48	0.68
DULUTH 9	0.56	0.56	0.48	0.65
UA	0.55	0.46	0.51	0.69
DULUTH Y	0.52	0.51	0.43	0.64

As we work to improve the ME method, we are also working to develop a cooperative strategy between several other methods as well, both knowledge-based and corpus-based.

References

1. Pedersen, T.: A decision tree of bigrams is an accurate predictor of word sense. In: Proceedings of the Second Annual Meeting of the North American Chapter of the Association for Computational Linguistics, Pittsburgh (2001) 79–86
2. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. The MIT Press, Cambridge, Massachusetts (1999)
3. Preiss, J., Yarowsky, D., eds.: Proceedings of SENSEVAL-2. In Preiss, J., Yarowsky, D., eds.: Proceedings of the 2nd International Workshop on Evaluating Word Sense Disambiguation Systems, Toulouse, France, ACL-SIGLEX (2001)
4. Ratnaparkhi, A.: Maximum Entropy Models for Natural Language Ambiguity Resolution. PhD thesis, University of Pennsylvania (1998)
5. Ng, H.T., Lee, H.B.: Integrating multiple knowledge sources to disambiguate word senses: An exemplar-based approach. In Joshi, A., Palmer, M., eds.: Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics, San Francisco, Morgan Kaufmann Publishers (1996)
6. Escudero, G., Màrquez, L., Rigau, G.: Boosting applied to word sense disambiguation. In: Proceedings of the 12th Conference on Machine Learning ECML2000, Barcelona, Spain (2000)

Linguistics in Large-Scale Web Search

Jon Atle Gulla, Per Gunnar Auran, and Knut Magne Risvik

Fast Search & Transfer ASA, Støperigata 2, N-0120 Oslo, Norway
{jon.atle.gulla,per.auran,knut.risvik}@fast.no

Abstract. In spite of intensive research on linguistic techniques in information retrieval, there are still few large-scale search engines that have taken full advantage of these techniques. This paper presents the integration of various linguistic techniques in one of the largest search engines on the Internet. The techniques include language identification, offensive content filtering, phrasing and anti-phrasing, normalization, and clustering. We go into some of the challenges of Internet search and discuss our experiences with these techniques.

1 Introduction

Search engines are today important to any user retrieving information on the Internet. The explosion of documents and the lack of directories on the Internet have made in increasingly difficult for users to retrieve the desired documents. 85% of Web users today claim to be using search engines or some kind of search tools to find information on the Web [1]. However, web search is faced with a number of challenges. Large volumes of data are distributed over many computers and platforms [2]. There are dynamic pages and dead links disturbing the search, and a large share of the documents are poorly structured or redundant. The proliferation of languages on the Web also makes it hard to introduce language-specific techniques to improve the situation. Still, there is usually enough information on the Web to answer even the weirdest question. The problem is not primarily to find documents, but to find documents that are relevant to the query and ranked in a meaningful way.

Linguistic techniques in information retrieval systems address the relevance of documents returned. They allow us to abstract away from the exact words used in the documents and queries and put more emphasis on the content of these representations. As such, they should help us close the gap between the user's expressed query and the representation of relevant documents.

Fast Search & Transfer (FAST) has one of the biggest search engines on the market, and its engine is integrated in portals like Lycos and T-Online as well as in enterprise solutions for IBM, eBay, and other large international companies. The FAST search engine holds a number of innovative linguistic techniques on AlltheWeb (www.alltheweb.com) that separate it from traditional search engines.

This paper discusses the introduction of language identification, offensive content reduction, normalization, query transformation, and clustering in FAST. Whereas Section 2 briefly presents the rationale of these linguistic techniques, we explain how the techniques were realized in Section 3. Conclusions are found in Section 4.

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 218–222, 2002.

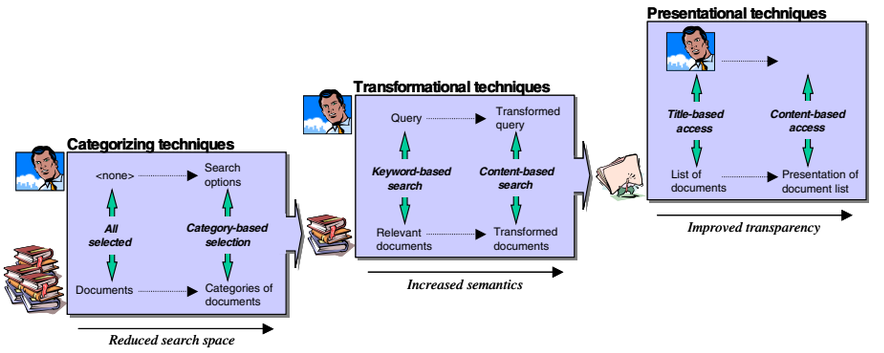


Fig. 1. Linguistic techniques in FAST fall into three categories

2 Why Linguistics in Search Engines

A fundamental issue in information retrieval is the relationship between queries and documents. The document's relevance with respect to a query is not necessarily decided on the basis of words common to both query and document. The document is relevant to the extent that its content satisfies the user's need for information. Linguistic techniques address the relevance of documents by exposing the underlying content or semantics of documents and queries. From a search perspective, the techniques fall into the following three categories (see Figure 1): (1) *Categorizing techniques*, (2) *transformational techniques,* and (3) *presentational techniques.*

Using categorizing techniques, the search engine can restrict the search space to only those documents that may be relevant to the search context. This can be documents of a certain language, documents on a certain topic, or documents of a certain quality. After identifying the category, the documents and their corresponding categories are added to the index.

Another line of thought is to transform queries or documents to representations that better facilitate document retrieval. This includes normalization of document and query text, and intelligent processing of query phrases. In general, transformational techniques expose semantic aspects of queries and documents that should be taken into account in the search process.

The last group of linguistic techniques comes into play when documents are presented to the user. Generating small summaries that are shown on the result page is one way of telling the user what these documents are about. Another technique, clustering, rearranges the result set according to pre-defined or generated categories. These presentational techniques improve the transparency of documents and speeds up the process of selecting the desired document from the result set.

3 Linguistic Components in FAST

The FAST approach is to use dictionary-driven linguistic techniques based on finite-state technology. This allows us to take full advantage of large lexical resources,

while maintaining an acceptable response time. In the following, we go into the realization of the linguistic techniques in the FAST search engine.

3.1 Language Identification and Offensive Content Reduction

The FAST language identifier has a dictionary that contains the relevant words in all relevant encodings for 52 languages. For languages with no clear word boundaries (e.g. Asian languages), bigram lists are used instead of words. The identifier can process HTML and plain text documents and identifies their language and encoding. When needed, we take the structure of HTML documents and the meta information into consideration. In some cases, we also use domain names like .DE for Germany to find the most probable language of a web document. This has to be done with caution, as there is no one-to-one mapping between domain names and languages.

For the current language identifier, both recall and precision are above 99% for documents of more than 20 words. However, there is a tremendous amount of documents with little or no text at all on the Internet. About 95-96% of the documents crawled are successfully tagged for language during indexing.

The offensive content reduction component employs standard text categorization techniques and relies on a large dictionary of offensive words and phrases. These dictionary entries are generated from large collections of offensive web pages and are associated with weights that specify their influence in the filtering process.

The current filter works for pornographic content in English, German, Italian, Spanish, and French. As many pornographic documents tend to contain keywords from several languages, the filter works reasonably well also for some other languages. The dictionary entries for each language span from a few hundred to several thousand.

3.2 Normalization

The goal of *normalization* is to map different but semantically equivalent words onto one canonical representation. The normalized words are then used instead of or in addition to the original words when documents are retrieved. This increases recall, as an exact match of the query terms and the words in the document is not needed.

Lemmatization is a normalization technique for reducing an inflected word to its lemma, which is the abstract representation of the word independent of time, person, case, etc. Whereas the infinitive form is taken as the lemma or *base form* for verbs, the indefinite singular nominative form is used for nouns. Adjectives and adverbs are often reduced to their absolute form in the indefinite nominative masculine context. For example, the base form of *written* is *write*, and the base form of *skies* is *sky*.

The most obvious lemmatization strategy is to replace all full forms of documents and queries with their base forms. However, this prevents us from being able to search for exact phrases in documents and the query language must be known. A more conservative approach is just to add the lemmas (base forms) to documents and queries. This introduces some asymmetry, as the base forms are only added for the words that are not equal to their base forms. A third strategy is to expand the document or the query with all inflected forms. The full form document expansion enables phrase search and can be done without knowing the language of the query, but increases the index size substantially and may lead to unexpected ranking results.

For a search engine like AllTheWeb, where the query language is not always known, it is difficult to lemmatize queries. Full form document expansion is used for Russian and Polish, as these languages are not too frequent on the web. For popular languages like German and French, base form document expansion is applied. Full replacement of inflected words with base forms for both documents and queries is only done for Japanese and Korean. Lemmatization requires that large full form dictionaries be available during indexing. Our dictionary for German, for example, contains 1,016,492 full forms. Whereas the indices for Russian and Polish increased by 600-800% after deploying lemmatization, the German index only increased by 5%.

3.3 Phrasing and Anti-phrasing

The query "*mutual information*" (explicit marking of phrase) gives us about 6,300 documents, and the top three documents are all about mutual information. Without explicit phrasing, more than a million documents are returned (AllTheWeb, March 2002). Since both *mutual* and *information* are used in so many other contexts, we get far more documents and none of the top ten documents are about mutual information.

FAST has a large phrase dictionary that is used to recognize and quote phrases in queries. This dictionary contains scientific expressions like "mutual information" above, but also person names, geographical names, song titles, organizations, etc. Some phrases, like *Albert Einstein*, should always be interpreted as a whole and are often referred to as hard phrases. For soft phrases like *data modeling,* the individual terms may also be used independently of each other with the same semantic content.

For a query like *New York Times sports*, both *New York* and *New York Times* are recognized as phrases. The longest phrase is preferred, and the query *"New York Times" sports* is consequently sent to the search engine. For a query like *New York art museum,* the phrase *New York* is found in the phrase list. A bigram for the last two terms ("art museum") are introduced as an optional phrase in the query. Whereas the hard phrases are assumed to be in the phrase list, soft phrasing is achieved by adding these additional bigrams at the end of the query.

In a similar vein, there are many queries that contain phrases that only disturb the search. Posting a query like *where can I find The Economist*, for example, the homepage of the magazine The Economist will not be found at all. If we remove the first part of the query, *where can I find*, The Economist shows up on top of the result list. Removing these irrelevant phrases in the beginning of the queries is called anti-phrasing and is done after phrasing, but before searching for documents.

Currently, we have several million phrases and thousands of anti-phrases available in English and German. Preliminary tests show that about 6% of the queries contain phrases, but only 0.2% contain anti-phrases.

3.4 Clustering

Traditional ranking of documents according to their relevancy to the query produces a flat list of ranked results. The process of generating hierarchical and more user-friendly presentations of result sets is referred to as *clustering*. Each cluster contains a list of documents or sub-clusters that pertain to the same topic.

The groups of documents with similar content effectively form a table-of-contents of the result set. We identify those documents in the result set that belong to the tax-

onomy of the Open Directory Project, and other documents in the result set are attempted mapped on-the-fly into this taxonomy. For documents that do not fit well into this hierarchy, an algorithm tries to detect topical clusters and devises descriptive names of these clusters. Large document vectors are used in this process.

The unsupervised classification is done without any special knowledge of the semantics of words or phrases. This means that the approach is largely language-independent, although having identified the language helps. If we know which language a document is in, we exploit this in the vectorization process to skip very frequent words that carry little or no information. Examples of such words in English are "the", and "to". The current version works best for English.

4 Conclusions

The linguistic techniques presented here serve different purposes, but also complement each other in important ways. Whereas normalization and anti-phrasing tend to increase the number of documents returned, language identification, phrasing, and offensive content reduction make the result set smaller and more focused. For almost all techniques, there is a need to augment the standard approach with Internet-specific features to take full advantage of their potential. A thorough evaluation of each individual technique on the Internet is very difficult, though we are now looking into test procedures in line with what has been presented in [3].

The FAST search engine today has an index of about 800 million text pages, about 115 million multimedia pictures and videos, and more than 2 million MP3 songs. The normal index is refreshed every 9-11 days, though there are also more than 2,000 online news sources that are continuously crawled and updated. Adding the queries coming from partner portals like Lycos and T-Online, the FAST search engine has a total of about 30 million queries per day.

References

1. Kobayashi, M. and Takeda, K. (2000). "Information Retrieval on the Web." *ACM Computing Surveys*, Vol. 32, No. 2, pp. 144-173. 2000.
2. Risvik, K. M. and R. Michelsen (2001). "Search Engines and Web Dynamics." To appear in Computer Network, special issue of Web Dynamics. 2002.
3. Craswell, N., D. Hawking, and K. Griffiths (2001). *Which search engine is best at finding airline site home pages?* Technical report 01/45. CSIRO, Australia, 2001.

Similarity Model and Term Association for Document Categorization

Huaizhong Kou[1] and Georges Gardarin[1,2]

[1] PRiSM Lab, University of Versailles
{Huaizhong.kou,Georges.Gardarin}@prism.uvsq.fr
[2] e-xmlmedia, France
{Georges.Gardarin}@e-xmlmedia.fr

Abstract. In the information retrieval and document categorization context, both Euclidean distance– and cosine-based similarity models are based on the assumption that term vectors are orthogonal. But this assumption is not true. Term associations are ignored in such similarity models. This paper analyzes the properties of term-document space, term-category space and category-document space. Then, without the assumption of term independence, we propose a new mathematical model to estimate the association between terms and define a ε-similarity model of documents. Here we make best use of existing category membership represented by corpus as much as possible, and the objective is to improve categorization performance. The empirical results been obtained by **k-NN** classifier over Reuters-21578 corpus show that utilization of term association can improve the effectiveness of categorization system and ε-similarity model outperforms than ones without term association.

1 Introduction

Document categorization is the procedure of assigning one or multiple predefined category labels to a free text document. Various mathematical models have been proposed to represent documents for categorizing documents, including a probabilistic model [2] based on the computation of relevance probabilities for the documents of a collection and the vector space model [6]. Based on vector representation model, a lots of algorithms have been intensively studied, including support vector machine (SVM) algorithm [1], k nearest neighbors (k-NN) algorithm [7]. The algorithms cited above are based on a similarity model by which two documents can be compared. There are two main similarity models: Euclidean distance models and cosine function models, where axis vectors are mathematically assumed orthogonal, that is, features are orthogonal. But this assumption does not hold in reality [4].There exists some association between terms. These two similarity models do not take into account such association .

In the context of document categorization, this paper proposes a mathematical model to estimate the term association without the assumption of orthogonal terms. Then such term associations are furthermore used in ε-similarity model of documents. The idea is as follows: each category is represented as a binary vector in category-document space whose elements indicate if a sample document belongs to it,

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 223–229, 2002.

then similarity between categories is estimated by Jaccard coefficient; the weights of feature terms in categories can be calculated by using x^2 algorithms, and feature terms are mapped as vectors in term-category space whose axis are categories, then association between feature terms can be estimated; such estimation of term association then are introduced in the similarity model of document vectors in term-document space.

2 Analysis of Similarity Models

2.1 General Notions

$\mathcal{D}$ is a set of example documents ; n is the number of documents in $\mathcal{D}$; $\mathcal{T}$ is a set of feature terms optimally selected from documents in $\mathcal{D}$, T the number of feature terms in $\mathcal{T}$; $\mathcal{C}$ a set of domain-specific categories, m the number of category in $\mathcal{C}$; f_{ij} is the number of times j^{th} term occurs in i^{th} document d_i, df_j the number of documents in which the j^{th} term occurs. Note that feature terms are very significant words to predicting category membership and representing the document contents. We also call feature term as term.

2.2 Document Vector Model

Under the vector space representation model, a document d_i is represented as term vector of the follow-ing form, noted also by $\mathbf{d}_i$:

$$d_i = (w_{i1}, w_{i2}, ..., w_{iT}) \qquad (2.1)$$

There exists different approaches to the calculation of w_{ij}, including tf-idf and tfc algorithm [4]. In the term-document space, each term corresponds to an axis vector and document to a point or vector. If we represent j^{th} axis vector corresponding to j^{th} term by t_j, then document vector dj can be represented as a linear combination of T axis vectors (2.2).

$$d_i = \sum_{j=1}^{T} w_{ij} \bullet t_j \quad (2.2)$$

where T axis vectors t_j (j=1,2,…,T) are linearly independent; otherwise a max linearly independent group of axis vectors can be determined by linear algebra theory and used in (2.2).

2.3 Similarity Model of Documents

Given two documents d_i and $d_j \in \mathcal{D}$ represented in the form (2.2), the similarity between them can be measured by the inner product [6]:

$$sim(d_i, d_j) = d_i \bullet d_j = \sum_{r,s=1}^{T} w_{ir} w_{js} (t_r \bullet t_s) = \sum_{r=1}^{T} w_{ir} w_{jr} (t_r \bullet t_r) + \sum_{\substack{r,s=1 \\ r \neq s}}^{T} w_{ir} w_{js} (t_r \bullet t_s) \quad (2.3)$$

From (2.3), we can see that computing the similarity between two document vectors does not only depend on documents but also the knowledge of the term correlation $t_r \bullet t_s$ for all term pairs. But the term association or correlation are not usually available a priori. In practice, term-correlation is often circumvented by assuming that the terms are not correlated, in which case the term vectors are orthogonal: $t_r \bullet t_s=0$ for $r \neq s$ and $t_r \bullet t_s=1$ for $r=s$. So, (2.3) can be rewritten as (2.4).

$$sim(d_i, d_j) = \sum_{r=1}^{T} w_{ir} w_{jr} \quad (2.4)$$

Now, if we call the first and the second part in (2.3) as orthogonal and non-orthogonal parts respectively, the non-orthogonal part describes the power that term correlation between terms in $\mathcal{T}$ plays rules for computing similarity. The non-orthogonal part of (2.3) disappears from (2.4) and association between terms are not at all taken into account in commonly used inner product model.

Again, on the basis of orthogonality assumption about terms, cosine similarity model (2.5) is also introduced as follows [6],

$$sim\ (d_i, d_j) = \cos\ (d_i, d_j) = \frac{d_i \bullet d_j}{\|d_i\| \cdot \|d_j\|} \quad (2.5)$$

where $\|d_i\|$ and $\|d_j\|$ are length of document d_i and d_j. Often, document vectors are normalized to be of unit length. This implies that (2.4) and (2.5) are same. Under this assumption, other similarity models are also proposed in information retrieval community, see [6]. Ignorance of term associations certainly effects system performance for example accuracy and effectiveness.

3 Mathematical Model for Term Association

On the basis of term-document matrix, one approach to estimation of term association is proposed [5]: each term vector firstly is represented as a linear combination of document vectors, then term association can be estimated under the assumption that the document vectors are themselves orthogonal. Such an assumption is clearly unreasonable for any kind of practical document collection [3]. Another approach uses term co-occurrence in training documents to estimate term association [5]. Under the vector space model, terms and documents are represented in the same vector space, and only the occurrences of terms in documents are used. In the document categorization context, there exists other important information about relationships between terms and categories, documents and categories. We attempt to make use of these information and introduce term-category space and document-category space. It's in the later two spaces where association between terms are estimated. We believe that term association estimated by such way could improve categorization performance.

From the point of view of document categorization, this section proposes a mathematical model to approximately estimate term association in (2.3) via two spaces: term-category space and document-category space. This model does not need the assumption made in[3].

3.1 Term-Category Space

Given a corpus, we can estimate the power that each term in $\mathcal{T}$ plays rules of category prediction. To do this, there exists several algorithms[7], for example, Information gain and CHI-Square χ^2 statistic. Then each term $t_r \in \mathcal{T}$ can be mapped to a vector represented as (3.1) in term-category space, in which categories correspond to axis.

$$t_r = (t_{r1}, t_{r2}, ..., t_{rm}) \qquad (3.1) \qquad\qquad t_r = \sum_{i=1}^{m} t_{ri} \bullet c_i \qquad (3.2)$$

In (3.1), t_{ri} (i=1,2,…,m) is the measure of category prediction power of the r^{th} term $t_r \in \mathcal{T}$ for category $c_i \in \mathcal{C}$ and is calculated by for example CHI-square algorithm in our implementation. Furthermore, without loss of generality, we can assume that m category vectors c_i are linearly independent; otherwise we can select one maximum linearly independent group to replace them. In this case, term vector can be represented as linear combination of category vectors like (3.2).

With term vector representation (3.2), similarity between two term $t_r, t_s \in \mathcal{T}$ can be calculated as inner product by (3.3). Different from the approach to estimation of term

$$tc\_sim(t_r, t_s) = t_r \bullet t_s = \sum_{l=1}^{m} t_{rl} t_{rl} (c_l \bullet c_l) + \sum_{\substack{k,l=1 \\ k \neq l}}^{m} t_{rk} t_{sl} (c_k \bullet c_l) \qquad (3.3)$$

association proposed in [6], (3.3) estimates term association at the level of category in term-category space. In the categorization context, we believe that such estimation (3.3) is reasonable. But it involves the similarity between two categories: $(c_k \bullet c_l)$. In order to calculate the similarity between categories, we go to category-document space where category similarity can be estimated approximately.

3.2 Category-Document Space

In the document categorization context, each category is characterized by the training documents that belong to it. We can represent each category as boolean vector of documents in category-document space where each training document corresponds to an axis. Given a category $c_r \in \mathcal{C}$, we have a boolean vector representation as (3.4)

$$c_r = (c_{r1}, c_{r2}, ..., c_{in}) \qquad (3.4)$$

and c_{ri}=1 if d_i belongs to c_r ; c_{ri}=0 otherwise. Hence category-document is Boolean vector space. In this case, we use Jaccard coefficient [5] to calculate similarity be-

$$cd\_sim(c_r, c_s) = \frac{|c_r \cap c_s|}{|c_r| + |c_s| - |c_r \cap c_s|} \qquad (3.5)$$

tween two category c_r and c_s as (3.5),where $|c_r|$ and $|c_s|$ are the number of documents of categories c_r and c_s respectively, $|c_r \cap c_s|$ is the number of common documents of

categories c_r and c_s. Obviously, any assumption is not needed for computing $cd\_sim(c_r,c_s)$, see (3.5). From (3.5), we obtain $cd\_sim(c_r,c_r)=1$ for every category $c_r \in \mathcal{C}$. Furthermore, if every training document at most belongs to one category, then $cd\_sim(c_r,c_s)=0$ for $r \neq s$.

3.3 Term Association

On the basis of (3.3) and (3.5), the association between two terms $t_r, t_s \in \mathcal{T}$, noted by $ass(t_r, t_s)$, is defined in (3.6). From the discussion in (3.5), we know that if every training document at most belongs to one category, the second part in (3.6) will be zero. In this case, we have (3.7).The result coincides with the case where category vectors are orthogonal in term-category space, See (3.3). Note that the association between terms by (3.6) is unbounded, we normalize $ass(t_r, t_s)$ so that association value drops in the interval [0,1] as (3.8). Now given a ε, $0 \leq \varepsilon \leq 1$, we define ε-association between two terms by (3.9).Obviously, if ε=0, ε-$ass(t_r, t_s)$ is equal to $ass(t_r, t_s)$. The function of ε-$ass(t_r, t_s)$ is to filter out those term pairs with relative weak associations.

$$ass(t_r, t_s) = \sum_{l=1}^{m} t_{rl} t_{sl} + \sum_{\substack{k,l=1 \\ k \neq l}}^{m} t_{rk} t_{sl} cd\_sim(c_k, c_l) \quad (3.6) \qquad ass(t_r, t_s) = \sum_{l=1}^{m} t_{rl} t_{sl} \quad (3.7)$$

$$ass(t_r, t_s) = \frac{ass(t_r, t_s)}{\max_{k,l}\{ass(t_k, t_l)\}} \quad (3.8) \qquad \varepsilon - ass(t_r, t_s) = \begin{cases} 0 & if\ ass(t_r, t_s) < \varepsilon \\ ass(t_r, t_s) & if\ ass(t_r, t_s) \geq \varepsilon \end{cases} \quad (3.9)$$

3.4 ε-Similarity Model

Now, by using ε-$ass(t_r, t_s)$ defined in (3.9) to approximate term association $t_r \bullet t_s$ in the non-orthogonal part of (2.3) and combining (2.4) and the non-orthogonal part of (2.3), we introduce ε-similarity model of document pair (d_i, d_j) in (3.10).

$$\varepsilon - sim(d_i, d_j) = \sum_{r=1}^{T} w_{ir} w_{jr} + \sum_{\substack{r,s=1 \\ r \neq s}}^{T} w_{ir} w_{js} \varepsilon\_ass(t_r, t_s) \quad (3.10)$$

ε-similarity model extends the simplest similar models in (2.4) and (2.5). It is clear that if ε =1 (3.10) will be same as (2.4) and (2.5). Compared to the simplest similarity models (2.4) and (2.5), we hope the introduction of term association in our ε-similarity model will improve categorization performance. The objective of ε is to filter out term pairs with low association.

4 Experiments and Analysis

Experiment design. We use Reuter-21578 collections as experiment benchmark. ApteMod split strategy is used but only 9805 news stores with news body are kept,

and both the news stories without news body and without category labels are removed. We chose 21 big categories that contain 6533 training documents and 2506 test document. After removing 319 stop-words, stemming is performed to convert words into stems, and14,743 unique terms are obtained. Then x^2 approach [7] is used to select 1100 terms with higher x^2 value. k-NN classifier is conducted by taking the value of k as 10,20,30,40,50,60,70 respectively. We set ε to be 0.6 for ε-similarity model, and the association values of only 2460 pairs of terms are used to calculate similarity of documents.

Table 1. Marco Average Performance

k	Recall		Precision		F1	
	sim	esim	sim	esim	sim	esim
10	0,643	0,647	0,864	0,865	0,715	0,718
20	0,661	0,663	0,862	0,861	0,726	0,726
30	0,649	0,653	0,863	0,863	0,716	0,717
40	0,692	0,695	0,874	0,874	0,755	0,757
50	0,693	0,697	0,885	0,886	0,759	0,76
60	0,696	0,7	0,887	0,888	0,76	0,763
70	0,698	0,702	0,893	0,893	0,762	0,764
avg	0,676	**0,679**	0,875	**0,876**	0,742	**0,744**

Evaluation of binary classification. Recall and precision are two commonly used evaluation measures of performance. Given a category, recall (r) is the proportion of correctly assigned documents to all documents belonging to the category and precision (p) one of correctly assigned documents to all assigned documents. In addition, Another measure, called $F_1(r, p)$ defined by combining them is checked. Macro-average and micro-average are also examined while 11-points Recall-Precision Graph is plotted by producing a score for each category-document pair. See [8].

Table 1 shows the results produced by k-NN with conventional similarity model (2.4) and ε-similarity model (3.10), where sim and esim represent the results by (2.4) and (3.10) respectively.

We can see that ε-similarity model always outperforms conventional similarity model for different k values even if it does slightly. This proves that utilization of term association really improves the effectiveness of categorization. One only achieves a little gain of performance by (3.10) against (2.4). The cause is that only a little parts of term pair associations (2460 vs. 604450 total number of term pairs) are used to calculate document similarity in our experiments.

Our experimental results shown that ε-similarity model performs better than cosine similarity model, in particular at the middle level of recall. These two similarity

models have two common parameters: the number of feature terms and the number k of nearest neighbors. Beside, ε-similarity model has the third parameter: term association threshold ε. The association threshold parameter ε in ε-similarity model can be tuned to furthermore improve its performance. Tuning performance by association threshold ε is an advantage of ε-similarity model.

5 Conclusion

In the context of document categorization, this paper addressed similarity model and term association. In practice, one assumes that term vectors are orthogonal in term-document vector space, the association between terms is omitted in inner product-based similarity model and cosine-based similarity model. But such assumption does not hold. By introducing two spaces at the more high level of concept: term-category space and document-category space, this paper proposed a mathematical model of estimating term associations, then term association by this mathematical model is integrated with the conventional cosine model of document similarity. Our experiments confirmed that utilization of association between terms can improve performance of categorization system.

Acknowledgements

Thanks to Professor Elisabeth Métais and anonymous reviewers for their suggestions.

References

1. Joahims, T., Text categorization with support vector machines: Learning with many relevant features. In the proceedings of the European conference on Machine learning, 1998, 137-142.
2. Lewis, D. D. & Ringuette, M., Comparison of two learning algorithms for text categorization. In Proceedings of the 3rd SDAIR,1994,81-93.
3. Raghavan V.V. & S.K.M. Wong., A Critical Analysis of the Vector space Model for information Retrieval, JASIS, 37:5, September 1986.
4. Salton, G. & Buckley, C., Term weighting approaches in automatic text retrieval, Information Processing and Management, Vol. 24, No.5, 1988,513-523.
5. Salton, G., Introduction to Modern Information Retrieval, 1983, McGraw-Hill.
6. Salton, G., Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley, Reading, Pennsylvania, 1989.
7. Yang,Y. & Pedersen, J. O., A Comparative Study on Feature Selection in Text Categorization, In the 14th Int. Conf. On Machine Learning, 1997, 412-420.
8. Yang, Y., An evaluation of statistical approaches to text categorization. Information Retrieval, 1(1),1999,69-90.

Omnibase: Uniform Access to Heterogeneous Data for Question Answering

Boris Katz, Sue Felshin, Deniz Yuret, Ali Ibrahim, Jimmy Lin,
Gregory Marton, Alton Jerome McFarland, and Baris Temelkuran

Artificial Intelligence Laboratory
200 Technology Square, Cambridge, MA 02139
boris@ai.mit.edu

Abstract. Although the World Wide Web contains a tremendous amount of information, the lack of uniform structure makes finding the right knowledge difficult. A solution is to turn the Web into a "virtual database" and to access it through natural language. We built Omnibase, a system that integrates heterogeneous data sources using an *object–property–value* model. With the help of Omnibase, our START natural language system can now access numerous heterogeneous data sources on the Web in a uniform manner, and answers millions of user questions with high precision.

1 Introduction

If someone is asked a question like "When did Rutherford Hayes become president of the U.S.?", he or she might locate a resource with the answer—say, a book on famous people, or a Web site about presidents—find the section for Rutherford B. Hayes, and look up the date of his inauguration. Millions of questions can be answered in this manner: by extracting an *object* (Rutherford Hayes) and a *property* (presidential term) from the question, finding a data source (e.g., the POTUS Web site, `http://www.ipl.org/ref/POTUS`) for that type of object, looking up the object's Web page, and extracting the *value* for the answer (see Figure 1).

The three main challenges in getting a computer to answer such questions are understanding the question, identifying where to find the information, and fetching the information itself. START [9,10] and Omnibase comprise our natural language question answering system[1] developed to addresses these challenges. START is responsible for understanding user questions and translating them into structured queries. Omnibase is a "virtual" database that provides a uniform interface to multiple Web knowledge sources, capable of executing the structured queries generated by START. Currently, our system can answer millions of questions about variety of topics such as almanac information (cities, countries, lakes, etc.; weather, demographics, economics, etc.), facts about people (birth dates, biographies, etc.), and so forth.

[1] `http://www.ai.mit.edu/projects/infolab/`

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 230–234, 2002.

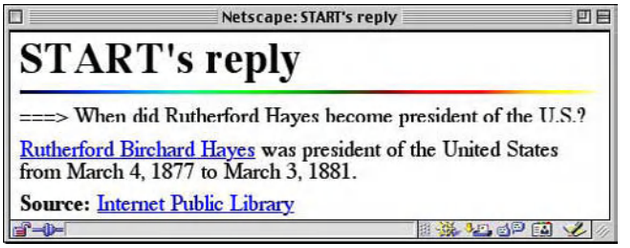


Fig. 1. The START system answering a question using Omnibase.

2 Data Model

Omnibase serves as a structured query interface to heterogeneous data on the World Wide Web. It is of course impossible to impose any uniform schema on the entire Web. To address this challenge, Omnibase adopts a stylized relational model which we call the "object–property–value" data model. Under this framework, data sources contain *objects* which have *properties*, and questions are translated into requests for the *value* of these properties.

Natural language commonly employs an 'of' relation or a possessive to express the relationship between an object and its property, e.g., "the director of La Strada" or "La Strada's director". The following table shows, however, that there are many alternative ways to ask for the value of a property of an object.

Question	**Object**	**Property**	**Value**
Who wrote the music for Star Wars?	Star Wars	composer	John Williams
Who invented dynamite?	dynamite	inventor	Alfred Nobel
How big is Costa Rica?	Costa Rica	area	51,100 sq. km.
How many people live in Kiribati?	Kiribati	population	94,149
What languages are spoken in Guernsey?	Guernsey	languages	English, French
Show me paintings by Monet.	Monet	works	[images]

Clearly, many other possible types of queries do not fall into the object–property–value model, such as questions about the relation between two objects (e.g., "How can I get from Boston to New York?"). However, our experiments reveal that in practice questions of the object–property–value type occur quite frequently. For example, just ten Web sources accessed through the object–property–value model turned out to be sufficient for handling 37% of TREC-9 and 47% of TREC-2001 questions from the QA track [14].

3 How the System Works

Suppose the user asks "Who directed gone with the wind?" START cannot analyze this question without first knowing that "Gone with the Wind" can be

treated as a single lexical item—otherwise, the question would make no more sense than, say, "Who hopped flown down the street?" Omnibase identifies the names of objects and the data sources they are associated with; for example, "Good Will Hunting" comes from a movie data source, "Taiwan" from a country data source, etc. Not only does this help START understand the user question (which can now be read as "who directed X"), but it also lets START know what data source contains the information, i.e., it should look in a movie database. In order to answer the question, START matches syntactic structures derived from the question with those derived from natural language annotations [10]. These annotations are machine-parseable sentences and phrases that serve as metadata to describe knowledge segments (in this case, Omnibase queries). A successful match triggers the execution of an Omnibase query. Because START performs this match at the syntactic level, linguistic machinery is utilized to achieve capabilities beyond simple keyword matching, for example, complex syntactic alternation involving verb arguments. More detailed descriptions of this technology can be found in [9,10].

With help from Omnibase, START translates user queries into a structured request (in the object–property–value model):

```
(get "imdb-movie" "Gone with the Wind (1939)" "DIRECTOR")
```

In this case, our natural language system needed to figure out that the user is asking about the DIRECTOR property of the object `"Gone with the Wind (1939)"`, and that this information can be found in the data source `imdb-movie`, corresponding to the Internet Movie Database.

Omnibase looks up the data source and property to find the associated script and applies the script to the object in order to retrieve the property value for the object[2]. The execution of the `imdb-movie` DIRECTOR script involves looking up a unique identifier for the movie (stored locally), fetching the correct page from the IMDb Website (via a CGI interface), and matching a textual landmark on the page (literal text and HTML tags) to find the director of the movie. As a result, the list of movie directors is returned:

```
(get "imdb-movie" "Gone with the Wind (1939)" "DIRECTOR") =>
("George Cukor" "Victor Fleming" "Sam Wood")
```

START then assembles the answer and presents it to the user either as a fragment of HTML or couched in natural language (Figure 2).

4 Related Work

The use of natural language interfaces to access relational databases can be traced back to the sixties; for a survey see [2].

[2] Such scripts are sometimes called wrappers [6].

Fig. 2. START's response to "Who directed Gone with the Wind" is shown above. The original Web page from which Omnibase extracts the answer is shown below.

The idea of applying database techniques to the World Wide Web is not new. Many existing systems, e.g., ARANEUS [3], ARIADNE [12], Information Manifold [11], LORE [15], TSIMMIS [7], and others, have attempted to integrate heterogeneous Web sources under a common interface. Unfortunately, queries to such systems must be formulated in SQL, Datalog, or some similar formal language, which render them inaccessible to the average user.

A well known disadvantage of data integration systems is the manual labor involved in writing wrappers [6]. Often, wrapper generation can be expedited by a well-designed authoring tool, e.g., [1,17]. Alternatively, machine learning techniques can automate the wrapper generation process [13,8,16,5] using annotated examples. Most promising is Semantic Web [4] research; if someday it can imbue ordinary Web documents with semantic annotations, integration of multiple knowledge sources could be accomplished effortlessly.

What makes Omnibase unique among these systems is its use of the object–property–value data model. Because this model corresponds naturally to both user questions and online content, the data integration task becomes more intuitive.

5 Contributions

We have built Omnibase as an abstraction layer over diverse, semi-structured, online content, centered around *object–property–value* queries. Because our data model is reflective of real-world user queries, we can achieve broad knowledge coverage with a reasonable amount of manual labor.

Omnibase has given START, our natural language question answering system, access to a wealth of information freely available on the World Wide Web. In the future, we intend to extend the data model and to automate the data integration process. We believe that structured access to online data sources will be a key component of any future natural language question answering system.

References

1. B. Adelberg. NoDoSE—a tool for semi-automatically extracting structured and semistructured data from text documents. *SIGMOD Record*, 27:283–294, 1998.
2. Ion Androutsopoulos, G. Ritchie, and P. Thanisch. Natural language interfaces to databases—an introduction. *Natural Language Engineering*, 1(1):29–81, 1995.
3. P. Atzeni, G. Mecca, and P. Merialdo. Semistructured and structured data in the Web: Going back and forth. In *Workshop on Management of Semistructured Data at PODS/SIGMOD'97*, 1997.
4. T. Berners-Lee. *Weaving the Web*. Harper, New York, 1999.
5. M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Automatically deriving structured knowledge bases from on-line dictionaries. Technical Report CMU-CS-98-122, Carnegie Mellon University, 1998.
6. D. Florescu, A. Levy, and A. Mendelzon. Database techniques for the World-Wide Web: A survey. *SIGMOD Record*, 27(3):59–74, 1998.
7. J. Hammer, H. Garcia-Molina, J. Cho, R. Aranha, and A. Crespo. Extracting semistructured information from the Web. In *Workshop on Management of Semistructured Data at PODS/SIGMOD'97*, 1997.
8. C. Hsu and C. Chang. Finite-state transducers for semi-structured text mining. In *IJCAI-99 Workshop on Text Mining*, 1999.
9. B. Katz. Using English for indexing and retrieving. In *RIAO '88*, 1988.
10. B. Katz. Annotating the World Wide Web using natural language. In *RIAO '97*, 1997.
11. T. Kirk, A. Levy, Y. Sagiv, and D. Srivastava. The Information Manifold. Technical report, AT&T Bell Laboratories, 1995.
12. C. Knoblock, S. Minton, J. Ambite, N. Ashish, I. Muslea, A. Philpot, and S. Tejada. The Ariadne approach to Web-based information integration. *International Journal on Cooperative Information Systems*, 10(1/2):145–169, 1999.
13. N. Kushmerick, D. Weld, and R. Doorenbos. Wrapper induction for information extraction. In *IJCAI-97*, 1997.
14. J. Lin. The Web as a resource for question answering: Perspectives and challenges. In *LREC2002*, 2002.
15. J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. Lore: A database management system for semistructured data. Technical report, Stanford University Database Group, February 1997.
16. I. Muslea, S. Minton, and C. Knoblock. A hierarchical approach to wrapper induction. In *3rd International Conference on Autonomous Agents*, 1999.
17. A. Sahuguet and F. Azavant. WysiWyg Web Wrapper Factory. In *WWW8*, 1999.

Automated Question Answering Using Question Templates That Cover the Conceptual Model of the Database

Eriks Sneiders

Department of Computer and Systems Sciences, Stockholm University and the Royal Institute of Technology, Forum 100, SE-164 40 Kista, Sweden
eriks@dsv.su.se

Abstract. The question-answering system developed by this research matches one-sentence-long user questions to a number of question templates that cover the conceptual model of the database and describe the concepts, their attributes, and the relationships in form of natural language questions. A question template resembles a frequently asked question (FAQ). Unlike a static FAQ, however, a question template may contain entity slots that are replaced by data instances from the underlying database. During the question-answering process, the system retrieves relevant data instances and question templates, and offers one or several interpretations of the original question. The user selects an interpretation to be answered.

1 Introduction

[1] groups natural language interfaces into pattern-matching systems, syntax-based systems, semantic grammar systems, and systems with intermediate query representations. The latter combines features of both syntax-based and semantic grammar systems. This short paper describes a prototype of a pattern-matching system, further called *question assistant*, which was first applied to the database on events in Stockholm, the capital of Sweden. The main advantage of a pattern matching system is its simplicity: no sophisticated processing of user questions is needed. The simplicity becomes essential for automated question answering on WWW because many small and medium-size websites cannot afford complex solutions that require much time and rare human skills to install and maintain the system.

A much broader description of this research can be found in [2].

2 Question Templates

The initial framework of this research was automated FAQ answering [3]. FAQ collections are created only for knowledge domains where concepts have rather few instances. No one is likely to write a separate FAQ for each data instance in a large

B. Andersson et al. (Eds.): NLDB 2002, LNCS 2553, pp. 235–239, 2002.

database. We can, however, benefit from the simplicity of automated FAQ answering in the task of creating a question-answering interface for a structured, e.g. relational, database.

Let us introduce a *question template* – a dynamic, parameterized FAQ as opposed to the traditional static FAQ. A question template is a question with entity slots – free space for data instances that represent the main concepts of the question. For example, "When does <performer> perform in <place>?" is a question template where <performer> and <place> are the entity slots. If we fill these slots with data instances that belong to the concepts, we get an ordinary question, e.g., "When does Depeche Mode perform in Globen?"

The question template's "answer" is created by the help of a *database query template* – a formal database query having entity slots for data instances, primarily primary keys. After the slots are filled, the template becomes an ordinary executable database query. The following could be a query template associated with the above question template:

```
SELECT t.time FROM timetable AS t WHERE t.eventid IN
  (SELECT e.eventid FROM event AS e WHERE
   e.placeid = <place> AND e.eventid IN
    (SELECT p.eventid FROM performer AS p WHERE
     p.performerid = <performer>))
```

Processing of a query template and executing the query returns raw data which needs to be formatted and complemented with wrapping text such as a header and footer. The wrapping text complemented with entity slots forms an *answer template*, e.g., "<performer> performs in <place> at time".

We can view a question template as a logical statement with a predicate having variable and fixed parameters, where the fixed parameters are entity slots:

$$\exists\ \text{variable}_1, \text{variable}_2, \ldots \text{variable}_n:$$
$$Q\ (\text{fixed}_1, \text{fixed}_2, \ldots \text{fixed}_m, \text{variable}_1, \text{variable}_2, \ldots \text{variable}_n)$$

If the question has an answer, the above logical statement is true. A database query template embodies the predicate Q, it implements the relationships between the parameters. During the question answering process, the values of the fixed parameters – fixed_1, fixed_2, … fixed_m – are bound to the user question, whereas the values of the variable parameters – variable_1, variable_2, … variable_n – are to be found. In order to answer the user question, the system finds all combinations of the variable parameters that retain the value of Q true.

Question and answer templates correspond to FAQs and their answers. A new concept, nonexistent in automated FAQ answering, is database query template which implements the relationships between the fixed and variable parameters. The templates are created manually with assistance of computerized tools (see [3] about creating an FAQ entry whose structure is similar to that of a question template) because this process requires analysis of the knowledge domain and understanding the structure of the underlying database. Today's technology does not allow automatic analysis of an arbitrary knowledge domain.

Answering a user question takes the following steps. The question assistant:

1. retrieves data instances that are relevant to the user question (see Section 4);
2. retrieves question templates that match the user question (the FAQ retrieval technique [3] is used);
3. combines the retrieved data instances and question templates, and creates one or several interpretations of the original question; the user selects a desired interpretation, and the question assistant answers it.

3 Question Templates and the Conceptual Model of the Database

The similarity between question templates and FAQs suggests that the mission of the question templates is to embody typical questions. Unlike the FAQs, however, the question templates are bound to the underlying structured, e.g. relational, database.

Conceptual modeling [4] is an activity that involves eliciting concepts, their attributes, relationships, and restrictions from a fuzzy knowledge domain. During the conceptualization process, information is transformed into sentences, sentences into elementary sentences, and elementary sentences into object-role pairs. A conceptual model describes which elementary sentences may enter and reside in the information system [5]. Entity slots in question templates are bound to the concepts, or entities, in the conceptual model while the templates themselves express the relationships between the concepts – those elementary sentences. Fig. 1 shows an example of binding a question template to a piece of a conceptual model.

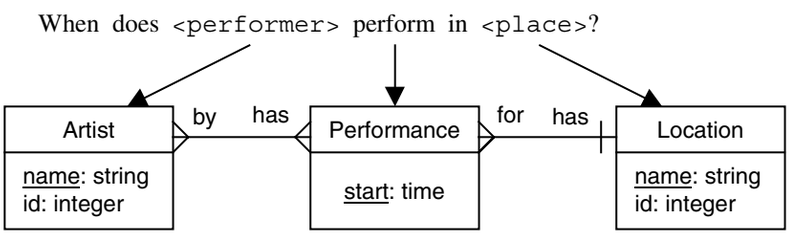


Fig. 1. Question template bound to a piece of a conceptual model

One question template serves a large number of data instances that pertain to its entity slots. If, however, we want to cover a new relationship or attribute, we have to create a new template. The relationships expressed in a question template are static unless we transform them into meta-concepts and treat them as instances in the corresponding entity slots.

During the question-answering process, the question assistant does *not* use the graphical representation of the conceptual model. Instead, it uses an embodiment of the conceptual model in a collection of question templates. The graphical representa-

tion helps when the question templates are created. Whatever information source is used in order to create the templates, they may not contradict the conceptual model as long as such a model is possible.

In conceptual modeling, ISA (”is a“) and PartOf are conventional relationships between concepts. The question assistant perceives these relationships as synonymy and context sensitive synonymy.

Establishing synonymy between several entities is an easy task – we define and use a parent entity. Context sensitive synonymy is slightly more complicated. Let us consider the question ”When does Depeche Mode perform in Globen?“ Because Globen is located in Stockholm (i.e., Globen is a part of Stockholm) and the band arrives from abroad, a more common question would be ”When does Depeche Mode come to Stockholm?“ In this context Globen can be exchanged with Stockholm, in some others cannot. ”How many seats are there in Globen?“ is a reasonable question, whereas ”How many seats are there in Stockholm?“ is not.

There are two options how to handle context sensitive synonymy. First, we can introduce separate equivalent question templates for both entities involved in the PartOf relationship in the context where the synonymy is appropriate. Second, we can define a parent entity and use it only in the context where the synonymy is appropriate.

4 Retrieval of Relevant Data Instances

Retrieval of data instances has three phases: query expansion, retrieval of candidate data instances, and examination of the candidate data instances.

The simplest *query expansion* is stemming of words, which allows matching inflections of the words. The question assistant uses a stem dictionary, which enables better control over the stems and works equally well with English and non-English words, as well as exotic proper names. The stem dictionary is extended by the repository of irregular forms of words and basic synonyms.

After the user query has been expanded, the question assistant selects a number of *data instances that are candidates* for closer examination. A candidate data instance contains at least one word represented in the expanded user query. The data index, which is analogous inverted index or inverted file in Information Retrieval, facilitates the process of selecting the candidates. In the index, a data instance is identified by its primary key and its entity name. Entity names bind the data instances to the entity slots in the question templates, database query templates, and answer templates. The entity names are linked to the physical data carriers – data tables and columns.

The question assistant *examines the candidate data instances* and concludes whether or not a given data instance is relevant to the user query. In the examination process, the question assistant uses entity-specific and data instance-specific rules. *Entity-specific rules* make use of the meaning of the data instance. For example, when the question assistant matches ”John Smith“ or ”Museum of Modern Arts in Stockholm“ to the user question, it knows that first is a person name and second is a place name. Such knowledge is helpful dealing with first and last names, prepositions, numbers, proper names, etc. Unfortunately, general rules tend to have exceptions.

Entity-specific rules cannot cope with particular synonyms such as pseudonyms. A *data instance-specific rule* is a list of phrases attached to a particular data instance. Wherever a data instance-specific rule is defined, it overrides the entity-specific rule.

5 Conclusions

The main contribution of this research is adapting the FAQ answering technique [3] to question answering using data in a structured, e.g. relational, database. When the question-answering system, called question assistant, receives a user question, it matches the question to a number of question templates – dynamic, parameterized "frequently asked questions". Unlike a static FAQ, a question template contains entity slots that are replaced by data instances from the underlying database. The entity slots are bound to the concepts, or entities, in the conceptual model of the database while the templates themselves express the relationships between these concepts in form of natural language sentences.

The main advantage of the question assistant is its simplicity. Being a pattern matching system, it requires no sophisticated processing of user questions. Maintenance of the system does not require rare human sills: the maintainer must have sufficient knowledge of the subject domain, a good command of English, and the ability to write database queries. The main disadvantages are limited sensitivity to the user input and generating interpretations as an intermediate step in the question-answering process.

The question assistant, like any existing natural language interface, is *not* recommended for building the only user interface of a database. The system is recommended in situations where answering of typical questions is appropriate, where the conventional keyword-based search retrieves too much irrelevant information.

References

1. Androutsopoulos, I., Ritchie, G. D., Thanisch, P.: Natural Language Interfaces to Databases – An Introduction. In: Journal of Natural Language Engineering, Vol. 1, No. 1, Cambridge University Press (1995)
2. Sneiders, E.: Automated Question Answering: Template-Based Approach. Ph.D. thesis. Department of Computer and Systems Sciences, Stockholm University and the Royal Institute of Technology, Sweden (2002)
3. Sneiders, E.: Automated FAQ Answering: Continued Experience with Shallow Language Understanding. In: Question Answering Systems. Papers from the 1999 AAAI Fall Symposium, November 5-7, North Falmouth, Massachusetts, USA. Technical Report FS-99-02. AAAI Press (1999) 97-107
4. Loucopoulos, P., Zicari, R. (eds.): Conceptual modeling, databases, and CASE: an integrated view of information systems development. John Wiley & Sons (1992)
5. Nijssen, G. M: Current Issues in Conceptual Schema Concepts. In: Nijssen, G. M (ed.) Architecture and Models in Data Base Management Systems. Proceedings of the IFIP Working Conference on Modelling in Data Base Management Systems, Nice, France, 3-7 January 1977. North-Holland Publishing Company (1977) 31-65

Author Index

Andreasen, Troels, 123
Anker Jensen, Per, 123
Auran, Per Gunnar, 218

Bekhouche, Dalila, 64
Berkovich, Aaron, 40
Bolshakov, Igor A., 172
Bontcheva, Kalina, 160
Brewster, Christopher, 203

Cachero, C., 13
Ciravegna, Fabio, 203
Cunningham, Hamish, 160

Dalianis, Hercules, 183
Delden, Sebastian van, 150
Düsterhöft, Antje, 52

Erdman Thomsen, Hanne, 123

Felshin, Sue, 230
Fischer Nilsson, Jørgen, 123

Gardarin, Georges, 223
Gelbukh, Alexander, 172
Gomez, Fernando, 150
Guan, Jihong, 97
Gulla, Jon Atle, 218

Hamza, Oana, 160
Hausser, Roland, 109

Ibrahim, Ali, 230

Katz, Boris, 230
Kedad, Zoubida, 137
Kou, Huaizhong, 223

Lammari, Nadira, 27
Langlais, Philippe, 191
Lin, Jimmy, 230

Martínez-Barco, P., 13
Martinovic, Miroslav, 208
Marton, Gregory, 230
Maynard, Diana, 160
McFarland, Alton Jerome, 230
Métais, Elisabeth, 27, 137
Meziane, Farid, 85
Moreda, P., 13
Muñoz, R., 13

Paggio, Patrizia, 123
Palomar, Manuel, 13, 213

Rezgui, Yacine, 85
Risvik, Knut Magne, 218
Romary, Laurent, 64

Saggion, Horacio, 160
Sampath, G., 208
Sandford Pedersen, Bolette, 123
Sneiders, Eriks, 235
Sokol, Dan, 40
Storey, Veda C., 1
Suárez, Armando, 213
Sugumaran, Vijayan, 1

Temelkuran, Baris, 230
Thalheim, Bernhard, 52
Thirunarayan, Krishnaprasad, 40
Todirascu, Amalia, 64

Ursu, Cris, 160

Wilks, Yorick, 160, 203

Yuret, Deniz, 230

Zhdanova, Anna V., 76
Zhou, Shuigeng, 97