Departamento de Tecnologías y Sistemas de Información

UNIVERSIDAD DE CASTILLA LA MANCHA

TESIS DOCTORAL

Fuzzy Approach to Conceptual Meaning Processing in Natural Language Documents

Autor: Andrés Soto Villaverde
Director: José Ángel Olivas Varela

Ciudad Real, October 24, 2008

Contents

Table index

Figure index

Resumen

El desarrollo de métodos para la Recuperación de Información partiendo del significado de la misma resulta vital para aumentar la relevancia de los documentos que recuperan los buscadores en la actualidad. En esta tesis se plantean diversos modelos y formulas que ayudan a desambiguar el significado de los términos usados por el usuario en una consulta. Uno de estos modelos se basa en el empleo de la sinonimia y la polisemia para identificar los conceptos más relevantes que aparecen en un documento y de esta manera, caracterizarlo, lo cual redunda favorablemente para determinar la relevancia del documento según los intereses de un usuario dado.

Otro de los modelos planteados en la tesis se basa en el uso de las definiciones que aparecen en el diccionario (i.e. WordNet) para, contabilizando las veces que aparecen los términos de la definición, contabilizar los conceptos asociados a dicha definición que aparecen en el documento.

En la tesis se plantea además un tercer modelo, similar al anterior, en el cual se contabiliza el numero de veces que aparecen las frases nominales incluidas en la definición de una palabra en el diccionario, en lugar de contar los términos que componen dicha definición como se planteo en el método anterior.

En la tesis se presentan los resultados obtenidos mediante algoritmos de clustering que emplean los modelos antes mencionados. Dichos algoritmos se aplicaron a colecciones de prueba conocidas como la SMART y la Reuters con resultados mejores que los obtenidos con los métodos clásicos.

El lenguaje natural (LN) es básicamente un sistema para describir percepciones, las cuales son intrínsecamente imprecisas. Para hacer frente a esta situación Zadeh plantea un nuevo enfoque denominado NL-Computation (Cómputo en Lenguaje Natural, LN), el cual emplea herramientas tales como las Restricciones Generalizadas (RG) y las formas prototípicas (FP). Dada una proposición en LN, si ésta se puede expresar mediante RG, entonces se puede considerar precisa, al menos en cierto grado.

La idea básica propuesta por Zadeh es la siguiente: dada una descripción de una percepción en LN, traducirla en una Restricción Generalizada (RG) para precisar de esta forma su significado. Luego, la RG se transforma en una forma prototípica (i.e. una protoforma, PtF), la cual es un modelo abstracto de RG. Luego, aplicando las reglas deductivas asociadas a las PtF, podemos deducir nueva información.

En la presente tesis, se analizan diferentes estructuras del LN tales como frases nominales, oraciones copulativas, comparativas y superlativas, destacando sus características sintéticas y semánticas. Dichas características permiten especificar restricciones con respecto a las entidades que aparecen involucradas en dichas oraciones. En la tesis se establecen metodologías para reconocer dichas estructuras en documentos en LN. También se propone una notación específica para representar dichas estructuras a manera de relaciones formales restrictivas.

Se ha desarrollado un programa que permite convertir oraciones expresadas mediante árboles sintácticos en estructuras orientadas a objeto, las cuales son utilizadas por el programa para detectar y procesar las oraciones y frases antes mencionadas. Posteriormente otro programa permite interpretar las estructuras orientadas a objeto antes mencionadas y obtener información con respecto a las características de las entidades que aparecen involucradas en dichas oraciones.

También se proponen en la tesis expresiones simbólicas que, a manera de formas prototípicas, resumen de una manera abstracta la estructura semántica de las oraciones y frases antes mencionadas. Se desarrollan ejemplos donde se evidencia el proceso de síntesis y de manipulación de dichas estructuras y se obtiene información que no aparecía reflejada en el documento original, lo cual podría ser aplicado en sistemas automáticos para responder preguntas (Question-Answering Systems)

Abstract

Development of methods for Information Retrieval based on conceptual aspects is vital to reduce the quantity of unimportant documents retrieved by today search engines. In this thesis, several methods and formulas which help to disambiguate the meaning of the terms used in the user queries are presented.

One of these models uses an approach based on synonymy and polysemy in order to identify the most relevant concepts that appear in a document. This way, the document could be better characterized, and its relevance could be better evaluated, according to user preferences.

Another model also introduced in this thesis calculates the frequency of the terms that appear in a dictionary definition in order to determine the frequency of the concept associated with that definition.

A third model is also presented here, which is similar to the previous one, but with one important difference: in spite of calculating the frequency of the terms that appear in a dictionary definition, it calculates the frequency of the nominal phrases which appears in a dictionary definition in order to determine the frequency of the concept associated with that definition.

After that, several results obtained by using those models combined with clustering algorithms are presented in the thesis. Those algorithms were applied to well known test collections as SMART and Reuters, with results that indicate a better performance than the classical approaches.

Natural Languages (NL) are basically a system for describing perceptions which are intrinsically imprecise. Zadeh proposed a new approach denominated NL-Computation (Natural Language Computation), which employs new tools as Generalized Constraints (GC) and protoforms (PtF). Assuming that a NL proposition could be expressed by GC, then it could be assumed precise, al least in certain degree.

The basic idea proposed by Zadeh is the following: given a description of a perception in NL, to translate it into a GC in order to make precise its meaning. Then the GC is transformed into a protoform, which is an abstract model of the GC. After that, applying the deductive rules associated with the PtF, new information could be deduced.

In this thesis, the characteristics of different NL structures such as noun phrases, copulative sentences, comparative sentences and superlative sentences are analyzed, emphasizing their main syntactic and semantic aspects. Those characteristics allow us to specify constraints with respect to the entities that appear involved on those sentences. Methodologies to recognize those structures in NL documents are presented. A specific formal notation to represent those structures as constraining relations is proposed

A program which allows transforming sentences expressed by parse trees into object oriented structures is presented. Those structures are used by the program to store and process conveniently the sentences and phrases previously mentioned. Later, another program interprets the O-O structures and provides information about the characteristics of the entities involved on those sentences.

In the thesis, symbolic expressions that, as prototypical forms, summarize the semantic structure of the sentences and phrases already mentioned are also proposed. Several examples have been developed to show how those structures could be synthesised and manipulated. It is also shown that we can obtain new information that was not present in the original text. Therefore these ideas could be used to develop Question Answering Systems.

Acknowledgements

This doctoral thesis is the result of many years of tenacity and determination, of research and study. Its materialization would not have been possible without the help and suggestions of many people, who showed me the way to do it in one form or another. Here, I want to express my gratitude to all of them.

First of all, my love and recognition to my parents, Enrique A. Soto and Ana Margarita Villaverde, who made me what I am, teaching me dedication, effort, laboriousness, perseverance, and self-confidence. To my whole family, who never stopped encouraging me to continue forward, to fight, and to overcome. To my brother, Enrique Soto, who revised different releases of this thesis. My deepest love and thanks to my wife, Mercedes E. Blanco Mujica, who patiently heard me talking about my plans and ideas, giving me her suggestions and support.

My gratitude goes to my headmaster, MEM Fidel Franco Cocón Pinto, director of the Department of Information Science, Del Carmen University (UNACAR), Campeche, Mexico, and other UNACAR authorities, who trusted in me and gave me his support to obtain the grant that make me possible to attend the PhD program at Castilla La Mancha University, Ciudad Real, Spain.

My deepest gratitude goes to my supervisor, Dr. Jose Angel Olivas Varela, leader of the SMILe group (Soft Management of Internet and Learning) at Castilla La Mancha University, who motivated me to work in this theme. I will be always grateful for his advice and helpful contributions during my research. Furthermore, my gratitude goes to the other SMILe members for their advice, suggestions, and friendship.

To all of you, thank you very much for allowing me to make this old dream come true.

Chapter 1
Introduction

1. Introduction

> "As we move further into the age of machine intelligence and automated reasoning, a daunting problem becomes harder and harder to master. How can we cope with the explosive growth in data, information, and knowledge? How can we locate and infer from decision-relevant information that is embedded in a large database that is unstructured, imprecise, and not totally reliable?"
>
> L. A. Zadeh [153]

The WWW has become a huge repository of human knowledge and information in a scale never seen before. Nobody could imagine its current size and complexity. However, the exponential growth of the Web makes really hard to retrieve all relevant information. In fact, to crawl the Web is perhaps the main bottleneck for Web search engines [8].

Before the introduction of Internet and the World Wide Web (WWW), the areas of Information Retrieval and Question Answering Systems were focus just to deal with more or less special purpose database systems, subject-specific expert systems and the like. The area of IR has grown well beyond its primary goals of indexing text and searching for useful documents in a closed collection with the introduction of the Web. The advent of the Web introduced new problems as: how to manage with the explosive growth in knowledge, information and data and how to locate decision-relevant information which is embedded in a large database and to infer new useful information from it.

Efficacy and relevance levels strongly depend on the fact that most crawlers just look for words or terms without considering their meaning. Those crawlers keep documents indexed by the terms contained in them. Terms are weighted by their frequency in the documents, thus more frequent terms are considered more important. Similarity between a query and a document is considered a function of the matching degree between the terms in the query and the terms in the document, according to the term frequency. Page ranking usually consider that document relevance directly depends on the number of links connected with the document page. Then, those search

systems work based on word matching instead of concept matching. Therefore, search methods should change from only considering lexicographical aspects to considering conceptual ones too [165, 8].

Recent works on the challenges of Web searching include the following problems [8]:

- The Web is full of low quality (syntactic and semantically) content, including noisy, unreliable and contradictory data. Hence, we have the problem of *identifying good quality content*.
- *Exploiting user feedback*, either from explicit user evaluation or implicitly from Web logs.
- *Improving the query language*, adding the context of the information needed.
- *Improving ranking*, in particular to make it dependent on the person posing the query. Relevance is based in personal judgments, so ranking based in user profiles or other user based context information can help.

"Soft computing[1] differs from conventional (hard) computing in that, unlike hard computing, it is tolerant of imprecision, uncertainty, partial truth, and approximation. The guiding principle of soft computing is: Exploit the tolerance for imprecision, uncertainty, partial truth, and approximation to achieve tractability, robustness and low solution cost. The basic ideas underlying soft computing in its current incarnation have links to many earlier influences, among them Zadeh's 1965 paper on fuzzy sets [237]; the 1973 paper on the analysis of complex systems and decision processes [238]; the 1976 paper on fuzzy-algorithmic approach [239], among others".

The Berkeley Initiative in Soft Computing (BISC) Program is the world-leading center for basic and applied research in soft computing, under the direction of Prof. Lotfi A. Zadeh. Some of the most striking achievements of BISC Program are: fuzzy sets and fuzzy logic reasoning, new soft computing algorithms making intelligent, semi-unsupervised use of large quantities of complex data, uncertainty analysis, perception-based decision analysis and decision support systems for risk analysis and management, computing with words, computational theory of perception (CTP), and precisiated natural language (PNL).

[1] BISC Soft Computing home page http://www-bisc.cs.berkeley.edu/

The successful applications of soft computing and the rapid growth of BISC suggest that the impact of soft computing will be felt increasingly in coming years. Soft computing is likely to play an especially important role in science and engineering, but eventually its influence may extend much farther. In many ways, soft computing represents a significant paradigm shift in the aims of computing - a shift which reflects the fact that the human mind, unlike present day computers, possesses a remarkable ability to store and process information which is pervasively imprecise, uncertain and lacking in categorization.

The Berkeley Initiative program Fuzzy Logic and the Internet (FLINT) [2] was formed in 2001. The initial objective of FLINT was to design an intelligent search engine with a high Machine Intelligence Quotient, in the Web context (WebMIQ) based on the advancement in the following areas:

- Add higher level deduction capability
- Precisiation of meaning
- A logic for approximate reasoning
- Information summarization
- Add content to the existing information
- Semantic Web development
- Ontology development
- Mobilization of the knowledge
- Machine-human interaction for better search
- Development of web-question and answering (WQ&A)
- Internet and web services
- Between the FLINT expectations are:
- To design an intelligent search engine with high WebMIQ with higher level deductive capability by 2010.

[2] Nikravesh, M., Fuzzy Logic and The Internet (FLINT) A Glimpse into the Future, http://www-bisc.eecs.berkeley.edu/

- To design an intelligent Q&A search engine with high Q&A MIQ with deductive capability by 2012.
- To add high level deductive capability to the existing search engine by 2010.
- To add Q&A capability to the existing search engine by 2012.

In his speech in the International Conference on Information Reuse and Integration [254], L. A. Zadeh said: "Existing search engines, with Google at the top, have many truly remarkable capabilities…..But what is not widely recognized is that there is a basic capability which existing search engines do not have: deduction capability, the capability to synthesize an answer to a query by drawing on bodies of information which reside in various parts of the knowledge base. By definition, a question-answering system, or a Q/A system for short, is a system which has deduction capability. Can a search engine be upgraded to a question-answering system through the use of …tools which are based on bivalent logic and probability theory? …the answer is: No."

As a result intelligent search engines with high WebMIQ with growing complexity and technological challenges are currently being developed. This requires new technology in terms of understanding, development, engineering design and visualization. While the technological expertise of each component becomes increasingly complex, there is a need for better integration of each component into a global model adequately capturing the imprecision and deduction capabilities.

Six years from the formation of the FLINT and 5 years from its International event at UC Berkeley, FLINT launches a new Initiative on August 10, 2006: Computational Intelligence for Information and Internet Search in the Interest of the Society (COINS)[3].

Zadeh, in his conferences [260, 261, 262] introduced a new frontier in computation: Computation with Information Described in Natural Language, or NL-Computation, for short. There, he said:

"Computation with information described in natural language is closely related to Computing with Words. NL-Computation is of intrinsic importance because much of human knowledge is described in natural language…. Computation with information described in natural language cannot be dealt with through the use of machinery of

[3] COINS http://www-bisc.eecs.berkeley.edu/FLINT/News.htm

natural language processing. The problem is semantic imprecision of natural languages."

NL Computation cannot be dealt with through the use of machinery of natural language processing, because of the semantic imprecision of natural languages. A natural language is basically a system for describing perceptions, which are intrinsically imprecise, reflecting the bounded ability of sensory organs, and ultimately the brain, to resolve detail and store information.

Zadeh approach to NL-Computation centres on what is referred to as generalized-constraint-based computation or GC-Computation for short. A fundamental thesis, which underlies NL-Computation, is that information may be interpreted as a generalized constraint.

According to Zadeh [262], NL-Computation involves three modules:

- Precisiation module, which precisiate the meaning of natural language elements by translating them into generalized constraints.
- Protoform module, which serves as an interface between Precisiation and Computation modules. Basically, its function is that of abstraction and summarization.
- Computation module which serves to deduce query answers.

Google, Yahoo and the other existing search engines are able to retrieve millions of page references in less than a second. But they do not have the capability to synthesize an answer to a query by drawing on bodies of information which reside in various parts of the knowledge base, that is, deduction capability. They are not able also to deal with the problems of world knowledge, assessment of relevance and deduction. Therefore new tools are required to deal with those necessities and problems. Some of these new tools proposed to do this are Generalized Constraints and Protoform Theory (PFT) [246].

The Soft Management of Internet and Learning (SMILe) research group [143] at Castilla-La Mancha University is deeply involved in the development of Information Retrieval methods for the World Wide Web based on conceptual characteristics of the information contained in documents. Several models and tools have been developed by the members of the group, such as:

- FISS (Fuzzy Interrelations and Synonymy based Searcher) Metasearcher [50, 141] which incorporates a soft clustering component whose similarity function considers the cooccurrence of concepts of the retrieved documents. The final result of the search process is a set of groups of conceptually related web pages.
- FIS-CRM (Fuzzy Interrelations and Synonymy Conceptual Representation Model) [50, 141] an extension of the vector space model, which provides a mechanism to represent the concepts contained in a document. This model is based on the study of two types of fuzzy interrelations (synonymy and generality).
- FzMail [169]: a portable and easy-to-use tool for automatically classifying emails into a fuzzy hierarchical structure. FzMail uses FIS-CRM to achieve a conceptual representation of the messages, and a softclustering algorithm for categorizing the documents and the defined folders.
- GUMSe [39, 142] a meta-search engine architecture based on agents, WordNet and the ontologies generated according to the documents recovered in each query and able to represent the user's most demanded ones.
- T-DiCoR (Three Dimensional Conceptual Representation) [140] a tool for representing the fuzzy relations among the most representative concepts of a domain. T-DiCoR shows the user a three-dimensional form, a graph with the form of molecule, where the nodes are the most relevant concepts of a domain and the edges show the relations that join them.

1.1. Scope and Goals

This thesis is enclosed into the research lines of the SMILe group previously mentioned. It is concerned with considering the conceptual aspects of the information contained in documents in order to improve search engine results. The approaches proposed into the thesis are more oriented to the meaning of the terms and their semantic in spite of the lexicographic aspects as the number of times that it appears in a document.

At the same time, the thesis is concerned with retrieving information from documents which can be used to characterize user preferences and, therefore, to construct a user profile which can be helpful in query expansion to retrieve more relevant information. It is also concerned with retrieving document information which can be used to deduce new pieces of information not previously contained into them.

This thesis has been partially supported by SCAIWEB PAC06-0059 project, JCCM, Spain, and TIN2007-67494 *F-META* project, MEC-FEDER, Spain. It has also been partially supported by UACAR-048, a PhD scholarship grant provided by SEP, Mexico. This thesis has been developed during the period 2006-2008. Partial results related with the thesis were presented in:

- XIV Spanish Congress about Fuzzy Technology and Fuzzy Logic, ESTYLF 2008, Mieres-Langreo, 2008.
- X National Congress of Mathematics and Computer Science COMPUMAT 2007, November 21-23, Holguín, Cuba, 2007.
- 5th Conference of the European Society for Fuzzy Logic and Technology EUSFLAT 07, Ostrava, Czech Republic, September 11-14, 2007.
- IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2007, Imperial College, London, UK, July 23-26, 2007.
- International Fuzzy Systems Association IFSA 2007 World Congress, Cancún, México, June 18-21, 2007.
- European Working Group on Fuzzy Sets workshop EUROFUSE'07, New Trends In Fuzzy Preference Modelling Jaen, Spain, April 11-13 2007.

- Eleven International Conference on Computer Aided Systems Theory, EUROCAST-2007, Las Palmas de Gran Canaria, Spain, February 12-16, 2007.
- International Symposium on Fuzzy and Rough Sets, ISFUROS 2006, Santa Clara, Cuba, December 2006.

Some partial results from the thesis were published in

- Theoretical Advances and Applications of Fuzzy Logic and Soft Computing, Advances in Soft Computing 42, Springer Verlag, 2007, pages 171-179.
- Granular Computing: At the Junction of Rough Sets and Fuzzy Sets, Studies in Fuzziness and Soft Computing, Springer Verlag Berlin / Heidelberg, Volume 224/2008, Pages 179-198.

The main goals of the thesis are:

1. To take into account the semantic of the words in order to obtain a higher level of relevance during searching, retrieval and management of documents in natural language. In this direction, the following relations between words are analyzed into the thesis:
    a. Synonymy and polysemy relations.
    b. The relation between a word and its definition; that is, between the term which is defined (i.e. definiendum in Latin), and the terms used in its definition (i.e. definiens also in Latin).
    c. Relations between word types like adjective and nouns, adverbs and adjectives, etc.
    d. Sentence structures like noun phrases, copula sentences, comparative sentences.
2. To develop mathematical models based on Fuzzy Logic, which reflect the aspects mentioned in goal 1).
3. To develop computational methods and algorithms which allow implementing the previously mentioned models.
4. To develop programs able:

   a. To process the information contained in documents collection like SMART and Reuters, taking into account the semantic aspects mentioned in goal 1.

   b. To elaborate indexes based on those semantic aspects which allowed an efficient retrieval of the information contained into the documents.

5. To carry out experiments with document collections such as SMART and Reuters, in order to evaluate the developed techniques.
6. To explore the possibility to extract information from documents expressed in natural language (NL) and to deduce new information, information which was not present in the original document.
7. To build a bridge between natural languages and mathematics by trying to precisiate the meaning of natural language propositions via constraints.
8. To identify patterns in sentences and phrases which allow us to represent them by formal relations.
9. To use those formal relations and knowledge bases to deduce new pieces of information in simple situations.
10. To investigate the application of Generalized Constraints and Protoforms as defined by Zadeh to represent the previously mentioned relations and to operate with them in order to infer new pieces of information.

1.2. Contributions

In pursuit of the previously mentioned goals, this thesis makes the following contributions:

1. Software which allows measuring the presence of concepts in documents by means of synonymy and polysemy has been developed. Based on the defined formulas, even though a certain term does not appear in a document, it is possible to estimate its presence according to the degree of synonymy shared with terms that do appear in the document.

2. Based on measuring the presence of the definiens (i.e. the terms used in its definition) of certain meaning in one or more documents another program was also developed, which allows measuring the presence of concepts in documents.
3. Programs using both models were developed to perform a clustering algorithm, so that documents within a cluster have high conceptual similarity. The experiment results demonstrate the quality and effectiveness of the models.
4. In this thesis, the possibility to extract information from documents expressed in natural language (NL) and to deduce new information, information which was not present in the original document is explored.
5. To build a bridge between natural languages and mathematics by trying to precisiate the meaning of natural language propositions via constraints is explored in the thesis.
6. Although natural languages are intrinsically imprecise, we try to identify patterns in sentences and phrases which allow us to represent them by formal relations.
7. Using the formal relations obtained, we will use them to obtain procedures which allow deducing new pieces of information are.
8. The role of those formal relations as Generalized Constraints and Protoforms will be analyzed as well as the rules which govern constraint propagation.
9. Software oriented to identify patterns in sentences and phrases and to represent them by formal relations will be developed and explained. This software will also allow manipulating those formal relations.

1.3. Outline

The thesis is structured in seven Chapters (including this introductory one). The chapters cover the main subdivisions of any research work: background, development and application. Each chapter begins with an overview and ends with a summary of the most important addressed aspects. The main topics of each section will be introduced in what follows.

- Chapter 1: Introduction. Explains the Scope and Goals of the present thesis, its Contributions, and an Outline of it.
- Chapter 2: State of the Art. Introduces general aspects about Information Retrieval and Question Answering Systems (QAS); Knowledge, Relevance, Deduction, Precisiation, and Perceptions; Computing with Words, and Granular Computing; Natural Language Processing.
- Chapter 3: Fuzzy models for Information Retrieval. Presents several fuzzy models developed as part of the thesis for IR: Fuzzy Model for Synonymy and Polysemy; Fuzzy Measure of Meaning Presence in Documents; Using Noun Phrases to Measure Concept Presence in Documents; User Profile clustering and Query expansion.
- Chapter 4: From Natural Languages to Generalized Constraints and Protoforms. Presents the possibility to use natural language structures (adjectives, adverbs, copular and comparative sentences) as constraints and how to translate them into Generalized Constraints first and then to Protoforms and to deduce new information.
- Chapter 5: Implementation aspects. Explains how were implemented the different fuzzy models defined for Information Retrieval and previously explained. The algorithms dedicated to recognize and extract the natural language constraints and its translation to Generalized Constraints and Protoforms are also explained.
- Chapter 6: Experimental Results. Presents the results obtained using the fuzzy models for Information Retrieval introduced in Chapter 3. The results obtained while extracting constraints from natural language documents are also presented.
- Chapter 7: Conclusions are dedicated to a review of the thesis as a whole by including a brief summary of the thesis main points, remarking its contributions, and proposing open questions for future work.

Finally, a list of references as well as a thematic index is also included. Appendixes provide additional material about implementation details.

Chapter 2
State of the Art

2. State of the Art

Most researches starts with a literature review of some sort. By literature reviews, researchers manage to identify and scope their future research activities. In the case of a PhD thesis, a systematic review "should identify the existing basis for the research student's work and make it clear where the proposed research fits into the current body of knowledge" [81].

This literature review is an attempt to identify and resume the information currently available about Generalized Constraints and Protoform Theory and its possible application to locate decision-relevant information embedded in the Web and to infer new useful information from it. This review will also serve to clarify where the proposed research fits into the current body of knowledge by pointing to the currently open research topics.

The reasons for this review are:

- To collect, organize and summarise the existing evidence concerning Generalized Constraints and Protoform Theory and its possible application in Question Answering Systems.
- To establish an appropriated background for the PhD research activities.

Therefore, the review objectives are to answer accurately the following questions:

- What is a Generalized Constraint? Which kinds of Generalized Constraints exist? What is a Generalized Constraint Language?
- Which is the meaning of Protoform? Which are the bases of Protoform Theory? What is a Protoform Language?
- Which is the relation between Generalized Constraint Language, Protoform Language and Natural Languages?
- Which is the relation of this theory with other related concepts and theories as Computing with Words, Granular Computing, Theory of Fuzzy Information Granulation, Computational Theory of Perceptions, Precisiated Natural Language?

- Which are the possible applications of Generalized Constraints and Protoform Theory to Question Answering Systems and Information Retrieval?

The searching strategy of the review has been to look for the relevant literature:

- directly by Google search engine.
- using Topic Alert service of ScienceDirect, Elsevier and Google Web Alert service about the topics:
  - Information Systems,
  - Computer Science,
  - Computational Intelligence,
  - Artificial Intelligence,
  - protoform fuzzy,
  - "prototypical form" fuzzy,
  - softcomputing "information retrieval", and
  - "fuzzy logic" "information retrieval".

The information collected that way is processed, studied and organized carefully. To keep it organized the following software products are used:

- MyWeb 2.0, Yahoo Search: helps to find, save, and share knowledge about web pages existing on the Web. It allows user :
  - To save page links so it is easy to re-find them
  - To connect with other users in order to share interesting links.
  - To browse tags to identify active taggers on any topic.
- Google Desktop, Google: helps the user:
  - To search for computer files as easily as to search for pages in the web with Google.

The main inclusion criterion used was to include all the bibliography (papers, abstract, and web pages) which refer to the terms prototypical forms or protoforms. Because those terms are also apply for car spare parts and even in the fictional world of

the Transformers toy line, then it was necessary to restrict it to the sense used in the Fuzzy Logic field. Some other bibliography about Information Retrieval and Question Answering Systems was also included in order to set the background terms used.

2.1. Information Retrieval

The importance of archiving and finding information has been realized by people from the Ancient times of the Humanity. With the advent of computers, to store large amounts of information became possible. Therefore, finding useful information from such collections became an aim. The field of Information Retrieval (IR) was born out of this necessity.

Information retrieval (IR) has changed considerably since Calvin Mooers coined the term at MIT in 1948-50. In 1945 Vannevar Bush gave birth to the idea of accessing large amounts of stored knowledge automatically [23]. In the 1950s, the idea materialized into more concrete descriptions of how archives of text could be searched automatically. Several works emerged in the mid 1950s based on the idea of searching text with a computer. H.P. Luhn in 1957 [108] proposed to use words as indexing units for documents and to measure word overlap as a criterion for retrieval.

In the 1960s, several key developments in the field happened. The development of the SMART system by Gerard Salton [179] was one of the most notable. The Cranfield evaluations done by Cyril Cleverdon and his group at the College of Aeronautics in Cranfield were other of those key developments [32]. The evaluation methodology for retrieval systems that they developed is still in use today.

The Internet and the World Wide Web are two of the great successes in the history of computing. The Internet is not a single homogeneous network, but an interconnected group of independently managed networks. It is sometimes called an information highway. The Web is a linked collection of information on many computers on the Internet around the world, known as web servers [5].

Many of the original concepts of the Internet came from Xerox's Palo Alto Research Centre. In the United States, universities were pioneers in expanding small local networks into campus-wide networks. The second source of network developments was the national networks, known as wide area networks (WAN). During the late 1980s the universities and the research communities converged to create the Internet that we know today. A key event was the 1986 decision by the National

Science Foundation to build a high-speed network backbone for the United States upon the Arpanet's technical achievements and thus set the standards for the Internet.

The 1970s and 1980s saw many developments built on the advances of the 1960s. Various models for doing document retrieval were developed and advances were made along all dimensions of the retrieval process. These new models/techniques were experimentally proven to be effective on small text collections (several thousand articles) available to researchers at the time. However, due to lack of availability of large text collections, the question whether these models and techniques would scale to larger corpora remained unanswered.

The situation changed in 1992 with the launch of Text Retrieval Conferences (TREC) [65] sponsored by US Government, which aims at encouraging research in IR from large text collections. With large text collections available under TREC, many new techniques were developed to do effective retrieval over large collections.

The area of IR has grown well beyond its primary goals of indexing text and searching for useful documents in a closed collection with the introduction of the World Wide Web (WWW) in the beginning of the 1990s. The web technology was developed about 1990 by Tim Berners-Lee and colleagues at CERN, the European research centre for high-energy physics in Switzerland. The Web became popular by the creation of a user interface, known as Mosaic, developed by Marc Andreessen and others at the University of Illinois, Urbana-Champaign in 1993. Numerous commercial versions of Mosaic followed, like the Netscape Navigator and Microsoft's Internet Explorer. These user interfaces were called web browsers, or simply browsers.

The algorithms developed in IR were the first ones to be employed for searching the World Wide Web from 1996 to 1998. Web search, however, matured into systems that take advantage of the cross linkage available on the web, and is not a focus of the present thesis.

WWW has become a huge repository of human knowledge and information in a scale never seen before. The basic reason for the success of the web is because it provides a convenient way to distribute information over the Internet. Individuals can publish information and users can access that information by themselves easily. Their computers were already connected to a local network and hence to the Internet. Since

the Internet covers the world, huge numbers of people had immediate access to this information.

Despite of it success, WWW has introduced new problems as the difficulty to find useful and relevant information on it. In the last years, the amount of information on the World Wide Web has increased enormously. Estimations of the International Data Corporation[4] indicates that between 1999 and 2003 the volume of data published in the Web was equivalent to all the information generated by the humanity from the antiquity to 1998. Web content in 2000 was of various terabytes (a terabyte or Tb are a billion of megabytes) of text, images, audio and video [7]. According to Lyman and Varian[5], between 1999 and 2002 new stored information grew about 30% a year. In 2002, the Web use to contain about 170 terabytes of information on its surface (i.e. fixed web pages), which means seventeen times the size of the USA Library of Congress print collections in volume. As Population Reference Bureau[6] registered a world population of around 6.3 billions in 2003, then almost 800 MB of recorded information was produced per person each year those years. Around January 2003, SearchEngineWatch.com[7] reported 319 million searches per day at the major search engines. Whois.Net, the Domain-Based Research Services[8], reported an increase of 30% in the number of domains registered from 32 millions in 2003 to 95 millions in June 2006.

Although Internet is still the newest information medium, it is the fastest growing medium of all times. World Wide Web (WWW) grows faster than the capacity to detect its changes. Connections inside it are dynamic and many of them became obsolete without being updated. The number of Internet users has more than doubled since the year 2000. In 2006, Internet was available to over 1 billion people worldwide according to Global Internet Statistics Overviews[9].

The computer network that interconnects the globe, in 2000 exceeded the 77 million computers connected in more than 220 countries. Web servers also grow

[4] International Data Corporation, http://www.idc.com/home.jhtml
[5] Lyman, Peter and Hal R. Varian, "How Much Information", University of California, Berkeley, 2003. http://www.sims.berkeley.edu/how-much-info-2003
[6] World Population Data Sheet. 2005, Population Reference Bureau (PRB). http://www.prb.org/
[7] Searches Per Day 2006. Danny Sullivan, Editor-In-Chief. http://searchenginewatch.com/reports/article.php/2156461
[8] Whois.Net: Domain-Based Research Services 2006. www.whois.net
[9] Global - Internet - Statistics Overviews, BuddeComm http://www.budde.com.au/

exponentially since 1993. It is estimated that at the end of 1999 there were at less seven million servers [7].

Following, different definitions of Information Retrieval from some important sources are shown:

- IR aims at modelling, designing, and implementing systems able to provide fast and effective content-based access to large amounts of information. The aim of an IR system is to estimate the relevance of information items to a user information need expressed in a query. This is a very hard and complex task, since it is pervaded with subjectivity, vagueness and imprecision [8].
- An information retrieval system does not inform (i.e. change the knowledge of) the user on the subject of his inquiry. It merely informs on the existence (or non-existence) and whereabouts of documents relating to his request [93, 166].
- Information retrieval (IR) is the art and science of searching for information in documents, searching for documents themselves, searching for metadata which describes documents, or searching within databases, whether relational stand alone databases or hypertext networked databases such as the Internet or intranets, for text, sound, images or data.
- The study of systems for indexing, searching, and recalling data, particularly text or other unstructured forms[10].
- The techniques of searching for data that have been stored in a computer[11].
- Information retrieval is usually used as a generic term to cover the access to and delivery of information from natural language databases by whatever method. Usually the information is delivered in the form of complete documents[12].
- The science and practice of identification and efficient use of recorded data[13].

[10] Glossary, Virtech E-Solutions-Search Engine Optimization www.virtechseo.com/seoglossary.htm
[11] Jargon Management, India Infoline Ltd. www.indiainfoline.com/bisc/acci.html
[12] SILK Project, Norwegian Directorate for Public Libraries. portal.bibliotekivest.no/terminology.htm
[13] Information Society Technologies (IST), European Sixth Framework Program (FP6), www.cordis.lu/ist/ka1/administrations/publications/glossary.htm

- Searching a body of information for objects that match a search query [5].
- Information retrieval (IR) deals with the representation, storage, organization of, and access to information items. The representation and organization of the information items should provide the user with easy access to the information in which he is interested [9].
- A field of specialization in computer science that looks at systematic ways of storing and retrieving data, including consideration of database design and implementation [138].

There are other terms as data retrieval, text retrieval, document retrieval, that could be confused with Information Retrieval. At first glance, data retrieval could be considered as a branch of Information Retrieval where the information stored and retrieved is just data. But the user of an IR system is concerned more with retrieving information about a subject than with retrieving data which satisfies a given query. While data retrieval should provide a solution to the user of a database system, it does not solve the problem of retrieving information about a subject or topic. A data retrieval system deals with data with a well defined structure and semantics. A data retrieval system should retrieve all objects which satisfy the user query. Therefore, a data retrieval system should not retrieve a single erroneous object.

An IR system, however, could retrieve inaccurate objects and small errors are unimportant. That is because IR systems usually deal with unstructured information in natural language, which is most of the time semantically ambiguous. To satisfy the user information need, an IR system must somehow ‘interpret’ the content of a collection of documents and rank them according to a degree of relevance to the user query. This content ‘interpretation’ involves extracting syntactic and semantic information from the document text and using this information to match the user query.

Document retrieval refers to the process of searching and retrieving documents stored in some database by the matching of user queries against the database records. These records could be any type of unstructured text, such as bibliographic records, newspaper articles, or paragraphs in a manual. By the other hand, text retrieval could also be considered as a branch of IR where the information is stored in text form.

Google, Yahoo, AOL Search and MSN Search are some of the most important Web search engines today. They are able to retrieve millions of page references in less

than a second. Therefore they have a high level of efficiency. Unfortunately, most of the information retrieved could be considered irrelevant. For that reason, efficacy level could be considered poor since a user could receive millions of documents for her/his query but just few of them are useful.

An information retrieval process begins when a user enters a query into the system, i.e. a formal statement of information needs. In information retrieval, a query usually identifies several objects that may match the query, probably with different degrees of relevancy. An object is an entity which keeps or stores information in some database. Depending on the application the objects may be, for example, text documents, images or videos.

Most IR systems compute a numeric score on how well each one of the objects matches the query, and rank the objects according to this value. The top ranking objects are then shown to the user. The process may then be iterated if the user wishes to refine the query.

Many different measures for evaluating the performance of information retrieval systems have been proposed. All those measures assume to have a document collection and a query, and that every document is known to be either relevant or non-relevant to a particular query. In practice queries may be ill-posed and there may be different shades of relevancy.

- **Precision** is the fraction of the documents retrieved that are relevant to the user's information need. Precision takes into account all retrieved documents.

$$precision = \frac{|relevant \cap retrieved|}{|retrieved|} \qquad 1$$

- **Recall** is the fraction of the documents that are relevant to the query that are successfully retrieved. It can be considered as *the probability that a relevant document is retrieved by the query*. It is also known as sensitivity. It is trivial to achieve a 100% recall by returning all documents in response to any query. Then, it is also necessary to take into account the number of non-relevant documents.

$$recall = \frac{|relevant \cap retrieved|}{relevant} \quad (2)$$

- **Fall-Out** is the proportion of non-relevant documents that are retrieved, out of all non-relevant documents available. It can be considered as *the probability that a non-relevant document is retrieved by the query*. It is trivial to achieve fall-out of 0% by returning zero documents in response to any query.

$$fall\_out = \frac{|non\_relevant \cap retrieved|}{|non\_relevant|} \quad (3)$$

- **F-measure** is the weighted harmonic mean of precision and recall, the traditional F-measure or balanced F-score. It is also known as the F_1 measure, because recall and precision are evenly weighted.

$$F = \frac{2 \cdot (precision \cdot recall)}{precision + recall} \quad (4)$$

The general formula for non-negative real ß is:

$$F = \frac{(1 + \beta^2) \cdot (precision \cdot recall)}{\beta^2 \cdot precision + recall} \quad (5)$$

The F-measure is based on van Rijsbergen effectiveness measure [166] so that F_β "measures the effectiveness of retrieval with respect to a user who attaches ß times as much importance to recall as precision". Then,

$$F_\beta = 1 - E \qquad 6$$

$$\text{where } E = 1 - \left(\frac{1}{\frac{\alpha}{precision} + \frac{1-\alpha}{recall}} \right)$$

$$\text{and } \alpha = 1 \Big/ \left(\beta^2 + 1\right)$$

- **Average precision** is the average of the precision after each relevant document is retrieved. Average precision emphasizes returning more relevant documents earlier, while the precision and recall are based on the whole list of documents returned by the system.

$$\mu_P = \frac{\sum_{r=1}^{N} \left(P_r \cdot rel_r\right)}{\left|relevant\right|} \qquad 7$$

Where r is the rank, N is the number of retrieved documents, rel( ) is a binary function on the relevance of a given rank, and P is the precision at a given cut-off rank.

Information Retrieval Models

The goal of **information retrieval** (IR) is to provide users with those documents that will satisfy their information need. Users have to formulate their information need in a form that can be understood by the retrieval mechanism. Likewise, the contents of large document collections need to be described in a form that allows the retrieval mechanism to identify the potentially relevant documents quickly. Several models have been proposed for Information Retrieval. Following the three most used of them are described [190]: the vector space model, the probabilistic models, and the inference network model.

Vector Space Model

In the vector space model, text is represented by a vector of *terms* [180]. Terms are typically words and phrases. If words are chosen as terms, then every word in the vocabulary becomes an independent dimension in a very high dimensional vector space.

Any text can then be represented by a vector in this space. If a term belongs to a text, it gets a non-zero value in the text-vector along the dimension corresponding to the term. Since any text contains a limited set of terms, most text vectors are very sparse.

To assign a numeric score to a document in correspondence with a query, the model measures the *similarity* between the query vector and the document vector. The similarity between two vectors is not inherent in the model. Typically, the angle between two vectors is used as a measure of divergence between the vectors, and cosine of the angle is used as the numeric similarity, since cosine has the nice property that it is 1.0 for identical vectors and 0.0 for orthogonal vectors. As an alternative, the inner or dot-product between two vectors is often used as a similarity measure. If all the vectors are forced to be unit length, then the cosine of the angle and the dot-product are equal. If D is the document vector and Q is the query vector, then the similarity of the document to the query can be represented as:

$$Sim(D,Q) = \sum_{t \in D \cap Q} w_{t,D} \cdot w_{t,Q} \qquad 8$$

where w_{tD} is the weight associated to t on vector D, and w_{tQ} is the weight of t on vector Q. Notice that w_{tD} is equal 0 (respectively w_{tQ} is equal 0) for any word not present in either the document or the query; thus $w_{tD.}\, w_{tQ}$ is also equal 0 for any term not included in the intersection between both vectors.

Probabilistic Models

This IR model family is based on *the probabilistic ranking principle* (PRP), which states that documents in a collection should be ranked by decreasing probability of their relevance to a query [166]. Since real probabilities are usually unknown, probabilistic IR models should be used to estimate the relevance probability of documents for a given query. The initial idea of probabilistic retrieval was proposed by Maron and Kuhns [113]. Following the common basis for these models [113, 209] will be described.

Let's suppose that there is a document D and a query Q. Thus, two possibilities exist:

- R : D is relevant to Q; otherwise
- $\bar{R}$: D is not relevant to Q.

The probability of relevance for D will be denoted by $P(R|D)$, i.e. the probability that a document is relevant whatever description it has. Since this ranking criteria is monotonic under log-odds transformation, documents could be ranked by

$$rank = \log \frac{P(R \mid D)}{P\left(\bar{R} \mid D\right)} = \log \frac{P(D \mid R) \cdot P(R)}{P\left(D \mid \bar{R}\right) \cdot P\left(\bar{R}\right)} = \log \frac{P(D \mid R)}{P\left(D \mid \bar{R}\right)} + \log \frac{P(L)}{P\left(\bar{L}\right)} \qquad 9$$

Assuming that the probability of relevance, P(R), is independent of the document under consideration and thus constant across documents, then P(R) and P($\bar{R}$) are just scaling factors for the final document scores and can be removed for ranking purposes from the above formulation. This further simplifies the above formulation and the idea of matching score is introduced:

$$MS = \log \frac{P(D \mid R)}{P\left(D \mid \bar{R}\right)} \qquad 10$$

In the simplest form of this model, by the *independence assumption*, terms are assumed to be mutually independent. Then P(D|R) could be re-written as a product of individual term probabilities:

$$P(D \mid R) = \prod_{t \in D \cap Q} P(t \mid R) \cdot \prod_{t \in \bar{D} \cap Q} (1 - P(t \mid R)) \qquad 11$$

Equation 11 is composed by two products: the first one considers the probability of presence of a term in relevant documents for those terms included both in the query and

in the document. The second product considers the probability of absence of a term in relevant documents for those terms that are present in the query and absent from the document. Then substituting P(t|R) by p and P(t|$\bar{R}$) by q, equation 10 is transformed into:

$$\log \frac{\prod_{t \in D \cap Q} p \cdot \prod_{t \in \bar{D} \cap Q} (1-p)}{\prod_{t \in D \cap Q} q \cdot \prod_{t \in \bar{D} \cap Q} (1-q)} \qquad 12$$

For a given query, we can add to this a constant

$$\log\left(\prod_{t \in Q} \frac{1-p}{1-q} \right) \qquad 13$$

to transform the ranking formula to use only the terms present in a document:

$$\log\left(\prod_{t \in D \cap Q} \frac{p \cdot (1-q)}{q \cdot (1-p)} \right) = \sum_{t \in D \cap Q} \log\left(\frac{p \cdot (1-q)}{q \cdot (1-p)} \right) \qquad 14$$

Different assumptions for estimation of p and q yield different document ranking functions.

Inference Network Model

The inference network model was proposed by Howard Turtle [215] in 1991. It is based on the Bayesian network mechanism [119]. A Bayesian network is an acyclic directed graph (i.e. a graph without cycles) that encodes probabilistic dependency relationships between random variables. The presentation of probability distributions as directed graphs makes it possible to analyse complex conditional independence assumptions by following a graph theoretic approach. In practice, the inference network model is comprised of four layers of nodes: document nodes, representation nodes,

query nodes and the information need node. Figure 1 shows a simplified inference network model.

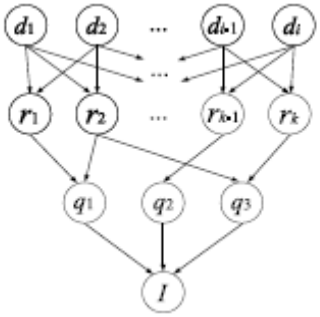


Figure 1 A simplified network model

All nodes in the network represent binary random variables. Let's concentrate into a graph subset to show how it works in theory, for instance the nodes r_2, q_1, q_3 and I. By the chain rule of probability, the joint probability of nodes r_2, q_1, q_3 and I is:

$$P(r_2, q_1, q_3, I) = P(r_2) \cdot P(q_1 \mid r_2) \cdot P(q_3 \mid r_2, q_1) \cdot P(I \mid r_2, q_1, q_3) \qquad 15$$

The dependence relations between random variables are suggested by the direction of the arcs. The event "information need is fulfilled" ($I = 1$) has two possible causes: either query node q_1 is true, or query node q_3 is true (notice that q_2 is not being considered). The two query nodes in turn depend on the representation node r_2. So, the model makes the following conditional independence assumptions.

$$P(r_2, q_1, q_3, I) = P(r_2) \cdot P(q_1 \mid r_2) \cdot P(q_3 \mid r_2) \cdot P(I \mid q_1, q_3) \qquad 16$$

On the right-hand side, the third probability measure is simplified because q_1 and q_3 are independent given their parent r_2. The last part $P(I|q_1, q_3)$ is simplified because I is independent of r_2 given its parents q_1 and q_3.

Unfortunately, straightforward use of the network is impractical if there are a large number of query nodes. The number of probabilities that have to be specified for a node

grows exponentially with its number of parents. For example, a network with n query nodes requires the specification of 2n+1 possible values for the information need node. For this reason, all network layers should use some kind of approximation. In Metzler and Croft [119] the following approximations are described: they assume for the document layer that only a single document is observed at a time, and for every single document a separate network is constructed for which the document layer is ignored. The representation node probabilities (which are effectively priors now, because the document layer is ignored) are estimated by some retrieval model for the representation layer of every network. Finally, the query nodes and the information need node are approximated by standard probability distributions defined by the believe operators. These operators combine probability values from representation nodes and other query nodes in a fixed manner. If the values of $P(q_1|r_2)$, and $P(q_3|r_2)$ are given by p_1 and p_2, then the calculation of $P(I|r_2)$ might be done by operators like *and*, *or*, *sum*, and *wsum*.

$$
\begin{aligned}
P_{and}(I \mid r_2) &= p_1 \cdot p_2 \\
P_{or}(I \mid r_2) &= 1 - ((1 - p_1)(1 - p_2)) \\
P_{sum}(I \mid r_2) &= (p_1 + p_2)/2 \\
P_{wsum}(I \mid r_2) &= w_1 \cdot p_1 + w_2 \cdot p_2
\end{aligned}
\qquad 17
$$

It can be shown that for these operators so-called link matrices exists, that is, for each operator there exists a definition of for instance $P(I|q_1, q_3)$ that can be computed as shown in equation 17. So, although the link matrix that belongs to the operator may be huge, it does not exist in practice and its result can be computed in linear time. One might argue though, that the approximations on each network layer make it questionable if the approach still deserves to be called a ‘Bayesian network model’.

2.2. Question Answering Systems (QAS)

A question-answering system may be viewed as a system which mechanizes question-answering. A search engine is a system which partially mechanizes question answering. A search engine is primarily a provider of topic relevant information. A search engine user exploits its capability to look for an answer to a question [254].

Question Answering Systems (QAS) are a special class of Information Retrieval Systems, where the system should be able to answer questions posed in natural language from a collection of documents. Search collections could include local document collections as well as the World Wide Web. By the other hand, QAS could be classified in:

- Closed-domain: which are oriented just to answer questions about some specific domain.
- Open domain: which are supposed to answer questions about almost any topic.

The first QAS were developed in the 1960s and they were basically controlled natural-language interfaces to expert systems oriented to specific domains like applications in Medicine. Some of the early Artificial Intelligence systems were QAS. Two of the most famous QAS at that time were BASEBALL and LUNAR. BASEBALL answered questions about the US baseball league over a period of one year, while LUNAR answered questions about the geological analysis of rocks returned by the Apollo moon missions. Both QAS were very effective in their chosen domains. 1970-1980 was a period of intense interest in question answering and expert systems [254].

In the late 1990s the annual Text Retrieval Conference (TREC)[14] included a question-answering track which has been running until the present. Systems participating in this competition were expected to answer questions on any topic by searching a corpus of text that varied from year to year. This competition fostered research and development in open-domain text-based question answering.

Current QAS use text documents in the World Wide Web as their underlying knowledge source and combine various natural language processing techniques to search for the answers. Today there is an increasing interest in the integration of QAS with web search. Ask.com[15] is an early example of such a system, and Google and Microsoft have started to integrate question-answering facilities in their search engines.

Many widely used special purpose QAS have limited deduction capability. Examples of such systems are driving direction systems, reservation systems, diagnostic systems and specialized expert systems, especially in the domain of medicine.

[14] Text Retrieval Conference (TREC) http://trec.nist.gov/
[15] Ask.com. http://www.ask.com/

Current QAS typically include a *question classifier module* that determines the type of question and the type of answer. After the question is analyzed, the system typically uses several modules that apply increasingly complex NLP techniques on a gradually reduced amount of text. Thus, a *document retrieval module* uses search engines to identify the documents or paragraphs in the document set that are likely to contain the answer. Subsequently a *filter* pre selects small text fragments that contain strings of the same type as the expected answer. For example, if the question is "Who invented Penicillin" the filter returns text that contain names of people. Finally, an ***answer extraction*** *module* looks for further clues in the text to determine if the answer candidate can indeed answer the question.

The Q&A Roadmap Committee of the USA National Institute of Standards and Technology (NIST)[16] provided a research roadmap [135] with several milestones in order to check the capabilities offered to the QAS users. Some of the features expected by the users are: Timeliness, Accuracy, Usability, Completeness and Relevance.

According to these features, answers to questions should be provided in real time, in spite of the number of users of the system. Answers should always be accurate, correct, therefore incorrect answers are not allowed. Answers should be in full form, short answers are not allowed. QAS should be able to explain its answers and must include ways of visualizing and navigating it Answers should be relevant within a specific context according to user characteristics.

The data source format should be transparent to the system in order to answer the user. The system should also be able to answer the user in any format required. New information should be incorporated to the system as soon as it is available in order to keep it updated.

Specific user needs should be kept into the system knowledge base, including also domain specific ontologies. Automatic acquisition of user needs and specifications allows to feedback the QAS. Interactive QAS allow the user to describe the context of the question. In order to be really accurate, world knowledge and common sense inference must be incorporated to the system.

Roadmap Committee [135] has identified a number of research issues in Q&A which are:

[16] Q&A Roadmap http://www-nlpir.nist.gov/projects/duc/papers/qa.Roadmap-paper_v2.doc

- **Question classes**. Different types of questions require the use of different strategies to find the answer. Question classes are arranged hierarchically in taxonomies. Speech Act Theory identifies different situations that can be used to categorize virtually all speech acts in conversations: questions (equivalent to interrogative), assertion, request/directive, reaction, expressive evaluation, commitment and declaration. These categories were abstracted from act theories in philosophy, linguistics and sociology. Research needs to be done to expand and consolidate these categories for the larger scope of open-domain question answering.

- **Question processing**. The same information request can be expressed in several different ways. A semantic model of question understanding and processing is needed, one that would recognize equivalent questions, regardless of the speech act or of the words, syntactic inter-relations or idiomatic forms. This model would enable the translation of a complex question into a series of simpler questions, would identify ambiguities and treat them in context or by interactive clarification. Question processing must allow for dialogues between the user and the system, forming a common ground of beliefs, intentions and understanding. New models of dialogue need to be developed, with well formulated semantic, that allow for open-domain NLP processing.

- Context and Q&A. Questions are usually asked within a context and answers are provided within that specific context. The context can be used to clarify a question, resolve ambiguities or keep track of an investigation performed through a series of questions. The notion of context is very complex and modelling context is not simple. A formal theory of the logic of contextual objects has been proposed by John McCarthy. Revisiting this theory and creating models for Q&A is necessary.

- Data sources for Q&A. Before a question can be answered, it must be known what knowledge sources are available. If the answer to a question is not present in the data sources, no matter how well we perform question processing, retrieval and extraction of the answer, we shall not obtain a correct result.

- Answer extraction. Answer extraction depends on the complexity of the question, on the answer type provided by question processing, on the actual data where the answer is searched, on the search method and on the question focus and context. The requested information might be present in a variety of heterogeneous data sources that must be searched by different retrieval methodologies. Several collections of texts might be organized into separate structures, with different index operators and retrieval techniques. Moreover, other information may be organized in databases, catalogues or may simply reside on the WWW, in HTML or XML format. Given that answer processing depends on such a large number of factors, research for answer processing should be tackled with a lot of care and given special importance.
- Answer formulation. The result of a QAS should be presented in a way as natural as possible. In some cases, simple extraction is sufficient. For example, when the question classification indicates that the answer type is a name (of a person, organization, shop or disease, etc), a quantity (monetary value, length, size, distance, etc) or a date (e.g. the answer to the question. In addition, the user should be allowed to input his/her own judgments or data. For other cases, the presentation of the answer may require the use of fusion techniques that combine the partial answers from multiple documents.
- Real time question answering. There is need for developing Q&A systems that are capable of extracting answers from large data sets in several seconds, regardless of the complexity of the question, the size and multitude of the data sources or the ambiguity of the question.
- Multi-lingual question answering. The ability of developing Q&A systems for other languages than English is very important. Moreover, the ability of finding answers in texts written in languages other than English, when an English question is asked is very important.
- Interactive Q&A. It is often the case that the information need is not well captured by a Q&A system, as the question processing part may fail to classify properly the question or the information needed for extracting and generating the answer is not easily retrieved. In such cases, the questioner

might want not only to reformulate the question, but (s)he might want to have a dialogue with the system.

- Advanced reasoning for Q&A. More sophisticated questioners expect answers which are outside the scope of written texts or structured databases. To upgrade a Q&A system with such capabilities, we need to integrate reasoning components operating on a variety of knowledge bases, encoding world knowledge and common-sense reasoning mechanisms as well as knowledge specific to a variety of domains. We also need to allow for the representation of scenarios of interest, capabilities of inferring new facts if required by the answer and a way of assembling all these facts and presenting them to the answer generator component. The potential benefits are:

- Customization: As answers are synthesized from a knowledge base, answers can be customized to the user's particular situation.

- Controllable level of detail: The level of detail can be dynamically controlled to suit the user's level of expertise, by controlling how much information from the knowledge base is included in the answer.

- Robustness: By inferring answers rather than extracting them, the Q&A system can respond to unanticipated questions and can resolve situations in which no answer could have been found in the sources of data.

- User profiling for Q&A. The user profile captures data about the questioner, comprising context data, domain of interest, reasoning schemes frequently used by the questioner, common ground established within different dialogues between the system and the user etc. The profile may be represented as a predefined template, where each template slot represents a different profile feature. Profile templates may be nested one within another. The Q&A system fills the template slots for each questioner that uses it.

Addition of deduction capability to a search engine cannot be met through the use of existing methods—methods which are based on bivalent logic and probability theory. To add deduction capability to a search engine it is necessary to generalize bivalent logic and generalize probability theory [254]

There are four major problems in upgrading a search engine to a QAS: World knowledge, Relevance, Deduction and Precisiation.

2.3. *Knowledge*

There is no single agreed definition of knowledge presently, nor any prospect of one. Philosophical debates in general start with Plato's formulation of knowledge as "justified true belief". Knowledge is defined variously as

- That which is known; the sum of what has been perceived, discovered, or inferred.
- Known facts, ideas, and skill that have been imparted.
- That which is known about a specific subject or situation or in total; facts and information.
- Expertise, and skills acquired through experience or education; the theoretical or practical understanding of a subject.
- Awareness or familiarity gained by experience of a fact or situation.

Lars Qvortrup [163] gives the following definition about knowledge:

> "What is knowledge? For me, a very simple, yet practical and applicable sociological definition of knowledge is that knowledge is confirmed observations. Observations may be confirmed over time or in society. When I observe something and then repeat my observation with the same result it becomes a confirmed observation and thus: personal knowledge. Similarly, when I observe something and another person can confirm this observation it becomes social knowledge.
>
> This implies that knowledge is not a quality of the world, but a quality of observing the world. Knowledge isn't something that we find "out there", but something that is created by observing the world and by comparing world observations over time and among different observers, bearing in mind, of course, that the observer is part of the observed world".

Representing and manipulating knowledge automatically is one of the outstanding research questions of our time, and the essential purpose of Artificial Intelligence [6]. The problems of representing and manipulating linguistic knowledge pale into

insignificance compared to the problems posed by real world knowledge. The real world knowledge includes common sense reasoning, as well as general knowledge, and facts about certain more specialized domains.

World knowledge is the knowledge which humans acquire through experience, communication and education [249, 253, 259]. Simple examples are:

- Few professors are rich
- There are no honest politicians
- It is not likely to rain in San Francisco in midsummer
- Most Swedes are tall
- There are no mountains in Holland
- Usually Princeton means Princeton University
- A person can have only one father

World knowledge plays a central role in search, assessment of relevance and deduction [263]. Centrality of world knowledge in human cognition entails its centrality in web intelligence and, especially, in assessment of relevance, summarization, knowledge organization, ontology, search and deduction. The problem with world knowledge is that it is, for the most part, perception-based. Perceptions—and especially perceptions of probabilities—are intrinsically imprecise, reflecting the fact that human sensory organs, and ultimately the brain, have a bounded ability to resolve detail and store information. Imprecision of perceptions stands in the way of using conventional techniques—techniques which are based on bivalent logic and probability theory—to deal with perception-based information. A further complication is that much of world knowledge is negative knowledge in the sense that it relates to what is impossible and/or non-existent. For example, “A person cannot have two fathers” and “Netherlands has no mountains”.

Alan Turing [214] in 1950, proposed a test to verify whether computer's capability to demonstrate intelligence. If the computer could successfully mimic a human during an informal exchange of text messages, then, for most practical purposes, it might be considered intelligent. This test is actually known as the Turing Test (TT).

Knowledge acquisition involves complex cognitive processes: perception, learning, communication, association and reasoning. In computer science, particularly artificial intelligence (AI), the primary aim is to store knowledge so that programs can process it and emulate human intelligence. AI researchers have borrowed knowledge representation theories from cognitive science. In AI a number of representations have been devised to structure information. Frames, rules and semantic networks are representation techniques which have originated from theories of human information processing. Since knowledge is used to achieve intelligent behaviour, the fundamental goal of knowledge representation is to represent knowledge in a manner as to draw conclusions (i.e. to infer new information) from knowledge.

2.3.1. Knowledge representation

Knowledge representation [185] is a sub area of Artificial Intelligence concerned with understanding, designing, and implementing ways of representing information in computers so that programs can use it

- to derive new information that is implied by it,
- to converse with people in natural languages,
- to plan future activities,
- to solve problems in areas that normally require human expertise.

Deriving information that is implied by the information already present is a form of reasoning.

The field of knowledge representation began, around 1958, with an investigation of how a computer might be able to represent and use common sense knowledge necessary to get from our house to the airport. One of the fundamental problems encountered became known as the general knowledge problem or the common sense knowledge problem. While researchers were aware that in an AI system, knowledge would have to be explicitly represented, they did not anticipate the vast amount of implicit knowledge we all share about the world and ourselves. Designers of AI systems did not consider producing rules like "If President Clinton is in Washington, then his left foot is also in Washington," or "If a father has a son, then the son is younger than the father and remains younger for his entire life." In retrospect, this is perhaps not surprising, because

the implicit nature of this knowledge in humans means that we all take it for granted, and never have to state it or consider it explicitly.

Once the problem was acknowledged, it soon became clear that it represented an enormous hurdle for the development of general purpose intelligent systems. One hope, or perhaps wishful thinking on the part of AI developers, was that all that was needed was a decent learning program, and this knowledge would be acquired by computers as automatically as it is acquired by humans. A central part of the common sense knowledge problem has to do with the issue of knowledge representation in artificial systems.

In the 1960s and 1970s, much knowledge representation research was concerned with representing and using information expressed in natural languages. Knowledge representation (KR) is most commonly used to refer to representations consisting of explicit objects (the class of all elephants, or Clyde a certain individual), and of assertions or claims about them ('Clyde is an elephant', or 'all elephants are grey'). Representing knowledge in such explicit form enables computers to draw conclusions from knowledge already stored ('Clyde is grey').

John Sowa in his book “Knowledge Representation” [206] consider that KR “is the application of logic and ontology to the task of constructing computable models for some domain”, applying theories and techniques from the fields of:

- *Logic* provides the formal structure and rules of inference.
- *Ontology* defines the kinds of things that exist in the application domain.
- *Computation* supports the applications that distinguish knowledge representation from pure philosophy.

Prolog, which emerged from the collaboration between Alain Colmerauer and Robert Kowalski, represents propositions and basic logic, and can derive conclusions from known premises. Colmerauer was working on natural language understanding, using logic to represent semantics and resolution for question-answering. During the summer of 1971, Colmerauer and Kowalski discovered that the clausal form of logic could be used to represent formal grammars and that resolution theorem provers could be used for parsing. In the following summer of 1972, Kowalski working again with Colmerauer, developed the procedural interpretation of implications, which later became formalised in Prolog.

Colmerauer, with Philippe Roussel, used this dual interpretation of clauses as the basis of Prolog, which was implemented in the summer and autumn of 1972. The first Prolog program was a French question-answering system. The use of Prolog as a practical programming language was given great momentum by the development of a compiler by David Warren in Edinburgh in 1977. Experiments demonstrated that Edinburgh Prolog could compete with the processing speed of Lisp programming languages. Edinburgh Prolog became the *de facto* standard and strongly influenced the definition of ISO standard Prolog.

Much of the modern development of Prolog came from the impetus of the Fifth Generation Computer Systems project (FGCS), which developed a variant of Prolog named *Kernel Language* for its first operating system. The FGCS project was an initiative by Japan's Ministry of International Trade and Industry, begun in 1982, to create a "fifth generation computer" which was supposed to perform much calculation using massive parallelism.

These days one of the implementations of Prolog most widely used in research and education is SWI-Prolog[17], as well as for commercial applications. It is an open source implementation that has been under continuous development since 1987. It has a rich set of features, libraries and developer tools [226].

XPCE [228] is a toolkit for developing graphical applications in Prolog and other interactive and dynamically typed languages. It has a dynamically typed object-oriented kernel. Methods can be defined in any language. Using XPCE, interactive Prolog applications can be written completely in Prolog.

2.3.2. Ontologies

The word "ontology" seems to generate a lot of controversy in discussions about AI. It has a long history in philosophy, in which it refers to the subject of existence. The word *ontology* comes from the Greek *ontos* for being and *logos* for word. The more traditional term is Aristotle's word *category*, which he used for classifying anything that can be said or *predicated* about anything. Ontology is a study of conceptions of reality and the nature of being; it is the study of existence. T. R. Gruber [60] defines ontology as “an explicit specification of a conceptualization”. J. Sowa, in his web page,[18] defines

[17] http://www.swi-prolog.org/
[18] http://www.jfsowa.com/

ontology as “a system of categories for classifying and talking about the things that are assumed to exist”.

Ontology [193] as a branch of philosophy is the science of what is, of the kinds and structures of the objects, properties and relations in every area of reality. In simple terms, it seeks the classification of entities. It seeks to describe or posit the basic categories and relationships of being or existence to define entities and types of entities within its framework. It is the science of what is, of the kinds and structures of the objects, properties and relations in every area of reality.

The subject of *ontology* [206] is the study of the *categories* of things that exist or may exist in some domain. The product of such a study, called *an ontology*, is a catalogue of the types of things that are assumed to exist in a particular domain of interest from the perspective of a person who uses some specific language for the purpose of talking about that domain. The types in the ontology represent the *predicates*, *word senses*, or *concept and relation types* of the language when used to discuss topics about that particular domain. The combination of logic with an ontology provides a language that can express relationships about the entities in the domain of interest.

In the context of knowledge sharing, T.R.Gruber [59] uses the term ontology to mean a *specification of a conceptualization*. That is, an ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents.

In the context of computer and information sciences, an ontology [61] defines a set of representational primitives with which to model a domain of knowledge or discourse. The representational primitives are typically classes (or sets), attributes (or properties), and relationships (or relations among class members). The definitions of the representational primitives include information about their meaning and constraints on their logically consistent application. Ontologies are typically specified in languages that allow abstraction away from data structures and implementation strategies; in practice, the languages of ontologies are closer in expressive power to first-order logic than languages used to model databases.

The ontology presented in Figure 2 is based on the book Knowledge Representation by John F. Sowa [206]. The basic categories and distinctions have been derived from a

variety of sources in logic, linguistics, philosophy, and artificial intelligence. The two most important influences have been the semiotics of Charles Sanders Peirce and the categories of existence of Alfred North Whitehead, who were pioneers in symbolic logic.

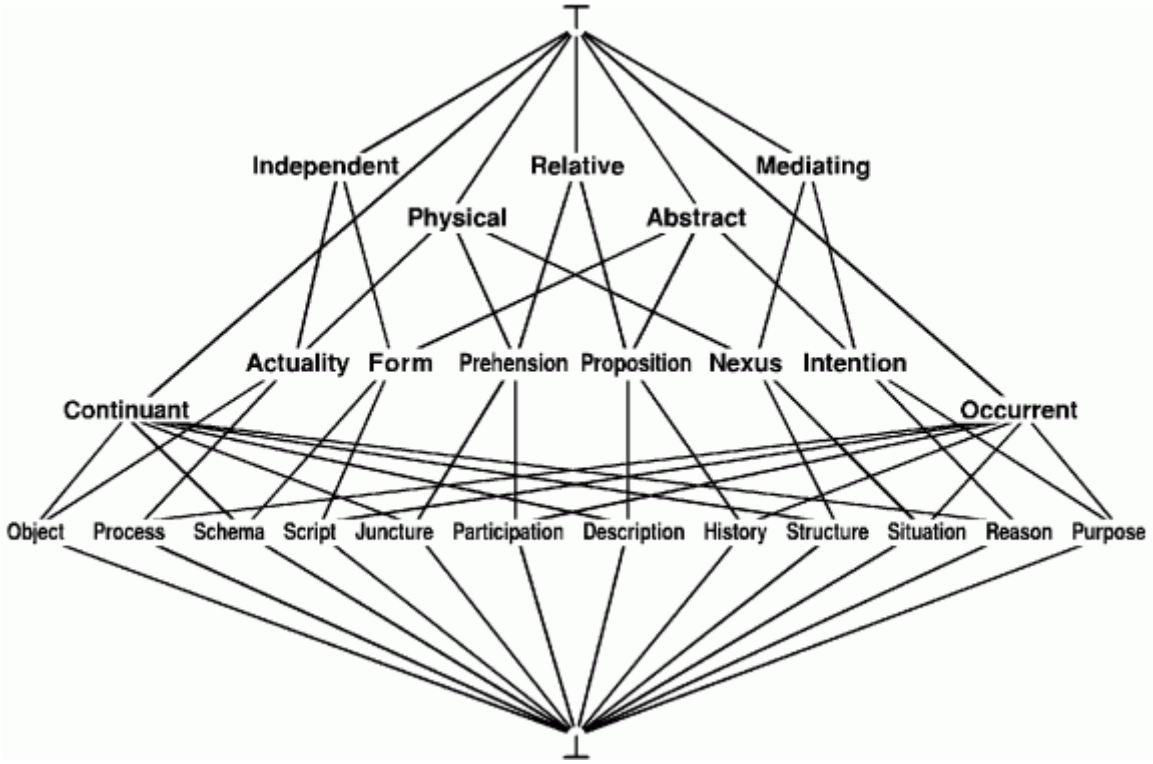


Figure 2 Hierarchy of top-level categories

2.3.3. SUMO

The Suggested Upper Merged Ontology [133, 134] (SUMO) and its domain ontologies form the largest formal public ontology in existence today. They are being used for research and applications in search, linguistics and reasoning. SUMO is the only formal ontology that has been mapped to the entire WordNet lexicon. SUMO is written in the Standard Upper Ontology Knowledge Interchange Format (SUO-KIF) language.

SUO-KIF[19] is a language with declarative semantics designed for use in the authoring and interchange of knowledge. It is possible to understand the meaning of expressions in SUO-KIF without appeal to an interpreter for manipulating those expressions. In this way, it differs from other languages that are based on specific interpreters, such as Emycin and Prolog.

[19] Adam Pease, Standard Upper Ontology Knowledge Interchange Format, 12/28/2007, http://sigmakee.cvs.sourceforge.net/*checkout*/sigmakee/sigma/suo-kif.pdf

2.3.4. Cyc

Beginning in the 1980s formal computer knowledge representation languages and systems arose. Major projects attempted to encode wide bodies of general knowledge; for example the Cyc project, managed by Cycorp, Inc[20]., went through a large encyclopedia, encoding not the information itself, but the information a reader would need in order to understand the encyclopedia: naive physics; notions of time, causality, motivation; commonplace objects and classes of objects.

The Cyc project is the largest sustained effort[21] to develop a broad coverage ontology with detailed axioms and definitions for each concept. It is the world's largest and most complete general knowledge base and commonsense reasoning engine. Over 100 person-years of effort have been spent on hand-crafting a hierarchy of 100,000 concept types with over a million associated axioms.

In 2004 [186], the full version of the KB contained over 2.5 million assertions (facts and rules) interrelating more than 155,000 concepts. Most of the assertions in the KB were intended to capture "commonsense" knowledge pertaining to the objects and events of everyday human life, such as buying and selling, kinship relations, household appliances, eating, office buildings, vehicles, time, and space.

The Cyc project attempts to assemble a comprehensive ontology and database of everyday common sense knowledge, with the goal of enabling AI applications to perform human-like reasoning [164]. The Cyc project, which came from encyclopedia, was started in 1984. Cyc is an attempt to do symbolic AI on a massive scale. It is not based on numerical methods or statistical probabilities, nor is it based on neural networks or fuzzy logic. All of the knowledge in Cyc is represented declaratively in the form of logical assertions. Recently, Cyc has been made available to AI researchers under a research-purposes license as ResearchCyc.

OpenCyc[22] is the open source version of the Cyc technology, which contains the full set of Cyc terms as well as millions of assertions. It includes simple statements of fact, rules about what conclusions to draw if certain statements of fact are satisfied (true),

[20] http://www.cyc.com/
[21] http://www.jfsowa.com/ontology/ontoshar.htm
[22] http://www.cyc.com/cyc/opencyc/overview

and rules about how to reason with certain types of facts and rules. New conclusions are derived by the inference engine using deductive reasoning.

Typical pieces of knowledge represented in the database are "Every tree is a plant" and "Plants die eventually". When asked whether trees die, the inference engine can draw the obvious conclusion and answer the question correctly. The Knowledge Base (KB) contains over a million human-defined assertions, rules or common sense ideas. These are formulated in the language CycL, which is based on predicate calculus and is similar to Lisp programming language.

Much of the current work on the Cyc project continues to be knowledge engineering, representing facts about the world by hand, and implementing efficient inference mechanisms on that knowledge. Its purpose is to break the software bottleneck once and for all by constructing a foundation of basic "common sense" knowledge that will enable a variety of knowledge-intensive products and services. Cyc is intended to provide a "deep" layer of understanding that can be used by other programs to make them more flexible. To date, Cyc has made possible ground-breaking pilot applications in the areas of heterogeneous database browsing and integration, captioned image retrieval, and natural language processing.

While great strides have been made in machine learning in the last few decades, automatically gathering useful, consistent knowledge in a machine-usable form is still a relatively unexplored research area. The original promise of the Cyc project – to provide a basis of real-world knowledge sufficient to support the sort of learning from language of which humans are capable – has not yet been fulfilled [116].

Some scientists[23] consider that “Cyc has had some useful applications, but none of them have been sufficiently successful to pay for the many millions of dollars that were invested in the project”. Over 700 person-years of effort have been spent in hand-crafting a hierarchy of 600,000 concept types with about two million associated axioms. But after 23 years of research and development, Cyc still cannot support language, learning, or reasoning at the level of a child[24].

Initial work [116] on a method of using a combination of Cyc and Google to assist in entering knowledge into Cyc is being developed. The long-term goal is automating the process of building a consistent, formalized representation of the world in the Cyc

[23] John F. Sowa, cited from a letter to standard-upper-ontology@listserv.ieee.org on Jan 29, 2008
[24] John F. Sowa, cited from a letter to cg@conceptualgraphs.org on Apr 17, 2007

knowledge base via machine learning. Comparatively shallow natural language parsing combined with the type constraint and relation knowledge in the Cyc system allows the retrieval, verification, and review of unconstrained facts at a higher rate than that achieved by human knowledge representation experts working unassisted.

2.3.5. OMCS Open Mind Common Sense

OMCS[25] is a MIT Media Lab project to give computers the capacity to understand and reason about the world as people do, by means of robust and scalable commonsense reasoning systems. They consider that no single technique is by itself powerful enough to deal with the broad range of domains every ordinary person can understand. Therefore, they are developing new types of reasoning technologies and cognitive architectures that support great procedural and representational diversity.

As long as practical commonsense reasoning systems require large quantities of knowledge about ordinary concepts such as objects, actions, events, goals, and places, they are developing new methods for acquiring commonsense knowledge, based on easy-to-use knowledge acquisition tools that let volunteers from all over the web collaborate to teach commonsense knowledge to our reasoning systems.

OMCS was built in the first half of the year 2000, and launched in September 2000. In 2004, OMCS had collected 700,000 commonsense facts about the properties and structures of ordinary objects, events, actions, and places from 15,000 people across the web. OMCS acquires world knowledge from a web-based community of instructors, in the form of structured stories. The underlying representation is based on natural language story templates built from different combinations of Wendy Lehnert's plot units [96, 97].

Open Mind Common Sense [188] is a commonsense knowledge acquisition system targeted at the general public. It is a web site that gathers facts, rules, stories, and descriptions using a variety of simple elicitation activities. In choosing to acquire knowledge in free-form natural language, OMCS team shifted the burden from the knowledge acquisition system to the methods for using the acquired knowledge. This way, they use the English items directly for reasoning and, use information extraction techniques to convert English items into more standard knowledge representations. They encourage knowledge to be supplied using templates rather than as free-form

[25] http://openmind.media.mit.edu/CommonsenseHome.htm

English text but, at the same time, allow users to extend the template library themselves. Therefore, the users can enter descriptions and simple stories extending across multiple sentences.

OMCS team has also incorporated several inference mechanisms into the acquisition cycle. The system induces inference rules from the knowledge that people have supplied, and these rules are used immediately to feed back inferences on entered items. They engage in three types of inference: Analogies over Concepts, Analogies over Relations and, Analogies as Inference Rules.

In order to build software that can deeply understand people and our problems, three commonsense knowledge bases [189] are being developed that take unconventional approaches to representing, acquiring, and reasoning with large quantities of commonsense knowledge. Each adopts a different approach — ConceptNet is a large-scale semantic network, LifeNet is a probabilistic graphical model, and StoryNet is a database of story-scripts. They use natural language as an essential part of their knowledge representation and employ alternative methods of reasoning and knowledge representation.

OMCS has gathered hundreds of thousands of small pieces of commonsense knowledge, and it continues to grow. Knowledge supplied by users in English must be parsed into a target representation that is as expressive as English itself. An option that has been largely overlooked within the field of knowledge representation is to use English itself as the knowledge representation [187]. The idea is that English itself can serve as the representation over which reasoning is done. The value of this approach is three-fold:

- to avoid having to impose a novel ontological structure on the universe beyond that which English has already supplied us.
- no need to create and learn a massive ontology, but just use English words and expressions.
- to avoid difficult and error prone “semantic interpretation” before reasoning can begin.

OMCS team consider those ideas helpful to build an inference system capable of reasoning with knowledge, and that new ways to make available the full expressive power of a natural language for commonsense reasoning will be found. They considered

to have explored only the very surface of ways to extract commonsense knowledge from the general public, and hope that others will be inspired to follow with new approaches.

2.3.6. WordNet

WordNet is a large lexical database of English, developed under the direction of George A. Miller [121, 122]. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words and concepts can be navigated with the browser. WordNet's structure makes it a useful tool for computational linguistics [4, 55, 56], natural language processing[26], information retrieval [21, 22, 194, 195], question answering systems [112] and others [117, 231].

John Sowa[27] considers that WordNet is the most widely used ontology for natural language processing, largely because it has long been easily accessible over the Internet, although it doesn't have as much detail as Cyc.

2.3.7. Semantic network

In 1909, Charles Pierce proposed a graphical notation of nodes and arcs called existential graphs that he called “the logic of the future” [176]. The first propositional semantic network to be implemented in AI [Sowa, 1992, Semantic Networks].was the MIND system, developed by Stuart Shapiro (1971). It later evolved into the *Semantic Network Processing System* (SNePS), which has been used to represent a wide range of features in natural language semantics.

A semantic network or net is a graphic notation for representing knowledge in patterns of interconnected nodes and arcs [205]. Each node represents a concept and arcs are used to define relations between the concepts. Computer implementations of semantic networks were first developed for artificial intelligence and machine translation, but earlier versions have long been used in philosophy, psychology, and linguistics. One of the most expressive and comprehensively described knowledge representation paradigms along the lines of semantic networks is MultiNet (an acronym for Multilayered Extended Semantic Networks).

[26] http://www.ai.sri.com/~harabagi/coling-acl98/acl_work/acl_work.html
[27] http://www.jfsowa.com/ontology/ontoshar.htm#s1

What is common to all semantic networks is a declarative graphic representation that can be used either to represent knowledge or to support automated systems for reasoning about knowledge. Some versions are highly informal, but other versions are formally defined systems of logic. Following are some of the most common kinds of semantic networks [204, 205].

- *Definitional networks* emphasize the *subtype* or *is-a* relation between a concept type and a newly defined subtype. They support the rule of *inheritance* for copying properties defined for a supertype to all of its subtypes.
- *Assertional networks* are designed to assert propositions. Some assertional netwoks have been proposed as models of the *conceptual structures* underlying natural language semantics.
- *Implicational networks* use implication as the primary relation for connecting nodes. They may be used to represent patterns of beliefs, causality, or inferences.
- *Executable networks* include some mechanism which can perform inferences, pass messages, or search for patterns and associations.
- *Learning networks* build or extend their representations by acquiring knowledge from examples, by adding and deleting nodes and arcs or by modifying numerical values, called *weights*, associated with the nodes and arcs.

2.3.8. Conceptual graphs

Conceptual graphs [202, 203, 206] are a variety of propositional semantic networks in which the relations are nested inside the propositional nodes. They evolved as a combination of the linguistic features of Tesnière's dependency graphs and the logical features of Peirce's existential graphs with strong influences from the work in artificial intelligence and computational linguistics [205].

Conceptual graphs (CGs) are a system of logic based on the existential graphs of C. S. Peirce and the semantic networks of artificial intelligence[28]. They express meaning in a form that is logically precise, humanly readable, and computationally tractable. With a direct mapping to language, conceptual graphs serve as an intermediate language for translating computer-oriented formalisms to and from natural languages. With their

[28] http://www.jfsowa.com/cg/

graphic representation, they serve as a readable, but formal design and specification language. CGs have been implemented in a variety of projects for information retrieval, database design, expert systems, and natural language processing.

> "A conceptual graph is a version of logic with a graphical representation. CGs were originally designed as a formal notation for representing the semantics of natural languages in a way that could be translated to logic and various computable forms" [29].

Conceptual graphs are formally defined in an abstract syntax that is independent of any notation. The formalism can be represented in several different concrete notations like:

- the graphical *Display Form* (DF),
- the formally defined *Conceptual Graph Interchange Form* (CGIF), and
- the compact, but readable *Linear Form* (LF).

Every CG could be represented in each of these three forms and is translated to a logically equivalent representation in predicate calculus and in the Knowledge Interchange Format (KIF) [72]. Following, the sentence: "a cat is on a mat" will be represented in each of these notations.

In the display form (DF), concepts are represented by rectangles: the concept [Cat] represents a instance of a cat, and [Mat] represents an instance of a mat. Conceptual relations are represented by circles or ovals: the conceptual relation (On) relates a cat to a mat. The arcs that link the relations to the concepts are represented by arrows: the first arc has an arrow pointing toward the relation, and the second arc has an arrow pointing away from the relation. If a relation has more than two arcs, the arcs are numbered.

In the linear form (LF), concepts are represented by square brackets instead of boxes, and the conceptual relations are represented by parentheses instead of circles:

[Cat] → (On) → [Mat].

[29] John Sowa in a letter to cg@cs.uah.edu 10 May 2006

Both DF and LF are designed for communication with humans or between humans and machines. For communication between machines, the conceptual graph interchange form (CGIF) has a syntax that uses *coreference labels* to represent the arcs:

[Cat: *x] [Mat: *y] (On ?x ?y)

The symbols *x and *y are called *defining labels*. The matching symbols ?x and ?y are the *bound labels* that indicate references to the same instance of a cat *x* or a mat *y*. To reduce the number of coreference labels, CGIF also permits concepts to be nested inside the relation nodes:

(On [Cat] [Mat])

For communication with systems that use other internal representations, CGIF can be translated to another logic-based formalism called the Knowledge Interchange Format (KIF):

(exists ((?x Cat) (?y Mat)) (On ?x ?y))

Although DF, LF, CGIF, and KIF look very different, their semantics is defined by the same logical foundations. They can all be translated to a statement of the following form in typed predicate calculus:

($*x*:Cat)($*y*:Mat)on(*x*,*y*).

Fuzzy Conceptual Graphs (FCG) extends simple CGs with linguistic labels defined by fuzzy sets as individual markers [131, 24, 25, 26].

2.3.9. AMINE

One of the more complete CG manipulation systems that is available and under active development is Amine[30]. It is a Multi-Layer Java Open Source Platform suited for the development of different types of intelligent systems as Knowledge-Based Systems, Ontology-Based Systems, Conceptual Graphs Based Applications, Natural Language Processing applications, and intelligent agents.

Amine is composed of four layers:

- The Kernel layer (Multi-Lingua Ontology Layer): which offers the possibility to create, edit and ask an ontology defined in terms of Conceptual Structures
- The Algebraic (structures and operations) layer

[30] http://amine-platform.sourceforge.net/

- The Programming layer which provides a dynamic ontology engine, PROLOG+CG that is object based and CG-based and, SYNERGY, a CG-activation and state-propagation based language.
- The Multi-Agents System layer

Conceptual Graphs (CGs) are being used for knowledge bases in systems as Conceptual Programming Environment (CPE) [158]. In 2004, the main form of interoperability was by using the CGIF interchange format. The knowledge system has now expanded not only into the storage and retrieval of graphs, but also into improving the performance of foundational reasoning operations.

The knowledge base in CPE is divided into two categories: 1) world knowledge and 2) prototype knowledge. The world knowledge comes from a common sense knowledge system that can perform non-monotonic reasoning, has the ability to change its mind and makes inferences when there is hardly enough information available to make any inference. The world knowledge describes objects and their relationship with other objects. The prototype knowledge is a higher order or abstract knowledge. Unlike the world knowledge, this knowledge is very dependent on the domain context. The prototype knowledge comes from the new story understanding system, but builds understanding that goes beyond the given text.

2.3.10. Frames

In the field of Artificial Intelligence, a frame is a data structure introduced by Marvin Minsky in the 70s [123] that can be used for knowledge representation.

- “A frame is a data-structure for representing a stereotyped situation like being in a certain kind of living room or going to a child's birthday party. Attached to each frame are several kinds of information. Some of this information is about how to use the frame. Some is about what one can expect to happen next. Some is about what to do if these expectations are not confirmed.
- We can think of a frame as a network of nodes and relations. The "top levels" of a frame are fixed, and represent things that are always true about the supposed situation. The lower levels have many terminals -- "slots" that must be filled by specific instances or data. Each terminal can specify

> conditions its assignments must meet. (The assignments themselves are usually smaller "sub- frames.") Simple conditions are specified by markers that might require a terminal assignment to be a person, an object of sufficient value, or a pointer to a sub-frame of a certain type. More complex conditions can specify relations among the things assigned to several terminals." [124]

Roughly similar to the object-oriented paradigm, they represent classes (called *frames*) with certain properties called *attributes* or *slots*. Slots may contain *values*, refer to other frames (*relations*) or contain *methods*. Frames are thus a machine-usable formalization of concepts or schemata.

Like many other knowledge representation systems and languages, frames are an attempt to resemble the way human beings are storing knowledge. It seems like we are storing our knowledge in rather large chunks, and that different chunks are highly interconnected. In frame-based knowledge representations knowledge describing a particular concept is organized as a frame. The frame usually contains a name and a set of slots.

The slots describe the frame with attribute-value pairs <slotname value> or alternatively a triple containing framename, slotname and value in some order. In many frame systems the slots are complex structures that have facets describing the properties of the slot. The value of a slot may be a primitive such as a text string or an integer, or it may be another frame. Most systems allow multiple values for slots and some systems support procedural attachments. These attachments can be used to compute the slot value, or they can be triggers used to make consistency checking or updates of other slots. The triggers can be trigged by updates on slots.

Reasoning in frame-systems is based on frame matching, inheritance and spreading activation. In most frame-based knowledge representations, inheritance is the central inference mechanism. The frames are organized as a hierarchy with some general concept as the root frame. Many systems support multiple inheritance.

The Knowledge Machine (KM) is a knowledge representation language and reasoning engine. The knowledge is represented as frames, but KM is also influenced by logic. This combination makes KM very expressive and provides it with a clear, formal semantics.

KL-ONE [182, 19] is a well known knowledge representation system in the tradition of semantic networks and frames. The system is an attempt to overcome semantic indistinctness in semantic network representations and to explicitly represent conceptual information as a structured inheritance network. It is built upon the idea of structured inheritance networks.

Frames in KL-ONE are called concepts, which form hierarchies using subsume-relations; in the KL-ONE terminology a super class is said to subsume its subclasses. Multiple inheritance is allowed. Actually a concept is said to be well-formed only if it inherits from more than one other concept. All concepts, except the top concept Thing, must have at least one super class.

In KL-ONE descriptions are separated into two basic classes of concepts: primitive and defined. Primitives are domain concepts that are not fully defined. This means that given all the properties of a concept, this is not sufficient to classify it. They may also be viewed as incomplete definitions. Using the same view, defined concepts are complete definitions. Given the properties of a concept, these are necessary and sufficient conditions to classify the concept.

The slot-concept is called roles and the values of the roles are role-fillers. There are several different types of roles to be used in different situations. The most common and important role type is the generic RoleSet which captures the fact that the role may be filled with more than one filler.

A famous editor for frame-based ontologies is Protégé. The Protégé-Frames editor[31] provides a full-fledged user interface and knowledge server to support users in constructing and storing frame-based domain ontologies, customizing data entry forms, and entering instance data. Protégé-Frames implement a knowledge model which is compatible with the Open Knowledge Base Connectivity protocol (OKBC). In this model, an ontology consists of a set of classes organized in a subsumption hierarchy to represent a domain's salient concepts, a set of slots associated to classes to describe their properties and relationships, and a set of instances of those classes - individual exemplars of the concepts that hold specific values for their properties.

In the fall of 2001, J. McCarthy, M. Minsky, and many others of the major established researchers in the area of commonsense knowledge and reasoning [118]

[31] http://protege.stanford.edu/overview/protege-frames.html

organized a meeting to discuss commonsense reasoning and what makes it difficult for computers to deal with. They contend that common-sense reasoning is too hard a problem to solve using any single artificial intelligence technique. For problems that require common sense, current computing systems fall far short of human ability. The most critical missing piece is the ability to deal with large amounts of knowledge of many kinds, and to make that knowledge effective in perception and problem solving.

Therefore, a multilevel architecture consisting of diverse reasoning and representation techniques that collaborate in order to allow the best techniques to be used for the many situations that arise in commonsense reasoning were proposed.

Meeting participants also proposed to use story understanding (i.e. understanding and answering questions about progressively harder children's texts) as a task for evaluating and scaling up a commonsense reasoning system. Common sense involved in a particular linguistic domain might be acquired by reading texts and conversing with humans [20]. In CPE [158] a story understanding system is being built using a multi-agent design. The primary goal is to perform in-depth story understanding by using both world and prototype knowledge through a knowledge base. The prototype knowledge builds understanding that goes beyond the given text.

2.4. Relevance, Deduction and Precisiation

> "A prerequisite to mechanization of question-answering is mechanization of natural language understanding, and a prerequisite to mechanization of natural language understanding is precisiation of meaning of concepts and proposition drawn from a natural language.
>
> To deal effectively with world knowledge, relevance, deduction and precisiation, new tools are needed."
>
> L.A.Zadeh [263]

Let's suppose that based on the following information [249]

(a) Pat is about ten years older than Carol; and

(b) Carol has two children: a son, in mid-twenties; and a daughter, in mid-thirties.

we want to answer the query q: How old is Pat?. Given (b), using world knowledge, it is possible to estimate Carol's age. Then, by (a), adding ten years to Carol's age, we can estimate to Pat's age.

One of the problems with the previous reasoning is world knowledge. World knowledge [259] plays an essential role in assessment of relevance, summarization, search and deduction, but much of world knowledge is perception-based, and perception-based information is intrinsically fuzzy.

But, there is another basic problem: the problem of relevance. Let's suppose that we have a collection of data which includes (a) and (b), and we have to search for and identify the data that are relevant to the query q. Both (a) and (b), by it selves, in isolation, are not relevant, but, in combination, they are relevant. It is not difficult to recognize that in a simple example, but having a large database, the problem of identifying the data which in combination are relevant to the query, could be very complex.

There is an extensive literature on relevance, and every search engine deals with relevance in its own way, some of them at a high level of sophistication. But what is quite obvious is that the problem of relevance assessment is quite complex and far from being solved. Indeed, the success of Google is due, in large measure, to its simple but ingenious page ranking algorithm for assessment of relevance.

Zadeh [263] considers two kinds of relevance: (a) question relevance and (b) topic relevance. Both are matters of degree. For example, if the available information is *p*: "Population of California is 37,000,000" and the question is *q*: "Number of cars in California?", then what is the degree of relevance of *p* to *q*?

Basically, there are two ways [263] of approaching assessment of relevance: (i) semantic; and (ii) statistical. In the car example, relevance of *p* to *q* is a matter of semantics and world knowledge. In existing search engines, relevance is largely a matter of statistics, involving counts of links and words, with little if any consideration of semantics. What should be noted is that assessment of topic relevance is more amendable to the use of statistical techniques. That explains why existing search engines are much better at assessment of topic relevance than question relevance.

But, what is really needed is a computing method for the degree of relevance based on the meaning of q and p, that is, a method which compute the semantic relevance. Existing search engines have a very limited capability to deal with semantic relevance.

Despite its importance, there is not a satisfactory definition of relevance in the literature, nor can the problem of assessment of relevance be considered to be on the way to solution. Relevance is a matter of degree, thus we can say: very relevant, quite relevant, slightly relevant, etc. Consequently, relevance is a fuzzy concept.

Underlying the problems of world knowledge, relevance and deduction is a very basic problem – the problem of natural language understanding. Much of world knowledge is expressed in a natural language, and natural language is basically a system for describing perceptions. Since perceptions are intrinsically imprecise, so are natural languages, especially in the realm of semantics.

Existing search engines have many remarkable capabilities. However, what is not among them is deduction capability that is the capability to answer a query by a synthesis of information which resides in various parts of the knowledge base [259], especially if it is perception-based information. Existing methods cannot deal with deduction from perception-based knowledge, knowledge which is both uncertain and imprecise [249]. The principal reason is that much of world knowledge is perception-based, and existing theories of knowledge representation and deduction provide no tools for this purpose.

To add deduction capability to a search engine it is necessary to (a) generalize bivalent logic; (b) generalize probability theory. Addition of deduction capability to a search engine is a highly complex problem, which is a major challenge to computer scientists and logicians [254]

Following, there are two basic examples. Assume that the question is *q* and the available information is *p*.

- *q*: What is the average height of Swedes?
- *p*: Most adult Swedes are tall.
- *p*: Usually Robert returns from work at about 6pm.
- *q*: What is the probability that Robert is at home at 6:15 pm?

Neither bivalent logic nor probability theories provide effective tools for dealing with problems of this type. The difficulty is centred on deduction from premises which are both uncertain and imprecise [263].

Underlying the problems of world knowledge, relevance and deduction is a very basic problem—the problem of natural language understanding. Much of world knowledge and web knowledge is expressed in a natural language. A natural language is basically a system for describing perceptions. Since perceptions are intrinsically imprecise, so are natural languages. A prerequisite to mechanization of question-answering is mechanization of natural language understanding. And a prerequisite to mechanization of natural language understanding is precisiation of meaning of concepts and proposition drawn from a natural language [254].

A concept which plays a key role in deduction is that of a protoform [263] – an abbreviation for prototypical form. An important concept which is related to the concept of a protoform is that of protoform equivalence [247]. The importance of the concept of protoform equivalence derives from the fact that it provides a basis for a mode of organization of knowledge, deduction and search in which what matters is the deep semantic structure rather than the surface structure and domain.

The rules of deduction are, basically, the rules which govern constraint propagation. Protoform-based deduction may be viewed as a generalization of deduction in classical symbolic logic. The principal difference is that in protoformal deduction each rule is associated with a computational part, and rules are large in number and are drawn from a wide variety of fields and methodologies.

Existing bivalent-logic-based methods of knowledge representation and deduction are of limited effectiveness in dealing with information which is imprecise or partially true. To deal with such information, bivalence must be abandoned and new tools [249] should be employed. Protoform-centered organization of knowledge and deduction plays a central role in Computing with Words (CW) and Precisiated Natural Language (PNL). PNL is based on fuzzy logic and has the capability to deal with partiality of certainty, partiality of possibility and partiality of truth. These are the capabilities that are needed to be able to draw on world knowledge for assessment of relevance, and for summarization, search and deduction.

A basic underlying problem in upgrading a search engine to a QAS is mechanization of natural language understanding. A prerequisite to mechanization of natural language understanding is precisiation of meaning [254].

Much of human knowledge is expressed in a natural language. Close relationship between human knowledge and natural language is one of the principal reasons why mechanization of natural language understanding has long been one of the important objectives of AI.

Over the years, impressive progress has been made toward achievement of this objective. But there is a fundamental limitation to what can be achieved through the use of commonly-employed methods of meaning representation. To understand the nature of the limitation, two facts have to be considered:

1) A natural language, NL, is basically a system for describing perceptions; and
2) Perceptions are intrinsically imprecise, reflecting the bounded ability of human sensory organs, and ultimately the brain, to manage with detail information. More specifically, perceptions are f-granular in the sense that

- the boundaries of perceived classes are not sharp (fuzzy); and
- the values of perceived attributes are granular.

Imprecision of perceptions is passed on to natural languages. What this implies is that imprecision of natural language semantics is rooted in imprecision of perceptions. Semantic imprecision of natural languages is not a problem for humans, but it is a major problem for machines.

To clarify what is meant by precisiation it is necessary to clarify what is meant by precise. Informally, precise may be interpreted in two different senses:

- precise in relation to value, or *v-precise*, for short; and
- precise in relation to meaning, or *m-precise*, for short.

To illustrate the difference, the following proposition is *v-imprecise* and *m-precise*.

> "*X* is a Gaussian random variable with mean *m* and variance σ2, where *m* and σ2 are precisely defined real numbers".

The same can be said about the proposition

"*X* is in the interval [*a*, *b*], when *a* and *b* are precisely defined real numbers".

On the other hand, the proposition

"Monika is young"

is both *v-imprecise* and *m-imprecise* so long as "young" is not defined precisely.

2.5. Computing with Words

Zadeh [238] introduced the concept of a linguistic variable, that is, a variable whose values are words rather than numbers. Computing with Words (CW) is a methodology in which the objects of computation are words and propositions drawn from a natural language, e.g.

- small, large, far, heavy,
- not very likely,
- the price of gas is low and declining,
- Berkeley is near San Francisco,
- it is very unlikely that there will be a significant increase in the price of oil in the near future, etc.

Computing with Words is inspired by the remarkable human capability to perform a wide variety of physical and mental tasks without any measurements and any computations. Familiar examples of such tasks are parking a car, driving in heavy traffic, playing golf, riding a bicycle, understanding speech and summarizing a story. Underlying this remarkable capability is the brain's crucial ability to manipulate perceptions – perceptions of distance, size, weight, color, speed, time, direction, force, number, truth, likelihood and other characteristics of physical and mental objects. Manipulation of perceptions plays a key role in human recognition, decision and execution processes.

The key points in Computing with Words are [267]:words are less precise than numbers

2) CW is less precise than Computing with Numbers (CN)

3) CW serves two major purposes

a) provides a machinery for dealing with problems in which precise information is not available

b) provides a machinery for dealing with problems in which precise information is available, but there is a tolerance for imprecision which can be exploited to achieve tractability, robustness, simplicity and low solution cost

Basically, there are four principal rationales for the use of CW [245]:

1) *The don't know rationale*. In this case, the values of variables and/or parameters are not known with sufficient precision to justify the use of conventional methods of numerical computing. An example is decision-making with poorly defined probabilities and utilities.

2) *The don't need rationale*. In this case, there is a tolerance for imprecision which can be exploited to achieve tractability, robustness, low solution cost and better rapport with reality. An example is the problem of parking a car.

3) *The can't solve rationale*. In this case, the problem cannot be solved through the use of numerical computing. An example is the problem of automation of driving in city traffic.

4) *The can't define rationale*. In this case, a concept that we wish to define is too complex to admit of definition in terms of a set of numerical criteria. A case in point is concept of causality. Causality is an instance of what may be called an amorphic concept.

In its traditional sense, computing involves for the most part manipulation of numbers and symbols. By contrast, humans employ mostly words in computing and reasoning, arriving at conclusions expressed as words from premises expressed in a natural language or having the form of mental perceptions. As used by humans, words have fuzzy denotations. The same applies to the role played by words in CW. [245]

A key aspect of Computing with Words is that it involves a fusion of natural languages and computation with fuzzy variables. In a natural language words play the role of labels of fuzzy granules. In CW, a proposition is viewed as an implicit fuzzy constraint on an implicit variable. Therefore, information is conveyed by constraining the values of variables. Furthermore, information is assumed to consist of a collection of propositions expressed in natural or synthetic language. Typically, such propositions

play the role of linguistic characterization of perceptions. The meaning of a proposition is the constraint which it represents.

In CW, we are given a collection of propositions expressed in a natural language which play the role of premises and constitute the *Initial Data Set* (IDS). The aim is to infer an answer to a query expressed in a natural language from the IDS. The answer (i.e. the computational result) will be a collection of propositions expressed in a natural language, referred to as the *Terminal Data Set* (TDS). To infer TDS from IDS the rules of inference in fuzzy logic are used for constraint propagation from premises to conclusions. First, the collection of propositions in IDS is translated into the Generalized Constraint Language. Then, the Fuzzy Logic inference rules are used to propagate the constraint from premises to conclusions. Finally, the TDS is obtained by translating the constraints to natural language by linguistic approximation.

There are two main rationales for computing with words. First, computing with words is a necessity when the available information is not precise enough to justify the use of numbers. And second, computing with words is advantageous when there is a tolerance for imprecision, uncertainty and partial truth that can be exploited to achieve tractability, robustness, low solution cost and better rapport with reality.

Inspired by the ways in which humans granulate human concepts~ we can proceed to granulate conceptual structures in various fields of science. In a sense, this is what motivates computing with words. An intriguing possibility is to granulate the conceptual structure of mathematics. This would lead to what may be called granular mathematics. Eventually, granular mathematics may evolve into a distinct branch of mathematics having close links to the real world. A subset of granular mathematics and a superset of computing with words is granular computing.

In the final analysis, fuzzy information granulation is central to fuzzy logic because it is central to human reasoning and concept formation. It is this aspect of Fuzzy IG that underlies its essential role in the conception and design of intelligent systems. In this regard, what is conclusive is that there are many, many tasks which humans can perform with ease and that no machine could perform without the use of fuzzy information granulation.

A typical example is the problem of estimation of age from voice. More specifically, consider a common situation where A gets a telephone call from B, whom A does not

know. After hearing B talk for 5-10 seconds, A would be able to form a rough estimate of B's age and express it as, say, "B *is old"* or *"It is very likely that B is old",* in which both age and probability play the role of linguistic, that is, f-granulated variables. Neither A nor any machine could come up with crisp estimates of B's age, e.g., "B *is* 63" or "The *probability that B is* 63 *is* 0.002". In this and similar cases, a machine would have to have a capability to process and reason with f-granulated information in order to come up with a machine solution to a problem that has a human solution expressed in terms of f-granulated variables. A related point is that, in everyday decision making, humans use that and only that information which is decision-relevant. For example, in playing golf, parking a car, picking up an object, etc., humans use fuzzy estimates of distance, velocity, angles, sizes, etc. In a pervasive way, decision relevant information is f-granular. To perform such everyday tasks as effortlessly as humans can, a machine must have a capability to process f-granular information. A conclusion which emerges from these examples is that fuzzy information granulation is an integral part of human cognition.

2.6. Granular Computing

As was stated earlier, a concept which plays a pivotal role in Computing with Words is that of a granule. Granulation is pervasive in human cognition. For example, the granules of Age are fuzzy sets labelled young, middle-aged and old. The granules of Height may be very short, short, medium, tall, and very tall. And the granules of Truth may be not true, quite true, not very true, very true, etc. The concept of granularity underlies the concept of a linguistic variable. The concept of a linguistic variable plays a pivotal role in almost all applications of fuzzy logic.

Typically, a granule is a fuzzy set of points drawn together by similarity. A word may be atomic, as in *young*, or composite, as in *not very young*. The denotation of a word may be a higher order predicate. Informally, a granule of a variable X is a clump of values of X which are drawn together by equivalence, similarity, proximity or functionality or maybe just because it is not possible to differentiate them. For example, a granule could be a fuzzy interval or a probability distribution.

Then, a granule, g, which is the denotation of a word, w, is viewed as a fuzzy constraint on a variable. As a simple illustration, consider the proposition *Mary is*

young, which may be a linguistic characterization of a perception. In this case, young is the label of a granule young. Note that for simplicity the same symbol is used both for a word and its denotation. The fuzzy set *young* plays the role of a fuzzy constraint on the age of Mary.

Granulation involves decomposition of whole into parts. Granulation of an object *A* leads to a collection of granules of *A*, with a granule being a clump of points (objects) drawn together by indistinguishability, similarity, proximity or functionality. For example, the time granules are the years, months, days, hours, minutes, etc. The granules of a human head are the forehead, nose, cheeks, ears, eyes, etc. In general, granulation is hierarchical in nature.

Modes of Information Granulation (IG), in which the granules are crisp (c-granular), play important roles in a wide variety of methods, approaches and techniques. Crisp IG, however, fails to reflect the fact that almost all human reasoning and concept formation is based in fuzzy granules (f-granular) rather than crisp ones (c-granular). The granules of a human head, for example, are fuzzy in the sense that the boundaries between cheeks, nose, forehead, ears, etc. are not sharply defined. Furthermore, the attributes of fuzzy granules, e.g., length of nose, are fuzzy, as are their values: long, short, very long, etc. The fuzziness of granules, as well as their attributes and values is characteristic of how humans manipulate information.

The Theory of Fuzzy Information Granulation (TFIG) is inspired by the informal ways in which humans granulate information and reason with it. However, the foundations of TFIG and its methodology are mathematical in nature. In this perspective, Fuzzy Information Granulation may be viewed as a mode of generalization which may be applied to any concept, method or theory. In human cognition, fuzziness of granules is a direct consequence of fuzziness of the concepts of indistinguishability, similarity, proximity and functionality. Furthermore, it is entailed by the finite capacity of the human mind and sensory organs to resolve detail and store information. In this perspective, fuzzy Information Granulation (fuzzy IG) may be viewed as a form of data compression. Fuzzy IG underlies the remarkable human ability to make rational decisions in an environment of imprecision, partial knowledge, partial certainty and partial truth.

Related to fuzzy IG are the following principal modes of generalization.

- Fuzzification (f-generalization). In this mode of generalization, a crisp set is replaced by a fuzzy set.
- Granulation (g-generalization). In this case, a set is partitioned into granules.
- Randomization (r-generalization). In this case, a variable is replaced by a random variable.
- Usualization (u-generalization). In this case, a proposition expressed as X is A is replaced with usually (X is A).

These and other modes of generalization may be employed in combination. A particularly important combination is the conjunction of fuzzification and granulation, referred to as f.g-generalization (or f-granulation or fuzzy granulation), which plays a pivotal role in the Theory of Fuzzy Information Granulation (TFIG) and Fuzzy Logic (FL). As a mode of generalization, f.g-generalization may be applied to any concept, method or theory. In particular, f.g-generalization in application to the basic concepts of variable, function and relation leads to the basic concepts of linguistic variable, fuzzy rule set and fuzzy graph in Fuzzy Logic. These concepts are unique to fuzzy logic and play a central role in its applications.

2.7. Perceptions

Perceptions have been an object of study in psychology for a long time. However, the idea of linking perceptions to computing with words is in a different spirit. An important point that should be noted is that classical logical systems such as propositional logic, predicate logic and modal logic, as well as AI-based techniques for natural language processing and knowledge representation, are concerned with propositions expressed in a natural language in a fundamental way. The main difference between such approaches and Computing with Words (CW) is that the methodology of CW – which is based on fuzzy logic – provides a much more expressive language for knowledge representation and much more versatile machinery for reasoning and computation [245].

Computing with Words provides a methodology for what may be called a Computational Theory Of Perceptions (CTP) – a theory which may have an important bearing on how humans make – and machines might make – perception-based rational

decisions in an environment of imprecision, uncertainty and partial truth.. In CTP perceptions and queries are expressed as propositions in a natural language. Then, propositions and queries are processed by CW-based methods to yield answers to queries [245].

A basic difference between perceptions and measurements is that, in general, measurements are crisp whereas perceptions are fuzzy. In CTP, words play the role of labels of perceptions and, more generally, perceptions are expressed as propositions in a NL [267] A proposition, *p*, in NL qualifies to be an object of a computation in CTP if *p* is in PNL

perception = descriptor(s) of perception

The basic idea underlying the relationship between CW and CTP is conceptually simple. More specifically, in CTP perceptions and queries are expressed as propositions in a natural language. Then, propositions and queries are processed by CW-based methods to yield answers to queries. CW-based techniques are employed to translate propositions expressed in NL into what is called the Generalized Constraint Language (GCL) [245].

To deal effectively with world knowledge, relevance, deduction and precisiation, new tools are needed. The principal new tools are: Precisiated Natural Language (PNL); Protoform Theory (PFT); and the Generalized Theory of Uncertainty (GTU). These tools are drawn from Fuzzy Logic (see Figure 3)

The centerpiece of the new tools is the concept of a Generalized Constraint. The importance of the concept of a Generalized Constraint derives from the fact that in PNL and GTU it serves as a basis for generalizing the universally accepted view that information is statistical in nature. More specifically, the point of departure in PNL and GTU is the fundamental premise that, in general, information could be represented as a system of generalized constraints, with statistical information constituting a special case. This, much more general, view of information is needed to deal effectively with world knowledge, relevance, deduction, precisiation and related problems. [254]

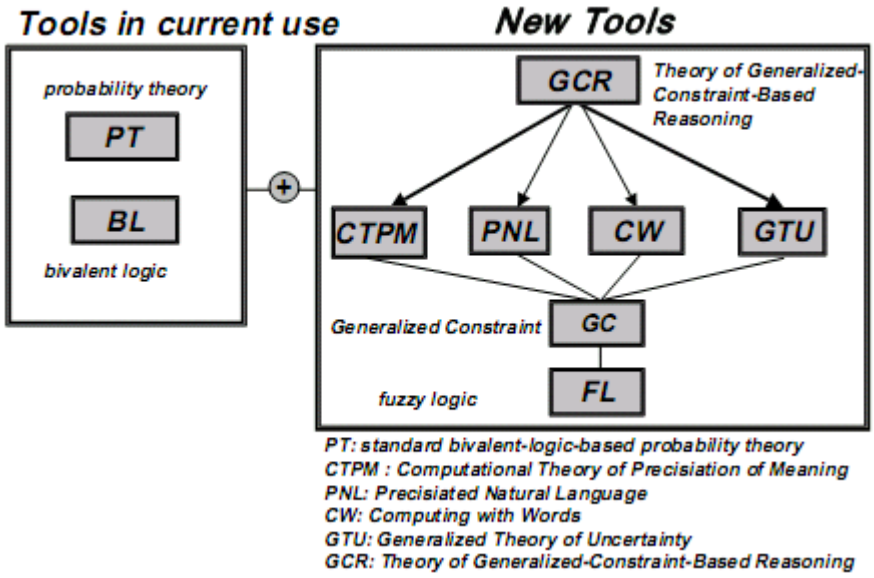


Figure 3 New tools organization

2.8. Natural Language Processing

Definition. A natural language (NL) is any of the languages naturally used by humans, i.e. not an artificial or man-made language such as a programming language. 'Natural language processing' (NLP) is a convenient description for all attempts to use computers to process natural language[32].

NLP is a subfield of artificial intelligence and linguistics, which studies the problems of automated generation and understanding of natural human languages. Natural language generation systems convert information from computer databases into normal-sounding human language, and natural language understanding systems convert samples of human language into more formal representations that are easier for computer programs to manipulate.

NLP is both a modern computational technology and a method of investigating and evaluating claims about human language itself[33]. Some prefer the term Computational Linguistics in order to capture this latter function, but NLP is a term that links back into the history of Artificial Intelligence (AI), the general study of cognitive function by

[32] P. Coxhead, Natural Language Processing & Applications, 2007 http://www.cs.bham.ac.uk/~pxc/nlpa/index.html
[33] Natural Language Processing Research Group, University of Sheffield Department of Computer Science http://nlp.shef.ac.uk/

computational processes, normally with an emphasis on the role of knowledge representations, that is to say the need for representations of our knowledge of the world in order to understand human language with computers.

NLP is the use of computers to process written and spoken language for some practical, useful, purpose: to translate languages, to get information from the web on text data banks so as to answer questions, to carry on conversations with machines, so as to get advice about, say, pensions and so on. These are only examples of major types of NLP, and there is also a huge range of lesser but interesting applications, e.g. getting a computer to decide if one newspaper story has been rewritten from another or not. NLP is not simply applications but the core technical methods and theories that the major tasks above divide up into, such as Machine Learning techniques, which is automating the construction and adaptation of machine dictionaries, modelling human agents' beliefs and desires etc. This last is closer to Artificial Intelligence, and is an essential component of NLP if computers are to engage in realistic conversations: they must, like us, have an internal model of the humans they converse with.

NLP includes: Speech synthesis, Speech recognition, Natural language understanding, Natural language generation, and Machine translation.

Natural language understanding is sometimes referred to as an AI-complete problem, because natural language recognition seems to require extensive knowledge about the outside world and the ability to manipulate it.

Goals and tasks. The goal of this field is to get computers to perform useful tasks involving human language, tasks like enabling human-machine communication, improving human-human communication, or simply doing useful processing of text or speech.

One goal of AI work in natural language is to enable communication between people and computers without resorting to memorization of complex commands and procedures. Automatic translation---enabling scientists, business people and just plain folks to interact easily with people around the world---is another goal. Both are just part of the broad field of AI and natural language[34], along with the cognitive science aspect of using computers to study how humans understand language.

[34] Association for the Advancement of Artificial Intelligence (AAAI), http://www.aaai.org/AITopics/pmwiki/pmwiki.php/AITopics/NaturalLanguage

Other language processing tasks, which is related to the Web, is Web-based question answering. This is a generalization of simple Web search, where instead of just typing keywords, a user might ask complete questions, ranging from easy to hard, like the following: How much Chinese silk was exported to England by the end of the 18th century? Answering complicated questions require extracting information that is embedded on a Web page, doing inference, or synthesizing and summarizing information from multiple sources or Web pages.

Problems. Some of the problems faced by natural language understanding systems are:

- **Word sense disambiguation**: Many words have more than one meaning. Most tasks in speech and language processing can be viewed as resolving **ambiguity** at one of these levels.
- **Syntactic ambiguity**: The grammar for natural languages is ambiguous. Choosing the most appropriate one usually requires semantic and contextual information.
- **Imperfect or irregular input**: Foreign or regional accents in speech; typing or grammatical errors in texts.
- **Speech acts and plans**: Sentences often don't mean what they literally say.

Evaluation. The goal of NLP evaluation is to measure one or more *qualities* of an algorithm or a system, in order to determine if (or to what extent) the system answers the goals of its designers, or the needs of its users. Research in NLP evaluation has received considerable attention, because the definition of proper evaluation criteria is one way to specify precisely an NLP problem, going thus beyond the vagueness of tasks defined only as *language understanding* or *language generation*. A precise set of evaluation criteria, which includes mainly evaluation data and evaluation metrics, enables several teams to compare their solutions to a given NLP problem.

Depending on the evaluation procedures, a number of distinctions are traditionally made in NLP evaluation.

- **Intrinsic vs. extrinsic evaluation**: Intrinsic evaluation considers an isolated NLP system and characterizes its performance mainly with respect to a gold standard result, pre-defined by the evaluators. Extrinsic evaluation considers

the NLP system in a more complex setting, either as an embedded system or serving a precise function for a human user. The extrinsic performance of the system is then characterized in terms of its utility with respect to the overall task of the complex system or the human user.

- **Black-box vs. glass-box evaluation**: Black-box evaluation requires one to run an NLP system on a given data set and to measure a number of parameters related to the quality of the process (speed, reliability, resource consumption) and, most importantly, to the quality of the result (e.g. the accuracy of data annotation or the fidelity of a translation). Glass-box evaluation looks at the design of the system, the algorithms that are implemented, the linguistic resources it uses (e.g. vocabulary size), etc. Given the complexity of NLP problems, it is often difficult to predict performance only on the basis of glass-box evaluation, but this type of evaluation is more informative with respect to error analysis or future developments of a system.

- **Automatic vs. manual evaluation**: In many cases, automatic procedures can be defined to evaluate an NLP system by comparing its output with the gold standard (or desired) one. Although the cost of producing the gold standard can be quite high, automatic evaluation can be repeated as often as needed without much additional costs (on the same input data). However, for many NLP problems, the definition of a gold standard is a complex task, and can prove impossible when inter-annotator agreement is insufficient. Manual evaluation is performed by human judges, which are instructed to estimate the quality of a system, or most often of a sample of its output, based on a number of criteria. Although, thanks to their linguistic competence, human judges can be considered as the reference for a number of language processing tasks, there is also considerable variation across their ratings. This is why automatic evaluation is sometimes referred to as objective evaluation, while the human kind appears to be more subjective.

Models.[35] Among the most important formal models or theories in language processing are [75] *state machines, rule systems, logic, probabilistic models, and vector-space models*. These models, in turn, lend themselves to a small number of algorithms, among the most important of which are *state space search* algorithms, such as *dynamic programming*, and machine learning algorithms, such as classifiers and Expectation-Maximization (EM) and other learning algorithms. Some of the variations of this basic model are *deterministic* and *non-deterministic finite-state automata* and *finite-state transducers.*

Closely related to these models are their declarative counterparts: formal rule systems. Among the more important ones, in both probabilistic and non probabilistic formulations, are *regular grammars* and *regular relations, context-free grammars,* and *feature-augmented grammars*. State machines and formal rule systems are the main tools used when dealing with knowledge of phonology, morphology, and syntax.

A third class of models that plays a critical role in capturing knowledge of language are models based on first-order logic, also known as the predicate calculus, as well as such related formalisms as lambda-calculus, feature structures, and semantic primitives. These logical representations have traditionally been used for modelling semantics and pragmatics, although more recent work has tended to focus on potentially more robust techniques drawn from non-logical lexical semantics.

Probabilistic models are crucial for capturing every kind of linguistic knowledge. Each of the other models (state machines, formal rule systems, and logic) can be augmented with probabilities. For example, the state machine can be augmented with probabilities to become the weighted automaton, or Markov model. Hidden Markov models or HMMs are used everywhere in the field, in part-of-speech tagging, speech recognition, dialogue understanding, text-to-speech, and machine translation.

Statistical natural language processing uses stochastic, probabilistic and statistical methods to resolve some of the difficulties discussed above, especially those which arise because longer sentences are highly ambiguous when processed with realistic grammars, yielding thousands or millions of possible analyses. Methods for

[35] Daniel Jurafsky, James H. Martin, Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, Second Edition, Pearson Prentice Hall, 2008

disambiguation often involve the use of corpora and Markov models. The technology for statistical NLP comes mainly from machine learning and data mining, both of which are fields of artificial intelligence that involve learning from data.

Finally, vector-space models, based on linear algebra, underlie information retrieval and many treatments of word meanings.

Processing language with any of these models typically involves a search through a space of states representing hypotheses about an input. In parsing, we search through a space of trees for the syntactic parse of an input sentence. For non-probabilistic tasks, such as tasks involving state machines, we use well-known graph algorithms such as depth-first search. For probabilistic tasks, we use heuristic variants such as best-first and A* search and rely on dynamic programming algorithms for computational tractability.

Following, a short overview of some of those models will be presented.

2.8.1. Context Free Grammar (CFG)

Context Free Grammar (CFG) [156] constitutes a system for defining the expressions of a language in terms of rules, which are recursive equations over expressions types, called nonterminals, and primitive expressions, called terminals. The term "context-free" expresses the fact that nonterminals can be rewritten without regard to the context in which they occur. A formal language is context-free if some context-free grammar generates it.

The standard notation for a context-free rule is

$N_0 \rightarrow V_1 \ldots V_n$

where N_0 is some nonterminal and the V_i are nonterminals or terminals. Such a rule has the following informal interpretation: “if expressions w_1, …, w_n match V1,….,Vn, respectively, then the single expression $w_1 \ldots w_n$ (the concatenation of the w_i) is itself of expression type N_0.”

Consider, for example, the following CFG for a fragment of English:

S → NP VP

NP → Det N OptRel

NP → PN

OptRel → e

OptRel → that VP

VP → TV NP

VP → IV

PN → terry

PN → shrdlu

Det → a

N → program

IV → halts

TV → writes

Table 1 Abbreviations

symbol	Abbreviates
S	Sentence
NP	Noun Phrase
VP	Verb Phrase
IV	Intransitive Verb
TV	Transitive Verb
PN	Proper Noun
Det	DETerminer
N	Noun
OptRel	OPTional RELative clause

Context-free grammars play a central role in the description and design of programming languages and compilers. They are also used for analyzing the syntax of natural languages.

Context-Free Grammar (CFG) or Phrase-Structure Grammar (PSG) is a formalism developed by Noam Chomsky [31], in the mid-1950s. A context-free grammar provides a simple and precise mechanism for describing the methods by which phrases in some natural language are built from smaller blocks, capturing the "block structure" of sentences in a natural way. Its simplicity makes the formalism amenable to rigorous mathematical study, but it comes at a price: important features of natural language syntax such as agreement and reference cannot be expressed in a natural way, or not at all.

Context-free grammars are simple enough to allow the construction of efficient parsing algorithms which, for a given string, determine whether and how it can be generated from the grammar. An Earley parser is an example of such an algorithm, while the widely used LR and LL parsers are more efficient algorithms that deal only with more restrictive subsets of context-free grammars.

A context-free grammar G is a 4-tuple $G = (V, \Sigma, R, S)$ where

- V is a finite set of *non-terminal* characters or variables. They represent different types of phrase or clause in the sentence.
- Σ is a finite set of *terminal*s, disjoint with V, which make up the actual content of the sentence.
- S is the start variable, used to represent the whole sentence (or program). It must be an element of V.
- R is a relation from V to $(V \cup \Sigma)^*$ such that there exist some element w of $(V \cup \Sigma)^*$ that (S, w) belongs to R.

In addition, R is a finite set. The members of R are called the *rule*s or *production*s of the grammar. The asterisk represents the Kleene star operation.

An obvious way to extend the context-free grammar formalism is to allow nonterminals to have arguments, the values of which are passed along within the rules. This allows natural language features such as agreement and reference, and programming language analogons such as the correct use and definition of identifiers, to be expressed in a natural way. E.g. we can now easily express that in English sentences, the subject and verb must agree in number.

In computer science, examples of this approach include affix grammars, attribute grammars, indexed grammars, and Van Wijngaarden two-level grammars. Similar extensions exist in linguistics.

2.8.2. Definite Clause Grammar

Definite clause grammars[36] (DCGs) are an extension of context free grammars that have proven useful for describing natural and formal languages, and that may be conveniently expressed and executed in Prolog. A Definite Clause Grammar rule in

[36] Definite Clause Grammars Baoqiu Cui 2000-04-23
http://www.cs.sunysb.edu/~sbprolog/manual1/node88.html

Prolog is executable because it is just a notational variant of a Prolog term that has the following general form:

Head -> *Body*.

with the declarative interpretation that ``a possible form for *Head* is *Body*". The procedural interpretation of a grammar rule is that it takes an input list of symbols or character codes, analyses some initial portion of that list, and produces the remaining portion (possibly enlarged) as output for further analysis.

Definite Clause Grammars[37] (DCGs) are convenient ways to represent grammatical relationships for various parsing applications. They can be used for natural language work, for creating formal command and programming languages.

Definite Clause Grammar (DCG) is a Prolog preprocessor that takes DCG grammar rules, and adds linked difference lists to the goals.

DCG provides a syntax for writing more readable grammar parsing rules, without including the linked difference lists. The DCG preprocessor takes the DCG rule and translates it into pure Prolog, adding the linked difference lists.

Difference lists are powerful tools for parsing applications, in which the input stream is represented by difference lists.

Difference lists are pairs of lists used to represent the list of elements (tokens, words, character codes, ...) being parsed. The actual list being represented is the 'difference' between the first list and the second list. For example, [the, cat] might be represented by the two lists [the, cat, chases, the, mouse] and [chases, the, mouse]. Or more generally, [the, cat | X] and X.

The use of difference lists[38] for parsing is so common in Prolog, that most Prologs contain additional syntactic sugaring that simplifies the syntax by hiding the difference lists from view. This syntax is called Definite Clause Grammar (DCG), and looks like normal Prolog, only the symbol ‘:-‘ is replaced with an arrow -->. The DCG representation is parsed and translated to normal Prolog with difference lists.

Using DCG, the 'sentence' predicate developed earlier would be phrased

[37] Amzi Inc. Definite Clause Grammars (DCGs), 2004, http://www.amzi.com/manuals/amzi7/pro/ref_dcg.htm,
[38] Diana Inkpen, Natural Language Processing in Prolog, University of Toronto, 2001, http://www.cs.toronto.edu/~dianaz/324/tut7.txt

sentence --> nounphrase, verbphrase.

This would be translated into normal Prolog, with difference lists. The above example would be translated into the following equivalent Prolog.

sentence(S1, S2):- nounphrase(S1, S3), verbphrase(S3, S2).

The key idea underlying difference lists is to represent the information about grammatical categories not as a single list, but as the difference between two lists. Think of the first list as *what needs to be consumed* (or if you prefer: the *input list*), and the second list as *what we should leave behind* (or: the *output list*). Viewed from this (rather procedural) perspective the difference list

[a,woman,shoots,a,man], [].

represents the sentence *a woman shoots a man* because it says: *If I consume all the symbols on the left, and leave behind the symbols on the right, I have the sentence I am interested in.*

That is: the sentence we are interested in is the difference between the contents of these two lists.

A definite clause is a Horn clause that has exactly one positive literal. A Horn clause without a positive literal is called a goal.

Horn clauses express a subset of statements of first-order logic. Programming language Prolog is built on top of Horn clauses. Prolog programs are comprised of definite clauses and any question in Prolog is a goal.

The programming language, Prolog, was born of a project aimed not at producing a programming language but at processing natural languages. The project gave rise to a preliminary version of Prolog at the end of 1971 and a more definitive version at the end of 1972.

Colmerauer [34, 35] was working on natural language understanding, using logic to represent semantics and using resolution for question-answering. During the summer of 1971, Colmerauer and Kowalski discovered that the clausal form of logic could be used to represent formal grammars and that resolution theorem provers could be used for parsing.

Colmerauer [156] started developing a language that could at the same time be used for language analysis and for implementing deductive question-answering mechanisms. It eventually became clear that a particular kind of linear resolution restricted to definite clauses had just the right goal-directness and efficiency, and also enough expressive power for linguistic rules and some important aspects of the question-answering problem. Their approach was first described as a tool for natural language processing applications.

DCGs has been used for processing Programming Languages (RDF[39], XML[40], HTML) [44], Controlled Natural Languages, and Natural Language [218, 219, 220, 221] [41][42], as well as Information Retrieval [47, 136, 151].

2.8.3. Montague grammar

Montague [148] was a logician and philosopher at UCLA who was one of the major figures in the early days of establishing formal approaches to natural language semantics. His seminal works on language [126, 127, 129] founded the theory known after his death as Montague grammar, one of the main starting points for the field of formal semantics [149]. In the late 1960's, he faced the project of developing a philosophically satisfactory and logically precise account of syntax, semantics, and pragmatics, encompassing both formal and natural languages. In 1970 he constructed the first popular formal analysis of quantification [11].

Montague grammar [147] is a theory of semantics, and of the relation of semantics to syntax. Classical Montague grammar had its roots in logic and the philosophy of language. The most constant features of the theory over time have been the focus on truth-conditional aspects of meaning, a model-theoretic conception of semantics, and the methodological centrality of the Principle of Compositionality: "The meaning of a whole is a function of the meanings of its parts and their mode of syntactic combination."

Montague's UG (Universal Grammar) [127] contains the most general statement of Montague's formal framework for the description of language. The central idea is that a

[39] SWI-Prolog RDF parser, http://www.swi-prolog.org/packages/rdf2pl.html
[40] Applications of DCG and Difference Lists, http://www.amzi.com/manuals/amzi7/pro/ref_dcg.htm
[41] Anthony Aaby, A Natural Language Processor, 1999, **http://moonbase.wwc.edu/~aabyan/LogicPgmg/CODE/DCG/doc/doc.html**
[42] Perpetual Amoah, Rose Luliana, Lila Ghemri, A Sentence Analyzer for Detecting Grammatical Errors, Texas Southern University Research week 2006, http://itscience.tsu.edu/ghemri/CREU/CREU.htm

grammar should be able to be cast in the following form: the syntax is an algebra, the semantics is an algebra, and there is a homomorphism mapping elements of the syntactic algebra onto elements of the semantic algebra. The nature of the elements of both the syntactic and the semantic algebras is open to variation; what is constrained by compositionality is the relation of the semantics to the syntax. This very general definition leaves a great deal of freedom as to nature of these algebras. In a logical language, the elements of the syntactic algebra can be the well-formed expressions. But for a natural language, ambiguity makes that impossible, since the homomorphism requirement means that each element of the syntactic algebra must be mapped onto a unique element of the semantic algebra. So for a natural language, the elements of the syntactic algebra are usually taken to be expressions together with disambiguating structural descriptions, typically trees of some sort.

It was the short but densely packed PTQ (The Proper Treatment of Quantification in Ordinary English) [129] that had the most impact on linguists and on the subsequent development of formal semantics. Montague grammar has often meant PTQ, but it is the broader algebraic framework of UG [127] that constitutes Montague's theory of grammar. Crucial features of that theory include the truth-conditional foundations of semantics, the algebraic interpretation of the principle of compositionality, and the power of a higher-order typed intensional logic.

The richness of Montague's logic was crucial for the possibility of a compositional semantic interpretation of independently motivated syntactic structure. Montague's use of a richly typed logic with lambda-abstraction made it possible for the first time to interpret noun phrases (NPs) like every man, the man, a man uniformly as semantic constituents ("generalized quantifiers"), something impossible with the tools of first-order logic. This was well illustrated in PTQ. PTQ also contained innovative treatments of quantifier scope and binding, intensional transitive verbs, phrasal conjunction, adverbial modification, and more.

2.8.4. Link Grammars

Most sentences of most natural languages have the property that if arcs are drawn connecting each pair of words that relate to each other, then the arcs will not cross. This well-known phenomenon is called planarity by Sleator and Temperley [192], and is the basis of Link Grammars, the formal language system they proposed.

Link grammars [89] are a formalism based on context-free grammatical for the description of natural language. They resemble dependency grammars and categorical grammars, although there are also many significant differences.

An algorithm for determining maximum-likelihood estimates of the parameters of these models is used. The language model differs from previous models based on stochastic context-free grammars in that it is highly lexical. In particular, the language model includes the familiar *n*-gram models as a natural subclass [89].

Link grammars have a unique combination of useful properties:

- In a link grammar each word of the lexicon is given a definition describing how it can be used in a sentence. The grammar is distributed among the words. Such a system is said to be *lexical*. This has several important advantages. It makes it easier to construct a large grammar, because a change in the definition of a word only affects the grammaticality of sentences involving that word. The grammar can easily be constructed incrementally. Furthermore, expressing the grammar of the irregular verbs of English is easy -- there's a separate definition for each word.
- In English, whether or not a noun needs a determiner is independent of whether it is used as a subject, an object, or even if it is part of a prepositional phrase. The algebraic notation developed for expressing a link grammar takes advantage of this orthogonality.
- Another interesting property of link grammars is that they have no explicit notion of constituents or categories. In most sentences parsed with our dictionaries, constituents can be seen to emerge as contiguous connected collections of words attached to the rest of the sentence by a particular type of link.

A link grammar consists of a set of words (the terminal symbols of the grammar), each of which has a linking requirement. A sequence of words is a sentence of the language defined by the grammar if there exists a way to draw arcs (i.e. links) among the words so as to satisfy the following conditions:

- Planarity: when drawn above the words, the links do not cross.
- Connectivity: all the words of the sequence are connected together by the links.
- Satisfaction: The links satisfy the linking requirements of each word in the sequence.

Categories as noun phrase, verb phrase, etc. play no role, thus the system is word-based.

The basic idea of Link Grammars [191, 192] is the following. Words are considered as blocks with connectors coming out. There are different types of connectors, which may point to the right or to the left. Right-pointing connectors are labeled "+", left-pointing connectors are labeled "-". Two connectors of the same type but different sign could form a "link". Words have rules about how their connectors can be connected up, that is, rules about what would constitute a valid use of that word. A valid sentence is one in which all the words present are used in a way which is valid according to their rules, and which also satisfies certain global rules.

A simple dictionary[43] entry would look like this:

blah: A+;

This means that if the word "blah" is used in a sentence, it must form an "A" link with another word; that is, there must be another word to the right of it with an "A-" connector. Otherwise the sentence is not valid. The expression following the colon is the "linking requirement" for the word.

A word may have more than one connector that has to be connected. This would be notated as

blah: A+ & B+;

A word may have a rule that either one of two (or one of several) connectors can be used, but exactly one must be used. In the dictionary, we notate this as

blah: A+ or B-;

[43] Davy Temperley Daniel Sleator John Lafferty, Link Gramma, 2007, http://www.link.cs.cmu.edu/link/index.html

This means that if the word can make an "A" link to the right or a "B" link to the left, its use in the sentence is valid; but it must make one or the other, and it can not make both.

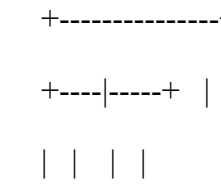
These rules can be combined and such expressions can be nested without limit. For example, consider the following notation:

blah: A+ or (B- & C+);

This means that the word must make either an "A" link to the right, or a "B" link to the left and a "C" link to the right. No other combination will be valid.

As well as these "word rules", which are specified in the dictionary, there are two other global rules which control how words can be connected.

First of all, links can not cross. This is the "crossing-links" (or "planarity") rule. For example, the following way of connecting these four words (connecting "cat" to "dog" and "horse" to "fish") would be illegal. The parser simply will not find such linkages.



cat horse dog fish

Secondly, all the words in a sentence must be indirectly connected to each other. This is the "connectivity" rule. Therefore the following way of connecting these four words would be illegal (if it was the entire linkage).

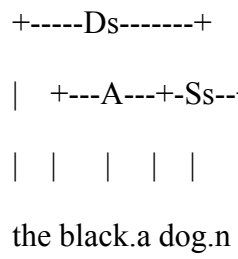
cat horse dog fish

A valid sentence is therefore one which can be linked up in a way that:

- All the words are used in a way that satisfies their linking requirements.
- The crossing- links and connectivity rules are not violated.

The structure assigned to a sentence by a link grammar is rather unlike any other grammatical system, although it is certainly related to dependency grammar. Rather than thinking in terms of syntactic functions (like subject or object) or constituents (like

"verb phrase"), one must think in terms of relationships between pairs of words. In the sentence below, for example, there is an "S" ("subject") relation between "dog" and "has"; a "PP" (past-participle) relationship between "has" and "gone"; and a "D" (determiner) relation between "the" and "dog". (Ignore the lower-case letters for the moment; they will be explained below.)



It may be seen, however, that parts of speech, syntactic functions, and constituents may be recovered from a link structure rather easily. For example, whatever word is on the left end of an "S" link is the subject of a clause (or the head word of the subject phrase); whatever is on the right end is the finite verb; whatever is on the left-end of a D link is a determiner; etc. Moreover, all nouns, verbs, and adjectives in the dictionary are subscripted (as ".n", ".v", or ".a"), so in these cases the syntactic category of the word is made explicit. The constituent structure of sentences, while not absolutely explicit, is also quite "close to the surface" in linkage structures.

2.8.5. Affix grammar

An affix grammar[44] is a particular kind of formal grammar used in computer science to describe computer languages. Affix grammars are context-free, meaning that the left hand sides of the language productions consist of a single symbol. Most commonly used programming languages (Java, C, etc.) can be described as a series of such productions. Most often, these are expressed using Backus–Naur form, which can be used to express an affix grammar.

Affix Grammars over a Finite Lattice (AGFLs) [84], a simple form of two-level grammars admitting quite efficient implementations, has been proposed as a formalism to express the syntax of natural languages.

An Affix Grammar over a Finite Lattice (AGFL) can be seen as a CF grammar extended with set-valued features (also called affixes or attributes) for expressing agreement between parts of speech. Those features form a finite categorization and

[44] Wikipedia http://en.wikipedia.org/wiki/Affix_grammar

express finite semantical categories like number, person, time, etc. Such grammars have been used extensively in classical Linguistics (albeit in a non-formal form) for more than two thousand years.

Affix grammars [62] establish long-range relations by duplicating information in an early stage; this information is, however, not part of the non-terminal name, but is passed as an independent parameter, an affix, which can, for instance, be an integer value. Normally these affixes are passed on to the members of a rule, until they are passed to a special kind of non-terminal, a primitive predicate. Rather than producing text, a primitive predicate contains a legality test. For a sentential form to be legal, all the legality tests in it have to succeed.

Conjugation and declination rules based on feature distinctions can be modelled easily in an AGFL. Indeed the original motivation for Affix Grammars was linguistic, and their first application was a generative grammar for a small part of English [132], which was presented to the Eratom-colloquium organized by Prof. E.W. Beth in 1962.

An AGFL consists of meta-rules and rules in arbitrary order. The meta-rules or affix rules are a collection of restricted Context Free rules, together forming the second level of the AGFL. Each affix rule defines the direct productions of a nonterminal affix. Such a direct production is either a terminal affix or a nonterminal affix, and recursion is not allowed. Consequently, a nonterminal affix has one or more terminal affixes as terminal productions, which must be all different.

Terminal affixes (or, rather, their representations) are written in small letters. Nonterminal affixes (or, rather, their names) are written with capital letters. Spaces may be used within nonterminal affixes to enhance readability.

Meta-rules can be recognized by the double colon separating their left-and right-hand side. The meta-rule

NUMB:: singular; plural.

defines the nonterminal affix NUMB to have two direct productions. These two productions are terminal affixes and therefore also its terminal productions.

The meta-rules may be seen as a type-system, in which the nonterminal affixes, with their respective domains, are the types. The domain of a nonterminal affix is the set of its terminal productions.

The rules of an AGFL are Context Free rules augmented with affixes. A rule consists of a left-hand-side, followed by a single colon, followed by a right-hand-side:

noun group (NUMB)

adjective, noun group (NUMB);

subst (NUMB)

The left-hand-side of a rule consists of a nonterminal symbol, the head, optionally followed by a list of affix expressions, the parameters, enclosed between brackets.

The right-hand-side of a rule consists of one or more alternatives, separated from one another by semicolons. An alternative is a possibly empty list of members, separated by commas. A member is either a terminal symbol or it is a call, which looks just like a left-hand-side. Nonterminal symbols can be written in small or large letters, and spaces can be used to enhance readability. A terminal symbol is written as its representation enclosed between quotes.

The possible values of an affix form a lattice. The lattice for the affix defined by

PERSON :: first; second; third.

can be depicted as in Figure 4

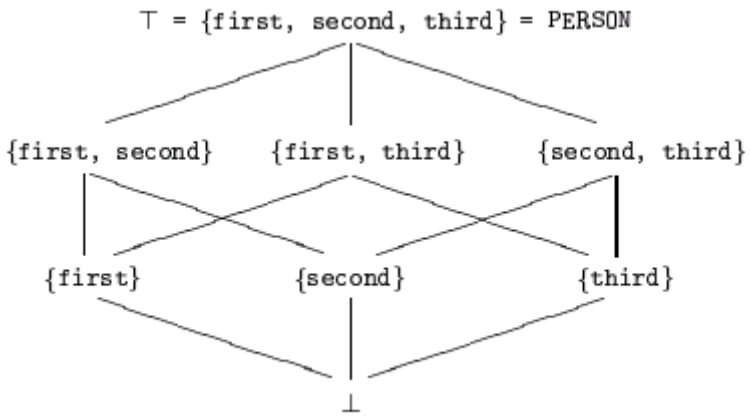


Figure 4 Lattice for the PERSON affix

AGFL is particularly suitable for the above purpose, for the following reasons:

- *Penalties* specified in the grammar can be used to influence the ranking of the various parsings.

- *Lexical probabilities* derived from word frequencies can be used to weed out unlikely analyses.
- Left-to-right *segment parsing,* in combination with penalties, allows robust analysis of incorrect input.
- *Transductions* can be specified in the grammar itself.

2.8.6. Constraint Grammar

Constraint Grammar is a methodological paradigm for Natural Language Processing (NLP) that was launched by Fred Karlsson in 1990 [76, 77]. Linguist-written, context dependent rules are compiled into a grammar that assigns grammatical tags to words or other tokens in running text. Typical tags address lemmatisation, inflexion, derivation, syntactic function, dependency, valency, case roles, semantic type etc. Typical CGs consist of thousands of rules, that are applied set-wise in progressive steps, covering ever more advanced levels of analysis. Within each level, safe rules are used before heuristic rules, and no rule is allowed to remove the last reading of a given kind, thus providing a high degree of robustness.

Constraint Grammar is a formalism to be used for parsing where the grammar statements are closer to real text sentences and more directly address some notorious parsing problems, especially ambiguity. The formalism is a linguistic one. It relies on transitional probabilities in an indirect way. The probabilities are not part of the description.

The descriptive statements, constraints, do not have the ordinary task of defining the notion of a 'correct sentence'. They are less categorical in nature, more closely tied to morphological features, and more directly geared towards the basic task of parsing, considering this task as one of inferring surface structure from a stream of concrete tokens in a basically bottom-up mode. . Constraints are formulated on the basis of extensive corpus studies. They may reflect absolute, rule like facts, or probabilistic tendencies where a certain risk is judged to be proper to take. Constraints of the former rule-like type are of course preferable.

CG taggers and parsers have written for a large variety of languages, routinely achieving accuracy F-scores for PoS (word class) of over 99%. A number of syntactic CG systems have reported F-scores of around 95% for syntactic function labels. CG systems can be used to create full syntactic trees in other formalisms by adding small,

non-terminal based phrase structure grammars or dependency grammars, and a number of corpus/treebank projects have used Constraint Grammar for automatic annotation. CG methodology has also used in a number of language technology applications, such as spell checkers and machine translation systems.

For most languages, a lexicon based morphological analyzer provides input to the first CG level, while the output of the last CG-level can be converted into syntactic tree structures by specially designed Phrase Structure Grammars (PSG's), using syntactic functions, not words, as terminals.

2.8.7. Head-driven phrase structure grammar

Head-driven phrase structure grammar[45] (HPSG) is a highly lexicalized, non-derivational, constraint-based, generative grammar theory developed by Carl Pollard and Ivan Sag in 1985 [159]. It uses a uniform formalism and is organized in a modular way which makes it attractive for natural language processing.

By non-derivational, it is meant that HPSG has no notion of deriving one structure or representation from another. Instead, different representations are just subparts of a single larger structure related by declarative constraints; thus, it is constraint based. It is said to be surface oriented because it provides a direct characterization of the surface order of elements in a sentence. Another major characteristic of HPSG is that it is highly lexicalist in that it makes use of a rich and complex lexicon in its representations.

HPSG puts a lot of emphasis on the precise mathematical modeling of linguistic entities. Because of its focus on precision, a lot of linguistic computer implementations are based in HPSG.

HPSG representations use feature structures, often written as attribute-value-matrixes (AVMs), to represent grammar principles, grammar rules and lexical entries.

An important concept in HPSG representations is that of a sign. A sign is a collection of information, including phonological, syntactic and semantic constraints, which is what is represented in AVMs. AVMs encode feature structures where each attribute (feature) has a type and is paired with a value. The notion of sign is formalized by being the type of every constituent admitted by HPSG, including both words and

[45] Pollard, Carl. 1997. *Lectures on the Foundations of HPSG*. Unpublished manuscript: Ohio State University. <http://www-csli.stanford.edu/~sag/L221a/cp-lec-notes.pdf>

phrases. Signs receive the subtypes *word* or *phrase* depending on their phrasal status. These subtypes differ in that they conform to different constraints.

HPSG uses Ferdinand de Saussure's notion of the sign. According to him, language is made up of signs and every sign has two sides:

- the signifier (French *signifiant*): the "shape" of a word, its phonic component, i.e. the sequence of letters or phonemes e.g. C-A-T
- the signified (French *signifié*): the ideational component, the concept or object that appears in our minds when we hear or read the signifiere.g. a small domesticated feline

(The signified is not to be confused with the referent. The former is a mental concept, the latter the actual object in the world).

Furthermore, Saussure separated speech acts (*la parole*) from the system of a language (*la langue*). *Parole* was the free will of the individual, whereas *langue* was regulated by the group, albeit unknowingly.

An HPSG grammar includes principles and grammar rules and lexicon entries which are normally not considered to belong to a grammar. The formalism is based on lexicalism. This means that the lexicon is more than just a list of entries; it is in itself richly structured. Individual entries are marked with types. Types form a hierarchy.

HPSG generates strings by combining signs, which are defined by their location within a type hierarchy and by their internal feature structure, represented by Attribute Value Matrices (AVMs). [160, 177] Features take types or lists of types as their values, and these values may in turn have their own feature structure. Grammatical rules are largely expressed through the constraints signs place on one another. A sign’s feature structure describes its phonological, syntactic, and semantic properties.

In the simplified AVM for the word “walks” (see Figure 5), the verb’s categorical information is divided into features that describe it (HEAD) and features that describe its arguments (VALENCE). In common notation, AVMs are written with features in upper case and types in italicized lower case. Numbered indices in an AVM represent token identical values.

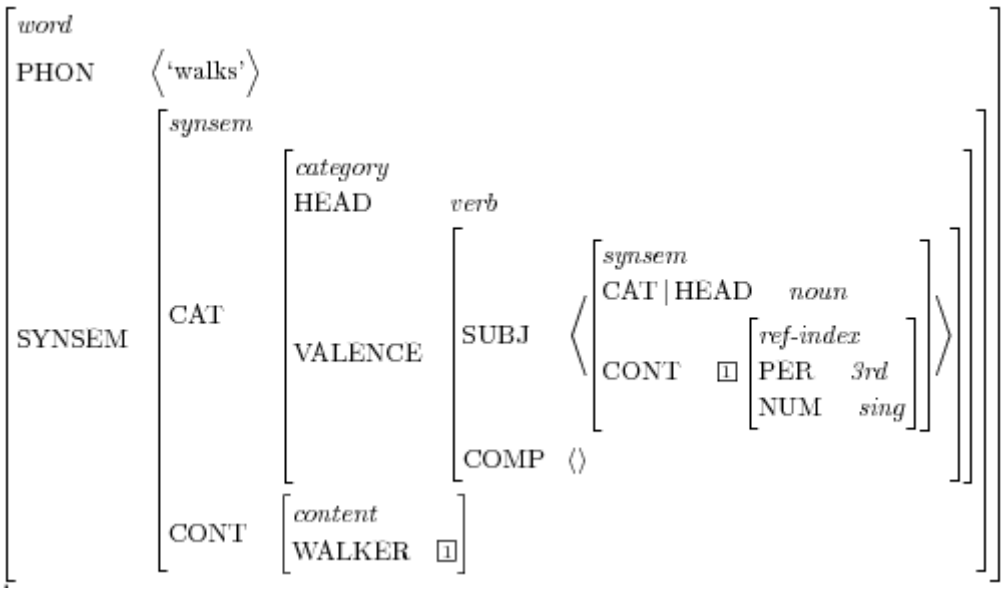


Figure 5 Simplified AVM for the word "walks"

"Walks" is a sign of type *word* with a head of type *verb*. As an intransitive verb, "walks" has no complement but requires a subject that is a third person singular noun. The semantic value of the subject (CONTENT) is co-indexed with the verb's only argument (the individual doing the walking). The AVM for the word "she" (see Figure 6) represents a sign with a SYNSEM value that could fulfill those requirements.

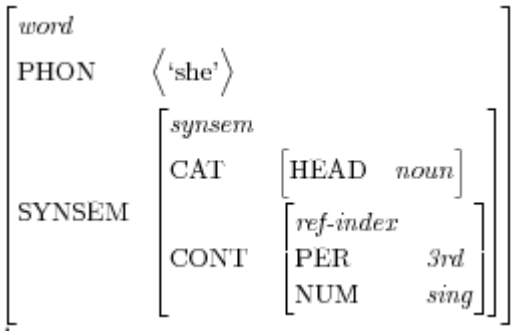


Figure 6 Simplified AVM for the word "she"

Signs of type *phrase* unify with one or more daughters and propagate information upward. The AVM in Figure 7 is for a *head-subj-phrase* that requires two daughters: the head daughter (a verb) and a non-head daughter that fulfills the verb's SUBJ constraints.

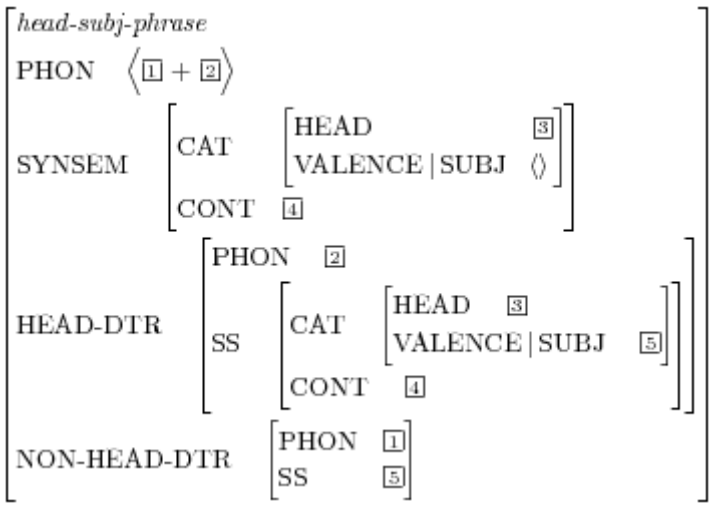


Figure 7 Simplified AVM for a head subj-phrase

The end result is a sign with a verb head, empty subcategorization features, and a phonological value that orders the two daughters.

2.8.8. SAX: Sequential Analyzer for syntaX and semantics

SAX (Sequential Analyzer for syntaX and semantics) [114, 115] is a syntactic analyzer based on logic programming. SAX employs a bottom-up and breadth-first parsing algorithm. The SAX grammar rules are basically written in Definite Clause Grammar (DCG). The SAX grammar rules are translated into a parsing program written in Prolog. SAX is implemented in SICStus Prolog Ver 0.7.

2.8.9. Link Grammar Parser

The parsing algorithm [58] is a natural extension of the original dynamic programming recognition algorithm which recursively counts the number of linkages between two words in the input sentence. The modified algorithm uses the notion of a *null link* in order to allow a connection between any pair of adjacent words, regardless of their dictionary definitions. The algorithm proceeds by making three dynamic programming passes. In the first pass, the input is parsed using the original algorithm which enforces the constraints on links to ensure grammaticality. In the second pass, the total *cost* of each substring of words is computed, where cost is determined by the number of null links necessary to parse the substring. The final pass counts the total number of parses with minimal cost. All of the original pruning techniques have natural counterparts in the robust algorithm.

The Link Grammar Parser[46] is a syntactic parser of English, based on link grammar, an original theory of English syntax. Given a sentence, the system assigns to it a syntactic structure, which consists of a set of labeled links connecting pairs of words. The parser also produces a "constituent" representation of a sentence (showing noun phrases, verb phrases, etc).

The system is written in generic C code, and runs on any platform with a C compiler. The parser has a dictionary of about 60000 word forms. It has coverage of a wide variety of syntactic constructions, including many rare and idiomatic ones.

The parser is robust; it is able to skip over portions of the sentence that it cannot understand, and assign some structure to the rest of the sentence. It is able to handle unknown vocabulary, and make intelligent guesses from context and spelling about the syntactic categories of unknown words. It has knowledge of capitalization, numerical expressions, and a variety of punctuation symbols.

The system has a "phrase-parser": a program which takes a "linkage" (the usual link grammar representation of a sentence, showing links connecting pairs of words) and derives a constituent or phrase-structure representation, showing conventional phrase categories such as noun phrase (NP), verb phrase (VP), prepositional phrase(PP), clause (S), and so on.

The constituent structure of a linkage can also be accessed as a tree data structure. Each node in this tree consists of a pair of integers, indicating a span of words in the sentence (the first and last word of the constituent), a character string indicating the "type" of constituent (NP, PP, etc.), and pointers to the node's children.

Another feature of the system is "morpho-guessing" of unknown words based on its context and its spelling. For example, words ending in "-ed" are assumed to be past-tense verbs; words ending in "-ing" are assumed to be present participles.

The phrase-parser has been tested against the Penn Treebank, a large database of newspaper text, where the parser gets about 75% of constituents correct. (The parser's "precision" score--the proportion of the constituents it finds that are correct--is about the same as its "recall score"--the proportion of correct constituents that it finds.) If the

[46] John Lafferty, The Link Parser Application Program Interface (API) *http://www.link.cs.cmu.edu/link/api/index.html* 2003

input is restricted to hard news and financial news, the parser gets about 82% of constituents correct.

2.8.10. EP4IR: English Phrases for Information Retrieval

The ``English Phrases for IR" (EP4IR)[47] grammar is a grammar of English concentrating on the description of the noun phrase and the verb phrase. The grammar provides detailed Part-Of-Speech information with a large lexicon. It is tailored towards Information Retrieval applications. The parsers generated from EP4IR are robust against badly formed input and unknown words. For each phrase, one or more analyses can be generated, in decreasing order of probability.

From the EP4IR grammar and lexicon, an English parser can be generated automatically using the AGFL system, which produces as its output not parse trees but Head/Modifier trees, binary dependency trees that can be unnested to Head/Modifier pairs.

The English Phrases for Information Retrieval (EP4IR) grammar started life in 1962 as the first affix grammar for a natural language[48] (English), developed, implemented as a generative device and presented to the EURATOM colloquium at the University of Amsterdam by two students, Lambert Meertens and Kees Koster.

EP4IR was revived in the early nineties of the previous century, cast into a modern notation (AGFL), extended and completed with a large lexicon. It describes not only the sentences of the language, but also their transduction to Head/Modifier trees.

Being especially directed towards IR applications, the EP4IR grammar [85] does not set out to give a linguistically impeccable “account” or all English sentences, but it describes mainly various forms of verb phrases (VP´s), each consisting in the application of a certain verbal part to certain noun phrases (NP´s) which occur as its complements. These phrases are transduced into HM frames, performing *syntactic normalization* by means or transformations during the transduction: Elements of the phrase are selected. The transformations are purely syntactical; they take no other information into account than the grammar, the lexicon and the input.

[47] English Phrases for IR EP4IR, http://www.agfl.cs.ru.nl/ep4ir/index.html
[48] English Phrases for Information Retrieval. The EP4IR grammar of English, Version 2.0, http://www.agfl.cs.ru.nl/ep4ir/grammar.html

The main practical limitations of the grammar however are caused by lexical and syntactical ambiguity. Based on the experiences obtained in research and teaching with the EAG (Extended Affix Grammar) formalism, in 1991 at University of Nijmegen, a simple form of Unification Grammars especially suited for the syntactic description of natural languages was defined: AGFL.

This formalism was implemented between 1991 and 1996 in the form of a parser generator with an efficient lexicon builder. In the period 2000/2001, the formalism was revised in the light of experience and was brought under the GNU General Public Licence.

AGFL is the first parser generator for natural languages available under the GNU Public License. On April 2002, Richard Stallman personally gave the green light.

The AGFL[49] system is a system written in C for the development of grammars for natural languages and the automatic generation of efficient parsers from such grammars.

The latest public release of AGFL is version 2.4 (Unix), version 2.5 (Unix) or version 2.3 (Windows, including Windows 2000 & XP).

2.8.11. Attribute Logic Engine (ALE)

Attribute-Logic Engine (ALE) is a logic programming language very similar to Prolog, except that its terms are typed feature structures. ALE is based on the typed attribute-value logic and associated grammar and constraint logic programming model developed in [28]. ALE integrates phrase structure parsing, semantic-head-driven generation and constraint logic programming with typed feature structures as terms. ALE [184] is an integrated system of definite clause logic programming and phrase structure parsing.

The terms involved in ALE grammars and logic programs are specified using a typed extension of Rounds-Kasper attribute-value logic, which includes variables, full disjunction, inequations, and functional descriptions. There is a strong type discipline enforced on descriptions, allowing many errors to be detected at compile-time.

The phrase structure system employs a bottom-up, all-paths dynamic chart parser. Parser performance is similar to that of the logic programming system.

[49] http://www.agfl.cs.ru.nl/about.html

The combination of typing and appropriateness allows **ALE** to compile extensively all of the basic operations it performs on typed feature structures. All operations and declarations in ALE use *typed feature structures* as terms. The most important basic operation is unification, the consistent combination of partial information from two or more feature structures.

The Attribute-Logic Engine[50] (ALE) is a long-term project surrounding the use of attribute-value logics to model constraint-based linguistic theories. The system includes constraint resolution, definite clause logic programming, and bottom-up chart parsing and head-driven generation in any combination.

ALE is a strongly typed language. Every structure must have a declared type. Types are defined by an inheritance structure and subtype relation. Basic representation scheme used is the typed feature structures. Types are assigned to appropriate feature value pairs. Type structure of ALE is very similar to the HPSG including properties like inheritance, nesting, and well-typedness.

ALE allows definition of general constraints on types. One can put restrictions on the feature structures of a particular type.

Another feature of ALE is the definite clauses in which all functionality of PROLOG definite clauses is provided with feature structure unification instead of simple term unification. One of the most distinct features of ALE is the support for phrase structure grammars. ALE provides phrase structure rules to be coded like Definite Clause Grammars of PROLOG. It has a built-in bottom-up chart parser in addition to feature structure unification. DCG's are top-down and depth-first. However ALE parser works in a combined manner asserting edges to chart right to left while applying rules left to right.

ALE is implemented in Prolog and compiles all basic operations over feature structures into Prolog code. ALE compilation can thus be viewed as a preprocessing step similar to YACC.

Attribute-Logic Engine (ALE) Version 3.2[51] is a freeware logic programming and grammar parsing and generation system written in Prolog by Bob Carpenter, and Gerald

[50] Bob Carpenter's Projects, http://www.colloquial.com/carp/Projects/index.html
[51] Attribute-Logic Engine (ALE) Home page http://www.cs.toronto.edu/~gpenn/ale.html

Penn. ALE 3.2 works with SICStus 3.8.6 and SWI 4.0. It has been widely adopted by over 150 research centers and universities for implementing feature-based grammars.

Arbitrary constraints may be attached to types, and types may be declared as having extensional structural identity conditions. Grammars may also interleave unification steps with logic program goal calls (as can be done in DCGs), thus allowing parsing to be interleaved with other system components. ALE was developed with an eye toward Head-Driven Phrase Structure Grammar (HPSG), but it can also execute PATR-II grammars, definite clause grammars (DCGs), Prolog, Prolog-II, and LOGIN programs, etc. With suitable coding, it can also execute several aspects of Lexical-Functional Grammar (LFG).

The Typed Feature Structure (TFS) representation formalism is an attempt to provide a synthesis of several of the key concepts of unification-based grammar formalisms (feature structures), knowledge representation languages (inheritance) and logic programming (logical variables and declarativity). The inheritance-based constraint architecture embodied in the TFS system integrates two computational paradigms: the *object-oriented* approach offers complex, recursive, possibly nested, record objects represented as typed feature structures with attribute-value restrictions and (in)equality constraints, and multiple inheritance; the *relational programming* approach offers declarativity, logical variables, non-determinism with backtracking, and existential query evaluation.

Typed feature structures [28, 155] are a generalization of the frames found in artificial intelligence classifiers and partial record structures in databases. They were first introduced by linguists to characterize natural language grammars in terms of well-formedness constraints. The logic of typed feature structures is strongly typed, with the types being arranged in a meet semi-lattice that organizes or classifies the information that the types encode.

Typed feature structures can also bear features. Every meet semi-lattice of types must come with a set of *appropriateness conditions* that specifies, for each type, the set of features for which every object of that type can and must have a value.

The usual graphic representation for feature structures is by means of Attribute-Value Matrixes (AVM). Attributes of a feature structure can have just simple or atomic

values like in Figure 8 or can also include nested structures like in Figure 9, where the value of the *head* attribute is another feature structure.

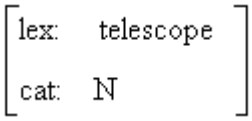


Figure 8 Feature structure with simple values

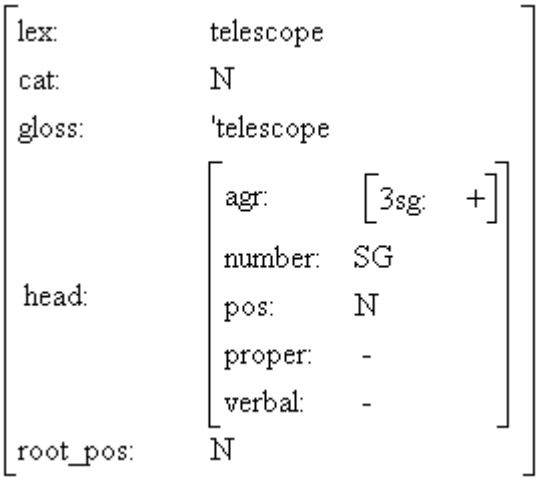


Figure 9 Feature structure with nested structures

The most important operations related with hierarchical systems of features structures are subsumption and unification.

2.8.12. NLTK

Natural Language Toolkit [17] or, more commonly, NLTK is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for the Python programming language. It provides many NLP data types, processing tasks, corpus samples and readers, together with animated algorithms, tutorials, and problem sets [106]. The NLTK project is led by Steven Bird, Edward Loper, and Ewan Klein.

The NLTK project began when Steven Bird was teaching at the University of Pennsylvania in 2001, and hired his star student, Edward Loper, to be teaching assistant. They agreed a plan for developing software infrastructure for NLP teaching that could be easily maintained over time.

NLTK Version 0.9.1, January 2008[52], contains:

[52] NLTK Web page http://nltk.sourceforge.net/index.php/News_Archive

- Data types include tokens, tags, chunks, trees, and feature structures.
- Software Modules are provided for: corpus readers (RTE, Movie Reviews, Question Classification, Brown Corpus), tokenizers, stemmers, taggers (regexp, n-gram, backoff, Brill, HMM), parsers (recursive descent, shift-reduce, chart, feature-based, probabilistic, etc), semantic interpretation, wordnet interface and similarity measures, clusterers (EM, k-means, etc), probability distributions, chatbots, demonstrations up to 48,000 lines of code.
- Corpora and Corpus Samples include: Reuters 21578 Corpus, ApteMod version, Movie Reviews corpus, Corpus for Recognising Textual Entailment (RTE), Brown Corpus, CMU Pronunciation Dictionary, CoNNL-2000 Chunking Corpus, Genesis, Gutenberg, NIST IEER Corpus, Presidential Addresses, Names, PP-Attachment Corpus, Senseval 2, TIMIT, Penn Treebank, the SIL Shoebox corpus format, Words (160 Mb of data).

NLTK is ideally suited to students who are learning NLP or conducting research in NLP or closely related areas. NLTK has been used successfully as a teaching tool, as an individual study tool, and as a platform for prototyping and building research systems [101, 105, and 178].

Chapter 3
Fuzzy Approaches for Information Retrieval

3. Fuzzy Approaches for Information Retrieval

WWW has become a huge repository of human knowledge and information in a scale never seen before. The basic reason for the success of the web is because it provides a convenient way to distribute information over the Internet. Individuals can publish information and users can access that information by themselves easily. Since the Internet covers the world, huge numbers of people had immediate access to this information.

Despite of it success, WWW has introduced new problems as the difficulty to find useful and relevant information on it. Google, Yahoo, AOL Search and MSN Search are some of the most important Web search engines today. They are able to retrieve millions of page references in less than a second. Therefore they have a high level of efficiency. Unfortunately, most of the information retrieved could be considered irrelevant. For that reason, efficacy level could be considered poor since a user could receive millions of documents for her/his query but just few of them are useful.

Efficacy and relevance level strongly depend on the fact that most crawlers just look for words or terms without considering their meaning. Terms are weighted by their frequency in the documents, thus more frequent terms are considered more important. Similarity between a query and a document is considered a function of the matching degree between the terms in the query and the terms in the document, according to the term frequency. Therefore search systems work based on word matching instead of concept matching. Therefore, search methods should change from only considering lexicographical aspects to considering conceptual ones too [165, 9].

In the following sections several new formulas developed for this thesis will be introduced. Some of these formulas allow measuring the presence of meanings or concepts by weighting the concurrence of synonymous terms in documents. Other formulas weight the presence of concepts by measuring the presence of words which allow describing those concepts, that is, by measuring the presence of the words included in the glossary definitions. Finally, an algorithm for query expansion which combines both one of the previously mentioned formulas is introduced.

3.1. Fuzzy Model for Synonymy and Polysemy.

FIS-CRM (Fuzzy Interrelations and Synonymy Conceptual Representation Model) [50, 51, 141, 169] is a methodology oriented towards processing the concepts contained in any kind of document, which can be considered an extension of the Vector Space Model (VSM), that uses the information stored in a fuzzy synonymy dictionary and fuzzy thematic ontologies. The dictionary provides the synonymy degree between pairs of synonyms and the ontologies bring the generality degree (hypernym, hyponym) between words. The generality degree value is calculated by the method proposed in [225]. The synonymy dictionary used in FIS-CRM was developed by S. Fernandez [48, 49]. It is an automatic implementation using Prolog of Blecua's Spanish dictionary of synonyms and antonyms [18] which include about 27 thousands words.

In this thesis, new formulas [200] based on those developed in FIS-CRM will be introduced. Therefore, it would be convenient to explain it with some extend. FIS-CRM approach is kept but a new version of the formulas is introduced in order to manage with synonymy and polysemy. With these new fuzzy formulas, the whole process of concept matching is simplified. As in FIS-CRM, although a certain term does not appear in a document, some degree of its presence could be estimated based on its degree of synonymy based on terms that do appear in the document. To measure the degree of concept presence in a document (or even in a document collection), a concept frequency formula is introduced. Finally, a method for expanding user queries is also presented, such that for each term in the original query, all of its synonyms by a certain meaning with maximum concept frequency are introduced.

Unlike FIS-CRM, in this thesis, a large lexical database of English, WordNet [121, 230] will be used as storage of synonymy relations for English language. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. WordNet distinguishes between nouns, verbs, adjectives and adverbs because they follow different grammatical rules. Every synset contains a group of synonymous words or collocations (a *collocation* is a sequence of words that go together to form a specific meaning, such as "car pool"); different senses of a word are in different synsets. WordNet also provides general definitions. The meaning of the synsets is further clarified with short defining *glosses,* which includes definitions and/or

example sentences. One of WordNet purposes is to support automatic text analysis and artificial intelligence applications.

As of 2006, the WordNet database contains about 150,000 words organized in over 115,000 synsets for a total of 207,000 word-sense pairs; in compressed form, it is about 12 megabytes in size. The database and software tools have been released under a BSD style license and can be downloaded and used freely. It also includes an ANSI Prolog version of the WordNet database.

The fundamental basis of FIS-CRM is to share the occurrences of a concept among the fuzzy synonyms that represent this concept and to give a weight to those words which represent a more general concept than the initial concept does. FIS-CRM constructs a vector space based on the number of occurrences of the terms contained in a set of documents. Afterwards, it readjusts the vector weights in order to represent concept occurrences, using for this purpose the information stored in the dictionary and ontologies. The readjusting process involves sharing the occurrences of a concept among the synonyms which converge to the concept and give a weight to the words that represent a more general concept than the contained ones. In this way, FIS-CRM readjusts the VSM vector weights in order to represent concept occurrences, using for this purpose the information stored in the dictionary and the ontologies.

Synonymy is usually conceived as a relation between expressions with identical or similar meaning. From the ancient times a controversy has existed about how to consider synonymy: if as an identity relation between language expressions or as a similarity relation. In FIS-CRM, synonymy is understood as a gradual, fuzzy relation between terms as in [48, 49].

Fuzzy sets were introduced in 1965 by L. A. Zadeh [237] A fuzzy set is a set without a crisp, clearly defined boundary. In classical set theory, an element either belongs or not to the set according to a crisp condition. In fuzzy set theory, elements could have only a partial degree of membership to the set. The membership function defines for each point in the input space its degree of membership, a number between 0 and 1. The input space is sometimes referred to as the universe of discourse.

Jaccard's coefficient is used in FIS-CRM to calculate the synonymy degree between two terms [169]. The method assumes that the set of synonyms of every sense of each

word is available stored in a synonymy dictionary [49]. Given two sets *X* and *Y*, their similarity is measured by:

$$sm(X,Y) = \frac{|X \cap Y|}{|X \cup Y|} \ . \qquad 18$$

On the other hand, let us consider two words w_1 and w_2 with m_i and m_j possible meanings respectively, where $1 \le i \le M_1$ and $1 \le j \le M_2$. Then $S(w, m_i)$ represents the set of synonyms provided by the dictionary for every entry w in the concrete meaning m_i. Then, the degree of synonymy *SD* between two words w_1 and w_2 by the meaning m_1 is defined in 19.

$$SD(w_1, m_1, w_2) = \max_{1 \le j \le M_2} \left(sm\left(S(w_1, m_1), S(w_2, m_j)\right)\right) . \qquad 19$$

S(w, m) represents the set of synonyms provided by the dictionary for a word w with meaning m and M_2 is the number of meanings m_j associated with word w_2.

A concept in FIS-CRM [169, 144] is not an absolute concept that has a meaning itself, i.e. there is not any kind of concept definition set or concept index. In FIS-CRM a concept is dynamically managed by means of the semantic areas of different words. Every word has a semantic area. The semantic area of a word pair is defined by the set of synonyms of that pair. The width of the semantic area of a word is intrinsic to the semantic shades of that word. Obviously, it can not be measured but if two overlapping semantic areas are compared, it could be assume that the one whose number of synonyms is larger should have a larger semantic area. The semantic area of a weak word (i.e. a word with several meanings) is the union of the semantic areas of each of its senses.

For example, if a term t_1 in a document is related to other more general term t_2 by means of a generality interrelation, the semantic area SA_1 of the first one will be included in the semantic area SA_2 of the second one. In this case, SA_1 is included in SA_2 with a membership degree equal to the generality degree between both terms, $GD(t_1, t_2)$.

$$GD(t_1, t_2) = \frac{O(t_1 \wedge t_2)}{O(t_1)} \quad . \qquad 20$$

Where $O(t_1)$ is the number of occurrences of t_1 and $O(t_1 \wedge t_2)$ is the number of co-occurrences of $t_1 \wedge t_2$.

In this case, it is considered that t_1 occurs once and the number of occurrences of the concept referred by t_2 is equal to $GD(t_1, t_2)$.

Considering a concept, obtained from the occurrences of various synonyms, as a fuzzy set, it is possible to define the membership degree of each one of the words that form the concept to the concept itself. Assuming that *m* words (synonyms each other) co-occur in a document, the membership degree of each term *ti* to the concept *C*, which they converge to, is:

$$\mu(t_i, C) = MIN_{j=1}^{m}\left(SD(t_i, t_j)\right) . \qquad 21$$

Once this value is defined, it is possible to define the number *N* of occurrences of a concept *C* (formed by the co-occurrence of *m* synonyms) in a query or document by the expression 22, in which w_i is the weight of the term t_i in the document, that in this case and in order to simplify, is the number of occurrences of the term t_i. The vector with the term weights is called the VSM vector.

$$N = \sum_{i=1}^{m} w_i * \mu(t_i, C) \quad . \qquad 22$$

After obtaining the weights w_i, FIS-CRM proceed to readjust the weights, by sharing the number of occurrences of each concept among the words of the synonyms set whose semantic area is more representative to that concept, obtaining FIS-CRM vectors based

on concept occurrences. Thus, a word may have a weight in the new vector even if it is not contained in it, as long as the referenced concept underlies the document.

The main handicap of the sharing process in FIS-CRM is managing with weak words (words with several meanings). Sense disambiguation of weak words is implicitly carried out by the sharing process. Three situations are distinguished depending on the implication of weak or strong words (words with only one meaning). So, there are three types of synonymy sharing:

- Readjustment occurrences among strong words: when one or more strong synonyms co-occur in a document or query.
- Readjustment occurrences among strong and weak words: when one or more strong synonyms co-occur with one or more weak synonyms.
- Readjustment occurrences among weak words: when one or more weak synonyms co-occur, without any strong synonym.

Readjustment occurrences among strong words. Let us consider a piece of the VSM vector (w_i), $1 \le i \le m$, containing several occurrences of m strong synonyms, where w_i reflects the number of occurrences of the term t_i (see Table 2). Let us assume that these synonyms converge to a concept C whose most suitable set of synonyms is formed by n strong terms.

Table 2 Readjustment among strong words

Terms	t_1	t_2	...	t_m	t_{m+1}	...	t_n
VSM vector	w_1	w_2	...	w_m	0	…	0
FIS-CRM vector	w'_1	w'_2	...	w'_m	w'_{m+1}	...	w'_n

Then, the FIS-CRM vector (w'_i), $1 \le i \le n$, would be obtained by equation 23 where w'_i is the readjusted weight of the term t_i.

$$w'_i = N * \mu(t_i, C) * \sqrt{\frac{1}{\sum_{i=1}^{n} \mu(t_i, C)^2}} \quad . \qquad 23$$

For example, let us suppose that the terms A and B are synonyms, which co-occur in a document with 2 and 3 occurrences respectively. And let us also suppose that the most suitable synonyms set they converge to contains the words C, D and E. Let us assume that the synonymy degrees among these terms are defined as shown in Table 3:

Table 3 Example of synonymy degrees

Terms	A	B	C	D	E
A		0.9	0.8	0.7	0.6
B			0.7	0.8	0.9
C				0.5	0.6
D					0.9
E					

Then, the VSM vector will be like the one below. In this case, the number of occurrences N of the concept formed by the co-occurrence of A and B is 4.5, obtained by the expression (5). The corresponding FIS-CRM vector is shown below in Table 4.

Table 4 Example of readjustment among strong words

Terms	*A*	*B*	*C*	*D*	*E*
VSM vector	2	3	0	0	0
FIS-CRM vector	2.35	2.35	1.83	1.83	1.56

Readjustment occurrences among strong and weak words. This type of adjustment is carried out when one or several weak synonyms co-occur in a document. Let us consider a piece of the VSM vector of a document containing m weak synonymous words, where w_i is the number of occurrences of the term t_i. In Table 5, the first m terms are the weak ones contained in the document. The next $f = n\text{-}m$ terms are the strong ones contained in the document. The last g = $p\text{-}n$ terms are the strong terms of the set of synonyms not contained in the document. In this case, the number N of concept occurrences, which the n synonyms converge to, is shared among the strong synonyms, from t_{m+1} to t_p.

Table 5 Readjustment among weak and strong words

Terms	t_1	t_2	...	t_m	t_{m+1}	...	t_n	t_{n+1}	...	t_p
VSM vector	w_1	w_2	...	w_m	w_{m+1}	…	w_n	0	…	0
FIS-CRM vector	0	0	...	0	w'_{m+1}	...	w'_n	w'_{n+1}	...	w'_p

It is important to point out that in order to calculate the number N, when managing the synonymy degree between two weak words, we must take into consideration the number that identifies the sense obtained by the disambiguation process. In the case of the synonymy degree between a strong word and a weak word it is implicitly disambiguated taking the value *SD*(*strong*, *weak*) as in 19.

The weights of the strong synonyms (from *tm+1* to *tp*) of the corresponding FIS-CRM vector are calculated by 23, assigning weight 0 (zero) to the first *m* terms (the weak ones). Then, the occurrences are shared only among the strong synonyms, leaving the weak terms without any weight.

Readjustment occurrences among weak words. This type of sharing is carried out when several weak synonyms co-occur and they do not have strong synonyms to share the occurrences of the concept they converge to. As in the previous cases, let us consider a piece of the VSM vector (w_i), $1 \leq i \leq m$, shown in Table 6, containing several occurrences of *m* weak synonyms, where w_i reflects the number of occurrences of the term t_i. And let us consider the set of *n* synonyms of the right disambiguated sense (all of them are weak terms). In this case, the number of occurrences of the concept to which the *m* weak terms converge is shared among all the synonyms of its right set of synonyms. In this case we should take the same considerations as the ones explained in the previous section about the identification of the number of the senses involved.

Table 6 Readjustment among strong words

Terms	t_1	t_2	...	t_m	t_{m+1}	...	t_n
VSM vector	w_1	w_2	...	w_m	0	…	0
FIS-CRM vector	w'_1	w'_2	...	w'_m	w'_{m+1}	...	w'_n

The approach of considering synonymy as an equivalence relation completely differs from that which considers it as a gradual relation. The latter one is closer to the behavior of synonymy in dictionaries, where it is possible to find synonyms which are

not equivalent. For example, *auto* and *automobile* share a common meaning: "a motor vehicle with four wheels; usually propelled by an internal combustion engine" [121]. But *automobile* has another meaning: as a verb, it means "to travel in an automobile". Therefore, *auto* and *automobile* are not equivalent terms, but similar. In what follows, synonymy will be considered an asymmetric relation.

Let V be a set of terms which belongs to a particular dictionary, and M the set of meanings associated to the terms in V. Therefore, each term in V has one or more meanings in M. According to WordNet, auto has only one meaning, while automobile has two (see Figure 10). On the other hand, each meaning in M has one or more terms associated in V. (see Figure 11).

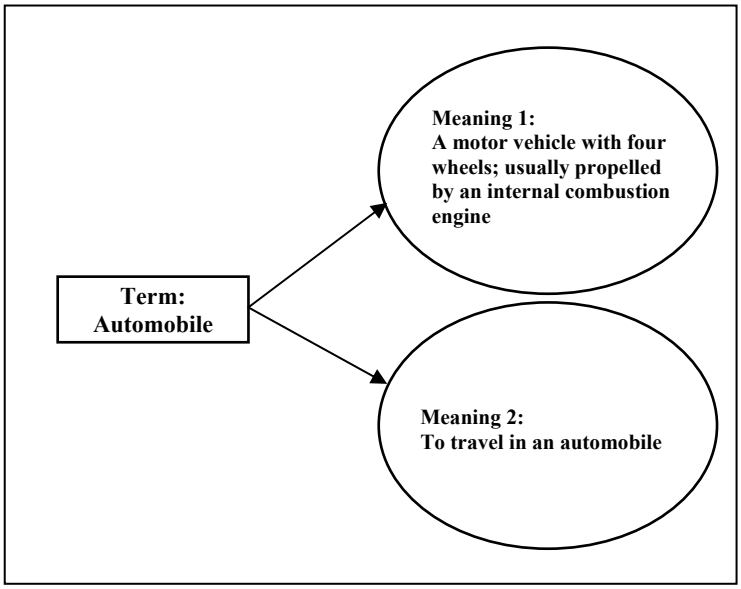


Figure 10: Each term has one or more meanings

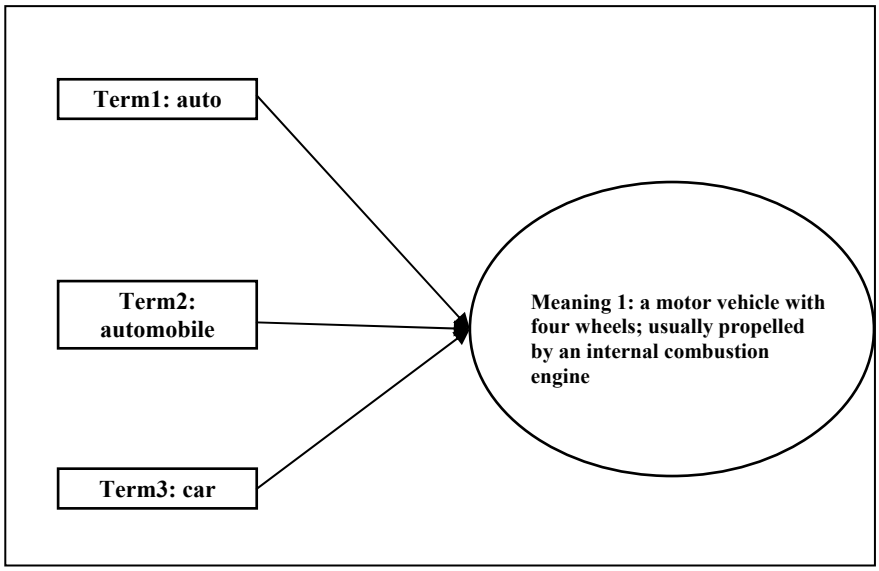


Figure 11 Several terms could share one meaning

Let *meaning* be a binary crisp relation such that *meaning* (*t, m*) = 1 if and only if there is a *t* in *V*, *m* in *M* such that *m* represents a meaning of term *t*.

$$meaning: V \times M \rightarrow \{0,1\} . \qquad 24$$

Let $M(t)$ be the set of different meanings associated with a certain term *t*.

$$M(t) = \{m \in M \,/\, meaning(t,m) = 1\} . \qquad 25$$

Polysemy is the capacity for a word or term to have multiple meanings. Terms with only one meaning are considered *strong*, while terms with several meanings are considered *weak*. [50, 169] consider that "the main handicap of the sharing process is managing weak words" and consider three situations depending on the implication of weak or strong words. In the above example, *auto* is a strong word, while *automobile* is weaker than *auto*, and *car* is even weaker because it has 5 meanings.

In order to manage all those situations in just one way, without having to manage case by case, an index $I_p(t)$ will be defined in 26 to represent the polysemy degree of term *t*. Therefore a strong term will have zero degree of polysemy, while weak terms will increase their degree of polysemy as increases their number of meanings. Let us

denote with $N_m(t)$ the number of meanings associated with the term t, that is, the number of elements of the set $M(t)$ defined above.

$$I_p : V \rightarrow [0, 1] \text{ where } I_p(t) = 1 - \frac{1}{N_m(t)} . \quad 26$$

Obviously, if t is a strong term (i.e. a term with only one meaning), then $N_m(t) = 1$ and $I_p(t) = 0$ which means that t is not polysemous, i.e. its polysemy degree is null (zero). On the other hand, as greater the number of meanings of t, the greater the polysemy degree and the closer the index $I_p(t)$ to 1. Therefore the polysemy index $I_p(t)$ is a measure of term weakness. At the same time, (1- $I_p(t)$) could be interpreted as a measure of the strength of the term t. Therefore $I_p(auto)$=0, $I_p(automobile)$= 0.5 and $I_p(car)$ = 0.8.

Polysemy index $I_p(t)$ could be interpreted as a measure of the term weakness (i.e. as the number of meanings of t increases, the term is weaker and the polysemy degree increases). Thus, we could define a complementary measure, the strength index I_S such that $I_S(t) = 1$ if t is a strong term, and is closer to zero (0) as the number of meanings of t increases (see equation 28)

$$I_S(t) = \frac{1}{N_m(t)} = 1 - I_p(t) . \quad 27$$

Let us define a fuzzy relation S, see equation 28, between two terms $t_1, t_2 \in V$ such that $S(t_1, t_2)$ expresses the degree of synonymy between the two terms:

$$S(t_1, t_2) = \frac{|M(t_1) \cap M(t_2)|}{|M(t_1)|} . \quad 28$$

Therefore,

- If $M(t_1) \cap M(t_2) = \varnothing$ there is not synonyms between them. Then $|M(t_1) \cap M(t_2)| = 0$ so that $S(t_1,t_2) = 0$, so the degree of synonymy between them is zero, i.e. there is not synonymy.
- If $M(t_1) \subseteq M(t_2)$ then t_2 includes all meanings of t_1 Therefore $M(t_1) \cap M(t_2) = M(t_1)$ so that $|M(t_1) \cap M(t_2)| = |M(t_1)|$ so that $S(t_1,t_2) = 1$, thus t_1 is a "full" synonym of t_2 (with the maximum degree).
- In other cases, when t_1 do not share some meanings with t_2, then $0 < |M(t_1) \cap M(t_2)| \leq |M(t_1)|$ so that $0 < S(t_1,t_2) \leq 1$, so the degree of synonymy varies.

That way, the degree of synonymy between auto and automobile will be 1, which means that the concept *auto* totally corresponds with the concept *automobile*. But, in the other way, the degree of synonymy between *automobile* and *auto* is just 0.5 because *automobile* just correspond with *auto* in half of the meanings.

Let us denote $T(m)$ the set of terms that share a meaning m:

$$T(m) = \{t \in V / meaning(t, m) = 1\} . \qquad 29$$

Then, for all m in M and t_1, t_2 in $T(m)$ so that $S(t_1, t_2) > 0$. Therefore, if the term t_2 appears in a particular document but the term t_1 does not, some degree of presence of term t_1 could be calculated for that particular document, considering the degree of synonymy between them.

Let us suppose for example that term "matching" appears 20 times in a document with 320 terms. According to WordNet [230], "matching" has two possible meanings:

- intentionally matched (m1)
- being two identical (m2)

The meaning *m1* is shared by two terms, T_1 = {matching, coordinated}, while *m2* is shared by four terms, T_2 = {matching, duplicate, twin, twinned}. Therefore, all of them share some degree of synonymy.

Consider D a collection of documents such that each document D_j is composed by terms from the vocabulary V:

$$D = \{D_1, D_2, D_3, ..., D_{nd}\} . \quad 30$$

Term frequency *tf*, see 31 is a well known measure [208, 209, 167, 168] of the importance of a term t_i in V within a document D_j where n_{ij} (resp. n_{kj}) is the number of occurrences of term t_i (resp. t_k) in the document D_j and n_{*j} is the number of terms in the same document.

$$tf_{ij} = \frac{n_{ij}}{\sum_k n_{kj}} = \frac{n_{ij}}{n_{*j}} . \quad 31$$

This measure is one of the most referenced in Information Retrieval, but it considers all terms in the same way, independently of their meaning. Therefore, it would be interesting to have a formula which allows measuring the importance within a document not only of a term but of a meaning.

According to the previous example about "matching", let us suppose two situations (see Table 7):

- "matching" appears 20 times in the document while the other synonyms do not appear in it.
- "matching" appears 20 times, "coordinated" appears 15 and the other synonyms do not appear.

Table 7 Calculating term frequency.

Term	Number of occurrences		Term frequency	
	a)	b)	a)	b)
matching	20	20	0.063	0.063
coordinated	0	15	0	0.047
duplicate	0	0	0	0
twin	0	0	0	0
twinned	0	0	0	0

Let us define a coefficient $R_j(m)$ which could be interpreted as a measure of the use of a meaning m in M in a document D_j based on the number of occurrences of the terms associated with that meaning.

$$R_j(m) = \sum_{t_i \in T(m)} \left(n_{ij} \cdot \left(1 - I_p(t_i)\right)\right) \qquad 32$$

$$= \sum_{t_i \in T(m)} \left(n_{ij} \cdot I_S(t_i)\right) = \sum_{t_i \in T(m)} \left(\frac{n_{ij}}{N_m(t_i)} \right) .$$

$R_j(m)$ relates the number of occurrences of the different terms (synonyms) associated with a certain meaning m, according with their respective polysemy degree. Thus, for strong terms (i.e. $N_m(t_i) = 1$ so $I_p(t_i) = 0$), any time the term occurs, it should be interpreted as a reference to its only meaning, so the total number of occurrences of t_i is added to $R_j(m)$. On the contrary, if the term is weak (i.e. $N_m(t_i) > 1$ therefore $0 < I_p(t_i) \leq 1$), then just a proportional part to the polysemy degree (weakness) of the term is added; then, as weaker the term, the lesser its contribution to $R_j(m)$.

On the other hand, it is easy to observe that if a term t_i has different meanings (i.e. $N_m(t_i) > 1$), then the number of occurrences of t_i will influence proportionally the corresponding values $R_j(m_i)$ for each one of the meanings m_i of t_i (i.e. m_i in $M(t_i)$).

Unfortunately, defined in that way, the value $R_j(m)$ could be difficult to analyze without knowing the corresponding value of R_j for the other meanings. Therefore, coefficient $Cf_j(m)$ is defined in equation 33 such that $0 \le Cf_j(m) \le 1$:

$$Cf_j(m) = \frac{R_j(m)}{\sum_k n_{kj}} = \frac{\sum_{t_i \in T(m)} \left(n_{ij} \cdot \left(1 - I_p(t_i)\right)\right)}{n_{*j}} = \frac{\sum_{t_i \in T(m)} \left(n_{ij} \cdot I_S(t_i)\right)}{n_{*j}} \quad . \qquad 33$$

That way, it is easy to compare the relative importance of the different meanings in a document D_j. As can be seen, the coefficient $Cf_j(m)$ formula resembles the term frequency one, consequently it will be called the *concept frequency* of meaning *m* in the document D_j.

Based on Table 7 the corresponding values of $R_j(m)$ and $Cf_j(m)$ for the meanings *m1* and *m2* are shown in Table 8. In situations a) and b), the terms "duplicate", "twin" and "twinned" are influenced by the concept frequency of *m2*. In situation b), the meaning *m1* is more influenced than *m2*, because synonyms, "matching" and "coordinated", do appear in the document.

Table 8 Estimating concept frequency

Meaning	**Situation a)**		**Situation b)**	
	Rj(m)	**Cfj(m)**	**Rj(m)**	**Cfj(m)**
m1	10	0.03	15	0.047
m2	10	0.03	10	0.03

This way, it is easy to calculate the concept frequency for a meaning *m* for two documents D_1 and D_2 and to compare them by some distance. A popular measure of similarity is the cosine of the angle between two vectors X_a and X_b. The cosine similarity is given by

$$sim(X_a, X_b) = \cos ine\left(\frac{X_a \cdot X_b}{\|X_a\| \cdot \|X_b\|} \right) \qquad 34$$

As the angle between the vectors shortens the cosine angle approaches 1, meaning that the two vectors are getting closer, meaning that the similarity of whatever is represented by the vectors increases.

Cosine measure is the most popular measure for text documents [54]. As the angle between the vectors shortens and the two vectors get closer the cosine angle approaches 1, meaning that the similarity of whatever is represented by the vectors increases.

By calculating concept frequencies $Cf_1(m)$ and $Cf_2(m)$ for all the meanings m in two documents D_1 and D_2, vectors Cf_1^M and Cf_2^M are obtained. Substituting them in equation 34, the similarity degree between both documents could be calculated this way.

Previously defined equations 31 to 33 could be easily extended for a whole collection of documents D. Therefore, term frequency of t_i in V for D would be defined in equation 35, where n_{i*} (resp. n_{k*}) is the number of occurrences of term t_i (resp. t_k) in the whole collection of documents D and n_{**} is the number of terms in the whole collection.

$$tf_{i*} = \frac{n_{i*}}{\sum_k n_{k*}} = \frac{n_{i*}}{n_{**}} \ . \qquad 35$$

Equation 32 will be redefined and a measure $RD(m)$ of the use of a meaning m in M in the whole collection D will be introduced:

$$RD(m) = \sum_{t_i \in T(m)} \left(n_{i*.} \left(1 - I_p(t_i) \right) \right) \qquad 36$$

$$= \sum_{t_i \in T(m)} \left(\frac{n_{i*}}{N_m(t_i)} \right) .$$

And then, the concept frequency coefficient $Cf_j(m)$ is also redefined for the whole collection D:

$$CfD(m) = \frac{RD(m)}{\sum_k n_{k*}} = \frac{\sum_{t_i \in T(m)} \left(n_{i*} \cdot \left(1 - I_p(t_i)\right)\right)}{n_{**}} = \frac{\sum_{t_i \in T(m)} \left(n_{i*} \cdot I_S(t_i)\right)}{n_{**}} \quad . \qquad 37$$

Once concept frequencies are calculated, the corresponding meanings could be considered ordered also.

$$m_i \geq m_j \Leftrightarrow CfD(m_i) \geq CfD(m_j) \quad . \qquad 38$$

3.2. Fuzzy Measure of Meaning Presence in Documents

Usually the meaning of a word is explained by a descriptive definition, a statement which captures the use, the function and the essence of a term or a concept[53]. A *definition* is a group of words which states the meaning of a term. This may either be the meaning which it bears in general use (*a descriptive definition*), or that which the speaker intends to impose upon it for the purpose of his or her discourse (*a stipulative definition*). The term to be defined is known as the *definiendum* (Latin: *that which is to be defined*). The group of words which defines it is known as the *definiens* (Latin: *that which is doing the defining*).

In this thesis, an approach to measure the presence in a document of a meaning or concept, based on the terms included in its definition, is used [213]. This approach assumes that terms included in the definition of a meaning constitute a set of keywords associated with the meaning essence. For example, the definition included in WordNet [230] says that an auto is "a motor vehicle with four wheels; usually propelled by an internal combustion engine". Then, the set of words {motor, vehicle, wheels; internal, combustion, engine} describe the essence of an auto. Therefore, we consider that by measuring the presence of those keywords in a document is a way to measure the presence of the meaning "auto".

Above some general definitions were already introduced:

- *V* a set of terms which belongs to a particular dictionary.

[53] Vaknin S., The Definition of Definitions http://samvak.tripod.com/define.html

- M the set of meanings associated to the terms in V.

Therefore, each term in V has one or more meanings in M and each meaning in M has one or more terms associated in V.

- $M(t)$ the set of different meanings associated with a certain term t.
- $N_m(t)$ the number of meanings associated with the term t.
- $T(m)$ the set of terms that share a meaning m.
- n_{ij} is the number of occurrences of term t_i in the document D_j.
- n_{*j} is the number of terms in the same document.

Based on this approach, a new definition will be introduced:

- $B(m)$ the set of terms that describe the essence of a meaning m

Then, the degree of presence of a meaning m in the document D_j could be measured by equation 39, except if $n_{i,j} = 0$ for all i such that t_i is in $B(m)$, that is, if none of the terms included in $B(m)$ appears in D_j . In this case, $Def_j(m)$ will be zero (0).

$$Def_j(m) = \frac{\sum_{t_i \in B(m)} n_{i,j}}{n_{*,j}}. \qquad 39$$

Defined in this way, Def_j is a function over the interval [0, 1], which could be interpreted as the membership function for the fuzzy set of meanings which are present in D_j. Then, it is easy to compare the relative importance of the different meanings in a document D_j.

This way, it is easy to calculate the degree of similarity between two documents D_1 and D_2. By calculating $Def_j(m)$ for all te meanings m, two vectors Def_1^M and Def_2^M are obtained. Given those vectors, it is possible to employ once more the cosine similarity to calculate the similarity between both documents.

Once $Def_j(m)$ is calculated for all m in M, then the corresponding meanings could be considered ordered by those values.

$$m_i \geq m_k \Leftrightarrow Def_j(m_i) \geq Def_j(m_k) \ . \qquad 40$$

Let's consider a document *D* and suppose that *auto* definiens appear in *D* in the proportion specified in Table 9

Table 9 Proportion of occurences of auto definiens

auto **definiens**	**ni,j**
motor	2
vehicle	1
four	2
wheels	3
internal	1
combustion	3
engine	3

Then *Def*(*auto*)=0.46

3.3. Using Noun Phrases to Measure the Presence of Concepts

As it was explained before the meaning of a word is usually explained by a descriptive definition, a statement which captures the use, the function and the essence of a concept[54]. In the previous section, the group of words which defines a concept was used just like that: as a group or bag of words, without considering any syntactic or semantic relation between them. In this section, we suggest to go one step further, considering the syntactic and semantic organization of the words, included in the definition of concepts or meanings. In this sense, we consider that the noun phrases included in the concept definition keep a closer relation with the essence of the definition than just the terms included in the same definition. In grammatical theory, a *noun phrase* (abbreviated *NP*) is a phrase whose head is a noun or a pronoun, optionally accompanied by a set of modifiers (determiners, adjectives, etc).

[54] Vaknin S., The Definition of Definitions http://samvak.tripod.com/define.html

For example, the definition of auto included in WordNet [230] says that an

> Auto: A motor vehicle with four wheels; usually propelled by an internal combustion engine.

Then, the bag of words and the set of noun phrases will be

BW(auto) = {motor, vehicle, wheels, internal, combustion, engine}

NP(auto) = {[motor vehicle], [four wheels], [internal combustion engine]}

It is easy to recognize that the word association forming noun phrases reveales better the essence of an auto. Let's consider another dictionary definition:

> Tiger: A large carnivorous feline mammal of Asia, having a tawny coat with transverse black stripes.

Then, the bag of words and the set of noun phrases will be

BW(tiger) = {carnivorous, feline, mammal, Asia, tawny, coat, transverse, black, stripes}

NP(tiger) = {[carnivorous feline mammal], Asia, [tawny coat], [transverse black stripes]}

In this section, we propose to measure not just the presence of the terms included in the definition of a concept, but also the presence in the document of the noun phrases included in the concept definition. This way, we consider that the perception of the meaning will be more accurate. It is important to emphasize that we are not considering to eliminate the bag of words but to complement it with the set of noun phrases and to measure both the terms and the noun phrases. Observe that although "transverse black stripes" is an important characteristic of tigers, black or transverse alone are not, so probably it could be not included into the bag of words.

Furthermore, we consider that, in the case of noun phrases with more than two words, like in "carnivorous feline mammal" and "transverse black stripes", other subdivisions should also be considered as, for example, carnivorous feline, carnivorous mammal, feline mammal, transverse stripes, black stripes. Those subdivisions should maintain the nouns as the subdivision head as in the former noun phrase.

Let's denote $Np(m)$ the set of noun phrases included in the definition of the meaning m, including the individual terms, the noun phrases properly and the noun phrase

subdivisions explained before. There will be word chains of length 1, 2, 3 or even longer. Observe that chains of lenght 1 correspond to independent terms.

Let's define

- nc_{ij} is the number of occurrences of chain C_i in the document D_j
- nc_{*j} is the number of chain in the same document

Then, the degree of presence of a meaning m in the document D_j could be measured by measuring the presence of NP chains using equation 41 such that $0 \leq NPf_j(m) \leq 1$.

$$NPf_j(m) = \frac{\sum_{C_i \in N_p(m)} nc_{i,j}}{nc_{*,j}}. \qquad 41$$

That way, it is easy to compare the relative importance of the different meanings in a document D_j.based on the presence of noun phrases associated with their definition.

By calculating concept frequencies $NPf_1(m)$ and $NPf_2(m)$ for all the meanings m in two documents D_1 and D_2, vectors NPf_1^M and NPf_2^M are obtained. Using the cosine similarity (equation 34), the similarity degree between both documents could be calculated this way.

Once concept frequencies are calculated, the corresponding meanings could be considered ordered also.

$$m_i \geq m_j \Leftrightarrow NPf(m_i) \geq NPf(m_j). \qquad 42$$

3.4. Query expansion

Under the bag of words model, if a relevant document does not contain the terms that are in the query, then that document will not be retrieved. The aim of query expansion is to reduce this query/document mismatch by expanding the query using words or phrases with a similar meaning or some other statistical relation to the set of relevant documents [1]. Query expansion is the process of reformulating a seed query to

improve retrieval performance in information retrieval operations. In the context of web search engines, query expansion involves evaluating a user's input and expanding the search query to match additional documents. Query expansion usually involves techniques such as:

- Finding synonyms of words, and searching for the synonyms as well
- Finding all the various morphological forms of words by stemming each word in the search query
- Fixing spelling errors and automatically searching for the corrected form or suggesting it in the results
- Reweighting the terms in the original query

Iterative searching is a natural approach to improve relevance level by using document collection frequency weights. Usually it is supposed that, some information is obtained about which documents are relevant and which others are not by an initial search. The information thus obtained can be used to modify the original query by adding new terms, selected from relevant documents to construct the new queries. This process is known as *query expansion* [168]. The power of relevance feedback comes not so much from re-weighting the original query terms, as from expanding the query by adding new search terms to it. Essentially, terms may be taken from the documents assessed as relevant.

Our approach is to use initially a collection of documents provided by the user as relevant, maybe from an initial search as told before or from the files the user keeps on hard disk or by links provided by some Web tool as Yahoo Search MyWeb Beta[55] or Google Bookmarks[56] for IE Toolbar Version 4. Based on those documents, it is possible to measure which meanings are more frequently used. Before, we have already defined two approaches in order to measure the presence of a meaning m in document D_j:

- By measuring the presence of synonyms of m by $Cf_j(m)$
- By measuring the presence of definiens of m by $NPf_j(m)$

Then it is possible to combine both measures in one expression by introducing some weighting factors that should be estimated experimentally for the document collection:

[55] http://myweb.yahoo.com/
[56] http://www.google.com/bookmarks/

$$P_j(m) = w_1 * Cf_j(m) + w_2 * NPf_j(m) \ . \qquad 43$$

Once $P_j(m)$ is calculated for all m in M, then the corresponding meanings could be considered ordered by those values.

$$m_i \geq m_k \Leftrightarrow P_j(m_i) \geq P_j(m_k) \ . \qquad 44$$

Therefore, the meaning m of a term t with the maximum presence $P(m)$ for collection D will be denoted:

$$\begin{aligned} &\max_m(t) = \max_{m_i \in M(t)}(m_i) \\ &m_i \geq m_k \Leftrightarrow P(m_i) \geq P(m_k) \end{aligned} \ . \qquad 45$$

Then, when the user makes a query Q, it is expanded to a new query Q_e defined:

$$Q_e = \bigcup_{t \in Q} T(\max_m(t)) \ . \qquad 46$$

Then, when the user make a query Q, it could be expanded to a new query Q_e such that for each term t in Q, all the terms associated with t by a maximum presence meaning will be included in Q_e, that is all the synonyms of t by a meaning m which has maximum presence membership.

3.4. Summary

Search engines are able today to retrieve millions of page references in less than a second, but, unfortunately, most of the information retrieved could be considered irrelevant. Irrelevance strongly depends on the fact that most crawlers just look for words or terms without considering their meaning. Terms are weighted by their frequency in the documents, thus more frequent terms are considered more important. Similarity between a query and a document is considered a function of the matching degree between the terms in the query and the terms in the document, according to the term frequency.

In this chapter several new formulas were introduced to measure the frequency of use of concepts in documents in spite of just to measure the frequency of use of terms. Some of these formulas are based on synonymy and polysemy concepts. They infer the frequency of use to a meaning or concept in a document by weighting the frequency of use of synonymous terms in a document.

Some other formulas introduced in this chapter are based on measuring the presence of the words included in glossary definitions in order to infer a reference to the meaning associated with those definitions. An improvement of this approach is also introduced, which proposes to measure the frequency of use of the noun phrases that appear in the glossary definition in spite of just measuring the frequency of use of the words that appear, independently one of each other, in the definition.

An algorithm for query expansion which combines both of the previously mentioned formulas was also introduced.

Chapter 4
From Natural Languages to Generalized Constraints and Protoforms

4. From Natural Languages to Generalized Constraints and Protoforms

> A concept which plays a pivotal role in fuzzy logic is that of generalized constraint. … serves as a basis for representation of and computation with propositions drawn from natural language. This is the province of NL-Computation … computation with information described in natural language.
>
> L. A. Zadeh [266]
>
> A concept which plays a key role in deduction is that of a protoform … an abstracted summary—a summary which serves to identify the deep semantic structure of the object to which it applies.
>
> L. A. Zadeh [258]

We are interested in the possibility to extract information from documents expressed in natural language (NL) and to deduce new information, information which was not present in the original document. Through the following chapter we are going to analyze the possibility to extract information from documents expressed in natural language (NL) and to deduce new information, information which was not present in the original document. The main problem with natural languages is that they are intrinsically imprecise. That is because natural languages are, basically, systems for describing perceptions, which are intrinsically imprecise as a consequence of the bounded ability of our brain, to resolve detail and store information. The key idea to precisiate the meaning of a proposition according to Zadeh [258] is to represent its meaning as a generalized constraint. Taking into account that not every proposition in NL is precisiable, Zadeh considers that a proposition is precisiable if it could be represented by generalized constraints.

A *constraint*, according to the dictionary, could be interpreted as something that limits or restricts or the proper act of limiting or condition of being limited. In mathematics, a constraint could be considered a condition that must satisfy a solution to an optimization problem. Usually, mathematical constraints are of two types: equality constraints and inequality constraints.

Constraints have been used as a methodological paradigm, Constraint Grammars [76, 77], for Natural Language Processing. Linguist-written, context dependent rules are

compiled into a grammar that assigns grammatical tags to words or other tokens in running text. Typical tags address lemmatisation, inflexion, derivation, syntactic function, dependency, valency, case roles, semantic type etc. The descriptive statements, constraints, do not have the ordinary task of defining the notion of a 'correct sentence'. They are less categorical in nature, more closely tied to morphological features, and more directly geared towards the basic task of parsing. Constraints are formulated on the basis of extensive corpus studies.

Other grammars as HPSG are also constraint-based, a lexicalist approach to grammatical theory that seeks to model human languages as systems of constraints[57]. An important concept in HPSG representations is that of a sign. A sign is a collection of information, including phonological, syntactic and semantic constraints, which is what is represented in attribute-value-matrixes (AVMs). AVMs encode feature structures where each attribute (feature) has a type and is paired with a value.

Most sentences of most natural languages have the property that if arcs are drawn connecting each pair of words that relate to each other, then the arcs will not cross. This well-known phenomenon is called planarity by Sleator and Temperley [192], and is the basis of Link Grammars, the formal language system that they proposed.

Link grammars [89] are a formalism based on context-free grammatical for the description of natural language. A link grammar consists of a set of words (the terminal symbols of the grammar), each of which has a linking requirement. A sequence of words is a sentence of the language defined by the grammar if there exists a way to draw arcs (which we shall hereafter call links) among the words

The structure assigned to a sentence by a link grammar is rather unlike any other grammatical system that we know of (although it is certainly related to dependency grammar). Rather than thinking in terms of syntactic functions (like subject or object) or constituents (like "verb phrase"), one must think in terms of relationships between pairs of words. It may be seen, however, that parts of speech, syntactic functions, and constituents may be recovered from a link structure rather easily.

A phrase parser is a component of the link grammar parser. It takes a linkage (as generated by the parser) and generates from it a constituent structure, showing conventional constituents such as noun phrases, verb phrases, and prepositional phrases,

[57] Stanford HPSG homepage http://hpsg.stanford.edu/ideas.html

a system which takes a "linkage" (the usual link grammar representation of a sentence, showing links connecting pairs of words) and derives a constituent or phrase-structure representation, showing conventional phrase categories such as noun phrase (NP), verb phrase (VP), prepositional phrase(PP), clause (S), and so on.

A prerequisite to mechanization of natural language understanding is to deal with the meaning imprecision of propositions drawn from a natural language. Zadeh[58] [263] holds that: "let *p* be a proposition in a NL. For *p* to be understood by a machine, it must be precisiated, expressed in a mathematically well-defined language". Zadeh[59] [263] also holds that "If *p* is a proposition or a concept, its precisiand, Pre*(p)*, is represented as a generalized constraint, GC. …..In this sense, the concept of a generalized constraint may be viewed as a bridge from natural languages to mathematics".

Abstraction of elements of Generalized Constraint Language gives rise to what is referred to as Protoform Language [249]. The set of protoforms of all precisiable propositions in NL, together with rules which govern propagation of generalized constraints, constitute what is called the Protoform Language [254].

The basic idea proposed by Zadeh [249] is: given a description of a perception (i.e. a proposition *p*) in NL, to translate it into a generalized constraint *GC*(*p*), a precisiation of its meaning. Then the generalized constraint *GC*(*p*) is transformed into a protoform *PF*(*p*), an abstraction of *GC*(*p*). After that, based on protoformal deduction rules, it is possible to deduce relevant information.

In order to apply Zadeh's ideas, it is necessary to identify propositions in NL and to transform them into GC. We consider that most noun phrases, copular sentences and comparative sentences act as constraints in a natural language context. Our approach is to do this by recognizing noun phrases in a NL document, considering that noun phrases act as constraints, specifying the noun involved. It is easy to realize that adjectives and adverbs constraint the meaning of nouns while describing them. Then, those noun phrases could be transformed into GC, which could be used later to deduce new pieces of information. We consider also that copular sentences and comparative sentences define relations that constraint the subject of those sentences.

[58] page 184
[59] page 185

Following different parts of the sentence are analyzed emphasizing those aspects that constraint the relations between the different parts and word types. The purpose is to identify those types of sentences, which could be "precisiated", according to Zadeh. This subset should form part of a Precisiated Natural Language (PNL), a subset of a natural language, composed of "precisiable" propositions [267]. In PNL a perception is equated to its description in a natural language. The point of departure in PNL is the assumption that the meaning of a proposition in a natural language may be represented as a generalized constraint [259]. Therefore, to identify the constraints that do exist in a natural language helps to describe them as generalized constraint.

Once we have formalized those constraints present in noun phrases, copular sentences and comparative sentences, we will analyze their deep semantic structure and summarize them via prototypical forms

4.1. About Constraints in Natural Language

Sentence: Ordinary conversation, personal letters, and even some types of professional writing (such as newspaper stories) consist almost entirely of simple sentences. Most people recognize a sentence as a unit which begins with a capital letter and ends with a full stop (period), a question mark, or an exclamation mark. Sentences have been defined notionally as units which express a "complete thought", though it is not at all clear what a "complete thought" is.

Syntactically, a sentence could be defined as a unit which consists of one or more clauses. Every complete sentence contains two parts: a subject and a predicate. The subject is what (or whom) the sentence is about, while the predicate tells something about the subject. Every subject is built around one noun or pronoun (or more) that is known as the simple subject. A predicate has at its centre a simple predicate, which is always the verb or verbs that link up with the subject.

Sentences may be classified according to their use in discourse in four types[60]:

1) Declarative sentences are by far the most common type. A declarative sentence simply states a fact or argument, without requiring either an answer or action from the reader. Declarative sentences are used to convey information or to make statements.

[60] Internet Grammar of English, University College London 1998, http://www.ucl.ac.uk/internet-grammar/function/sentpatt.htm

2) Interrogative sentences are used in asking questions. There are three basic types:
   a) yes/no interrogatives elicit a response which is either yes or no,
   b) alternative interrogatives offer two or more alternative responses,
   c) wh- interrogatives are introduced by a wh- word, and they elicit an open-ended response.
3) Imperative sentences are used in issuing orders or directives. In an imperative sentence, the main verb is in base form.
4) Exclamative sentences are used to make exclamations, to communicate strong emphasis or emotion.

A sentence could be defined syntactically, as a unit which consists of one or more clauses. There, a Simple Sentence contains only one clause, while a Complex Sentence is a sentence which contains at least one subordinate clause. Finally, a Compound Sentence consists of two clauses which are coordinated with each other. By using subordination and coordination, sentences can potentially be infinitely long, but in all cases they can be analysed as one or more clauses.

Sentences, clauses, phrases, and words constitute what is called the grammatical hierarchy, which can be represented schematically as follows:

- **sentences** consist of one or more...
- **clauses** consist of one or more...
- **phrases** consist of one or more...
- words

The most familiar grammatical function is the Subject, which could be interpreted as the element that performs the "action" denoted by the verb. Having identified the Subject, the remainder of the sentence tells us what the Subject does or did. This string is denoted as the Predicate of the sentence, so the Subject performs the action described in the Predicate.

However, there are some problems in defining verbs as "action" words, and in defining the Subject as the "performer" of the action. Therefore, the grammatical Subject has a number of characteristics[61] which have to be taken into account:

- Subject-Verb Inversion: In a declarative sentence, the Subject comes before the verb.
- Position of the Subject: In a declarative sentence, the Subject is usually the *first* constituent.
- Subject-verb Agreement: Subject-verb agreement or concord relates to number agreement (singular or plural) between the Subject and the verb which follows it.
- Subjective Pronouns: The pronouns *I, he/she/it, we, they*, always function as Subjects.

All of these restrictions should be taking into account in order to identify correctly the subject of a sentence. We just emphasize about declarative sentences because they are the most common type.

Phrase. Phrases are used to add information to a sentence and can perform the functions of a subject, an object, a subject or object complement. A phrase may function as a verb, noun, an adverb, or an adjective. A phrase is a group of two or more grammatically linked words without a subject and predicate.

There, we have a basic three-part structure:

<pre-Head string> <Head> <post-Head string>

The central element in a phrase is called the Head of the phrase. A phrase could consist minimally of a head element. In longer phrases, a string of elements may appear before or after the head, which are denoted as the pre-Head and post-Head string. Phrases are usually categorized in five types: noun phrase (NP), verb phrase (VP), adjective phrase (AdjP), adverb phrase (AdvP) and prepositional phrase (PreP).

Noun Phrase. Noun Phrases are by far the most common type as they are the most common realisations of the Subject, Direct Object, and Indirect Object. They can also

[61] Internet Grammar of English, University College London 1998, http://www.ucl.ac.uk/internet-grammar/function/sentpatt.htm

used as Adjuncts, in which case, they generally refer to time. A noun phrase consists of a pronoun or noun with any associated modifiers, including adjectives, adjective phrases, adjective clauses, and other nouns in the possessive case. Examples:

- [NP **I**] like coffee
- The waitress gave [NP **me**] the wrong dessert
- [NP **This**] is my car
- [NP **Those** *who arrive late*] cannot be admitted until the interval
- [NP **Two** of my guests] have arrived
- [NP The **first** to arrive] was John

A noun phrase has a noun as its Head (see Table 10). This noun could be a common or a proper noun or a pronoun. If the Head is a pronoun, the NP will generally consist of the Head only. This is because pronouns do not take determiners or adjectives, so there will be no pre-Head string. However, with some pronouns, there may be a post-Head string

Since some verbals can act as nouns, these also can form the nucleus of a noun phrase. However, since verbals are formed from verbs, they can also take direct objects and can be modified by adverbs. A gerund phrase or infinitive phrase, then, is a noun phrase consisting of a verbal, its modifiers (both adjectives and adverbs), and its objects. Similarly, numerals, as a subclass of nouns, can be the Head of an NP.

Table 10 Structure of a Noun Phrase

pre-Head	Head	post-Head
[NP *the small*	*children*	*at the window*]

Determiners and adjective phrases usually constitute the pre-Head string. The post-Head string in an NP can be indefinitely long. Complements also occur in NP. Typical Complements in NP are: PreP and clauses.

Determiners occur before nouns, and they indicate the kind of reference which the nouns have. Depending on their relative position before a noun, three classes (see Table 11) of determiners are distinguished[62]:

Table 11 Determiner classification

	Pre determiner	**Central Determiner**	**Post determiner**	**Noun**
I met	*all*	*my*	*many*	friends

Predeterminers specify quantity in the noun which follows them, and they are of three major types. They do not normally co-occur:

- "Multiplying" expressions, including expressions ending in times like: twice my salary, double my salary, ten times my salary.
- Fractions: half my salary, one-third my salary.
- The words all and both: all my salary, both my salaries.

The most common central determiners are the articles *the* and *a/an*: all *the* book, half *a* chapter. Possessives and Demonstratives, too, are central determiners: all *my* money, all *your* money, all *their* money, all *these* problems.

The postdeterminer slot is occupied by:

- Cardinal and ordinal numerals: the two children, his fourth birthday,
- Other quantifying expressions: my many friends, our several achievements,
- General ordinals: my next project, our last meeting, your previous remark

Unlike predeterminers, postdeterminers can co-occur: my *next two* projects, *several other* people.

The examples in Table 8 show how noun phrases can grow in legth, while their structure remains fairly clear.

[62] Internet Grammar of English, University College London 1998, http://www.ucl.ac.uk/internet-grammar/function/sentpatt.htm

Table 12 Noun phrase structure examples

Noun Phrases						
Noun phrase structure						Verb phrase
Pre determiner	Determiner	Post determiner	Pre modifier	Head	Post modifier	(not part of noun phrase)
				Buns		are for sale.
	The			buns		are for sale.
All	the		currant	buns		are for sale.
Not quite all	the		currant	buns		are for sale.
Not quite all	the		hot tasty currant	buns		are for sale.
Not quite all	the		hot tasty currant	buns	on the table	are for sale.
Not quite all	the	many	hot tasty currant	buns	on show on the table	are for sale.
Not quite all	the	very many	fine hot tasty currant	buns	which I cooked	are for sale.

Verb Phrase. A Verb Phrase consists of a verb, its direct and/or indirect objects, and any adverb, adverb phrases, or adverb clauses which happen to modify it. The predicate of a clause or sentence is always a verb phrase.

In a Verb Phrase (VP), the Head is always a verb. The pre-Head string, if any, will be a 'negative' word such as not or never, or an adverb phrase as it is shown in the following examples:

[VP not compose an aria]

[VP never compose an aria]

Paul [VP deliberately broke the window]

Many verb Heads must be followed by a post-Head string. The post-Head string completes the meaning of the Head. In functional terms, we refer to this string as the Complement of the Head. The string which completes the meaning of the Head is not always a Direct Object. It could be an Indirect Object. Typical Complements in VP are: NP, clause, PreP

- My son [VP made a cake]
- We [VP keep pigeons]

- I [VP recommend the fish]

Verbs which require a post-Head string are called transitive verbs. The post-Head string, in these examples, is called the Direct Object. In contrast, some verbs are never followed by a direct object.

These are known as intransitive verbs. However, most verbs in English can be both transitive and intransitive, so it is perhaps more accurate to refer to transitive and intransitive uses of a verb.

In addition to the transitive verb and the intransitive verb, there is a third kind of verb called a linking verb. The word (or phrase) which follows a linking verb is called not an object, but a subject complement. The most common linking verb is "to be". Other linking verbs are: become seem, appear, feel, grow, look, smell, taste, and sound, among others.

The verb "to be" is known as a Copular verb. It takes a special type of Complement which is identified as a Copular Complement.

In verb phrases, a wide range of Complements can appear, but in all cases there is a strong syntactic link between the Complement and the Head. The Complement is that part of the VP which is required to complete the meaning of the Head.

Adjective Phrase (AdjP). An Adjective Phrase is any phrase which modifies a noun or pronoun. Adjective phrases are often constructed using participles or prepositions together with their objects. The prepositional phrase acts as an adjective modifying the noun.

- Susan is [AP clever]
- The doctor is [AP very late]
- My sister is [AP fond of animals]

In an AdjP, the Head word is an adjective. The pre-Head string in an AdjP is most commonly an adverb phrase such as very or extremely. Adjective Heads may be followed by a post-Head string. Typical Complements in AdjP are: clause, PreP.

Adverb Phrase (AdvP). In an Adverb Phrase, the Head word is an adverb. Most commonly, the pre-Head string is another adverb phrase. In AdvPs, there is usually no post-Head string. Typical Complements in AdvP are: PreP.

- He graduated [AdvP very recently]
- She left [AdvP quite suddenly]

Prepositional Phrase (PP). A Prepositional Phrase can also be an adverb phrase, functioning as an adverb modifying the verb. Prepositional phrases usually consist of a preposition acting as the Head and a post-Head string only. Typical Complements in PP are: NP, PreP.

- [PP through the window]
- [PP over the bar]
- [PP across the line]
- [PP after midnight]

Complements and adjuncts can occur in all types of phrases (see Table 13 and Table 14)

Table 13 Examples of typical complements

Phrase Type	**Head**	**Typical Complements**	**Examples**
Noun Phrase (NP)	noun	PP clause	respect *for human rights* the realisation *that nothing has changed*
Verb Phrase (VP)	verb	NP clause PP	David plays *the piano* They realised *that nothing has changed* She looked *at the moon*
Adjective Phrase (AP)	adjective	Clause PP	easy *to read* fond *of biscuits*
Adverb Phrase (AdvP)	adverb	PP	luckily *for me*
Prepositional Phrase (PP)	preposition	NP PP	in *the room* from *behind the wall*

Table 14 Examples of typical adjunts

Phrase Type	Head	Typical Adjuncts	Examples
Noun Phrase (NP)	noun	PP AP clause	the books *on the shelf* the *old* lady cocoa, *which is made from cacao beans*
Verb Phrase (VP)	verb	Adv. PP	she *rapidly* lost interest he stood *on the patio*
Adjective Phrase (AP)	adjective	AdvP	it was *terribly* difficult
Prepositional Phrase (PP)	preposition	AdvP	*completely* out of control

Clauses. Every sentence consists of one or more clauses, therefore clauses are the building blocks of sentences. Clauses consist of groups of grammatically-linked words with a subject and predicate. They should contain at least a verb phrase. Clauses are the building blocks of sentences: every sentence consists of one or more clauses.

A clause can stand alone as a sentence, so it is an independent clause. Some clauses, however, cannot stand alone as sentences: in this case, they are dependent clauses or subordinate clauses.

Clauses are either finite or nonfinite, depending on the Verbs they contain. Verbs (and therefore the VPs and clauses that contain them) are either Finite or Nonfinite. Finite verb phrases carry tense (present tense or past tense), and the clauses containing them are Finite Clauses.

- She writes home every day --> (finite clause -- present tense verb)
- She wrote home yesterday --> (finite clause -- past tense verb)

Nonfinite verb phrases and clauses do not carry tense. Their main verb is either to-infinitive, bare infinitive, -ed form or -ing form. Examples:

- David loves [to play the piano]
- We made [David play the piano]
- [Written in 1864], it soon became a classic
- [Leaving home] can be very traumatic

Subordinate clauses may be finite or nonfinite. Many subordinate clauses are named after the form of the verb which they contain

- To-Infinitive Clause: You must book early [to secure a seat]
- Bare Infinitive Clause: They made [the professor forget his notes]
- -ing Participle Clause: His hobby is [collecting old photographs]
- -ed Participle Clause: [Rejected by his parents], the boy turned to a life of crime

Sometimes clauses are named after its first element:

- If-Clause: I'll be there at nine [if I catch the early train] if-clauses are sometimes called conditional clauses
- That-Clause: David thinks [that we should have a meeting]

Another important type of subordinate clause is the Relative Clause. Relative clauses are generally introduced by a relative pronoun, such as who or which --> The man [who lives beside us] is ill.

Martha Kolln and Robert Funk [87] identify ten basic sentence patterns in English syntax. It is helpful to think of a sentence as a series of slots. Each of the ten basic patterns (see Table 15) begins with a noun phrase in the subject slot, followed by one, two, or three slots in the predicate.

Table 15 Basic sentence patterns[63]

Subject	**Verb**	**Indirect Object**	**Direct Object**
[NP The team]	[VP is		[AdvP outside]]
[NP The team]	[VP is		[AdjP good]]
[NP (1) That team]	[VP is		[NP1 the Raiders]]
[NP The child]	[VP seems		[AdjP honest]]
[NP1 The children]	[VP became		[NP1 foster kids]]
[NP The club members]	[VP arrived		
[NP1 The woman]	[VP passed		[NP2 the test]]
[NP1 The players]	[VP gave	[NP2 the other team]	[NP3 the ball]]
[NP1 The members]	[VP find	[NP2 the club]	[AdjP interesting]]
[NP1 She]	[VP considers	[NP2 her teacher]	[NP2 a genius]]

The first three patterns are *be* patterns. The number of slots in the predicate is two. The first slot contains the main, or predicating verb, which is a form of *be*. Some examples of forms of *be* are *is, am, are, was, were, being,* and *been*. Expanded forms include *have been, was being, might be,* and *will be*. What follows the main verb in the subject complement determines which pattern the sentence is.

Patterns 4 and 5 contain two slots in the predicate, just as in Patterns 1-3. These patterns contain a linking verb followed by a subject complement. Linking verbs that commonly appear in Pattern IV are verbs of the senses such as *taste, smell, feel, sound,* and *look.* Others include *turn, appear, become, get, remain,* and *prove.* Some of these verbs also are used in Pattern 5.

[63] The numbers in parentheses in some patterns show relationships between noun phrases. If the numbers are identical, the noun phrases have the same referent.

No complement follows the verb in pattern 6, the Intransitive Verb Pattern; however, the verb may be followed by adverbial information answering questions such as: When? Where? Why? How? How long?

The four last patterns, the Transitive Verb Patterns, have one thing in common: each contains a direct object--a noun phrase that often refers to the object of a verb's action.

The term "Complement" is not simply another word for the "post-Head string" -- post-Head strings are not always Complements. This is because the post-Head string is not always required to complete the meaning of the Head. Adjuncts are optional elements, since their omission still leaves a complete sentence. They may convey information about how, when, or where something happened. Many types of constituents can function as Adjuncts.

Adjuncts are syntactically peripheral to the rest of the sentence. They may occur at the beginning and at the end of a sentence, and they may occur in all three of the patterns above: NP, AdvP, PreP, clauses. The following patterns (see Table 16) are essentially a conflation of the previous one, with Adjuncts added. Adjuncts are bracketed to show that they are optional. Strictly speaking, Objects are also optional, since they are only required by monotransitive and ditransitive verbs.

Table 16 Sentence patterns including adjuncts

(Adjunct)	**Subject**	**Verb**	**Indirect Object**	**Direct Object**	**(Adjunct)**
Usually	*David*	*sings*			*in the bath*
Unfortunately	*the professor*	*wants*		*to retire*	*this year*
At the start of the trial	*the judge*	*showed*	*the jury*	*the photographs*	*in a private chamber*

4.2. From Constraints in Natural Language to Generalized Constraints

The point of departure in the Theory of Fuzzy Information Granulation (TFIG) is the concept of a Generalized Constraint (GC). A granule is characterized by a GC which defines it.

The concept of Generalized Constraint provides a basis for a classification of fuzzy granules. More specifically, in the theory of fuzzy IG a granule, *G*, is viewed as a clump of points characterized by a generalized constraint. Thus, $G = \{X \mid X \text{ isr } R\}$.

In this context, the type of a granule is determined by the type of constraint which defines it. In particular, possibilistic, veristic and probabilistic granules are defined, respectively, by possibilistic, veristic and probabilistic constraints. For example:

- granule $G = \{X \mid X \text{ is small}\}$ is a possibilistic granule.
- granule $G = \{X \mid X \text{ isv small}\}$ is a veristic granule
- granule $G = [X \mid X \text{ isp } N(m, \sigma^2)\}$ is a probabilistic (Gaussian) granule.

The Theory of Precisiation of Meaning (TPM) may be viewed as an attempt to construct a conceptual framework for dealing with issues like the concept of precision, precisiation of natural languages and definition of concepts [256]. In TPM, precise is interpreted as *m-precise*, and precisiation of a proposition, *p*, is interpreted as precisiation of the meaning of *p*. In this perspective, in TPM, precisiation of *p* is interpreted as translation of *p* into the Generalized Constraint Language (GCL), that is, expressing the meaning of *p* as a generalized constraint, GC(*p*), which is referred to as the precisiand of *p* [255].

For example, if *p* is *Monika is young*, then, in annotated form, its precisiand would be expressed as *X*/*Age*(*Monika*) is *R*/*young*, when *young* is defined as a fuzzy set. More generally, a precisiand is an annotated instance of a generalized constraint.

What is important to note is that precisiation of *p* presupposes that the meaning of *p* is understood. For example, if I am told that *p*: *It is very warm*, I understand what *p* means but what I may want is a precisiation of *p*. The same applies to propositions such as “Use with adequate ventilation,” “Unemployment is high,” “Most Swedes are tall,” as well as concepts such as mountain, valley, edge, obesity, relevance and causality.

Basically, a natural language (NL) is a system for describing perceptions. Precisiated Natural Language (PNL) is a sub language of precisiable propositions in NL, which primary function is serving as a part of NL which admits precisiation. A proposition, *p*, in NL is precisiable if it is translatable into a precisiated language [267].

The concept of a Generalized Constraint plays a key role in the Theory of Precisiation of Meaning by providing a basis for precisiation of meaning [256]. More specifically, if *p* is a proposition or a concept, its precisiand, *Pre*(*p*), is represented as a Generalized Constraint, *GC*. Thus, *Pre*(*p*)=*GC*. In this sense, the concept of a generalized constraint may be viewed on a bridge from natural languages to mathematics. To clarify the issue, for *p* to be understood by a machine, it must be precisiated that is, expressed in a mathematically well-defined language. A precisiated form of *p*, *Pre*(*p*), will be referred to as a precisiand of *p* and will be denoted as *p**. The precisiand p* is a Generalized Constraint form (GC-form), an element of Generalized Constraint Language (GCL). In the case of PNL, the precisiation language is the Generalized Constraint Language (GCL).

Table 17 Using Generalized Constraints for meaning precisiation

proposition in NL	**precisiation**
p	*p** (GC-form)
most Swedes are tall	S Count (tall.Swedes/Swedes) is most

The principal modes of generalization in TFIG are

- fuzzification (f-generalization);
- granulation (g-generalization); and
- fuzzy granulation (f.g-generalization), which is a combination of fuzzification and granulation.

F.g-generalization underlies the basic concepts of linguistic variable, fuzzy if-then rule and fuzzy graph. These concepts have long played a major role in the applications of fuzzy logic and differentiate fuzzy logic from other methodologies for dealing with imprecision and uncertainty. What is important to recognize is that no methodology other than fuzzy logic provides machinery for fuzzy information granulation.

A typical constraint is an expression of the form $X \in C$, where X is the constrained variable and C is the set of values which X is allowed to take. A typical constraint is hard (inelastic) in the sense that if u is a value of X then u satisfies the constraint if and only if $u \in C$.

Let X be a variable which takes values in a universe of discourse U. A generalized constraint, GC, is defined as an expression of the form GC: X *isr* R where X is the constrained variable; R is a constraining relation which, in general, is nonbivalent; and r is an indexing variable which identifies the modality by which R constrains X, that is, its semantics. R will be referred to as a granular value of X. In the above example, a proposition p: *Monika is young* in NL is transformed into a generalized constraint $GC(p)$: *Age*(*Monika*) *isr young* in GCL, where *young* is a fuzzy relation characterized by a membership function m, with $m(u)$ representing the degree to which a numerical value of age, u, fits the description of age as *young* [249].

The principal types of constraints and the values of the modality r which define them are the following [263]:

- *Equality constraint*: r = e. Example: X ise a means that X = a.
- *Possibilistic constraint*: r = blank. In this case, if R is a fuzzy set with membership function μR: U → [0, 1], and X is a disjunctive (possibilistic) variable, that is, a variable which cannot be assigned two or more values in U simultaneously, then

X is R means that R is the possibility distribution of X. More specifically,

X is R → Poss{X = u} = μ_R (u), u∈U.

Example: X is small. Means that Poss{X = u} = μ_{small}(u).

The simplest value, r = blank, was chosen to define possibilistic constraints, because constraints induced by propositions expressed in a natural language are very often possibilistic in nature.

- *Veristic constraint*: r = v. In this case, if R is a fuzzy set with membership function μR and X is a conjunctive (veristic) variable, that is, a variable which can be assigned two or more values in U simultaneously, then

X isv R→. Ver{X = u} = μR(U), μ∈U,

where Ver{X = u} is the verity (truth value) of X = u.

An example of a veristic constraint is the following. Let U be the universe of natural languages and let X denote the fluency of an individual in English, French and German. Then, X isv (1.0 English + 0.8 French + 0.6 Italian) means that the degrees of fluency of X in English, French and Italian are 1.0, 0.8 and 0.6, respectively.

NOTE that, in the case of a possibilistic constraint, the fuzzy set R plays the role of a possibility distribution. In the possibilistic interpretation, the grades of membership are possibilities. Since in most cases constraints are possibilistic, the default assumption is that a fuzzy set plays the role of a possibility distribution. When dealing with partial (not complete) knowledge, the constraints are possibilistic.

Example: Mary is young → Age(Mary) is young

In the case of a veristic constraint, the fuzzy set R plays the role of a verity distribution. Therefore, any fuzzy and crisp set R admits of two different interpretations. In the veristic interpretation the grades of membership are verities (truth values). When dealing with partial truth, the constraints are veristic.

Example: Robert is fluent in English, French and Italian → Fluency(Robert) isv (1/English + 0.8/French + 0.6/Italian).

- *Probabilistic constraint*: r = p. In this case, X isp R means that X is a random variable and R is the probability distribution (or density) of X. For example, X isp N(m, σ2) means that X is a random variable with Normal distribution and mean m and variance σ2. Similarly, X isp (0.2\a + 0.4\b + 0.4\c) means that X takes the values a, b, c with respective probabilities 0.2, 0.4 and 0.4.

Probability value constraint, r = λ. In this case, X isλ R signifies that what is constrained is the probability of a specified event, X is A. More specifically,

X isλ R →Prob{X is A} is R.

Example: if A = small and R = likely, then X isλ likely means that

Prob{X is small} is likely.

- *Random set constraint*, r = rs. In this case, X isrs R is a composite constraint which is a combination of probabilistic and possibilistic (or veristic) constraints. In a schematic form, a random set constraint may be represented as

  Y isp P

  (X,Y) is Q

  _____

  X isrs R

  Or

  Y isp P

  (X,Y) isv Q

  _____

  X isrs R '

  where ***Q*** is a joint possibilistic (or veristic) constraint on ***X*** and ***Y***, and ***R*** is a random set, that is, a set-valued random variable.

- *Fuzzy graph constraint, r = fg.* In this case, in *X isfg R, X* is a function and *R* is a fuzzy graph approximation to *X*. More specifically, if *X* is a function, *X: U → V,* defined by a fuzzy rule set

  If u is A_1, then v is B_1

  if u is A_2, then v is B_2

  ……

  if u is A_n, then v is B_n

  where A_1 and B_1 are linguistic values of *u* and *v*, then *R* is the fuzzy graph $R=A_1xB_1 + ... +A_nXB_n$ where $A_i \times B_i$, i = 1 . . . . , n, is the cartesian product of A_i and B_i and + represents disjunction or, more generally, an s-norm.

In addition to the types of constraints defined above there are many others that are more specialized and less common. A question that arises is: What purpose is served by having a large variety of constraints to choose from.

A basic reason is that, in a general setting, information may be viewed as a constraint on a variable. For example, the proposition "Mary is young", conveys information about Mary's age by constraining the values that the variable Age (Mary) can take. Similarly, the proposition "Most Swedes are tall" may be interpreted as a possibilistic constraint on the proportion of tall Swedes, that is,

most Swedes are tall → Proportion (tall Swedes/Swedes) is most

in which the fuzzy quantifier *most* plays the role of a fuzzy number

More generally, in the context of computing with words, a basic assumption is that a proposition, *p*, expressed in a natural language may be interpreted as a generalized constraint *p* → *X isr R*. In this interpretation, *X isr R* is the *canonical form* of *p*. The function of the canonical form is to place in evidence (i.e., to make explicit) the implicit constraint which *p* represents.

In Computing with Words, the depth of explicitation of a proposition is a measure of the effort involved in explicitating *p*, that is, translating, *p* into its canonical form. In this sense, the proposition *X isr R* is a surface constraint (depth = zero). As shown in Figure 12, the depth of explication increases in the downward direction. Thus, a proposition such as "Mary is young" is shallow, whereas "it is not very likely that there will be a significant increase in the price of oil in the near future" is not.

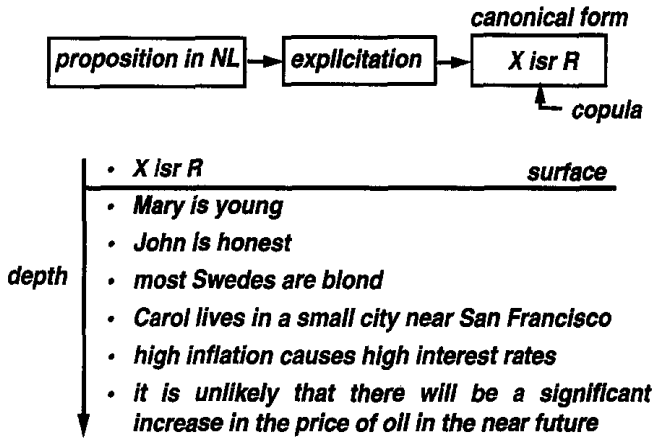


Figure 12: Depth of explication of propositions in a natural language

What we see, then, is that the information conveyed by a proposition expressed in a natural language is, in general, too complex to admit of representation as a simple, crisp constraint. This is the main reason why in representing the meaning of a proposition expressed in a natural language we need a wide variety of constraints which are subsumed under the rubric of generalized constraints.

The rationale for constructing a large variety of constraints is that conventional crisp constraints are incapable of representing the meaning of propositions expressed in a natural language -- most of which are intrinsically imprecise -- in a form that lends itself

to computation. The elements of GCL are composite GCs which are formed from generic GCs by combination, modification and qualification.

By construction, the Generalized Constraint Language is maximally expressive, which means that PNL is the largest subset of a NL which admits precisiation. Informally, this implication serves as a basis for the conclusion that if a concept cannot be defined in terms of PNL, then it is indefinable or, synonymously, amorphic [242] PNL is equipped with [249]:

- a dictionary from NL to GCL;
- a dictionary from GCL to Prototypical form Language (PFL) and a collection of deduction rules (rules of generalized constrained propagation) expressed in PFL.The principal components of PNL are:perception description language
- knowledge representation language
- definition language
- specification language
- deduction language

4.2.1. Introducing Adjectives and Adverbs as Constraints.

"The subject of *ontology* is the study of the *categories* of things that exist or may exist in some domain. The product of such a study, called *an ontology*, is a catalogue of the types of things that are assumed to exist in a domain of interest ***D*** from the perspective of a person who uses a language ***L*** for the purpose of talking about ***D***. The types in the ontology represent the *predicates*, *word senses*, or *concept and relation types* of the language ***L*** when used to discuss topics in the domain ***D***."

J. Sowa, "Knowledge Representation" [206]

We consider that adjectives and adverbs constraint the meaning of the nouns they are associated with. Through this epigraph we are going to analyze the role of adjectives and adverbs in natural language sentences and how they contribute to describe and explicitate the concepts expressed by noun phrases, and sentences. Afterthat, noun phrases will be represented by generalized constraints.

Each one of the things in the world can be thought of as having a value on each of different characteristics such as size, colour, shape, taste, consistency, etc. Each attribute has a value on some dimension. To distinguish one thing from another, it is necessary to deal with these dimensions.

Lexicons define nouns to designate categories of things, which share a whole set of attributes that distinguish and characterize them. Common nouns represent a strong tendency for particular sets of values on different dimensions to co-occur. Proper nouns are nouns representing unique entities (such as London, Universe or John), as distinguished from common nouns which describe a class of entities (such as city, planet or person).

The noun *apple* itself cannot be said to refer to a particular apple; rather than it designates a whole category, the category *apple*, which includes many possible individual apples. An *apple* isn't just an object of a particular shape; it has a characteristic range of sizes, tastes, consistencies, and locations. In other words, the category *apple* is a whole cluster of co-occurring features.

In a noun phrase like *red apple*, the word *red* is an attribute that characterizes some members of the *apple* category, one possible value on a conceptual dimension, *colour*. The phrase attributes redness[64] to the apple that is being referred to. The word designating the attribute *red* in the example is an adjective.

In a noun phrase like *extremely big apple*, the words *extremely big* describe an attribute that characterizes some members of the *apple* category, one possible value on a conceptual dimension, *size*. The phrase attributes a big size to the apple that is being referred to. The word designating the attribute *big* in the example is an *adjective*, while the word *extremely* is an adverb modifying the adjective.

Gasser [52] proposes that an English attributive phrase consisting of an adjective *Adj* (i.e. *young*) designating an attribute *Att* (i.e. *age*) followed by a noun *N* (i.e. *Monika*) designating a thing category C_1 (i.e. *person*) designates the subcategory of C_2 (*young person*) whose members have attribute *Att*. The same analysis could be applied to a phrase like *very young Monika*, where the words *very young* designate a subcategory C_3 (*very young person*) of the previous one.

[64] Webster 1913 Dictionary. Patrick J. Cassidy, 1913. *Answers.com* 30 Apr. 2007. http://www.answers.com/topic/redness-2

J. Sowa [206] considers that the properties of permanent entities (i.e. continuants), called *attributes*, are usually described by adjectives. Those attributes include colours, shapes, sizes, and weights. Continuants are commonly expressed by nouns. Attributes are categories, defined by purely semantic distinctions, which have a strong correlation with the syntactic categories of natural languages. This relation between an entity and its attribute can be expressed by the following notation: Entity has_attrib Property. Example: The rose is red → the_rose *has_attrib* red. In order to identify, in a more precisely way, the entity designated by a noun phrase, we adopted the convention to link together, by hyphens, all the words that compose the noun phrase.

But many of the adjectives could be applied to large subsets of nouns: *red apple*, *red raspberry*, *red stone*, *red skin*; *wet apple*, *wet stone*, *wet tree*, *and wet skin*. In those cases, red and wet do not have the same connotation, because both of them, red and wet, do not have a precise meaning. Therefore, they could be interpreted as fuzzy values, associating with them a degree of membership to certain set. The previous example, The rose is red will be denoted by *the_rose has_attrib red degree m*, with $m > 0$, which means that the entity the_rose has a property red with a membership degree $m > 0$. Then the notation will be finally defined as *Entity has_attrib Property degree Degree*.

In English, characteristics are usually expressed by nouns, such as *shape, colour, length,* and *weight* and could be considered types of properties. This relation between an entity and some characteristic can be expressed by: *Entity has_chrc Characteristic*. Example: The colour of the rose → *the_rose has_chrc colour*. Using a common knowledge base as Cyc[65] or Concept Net [103, 104], it is possible to associate *red* with *colour*, as a characteristic of *rose*, by the relation *red isA colour*. Usually, characteristics are instantiated by entity properties, like attributes. Therefore, this relation will be expressed here by the following notation: *Entity has_chrc Characteristic value Property*. Example: The rose is red → *the_rose has_chrc colour value red*.

According to Zadeh [243] the key idea underlying Constraint-Centred Semantics of Natural Languages (CSNL) is that the meaning of a proposition in Natural Language, *p*, can be represented as a generalized constraint on a variable. A generalized constraint is represented as *X* isr *R*, where isr is a variable copula that defines the way in which *R* constrains *X* by a modality *r*. When the constraint is possibilistic, it leads to the expression *X* is *R*, in which *R* is a fuzzy relation that constrains *X* by playing the role of

[65] OpenCyc http://www.cyc.com/cyc/opencyc/overview

the possibility distribution of *X*. Constraints induced by propositions expressed in a natural language are, for the most part, possibilistic in nature [241].

If R is a fuzzy set with membership function

m_R: U → [0, 1], and X is R → Poss{X = u} = m_R(u), u∈U.

As a simple example of a possibilistic constraint consider the proposition

p: Mary is young.

the meaning of p would be represented as

p → Age(Mary) is young.

In this case,

Poss{ Age(Mary) = u} = m_{young}(u).

The membership function of young defines the possibility distribution of Age(Mary). More specifically, if the grade of membership of, say, 25 in young is 0.8, then the possibility that Mary is 25 given that Mary is young is 0.8. That is Poss{Age(Mary) = 25} = 0.8

Age can be described as very young, young, middle aged, old, and very old, where very young, young, and so on, could be considered age granules, i.e. clumps of attribute-values which are drawn together by indistinguishability, equivalence, similarity, proximity or functionality. The granules are associated with fuzzy attributes, for example, length, color, age, etc. In turn, fuzzy attributes can have fuzzy values; for example, in the case of the fuzzy attribute *length (hair)*, the fuzzy values could be long, short, very long, and so on. The fuzziness of granules, their attributes, and their values is characteristic of the ways in which human concepts are formed, organized, and manipulated [243].

Therefore, in order to represent correctly this relation the following notation is introduced here: *Entity has_chrc Characteristic value Property modality M*. Example:

'Mary' has_chrc age value young degree m modality possibilistic with degree m > 0. We used the apostrophe ('') this time to indicate a string, differentiating it from a variable, but since we are not programming, we are not going to use them usually.
As long as the possibilistic constraint will be assumed by default, the corresponding modality argument could be omitted, so the relation *'Mary' has_chrc age value young*

will be assumed with modality = *possibilistic*. If Mary were exactly 25 years old, then an equality constraint would be used: *'Mary' has_chrc age value 25 degree m modality equality*.

Adjectives constitute a lexical category which includes those words that describe or modify an attribute of a noun or pronoun [68]. Adjectives are usually placed just before the words they qualify: cold weather, large windows, violent storms, shy child, blue notebook, rotten apple, four wheels, and another table. The most widely recognized adjectives are those words, such as big, old, and tired which describe people, places, or things.

In English, most adjectives can occur both before and after a noun. Adjectives before the noun are called *attributive* adjectives. An *attributive* adjective is part of the noun phrase headed by the noun it modifies. For example, in the noun phrase *the young girl*, the attributive adjective *young* modifies the noun *girl*, which heads the phrase.

Adjectives after the noun are called *predicative* adjectives. Predicative adjectives do not occur *immediately* after the noun. Instead, they follow a verb. A *predicative* adjective is the complement of a verb that links it to the noun. For example, in *the girl is young*, the predicative adjective *young* is linked by the verb *is* to the noun *girl*, which it modifies.

Most attributive adjectives denote some attribute of the noun which they modify. Most adjective-noun sequences can be loosely reformulated as predicative adjectives (see Table 18 below). Therefore we can consider that both sequences could be interpreted in the same way proposed by Gasser for an English attributive phrase.

In each case the adjective denotes an attribute or quality of the noun, just the way it is shown in the reformulations. Adjectives of this type are known as *inherent adjectives*[66]. The attribute they denote is inherent in the noun which they modify.

Therefore, in most cases, it is possible to transform a phrase like "cold weather" into a sentence like "the weather is cold", meaning that "the temperature of the weather is cold" which can be expressed by the generalized constraint Temperature(weather) is cold or the equivalent representation: has_chrc(weather, temperature, cold, m) with m>0.

[66] The Internet Grammar of English, UCL http://www.ucl.ac.uk/internet-grammar/adjectiv/inherent.htm

Table 18 Adjectives used attributively and predicatively

Attributive adjective	**Predicative adjective**	**Generalized Constraint**
The young girl	The girl is young	Age(girl) is young
The red car	The car is red	Color(car) is red
The big elephant	The elephant is big	Size(elephant) is big
This high mountain	This mountain is high	Height(mountain) is high
This wide river	This river is wide	Width(river) is wide

When the sentence *Monika is young* is analyzed, the usual interpretation is to assume that *Monika* is a person, who is *young*. In this case, *young* refers to *Monika*'s *age*. Let's suppose now that the sentence is *Candy is young*. Is *Candy* a *person* or a *cat* or a *dog*? Often the interpretation of the adjective depends on the context: on the modified noun, on other words that occur before or after the phrase, or on the situation in which the phrase is uttered.

Most adjectives have relative, rather than absolute, meanings. Consider a continuous dimension, like size, darkness, age, or crispness. There are many possible values on such dimensions. Consider the designatum 'old'. The statement *Emmy is old* places *Emmy* on the 'old' side of a boundary determining 'old' and 'not old' subclasses within the an 'age' class. Old, by virtue of the 'not old' subclass, has simultaneously a classifying and a comparing function. The boundary between 'old' and 'not old' often corresponds to what is taken by encoder and decoder to be 'average' or 'expected' or 'normal', that is possibilistic.

In English, the age dimension has adjectives for the two poles of the dimension, young and old. Neither of these adjectives has an absolute meaning; their precise meanings depend on how they are used. Compare the meanings of *a young man* and *a young cat*. In both cases, *young* means that the subject associated to the adjective has an attribute value 'close to the young end' of the *age* dimension. But the standard of

comparison depends on the concrete subject, i.e. *a man* or *a cat*, the word *young* refers to. The same happens with adjectives like tall and short, fat and thin, crisp and mushy.

Using one adjective or the other means to be closer to that end of the dimension, based on some standard of comparison. Observe that, even though young could be interpreted the same as 'the young end' of the *age* dimension, *very young*, *extremely young* and *too young* are even closer to it. In those cases, the adverbs *very*, *extremely* and *too* just modify the adjective young, defining new granules in the age dimension.

Therefore, Mary is young, means that Mary is closer to the end of "*youngness*"; she has this characteristic in a high degree. But if Tom is very young, then Tom will have this characteristic in a higher degree than Mary. As the antonym of *young* is *old*, therefore *Monika* has a low degree of oldness. Let's suppose also that Jane is extremely young and John is too young, then

'Mary' has_chrc age value young degree m1

'Tom' has_chrc age value very_young degree m2

'Jane' has_chrc age value extremely_young degree m3

'John' has_chrc age value too_young degree m4

where m1 < m2 < m3 < m4.

Adverbs as adjective-modifiers typically express something about the degree of the adjective, such as `very'. These adverbs are usually called *degree adverbs*[5] for obvious reasons.

- Monika is **very** *young.*
- His poetry is **very** *beautiful*.
- The meaning of this passage is **abundantly** *clear*.
- That sign is **hardly** *visible*.

The value expressed by the adjective (i.e. *young*) is transformed by the adverb (i.e. *very*) in a new level, a new value. In general, it holds that adverbs that end in *-ly* can be modified by the same adverbs that can modify the adjective that results from removing the *-ly* ending.

- He writes **very** *clearly*.

- The sun came out **quite** *suddenly*.
- This species is the **slightly** *slower* growing one.

It is possible to define a partial order between different adverbs like extremely is stronger than very, meaning that the degree of an adjective modified with extremely is stronger than the adjective modified with very. For example, if the adjective modified is old, we can considered that extremely old > very old.

The term dimension has various different, although related, meanings in common usage, in mathematics, and in physics. In common usage, a dimension is a measurable aspect of an object. The most commonly used dimensions give the measurements describing the size of a roughly block-shaped object: length, width, and height. However, dimensions can also concern other physical aspects, such as the mass and electric charge of an object, or even, in a context where cost is relevant, an economic aspect such as its price.

One of the meanings of the term "dimension" in physics relates to the *nature* of a measurable quantity. In general, physical measurements that must be expressed in units of measurement, and quantities obtained by such measurements are dimensionful. An example of a dimension is length, which is the dimension for measurements expressed in units of length, be they meters, nautical miles, or light-years. Another example is time, whether the measurement is expressed in seconds or in hours.

In the physical sciences, measurement is most commonly thought of as the ratio of some physical quantity to a standard quantity of the same type, thus a measurement of length is the ratio of a physical length to some standard length, such as a standard meter. Measurements are usually given in terms of a real number times a unit of measurement, for example 2.53 meters.

In mathematics the concept of a measure generalizes notions such as "length", "area", and "volume". Informally, given some base set, a "measure" is any consistent assignment of "sizes" to some of the subsets of the base set. Depending on the application, the "size" of a subset may be interpreted as its physical size, the amount of something that lies within the subset, or the probability that some random process will yield a result within the subset.

The magnitude of a mathematical object is its size: a property by which it can be larger or smaller than other objects of the same kind; in technical terms, an ordering of

the class of objects to which it belongs. It is the property of relative size or extent (whether large or small) of an object, its relative importance.

Table 19 Examples of scalar adjectives

Pole A	Pole B	Dimension	Units
young	old	age	Year, month, day
small	big	size	Inch, foot, meter
low	high	height	Inch, foot, meter
thin	fat	weight	Pound, kilogram

An essential difference between measurements and perceptions is that in general, measurements are crisp, whereas perceptions are fuzzy. Thus, perceptions represented by adjectives are, in general, both fuzzy and granular, where a granule is a clump of attribute-values which are drawn together by indistinguishability, equivalence, similarity, proximity or functionality.

Table 20 Attributes and fuzzy values

Attribute	**Values**
temperature	warm, cold, very warm, much warmer
time	soon, about one hour, not much later
distance	near, far, much farther, not very far
speed	fast, slow, much faster
length	long, short, very long
color	red, blue, green, yellow
age	young, middle-aged, old, very old
size	small, big, very big

Some authors[67] classify adjectives as gradable and non-gradable, according to the following analysis. Gradable adjectives express qualities which can be measured on a scale such as size or value, e.g. ‘very tall’, ‘very good’, but not *‘very huge’. Gradable adjectives can be modified by degree adverbs like ‘very’ or ‘so’. They can also be made into comparatives and superlatives. Non-gradable adjectives express qualities that cannot be intensified by using degree adverbs such as ‘very’, e.g. *‘very male’. The

[67] Department of English Language, University of Glasgow, http://www.gla.ac.uk/englishlanguage/

qualities expressed by non-gradable adjectives tend to be absolute, and they often fall into pairs, e.g. 'male/female', 'married/single', 'black/white', 'true/false'.

Many adjectives describe qualities that can be measured in degrees, such as size, beauty, age, etc. These adjectives are often called gradable adjectives, because they can be used in comparative or superlative forms, or with grading adverbs such as very or extremely, to show that a person or thing has more or less of a particular quality.

Other authors as C. Paradis[68] propose a different categorization of adjectives considering that gradable adjectives fall into three categories [146]:

1) Scalar adjectives: *long, good, nasty*

2) Extreme adjectives: terrible, brilliant, disastrous

3) Limit adjectives: *dead, true, identical*

For Paradis, non- gradable adjectives, such as '*daily* newspaper', '*classical* ballet' and '*pictorial* atlas', are not associated with gradability at all. They are typically categorizing and do not combine with degree modifiers: it is not possible to say 'a *very daily* newspaper', 'an *absolutely daily* newspaper', 'a *fairly classical* ballet', 'a *completely pictorial* atlas'.

Adjectives such as big, little, crisp, mushy, dark, and light that designate values on continuous dimensions are called *scalar adjectives* by Gasser [52]. Scalar adjectives do not normally designate absolute values or ranges of values. Rather their meanings are relative to a standard provided by the context, which require applying the so called common knowledge. The relative nature of scalar adjectives allows us to use the same adjectives for things with all sorts of values on the relevant dimensions. More importantly, scalar adjectives are probably relative because it is the relative value of things that matters, corresponding with the imprecision of human perceptions. Then, knowledge bases as Cyc[69] or ConceptNet [104] which includes more than 1.6 million binary-relational assertions are required.

Scalar adjectives combine with scalar degree modifiers (*fairly long, very good, terribly nasty*). The mode of oppositeness that is characteristic of scalar adjectives is antonymy, defining in this way the two poles of the dimension. Therefore, scalar degree modifiers define degrees of membership to the fuzzy granule defined by the unmarked

[68] Department of English, Lund University
[69] OpenCyc http://www.cyc.com/cyc/opencyc/overview

pole. The unmarked member is also the global member of the opposition [88]. For example, in the pair old and young, old is the global, unmarked adjective. In this thesis the antonym relation is represented with the following notation:

antonym(PoleA, MeaningA, PoleB, MeaningB).

where PoleA and PoleB are the two terms and MeaningA and MeaningB are their corresponding meanings in a similar form as the one used by WordNet, which includes almost 8,000 pairs of terms. In our representation, PoleB represents the unmarked adjective.

Extreme adjectives combine with reinforcing totality modifiers (*absolutely terrible, totally brilliant, utterly disastrous*). Like scalar adjectives, extreme adjectives too are antonymic and conceptualized according to a scale. An example is the scale of merit where the extreme adjectives *terrible* and *excellent* appear at the opposite extremes. Extreme adjectives differ from scalar adjectives in that they do not represent a range on a scale. They represent the ultimate point of a scale.

Finally, limit adjectives combine with totality modifiers (*completely dead, absolutely true, almost identical*). Limit adjectives are logically different from scalar and extreme adjectives in that they are not associated with a scale but conceptualized in terms of 'either-or'. Limit adjectives are complementary. They do not occur in the comparative or the superlative. They are absolute and divide some conceptual domain into two distinct parts. They are thus not susceptible to being laid out on a scale.

In general, it is not easy to recognize if some adjective like *young* is really evaluating (i.e. giving value) to an attribute as *age* (see Table 21). Therefore, without enough information, it would be better to consider that each one of these adjectives defines a dimension with two poles denoted by their antonyms: mushy and crisp, young and old, dark and light, big and little, and so on.

Table 21 Different meanings of young and old

Sentence	**Refers to:**
It is young vine	Freshness and vitality
It is young corn	Not mature
It is an old tradition	Time, not age
He is an old student	Time of graduation

Taking into account the previous reasoning, it is possible to use some common knowledge bases as OpenCyc[70] or ConceptNet[71], containing relations between adjectives, nouns and their attributes in order to make it possible to know, for example, when young refers to age, or to freshness, or to time, etc.

For antonymic relations another useful database is WordNet [121, 122] which contains 7993 pairs of terms. In WordNet, adjectives are organized into clusters containing one or more head synsets (sets of synonymous terms) and optional satellite synsets. Most adjective clusters contain two antonymous parts.

Dixon [43] distinguishes between two kinds of semantic opposition for adjectives:

1) **Antonymy:** An antonym pair is relative to some implicit norm, so they do not provide absolute descriptions. An example is a pair such as large and small. Antonyms occur frequently in comparative constructions and then establish a converse relation: if "A is longer than B", then "B is shorter than A".

2) **Complementarity:** Complements are distinguished from antonyms by the fact that the denial of one term implies the assertion of the other and vice versa. An example is the relation between married and single. True complements cannot occur in comparative constructions; they give complete descriptions.

In his analysis of inter-language class correspondences, Dixon [42] identifies seven subclasses of adjectives, arguing that “each semantic type has its own particular norm and extensional grammatical properties”, the latter also including its position in a string of adjectives (see Table 22), were increasing numbers indicate increasing distance from the noun[72].

As Dixon claims that adjectives denoting origin, composition, purpose or beneficiary of the head noun are post-adjectival modifiers, these adjectives could be coded with 0. Even though these adjectives are not part of Dixon’s definition of proper adjectives, they can well be predicted along Dixon’s lines always to be the ones closest to the noun [232].

[70] OpenCyc contains the full set of Cyc terms as well as millions of assertions http://www.opencyc.org/
[71] the full version of the common sense knowledgebase of ConceptNet contains 1.6 million assertions http://web.media.mit.edu/~hugo/conceptnet/
[72] Linguistic aspects of lexical, EAGLES Preliminary Recommendations on Semantic Encoding Interim Report, Expert Advisory Group on Language Engineering Standards (EAGLES), http://www.ilc.cnr.it/EAGLES96/rep2/node12.html

Table 22 Dixon's semantic classes and their hypothesized order

Code	Semantic class	Examples
1	dimension	big, long, short
2	physical property	hard, sweet, strong, ill
3	speed	quick, slow, fast
4	age	new, young, old
5	color	red, white, black
6	value	good, bad
7	difficulty	easy, difficult
8	qualification	definite (probable), possible (possible), usual (usual), likely (likely), sure (sure), correct (appropriate)
9	human propensity	fond (fond), angry (angry, jealous), happy (anxious, happy), unsure (certain), eager (eager, ready), clever (clever, stupid, generous)
10	similarity	similar, different

Dimensional adjectives can occur in antonym pairs (strong marking). The positive member of each pair is the unmarked member e.g. long, high, heavy etc, which is used in a neutral question (How long is the stick?). The nominalisation which describes this parameter is derived from the unmarked form (What is the length of the road?).

Markedness [109] has been used as cover term for several related phenomena which distinguish the marked member of an antonym pair from the unmarked member. It has been noted that if the name of the semantic scale is morphologically related to one of the antonyms, it is related to the unmarked member, so for example, the name of the scale of LENGTH is related to the unmarked *long* rather than the marked *short*.

Committedness [109] is another criterion which has been proposed to define markedness: the uncommitted member of an antonym pair is said to be unmarked and the committed member is said to be marked, so old is unmarked, while young is marked. Committedness involves an adjective's behaviour in questions. An adjective is said to be committed if it implies a particular value when used in a question, and

impartial or uncommitted if it does not have such an implication. For example, tall is uncommitted in a question like "How tall is Pat?" This question is neutral and can be used whether or not the speaker knows Pat's approximate height and whether Pat is tall, short or of average height. In contrast, the adjective short is committed; a speaker would only ask "How short is Pat?" if there is some reason to believe that Pat is shorter than average height. Many pairs of gradable antonyms contain one committed term and one uncommitted, e.g., old/young, heavy/light, fast/slow; many other pairs are made up of two committed terms, e.g., innocent/guilty, beautiful/ugly, happy/sad.

In WordNet, adjectives are divided in two main classes, which are said to account for the majority of adjectives: *ascriptive* which are considered to ascribe a value of an attribute to a noun and *non-ascriptive* which are similar to nouns used as modifiers. Ascriptive adjectives are organized in terms of antonymy and synonymy while non-ascriptive ones are considered as stylistic variants of modifying nouns and are cross-referenced to the noun files. For example, *astral* and *stellar* have the meaning of *pertaining to a star or stars*.

Gradation[73] is not indicated in *WordNet* because it is not often lexicalized in English. Restrictions on syntactic position (for **p**renominal-only and **p**ostnominal-only adjectives) are directly encoded in the word, as they cannot be inferred from the head word in the cluster.

WordNet does not say anything about the way senses are related: adjectives have as much senses as synsets. Moreover, it does not provide the means to predict grammatical properties from the representation (complementation, alternations, selective restriction). These two features distinguish the relational approach of WordNet from a Generative approach, like Generative Lexicon.

Generative Lexicon [162] is a theory of linguistic semantics which focuses on the distributed nature of compositionality in natural language. Generative Lexicon focuses on the two aspects neglected in WordNet: (1) how the different adjectival senses are related and how they can be derived compositionally from the representations of the noun and the adjective and (2) the syntax-semantics interface. In this theory, the

[73] EAGLES Preliminary Recommendations on Semantic Encoding Interim Report, Expert Advisory Group on Language Engineering Standards (EAGLES), http://www.ilc.cnr.it/EAGLES96/rep2/node12.html

adjectival polymorphism is explained by richer representations of adjectives and nouns (the qualia structure) and the way they combine together.

The qualia structure is defined as the modes of explanation associated with a word or phrase in the language:

1) *formal*: the basic category of which distinguishes the meaning of a word within a larger domain;
2) *constitutive*: the relation between an object and its constituent parts;
3) *telic*: the purpose or function of the object, if there is one;
4) *agentive*: the factors involved in the object's origins or coming into being.

Henry and Bassac [66] have already implemented a free software toolkit designed for the construction, maintenance and collaborative use of a Generative Lexicon. This toolkit has already been used in the construction of an NLP application, although the tests were carried out on a limited number of lexicon entries: just one paper One of the limitations of the system is the size of the lexicon and the obligation to encode the lexicon information manually. They plan to use corpora or systems such as WordNet and its ontology to automatically acquire relevant information, although they considered it a fairly difficult task.

English allows nouns to be used adjectivally (i.e., in function they are "adjectives", in structure they are nouns). Examples: *apple pie*, *desk chair*, *axe handle*, *rally car*, *corner table*, *tire company*. Here, the first word modifies the second, that is, it tells us something further about it. For example, a *rally car* is a *car* which is driven in rallies. These modifiers occur in the same position as adjectives, but they are not. Like simple nouns, these phrases designate categories of things. In each case, the category is a subcategory of the category that is designated by the second noun. What is not easy to deduce is the role the first noun plays, the one that acts as an adjective. In fact it appears that there are few limits on what sort of conceptual relation can be behind the meaning of a noun phrase like that in English (see Table 23).

Therefore, Gasser [52] describes the meaning of those kinds of noun phrases in the following way: an English phrase consisting of a noun *A* (i.e. *apple*) designating a thing category *CA* followed by a noun *B* (i.e. *pie*) designating a thing category *CB* designates a subcategory of *CB*, denoted by *CAB*, (i.e. *apple pie*) whose members are related in a

particular way to at least some members of *CA*. In other words, it defines a new category (i.e. *apple pie*) as a subcategory of the one designated by the second noun (i.e. *pie*).

Table 23 Nouns used adjectivally

Noun + noun	Interpretation	Subcategory relation
apple pie	pie made with apple	apple pie is_a pie.
pie apple	apple used for pies	pie apple is_a apple.
desk chair	a chair used for the desk	desk chair is_a chair.
rally car	a car which is driven in rallies	rally car is_a car

The majority of adjectives denote a state or condition, which may generally be considered permanent, like big, red and small. They are denominated *stative*[74] *adjectives*. In contrast, *dynamic adjectives* denote attributes which are, to some extent, under the control of the one that possesses them. They are characterized by action or forcefulness or force of personality, expressing action rather than a state of being. For instance, *brave* and *calm* denote attributes which may not always be in evidence unlike *red*, for example.

Adverbs are a part of the speech, used to modify any other language element: verbs, adjectives (including numbers), clauses, sentences and other adverbs, except for nouns. Taken as a whole, the adverb class is the most diverse of all the word classes, and its members exhibit a very wide range of forms and functions. For example, some adverbs can be used to modify an entire sentence, whereas others can not. Many semantic classifications of adverbs have been made, as circumstantial adverbs (i.e. Manner, Time and Place are the most distinctive classes of adverbs), Additives, Exclusives, and Particularisers, *Wh-* Adverbs, Sentence Adverbs, etc. Therefore, in those cases, the adverb is an attribute of the verb, of the adjective, of the clause or of the sentence, respectevily.

Examples:

- David [VP is [AdvP *extremely* [AP clever]]
- Mary [VP sings [AdvP *beautifully*]]

[74] WordNet 1.7.1. Princeton University, 2001. Answers.com 01 May. 2007. http://www.answers.com/topic/stative

In the first example, *extremely* modifies *clever* and it is just possible to consider it as another value in the same dimension as clever. On the contrary, *beautifully* describe how Mary sings, that is, how she acts, thus it should be interpreted as an attribute of the action, specifying the action. According to J.Sowa [206], a manner is a property of some process, which is usually expressed by adverbs, such as *quickly, boldly,* and *tentatively*. Therefore, it is possible to consider that Mary's sing has property beautiful, which could be represented by a relation as Mary *do* sing *manner* beautifully, where *do* and *manner* are special operators.

Adverbs typically answer such questions as *how?*, *when?*, *where?*, *in what way?*, or *how often?* This function is called the adverbial function, and it is realised not just by single words (i.e., adverbs) but by adverbial phrases and adverbial clauses.

Adverbs and adjectives have important characteristics in common:

- Their gradability.
- They have comparative and superlative forms.

The words *early, far, fast, hard,* and *late* together with their comparative and superlative forms, can be both adverbs and adjectives (see Table 24):

Table 24 Use of late as adjective and adverb

Adjective	**Adverb**
Robert catches the late train	Robert returns home late.

The comparative *better* and the superlative *best*, as well as some words denoting time intervals (*daily*, *weekly*, *monthly*), can also be adverbs or adjectives, depending on how they are used.

Like adjectives (see Table 25), many adverbs are *gradable*[75], and they can be modified by *very* or *extremely*. The modifying words *very* and *extremely* are themselves adverbs. They are called *degree adverbs*[5] because they specify the degree to which an adjective or another adverb applies. Degree adverbs include *almost*, *barely*, *entirely*, *highly*, *quite*, *slightly*, *totally*, and *utterly*. It should be noticed that there exist some kind of order between those adverbs (i.e. extremely old means older than very old), but this

[75] University College London, http://www.ucl.ac.uk/internet-grammar/adjectiv/inherent.htm

order is not total but partial, because of the inherent human imprecision transferred to natural language.

Table 25 Examples of gradable adverbs

softly	very *softly*
suddenly	very *suddenly*
slowly	extremely *slowly*

4.2.2. Introducing Copular Sentences as Constraints

In English, most adjectives can occur both before and after a noun. Adjectives after the noun are called *predicative* adjectives. Predicative adjectives do not occur *immediately* after the noun. Instead, they follow a verb. A *predicative* adjective is the complement of a verb that links it to the noun. For example, in *the girl is young*, the predicative adjective *young* is linked by the verb *is* to the noun *girl*, which it modifies.

Linking verbs or copular verbs link a subject to a complement. Linking verbs must be followed by a complement in order to make the sentence complete. The complement can be a subject complement or an adverbial, and occurs in two sentence types which are of the Subject-Verb-Complement (SVC) and Subject-Verb-Adverbial (SVA) pattern. Although it might not itself express an action or condition, it serves to equate (or associate) the subject with the predicate. The main linking verb that allows an adverbial as complementation is *to be*. The most common adverbials are place and time adverbials.

The main copular verb in English is the verb "to be", although there are others as "to become", "to get", "to feel", and "to seem". Some grammarians refer to the verb *be* as "the copula" since this is its main function in English. All the forms of *be* can *be* used as a linking verb. *Be* is the main verb of the sentence, rather than the auxiliary and is used in both SVC and SVA patterns.

From one perspective, the copula always relates two things as subsets.

- John is a doctor. As long as the proper nouns represent unique entities, they could be interpreted as specific elements of some set or category, while common nouns describe a whole class or category of entities. Therefore we will represent this kind of sentence by the relation:

John isA doctor.

where John defines an element of the doctor category.

- John and Mary are doctors. We consider that the plural form of a noun defines a set of entities of the same type or category. Therefore, the previous sentence defines the following relations:

  doctors isASetOf doctor.

  John isMemberOf doctors.

  Mary isMemberOf doctors.

  [John, Mary] isASetOf doctor.

- Doctors are educated: includes the set of doctors in the set of those persons who are educated, those who have the attribute educated.

  doctors has_attrib educated.

Copular sentences are the most frequent in English. Martha Kolln and Robert Funk [87] identify ten basic sentence patterns in English syntax; half of them are patterns corresponding to copular sentences and 3 of them are specific patterns for the verb "to be"

- The complement is an adjective: *The team is good*. Includes an object from class team into the subclass good team (i.e. those teams with attribute good)

  the_team has_attrib good.

- The complement is an adverb: *The car is outside*. Includes an object from class car into the class of objects with position outside.

  the_car has_chrc place value outside.

- The complement is a noun phrase:

  The Raiders is the winning team. Includes an object which represent a set of persons identified as Raiders (a proper noun) as element of the subclass *winning team* (i.e. those teams with attribute *winning*)

  winning_team subClassOf team.

  the_Raiders isA winning_team.

Arthur Conan Doyle was a prolific writer. The subject is a proper noun; therefore it defines a specific entity. The predicate has another noun phrase with a common noun that is a category

Arthur_Conan_Doyle was a_prolific_writer.

a_prolific_writer subClassOf writer.

a_prolific_writer has_attrib prolific.

4.2.3. Introducing Comparative Sentences as Constraints

Another important type of sentences are comparative sentences, which consist of two clauses joined together by a comparative formula defined by two words (*Mary is more intelligent than Tom is*). The comparative is the form of an adjective or adverb which denotes the degree or grade by which a person, thing, or other entity has a property or quality greater or less in extent than that of another.

The structure of a comparative in English consists normally of the positive form of the adjective or adverb, plus the suffix -er, or the modifier "more" (or "less") before the adjective or adverb (in the case of polysyllabic words borrowed from foreign languages). The form is usually completed by "than" and the noun which is being compared, e.g. "he is taller than his father is", or "the village is less picturesque than the town nearby". "Than" is used as a subordinating conjunction to introduce the second element of a comparative sentence while the first element expresses the difference ("our new house is larger *than* the old one").

Usage prescriptionists [229] apply a number of rules concerning *than* that those who do not wish to be edited or corrected may wish to note. These prescriptive grammarians say that *than* is a preposition that invariably governs the oblique case i.e. a noun case that is used generally when a noun is the object of a sentence or a preposition.

Than, as used in comparatives, has traditionally been considered a conjunction; as such, for subject comparison, the pronouns after *than* should take the subjective case. In other words, "He's taller than *I*," not "He's taller than *me*". On the other hand, for direct or indirect object comparison, the pronouns should be objective: "I've never worked with a more difficult client than *him*."

Opponents argue that people have been treating *than* as a preposition for centuries. Let’s consider the following examples from well known English and American writers:

- William Shakespeare, whose 1600 play Julius Caesar contains the line: “A man no mightier than thyself or me. . .”
- Samuel Johnson, who wrote: “No man had ever more discernment than him, in finding out the ridiculous.”
- Matthew Prior, *Better Answer*: "For thou art a girl as much brighter than her,/ As he was a poet sublimer than me."
- Samuel Richardson's *Clarissa*, 1.10.58, "I am fitter for this world than you, you for the next than me."
- Lord Byron's letter of 2 November 1804, "Lord Delawarr is considerably younger than me."
- Robert Southey, *Well of St. Keyne*, 51: "She had been wiser than me,/ For she took a bottle to Church."
- William Faulkner's *Reivers*, 4.82: "Let Lucius get out . . . He's younger than me and stouter too for his size."

In Casual, Impromptu, and some Informal use, however, *than* functions as both conjunction and preposition.

Comparison is an inflection not possessed by nouns and pronouns: it belongs to adjectives and adverbs (see Table 26). The degree of comparison of an adjective describes the relational value of an adjective or adjectival expression. An adjective may simply describe a quality (absolute degree); may compare the quality to that of another of its kind (comparative degree); may compare the quality to many or all others (superlative degree).

As well as adjectives, too, adverbs are inflected in terms of comparison. The comparative and superlative forms of adverbs are sometimes generated by adding *-er* and *-est*. However, most adverbs form the comparative using *more* and the superlative using *most*.

Table 26 Degrees of comparison

Absolute	Comparative	Superlative
John is young	Mary is younger than Tom	John is the youngest of all
John is intelligent	Mary is more intelligent than Tom	Mary is the most intelligent of all
John works hard	Mary works harder than Tom	Mary works hardest of all
John travels *frequently*	Mary travels more *frequently* than Tom	Mary travels the *most frequently of all*

In the formation of comparatives and superlatives, some adverbs are irregular (see Table 27).

Table 27 Examples of irregular adverbs

Adverb	Comparative	Superlative
well	*better*	*best*
badly	*worse*	*worst*
little	*less*	*least*
much	*more*	*most*

Empirical psychology has established the fact that we can know a quality only by means of its contrast with or similarity to another. By contrast and agreement a thing is referred to a correlate, if this term may be used in a wider sense than usual. The occasion of the introduction of the conception of reference to a ground is the reference to a correlate, and this is, therefore, the next conception in order.. [154].

When two objects are placed side by side, some differences between them as to size, weight, colour, etc could be noticed. Thus, it could be said that a cow is *larger* than a sheep, gold is *heavier* than iron, a sapphire is *bluer* than the sky. All these have certain qualities; and when comparing the objects, it is done by means of their qualities,—cow and sheep by the quality of largeness, or size; gold and iron by the quality of heaviness, or weight, etc.,—but not the same degree, or amount, of the quality. The degrees belong to any beings or ideas that may be known or conceived of as possessing quality.

Therefore, in the sentence: “Tom is older than Mary”, according to the previous reasoning, Tom’s age is greater than Mary’s age, which is equivalent to Tom’s age - Mary’s age is greater than 0 (zero). Or, if it is not possible to associate old with age, it is

always possible to interpret it as Tom's oldness is greater than Mary's oldness. Knowing that young and old are antonyms, the previous relation can be transformed into Mary's youngness is greater than Tom's youngness.

The statement *Tom is older than Mary* does not place either *Tom* or *Mary* in either an 'old' or in a 'not old' subclass; they could be twin, new-born babies. Older refers to one entity having 'greater age' than that of a stated entity. The statement "Tom and Mary were older than their brother" can be made either they are infants or when they reach whatever age; that is, irrespective of any 'normal', 'average' or 'expected' age. And, furthermore, the entities may be singular or plural.

Some grammarians [53] have defined a comparison prototype X is q-er than Y, in order to represent those kind of sentences. Following, some other definitions to represent them with a different approach are introduced in this thesis.

Let's consider the following sentence S_1: *Mary is younger than Tom*. There we have two entities (Mary and Tom) linked by a verb (is) where, one of them (Mary) has a quality (young) in higher degree than the other entity (Tom). Then we have a comparative relation between two entities based on a certain specific attribute that both of them are assumed to have but with different degrees. If we denote Mary's age and Tom's age as fuzzy numbers Ma and Ta respectively and R the comparative relation, we can write it down as: *R(Ma, Ta) is younger*, where *R* is a binary fuzzy relation. Substituting *R(Ma, Ta)* by a certain variable X, we obtain according to Zadeh definition of Generalized Constraint that X is younger, where X is the constrained variable, and *younger* will be referred to as a granular value of *X,* a fuzzy relation characterized by a membership function *m*, with $m(u)$ representing the degree to which *X* fits the description *younger*.

A similar situation occurs with the sentence S_2: *Mary is more intelligent than Tom*. Here, once again, both Mary and Tom have the same characteristic, intelligence, but Mary has it in a greater degree than Tom. In this case, as in many others, it is not possible to use fuzzy numbers, because intelligence is not a number as age (at least we use the IQ, Intelligence Quotient).

In order to unify the notation, we will consider that comparative forms like younger, bigger, taller, etc are equivalent, short ways to express "more young", "more big", "more tall", etc. Then, the following expression will be introduced

compare(Mary, is, more, young, Tom, M)

which can be interpreted in the same way as was explained before and M is the membership degree associated with the relation *R(Ma, Ta)*.

Furthermore, we will consider that this expression represents the following set of statements: Mary's age is young with degree D1, Tom's age is young with degree D2, M = D1 – D2 as long as D1 > D2; otherwise M = 0.

In order to generalize the previous notation, let's consider E1, E2 two variables which identify two entities. Then, once again, those entities will be linked by a verb (is) where, one of them (E1) has a quality (Adj) in higher degree than the other entity (E2).

compare(E1,is, more, Adj, E2, M)

where it represents the following set of statements: E1 has_attrib Adj with degree D1, E2 has_attrib Adj with degree D2, M = D1 – D2 as long as D1 > D2; otherwise M = 0.

Similarly, an equivalent expression could be defined using *less* in place of *more*, meaning the reverse relation.

compare(E1,is, less, Adj, E2, M)

where it represents the following set of statements: E1 has_attrib Adj with degree D1, E2 has_attrib Adj with degree D2, M = D2 – D1 as long as D1 < D2; otherwise M = 0.

There are also other forms of comparison to show difference between two people, things or events. *Less* is used with uncountable nouns, like *money* or *work* or *travel*, and *fewer* is used with countable plural nouns, like *coins* or *jobs* or *trips*. *Less* is the comparative form of *little***,** and *fewer* is the comparative form of *few*. *More* can also be used in comparative forms with countable and uncountable nouns.

a) John has *more* money than Frank.

compare(John, has, more, money, Frank, M).

Here we introduce a new operator has, which we used for properties (i.e. possessions) in spite of attributes and characteristics. Therefore the previous expression will represent the following set of statements: John has money with degree D1, Frank has money with degree D2, M = D1 – D2 as long as D1 > D2; otherwise M = 0.

b) I've *less* money in the bank than I had last year.

In this case, we consider that "in the bank" is an attribute of money, a specification about which money we are talking about. Then, "money in the bank" will be considered the possesion. By the other hand, the two entities are represented by two instances of the same object, one in the present, now, and one in the past, last year. This means that we will have two objects "I", one with the attribute "now" and the other with the attribute "last year". Let's identify them in this way: 'I_now' and 'I_last_year'. Then the previous expression will represent the following set of statements: 'I_now' have money_in_the_bank degree D1, 'I_last_year' had money_in_the_bank degree D2, M = D2 – D1 as long as D1 < D2; otherwise M = 0.

We consider that has and have are equivalent operators with the same meaning. By default, time (or date) is assumed current time. If the verb expresses past tense, time is assumed before current time. Observe that:

this_year subClassOf year.

last_year subClassOf year.

year subClassOf timeUnit.

this_year means current_year.

current_year is Y and current_date(Y, M, D).

last_year is L and precede(L, Y).

Comparison can also be established to show no difference between two entities, using the AS – AS pattern.

- She has as many brothers as sisters.
- He has as much courage as you do.
- They take as few risks as possible.
- He knows as little English as they do.

For adjectives which form the comparative with *more*, either the construction *less* **...** *than* or the construction *not as* **...** *as* may be used. The construction *not as ... as* is somewhat less formal than the construction *less ... than*.

For instance, the two sentences in each of the following pairs have the same meaning.

- Formal: The red bicycle is less expensive *than* the blue one.
- Informal: The red bicycle is *not as* expensive *as* the blue one.

Following, a schematic resume[76] of the different comparison situations, including examples and the previously defined notation is presented:

1. To compare the difference between two people, things or events.
    a. Pattern: ADJ/ADV –er + THAN
        i. ADJ: Mary is younger than Tom. compare(Mary, is, more, young, Tom, M).
        ii. ADV: Mary works harder than Tom.

        In the case of comparisons were the verb is an action verb, like in this case, the comparison expresses how the entities execute the action, it refers about the manner they act. Then the expression will be

        compare(Mary, works, more, hard, Tom, M).

        Therefore the previous expression will represent the following set of statements: Mary works manner hard with degree D1, Tom works manner hard with degree D2, M = D1 – D2 as long as D1 > D2; otherwise M = 0.

    b. Pattern: MORE + ADJ/ADV/NOUN + THAN
        i. ADJ: Mary is more intelligent than Tom. compare(Mary, is, more, intelligent, Tom, M).
        ii. ADV: Mary travels more frequently than Tom. compare(Mary, travels, more, frequently, Tom, M).
        iii. Countable NOUN: Eloise has more children than Chantal. compare(Eloise, has, more, children, Chantal, M).
        iv. Uncountable NOUN: Eloise has more money than Chantal. compare(Eloise, has, more, money, Chantal, M).

[76] Online English Grammar, **http://www.edufind.com/english/grammar/index.cfm**

c. Pattern: LESS + ADJ/ADV/NOUN + THAN

i. ADJ: Tom is less intelligent than Mary. compare(Tom, is, less, intelligent, Mary, M).

ii. ADV: Tom travels less *frequently* than Mary. compare(Tom, travels, less, frequently, Mary, M).

iii. Uncountable noun: Chantal has less money than Eloise. compare(Chantal, has, less, money, Eloise, M).

d. Pattern: FEWER + NOUN + THAN

i. Countable noun: Chantal has fewer children than Eloise. compare(Chantal, has, less, children, Eloise, M).

2. To compare people, places, events or things, when there is no difference.

a. Pattern: AS + Adj/Adv + AS.

i. ADJ: Peter is as old as John. compare(Peter, is, as_as, old, John, M).

ii. ADV: Peter runs as fast as John. compare(Peter, runs, as_as, fast, John, M).

In this case, the operator as_as identify the situation. Therefore the expression will represent the following set of statements: Peter runs manner fast with degree D1, John runs manner fast with degree D2, M = D1 – D2 as long as D1 = D2; otherwise M = 0.

3. To compare the difference between two people, things or events.

a. Pattern: NOT AS + Adj/Adv + AS.

i. ADJ: Mont Blanc is not as high as Mount Everest. compare(Mont_Blanc, is, not_as, high, Mount_Everest, M).

ii. ADV: John does not work as hard as Tom. compare(John, work, not_as, hard, Tom, M).

4. To show no difference with countable nouns:

a. Patterns: AS MANY + NOUN + AS / AS FEW + NOUN + AS

i. They have as many children as us. compare(they, have, as_as, children, us, M). Here the children act as a property because of the verb have and the comparison could be implemented the same way than in the case as_as.

ii. Tom has as few books as Jane. compare(Tom, has, as_as, books, Jane, M). It is the same situation as the previous one.

5. To show no difference with uncountable nouns

   a. Patterns: AS MUCH + NOUN + AS / AS LITTLE + NOUN + AS

      i. John eats as much food as Peter. compare(John, eats, as_as, food, Peter, M).

      ii. Jim has as little food as Sam. compare(Jim, has, as_as, food, Sam, M).

A short resume of the main types of comparative sentences is shown in Table 28.

Table 28 Constraint representation of comparative sentences

Notation	Example
compare(E1,is, more, Adj, E2, M)	*Mary is younger than Tom.* *Mary is more intelligent than Tom.*
compare(E1,is, less, Adj, E2, M)	Tom is less intelligent than Mary.
compare(E1, is, as_as, Adj, E2, M)	Peter is as old as John.
compare(E1, is, not_as, Adj, E2, M)	Mont Blanc is not as high as Mount Everest.
compare(E1,do, more, Adj, E2, M)	Mary works harder than Tom *Mary travels more frequently than Tom*
compare(E1,do, less, Adv, E2, M)	Tom travels less *frequently* than Mary.
compare(E1, do, as_as, Adv, E2, M)	Peter runs as fast as John.
compare(E1, do, not_as, Adv, E2, M)	John does not work as hard as Tom.
compare(E1,has, more, Prop, E2, M)	Eloise has more children than Chantal Eloise has more money than Chantal.
compare(E1, has, less, Prop, E2, M)	Chantal has less money than Eloise. Chantal has fewer children than Eloise
compare(E1, have, as_as, Prop, E2, M)	They have as many children as us. Tom has as few books as Jane. Jim has as little food as Sam.
compare(E1, do, as_as, Prop, E2, M)	John eats as much food as Peter.

4.2.4. Superlative form

The superlative form of an adjective is used to describe something which possesses a characteristic in the greatest degree. The superlative forms of adjectives are usually preceded by **the**, and followed by the nouns they modify. In the following examples, the superlative forms of the adjectives are underlined.

- Louis is the youngest boy in our class.
- She is the best actress I have ever seen.

It should be noted that the noun following the superlative form of an adjective is often omitted, when it is obvious what is meant. This is illustrated in the following examples.

- That star is the brightest.
- These cookies are the best.

The superlative form of adjectives which do not use endings is formed by placing the word **most** before the positive form of the adjective. For example

Tom is the oldest in his classroom.

Adjectives which form the superlative with the adverb **most** are used in the same constructions as adjectives which form the superlative with the ending **est**. For example:

1. Mary is the most intelligent child in the family.
2. The Lord of the Rings is the most interesting book I have ever read.

As we did it before with comparative sentences to unify the notation, we will consider that superlative forms like youngest, biggest, tallest, etc are equivalent, short ways to express "most young", "most big", "most tall", etc. Then the following expression will be introduced.

superlative(Mary, is, most, intelligent, child_in_the_family, M)

where child_in_the_family is a set of childs with the attribute in_the_family. Then, the interpretation of that expression is that Mary has_attrib intelligent degree M, where M > Mi, for every C isMemberOf child_in_the_family and C has_attrib intelligent degree Mi.

An equivalent definition could be introduced for superlative sentences using least. Example: John is the least intelligent in the office. The expression will be:

superlative(John, is, least, intelligent, in_the_office, M)

where in_the_office is a set of person with the attribute in_the_office. Then, the interpretation of that expression is that John has_attrib intelligent degree M, where M < Mi, for every C isMemberOf in_the_office and C has_attrib intelligent degree Mi.

Following, we present a schematic resume of different superlative situations, including examples[77].

1. Comparing *more than two* things, persons, or events: in the world, in the class, in the group, of all
    b. Pattern: ADJ/ADV suffix –est
        i. ADJ: John is the youngest in the group.
        ii. ADV: Mary works the hardest of all the students.
    c. Pattern: MOST + ADJ/ADV

[77] Online English Grammar, **http://www.edufind.com/english/grammar/index.cfm**

i. ADJ: Mary is the most intelligent of the daughters.

ii. ADV: Mary travels the most frequently in the classroom.

d. Pattern: LEAST + ADJ/ADV

i. John is the least intelligent of the siblings.

ii. John travels the least frequently of all his friends.

A short resume of the main types of comparative sentences is shown in Table 29.

Table 29 Constraint representation of superlative sentences

Notation	Example
superlative(E1, is, most, Adj, Group, M) superlative(E1, is, least, Adj, Group, M)	Mary is the youngest child in the family. Mary is the most intelligent child in the family. John is the least intelligent in the office.
superlative(E1, do, most, Adv, Group, M)	Mary works the hardest of all the students.
superlative(E1, do, most, Adv, Group, M) superlative(E1, do, least, Adv, Group, M)	Mary travels the most frequently in the classroom. John travels the least frequently of all his friends

4.2.5. Application examples

The Bronte sisters

Let's consider that we are interested in searching for information about the Bronte sisters using Google. Our first search with the words Bronte sisters produces 686.000 links. The first one of them corresponds to the encyclopedia Wikipedia. Between the first ten links are several pages dedicated to book promotion and sales, some links correspond to the museum dedicated to their memory, another encyclopedia, etc. If we search for the string "Bronte sisters", we obtain 288.000 links, what is a normal situation because we are restricting the search. Some other search results are shown in Table 30.

Table 30 Results by searching about the Bronte sisters

Search for	Links obtained without ""	Links obtained with ""
Bronte sisters	686.000	288.000
Bronte sister	1.040.000	5.030
eldest Bronte sister	55.300	8
youngest Bronte sister	160.000	102
second eldest Bronte sister	6460	0

Some of the Web pages found have been included as examples in Appendix B. Following we are going to analyze some of the sentences found in them in more detail in order to analyze in more detail the formal relations previously proposed in the present chapter. The page about Charlotte Brontë[78] from the Wikipedia begins with the following sentence:

> "Charlotte Brontë (pronounced /□brɒnti/) (April 21, 1816 – March 31, 1855) was a British novelist, the eldest of the three famous Brontë sisters whose novels have become standards of English literature."

Let's analyze the different phrases included in this complex sentence. The nucleus of the sentence is that "Charlotte Brontë was a British novelist". This is a copula sentence. There is a noun phrase, Charlotte Brontë, with two proper nouns, Charlotte and Brontë. Looking at YAGO[79] categories [211, 212], we can find that Brontë is a family name (Brontë isA family_name) and Charlotte is a given name (Charlotte isA given_name). Therefore, we can conclude that Charlotte Brontë is a person, which belongs to the Brontë family, another category from Wikipedia.

The verb phrase "was a British novelist" includes the copula verb "was" and another noun phrase "a British novelist", which shows, according to Gasser [52], that Charlotte belongs to the category "British novelist", which in turn is a subcategory of "novelist". This interpretation can be also corroborated by looking at the Wikipedia ontology. Summarizing:

[78] Charlotte Brontë, http://en.wikipedia.org/wiki/Charlotte_Brontë last modified on 1 September 2008, at 19:26
[79] YAGO knowledge base http://www.mpii.mpg.de/~suchanek/yago

```
Brontë isA family_name.
Charlotte isA given_name.
Charlotte Brontë means Charlotte_Brontë.
Charlotte_Brontë isA person.
Charlotte_Brontë isA British_novelist.
British_novelist subClassOf novelist.
```

A special identifier for Charlotte Brontë, Charlotte_Brontë, has been introduced in order to simplify its identification. Observe that while Charlotte_Brontë isA British_novelist (because Charlotte Brontë is a proper noun, identifying some entity), British_novelist subClassOf novelist (because novelist is a common name, and therefore is British novelist).

The following clause will be also introduced as a heuristic feature given by human common practice: to designate persons by their given name.

Charlotte means Charlotte_Brontë.

Let's look at the other clause included in the cited sentence:

"Charlotte Brontë was the eldest of the three famous Brontë sisters whose novels have become standards of English literature".

The sentence is still a copula sentence, and the subject is the same, so let's look at the second noun phrase, included into the verb phrase: "the eldest of the three famous Brontë sisters whose novels have become standards of English literature". The head of the noun phrase is "eldest", saying that Charlotte is the eldest of some set, which is a comparative sentence. As a comparative sentence, we have two entities involved, Charlotte by one side, and "the three famous Brontë sisters whose novels have become standards of English literature", which identify the set of persons from which Charlotte is the eldest. This set is identified by a noun phrase, which head is "Brontë sisters", pre modified by "the three famous" and post modified by "whose novels have become standards of English literature". Analyzing the noun phrase head, it is possible to deduce that there are a proper noun, Brontë, and a common noun, sisters. We already know that Brontë isA family_name. As we did it before with British novelist (subClassOf novelist), following the same reasoning, we could say that:

Brontë sisters means Brontë_sisters
Brontë_sisters subClassOf sisters.
Brontë_sisters isSet [].
Charlotte_Brontë isA Brontë_sisters
Charlotte_Brontë isMemberOf Brontë_sisters.

As long as Brontë sisters is a plural form it designs a set of sisters, to which belongs Charlotte as long as she is a Brontë sister.

Analyzing the string that precedes and modifies the noun phrase head, we get that Bronte sisters have an attribute, famous, and a cardinal numeral commonly used in expressing how many objects are referred to. Therefore we can conclude that

Brontë_sisters has_attrib famous
Brontë_sisters has_chrc cardinal_number three.

Therefore, we now know that the set of Bronte sisters is composed by 3 sisters and Charlotte is one of them. Therefore, Charlotte Brontë has also the attribute famous.

Analyzing the post head string "whose novels have become standards of English literature", we could identify a relative clause, introduced by the relative pronoun whose, which indicates possession, so we could get that the Brontë_sisters has novels. These entities, the novels, will be considered as an independent set of entities:

Brontë_sisters_novels subClassOf novels
Brontë_sisters_novels isSet []
Brontë_sisters has novels Brontë_sisters_novels.

The relative clause "whose novels have become standards of English literature" has a verb phrase which includes another noun phrase "standards of English literature". Therefore, once again, we have that:

English_literature means English literature
English_literature subClassOf literature

As long as the noun phrase "standards of English literature" could be reordered as English literature standards. Then we have that:

English_literature_standards means English_literature standards
English_literature_standards subClassOf standards
English_literature_standards isSet []

And we have that the set of novels Bronte sisters:

Brontë_sisters_novels subClassOf English_literature_standards

Now that we have determined who are the Bronte sisters, we can return to analyze the second clause of the cited sentence: "Charlotte Brontë was the eldest of the three famous Brontë sisters whose…." from where we get that:

superlative(Charlotte_Brontë, is, most, old, Brontë_sisters, D1)

where D1 is the corresponding membership degree.

There are still two pieces of the original sentence that we have not analyzed and we will not do it. Those pieces, "(pronounced /□brɒnti/)" and "(April 21, 1816 – March 31, 1855)", are fragments, between parenthesis, with special meaning, so we will consider that the computer is not able to process them. It is important to remark that we are not assuming that the computer is able to successfully understand all the sentences included in a document in natural language. Therefore we assume that there will be sentences, paragraphs and even documents that the computer can not interpret. Others may be erroneously interpreted. In other cases, the information could not be useful as is the first fragment about the pronunciation. Other cases require special interpretation as is the case of the second fragment, which specify the beginning and end points of Charlotte's life.

Searching for the eldest Bronte sister, we found Bibliomania page[80], where is possible to find the following sentence:

"Charlotte was the eldest of the three literary Brontë sisters"

The noun phrase "the three literary Brontë sisters" could be reformulated:

literary_Brontë_sisters isSet [].
literary_Brontë_sisters has_chrc cardinal_number three.

as we did it before.

superlative(Charlotte_Brontë, was, most, old, literary_Brontë_sisters, D2)

According to our reasoning algorithm,

literary_Brontë_sisters isSubclassOf Brontë_sisters

[80] Bibliomania, http://www.bibliomania.com/0/0/9/frameset.html

which is also a set with cardinality three. So, it looks clear that we are in the right way, that we have the right interpretation.

But the situation gets twisted when we found the book "The lonely dreamer" [46] with the sentence (page 80):

"Marie had before this considered the eldest Brontë sister"

From where we get that

Marie isA Brontë_sister superlative(Marie_Brontë, had_considered, most, old, Brontë_sisters, D3)

Who is that Marie considered the eldest Brontë sister? It was not Charlotte the eldest of the Brontë sister? It should be pointed out that Brontë_sisters is not the same set as literary_Brontë_sisters although the second should be a subset of the first one.

In Emily Bronte's page[81] we found the following sentence:

"Emily was the second eldest of the three surviving Brontë sisters, being younger than Charlotte and older than Anne"

Analysing it, we found another set, three_surviving_Brontë_sisters, made up of

three_surviving_Brontë_sisters isSet [Charlotte, Emily, Anne]

with the same elements as before, but with a new attribute "surviving". The sentence also explains that:

comparative(Emily, was, more, young, Charlotte, D2) comparative(Emily, was, more, old, Anne, D3)

Searching into the Bronte's page from Wikipedia, we could find other interesting sentences:

"In 1824 the four eldest Brontë daughters were enrolled as pupils at the Clergy Daughter's School at Cowan Bridge. The following year Maria and Elizabeth, the two eldest daughters, became ill, left the school and died; Charlotte and Emily were brought home."

The first sentence defines another set: the four eldest Brontë daughters, which is a set of daughters, which should be sisters each other.

[81] Emily Bronte's page http://en.wikipedia.org/wiki/Emily_Brontë

```
four_eldest_Brontë_daughters isSet []
four_eldest_Brontë_daughters has_chrc cardinal_number four.
X isDaughterOf B and Y isDaughterOf B => X is SisterOf Y.
four_eldest_Brontë_daughters enrolled atPlace Clergy_Daughter's_School inYear 1824
```

The second sentence says that Maria and Elizabeth, the two eldest daughters, died, leaving alive Charlotte and Emily, whom we already know that, are two of the three surviving Bronte sisters. It does not say that they are the four eldest Bronte daughters but it looks normal to assume it, by context. Therefore

```
Maria isA Bronte_daughters.
Elizabeth isA Bronte_daughters.
Charlotte isA Bronte_daughters.
Emily isA Bronte_daughters.
four_eldest_Brontë_daughters isSet [Maria, Elizabeth, Charlotte, Emily]
```

From this sentence we also obtain that

superlative(Maria, is, most, old, four_eldest_Bronte_daughters, D4)

If we exclude Maria from the set, we obtain a new set, where Elizabeth should be the eldest:

```
send( four_eldest_Bronte_daughters,
    substract([Maria], four_eldest_Bronte_daughters2) )
    superlative(Elizabeth, is, most, old, four_eldest_Bronte_daughters2, D5).
```

If we did it once more, we obtain a new set, made up by Charlotte and Emily.

Furthermore, we also know that Maria and Elizabeth died

```
Maria died onDate following_year
Elizabeth died onDate following_year
```

Therefore Charlotte and Emily became the two surviving daughters that were enrolled at Clergy Daughter's School. And Charlotte became the eldest of them at that time. What time (year) is that? If the previous sentence is about something that happens in 1824, we could consider that it sets the current time at that moment.

current_time(1824).

Therefore, the following year is one year after the current year, 1825.

following_year isA year.
year isA time_unit.
time_unit isA unit.
following_year has_value 1825.

By other side, we already know that Charlotte, Emily, Anne were the three surviving Brontë sisters. And we also know that Anne was younger than Emily, which was younger than Charlotte. Therefore, Anne is the youngest of them. That information could also be confirmed by searching for "youngest Bronte sister".

Why Anne was not enrolled as pupil at the Clergy Daughter's School? Maybe because just the four eldest Bronte daughters were enrolled. Maybe because she was too young. How old was Anne at that time? Looking for her born date in YAGO, we can find that

Anne born inYear(1820). => Anne has_chrc age 4 inYear(1824).

Therefore in 1824, Anne was 4 years old. Figure 13 shows a graph of the main events on the life of Anne Bronte

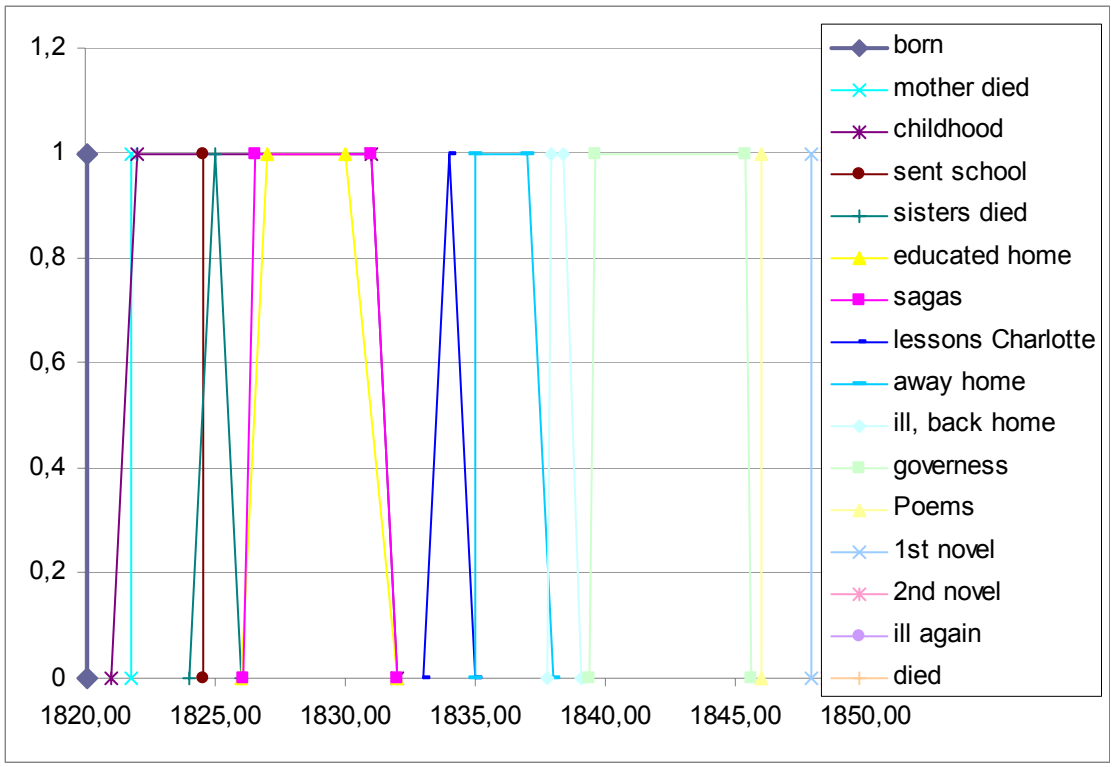


Figure 13 Main events on Anne Bronte's life

How old were the other daughters in 1824? Looking also at YAGO [211, 212], we get that

Maria born inYear(1814). => Maria has_chrc age 10 inYear(1824) Elizabeth born inYear(1815). => Elizabeth has_chrc age 9 inYear(1824) Charlotte born inYear(1816). => Charlotte has_chrc age 8 inYear(1824) Emily born inYear(1818). => Maria has_chrc age 6 inYear(1824)

4.3. From Generalized Constraints to Protoforms

Recently, Zadeh [246, 248, 250] has introduced the idea of protoforms. These protoforms are local reasoning patterns based on the structure of the component propositions. A protoform consists of a collection of premises *P1, ... , Pq* and a consequent *Q* and is of the nature (*P*1, ... ,*Pq*) |— *Q*. Then having a knowledge base with propositions whose structure matches the premise of a protoform, it is possible to infer the consequent.

Among the many concepts related with the problem of locating and inferring from decision-relevant information embedded in a large database, there are four concepts that stand out in importance: 1) search, 2) precisiation and 3) deduction. In relation to these concepts, a basic underlying concept is that of 4) protoform — a concept which is centred on the confluence of abstraction and summarization. Informally, a protoform — abbreviation of prototypical form — is an abstracted summary. More specifically, a protoform is a symbolic expression which defines the deep semantic structure of a construct such as a proposition, command, question, scenario, or a system of such constructs [258].

Yager [233] points out QASs differ from search engines and other information-seeking applications by having a deduction capability, an ability to answer questions by a synthesis of information residing in different parts of its knowledge base. This capability requires appropriate representation of various types of human knowledge, rules for manipulating this knowledge, and a framework for providing a global plan for appropriately mobilizing the knowledge in response to a query. Protoforms are analysed as an aid to deduction and local manipulation of knowledge.

Informally, a protoform, A, of an object, B, written as A=PF(B), is an abstracted summary of B. The primary function of *PF*(*B*) is to place in evidence the deep semantic structure of *B* [254] Usually, B is a lexical entity such as a proposition, question, command, scenario, decision problem, etc. More generally, B may be a relation, system,

geometrical form or an object of arbitrary complexity. Usually, A is a symbolic expression, but, like B, it may be a complex object. Following, our attention will be focused on protoforms of propositions, with PF(p) denoting a protoform of p. (see Figure 14).

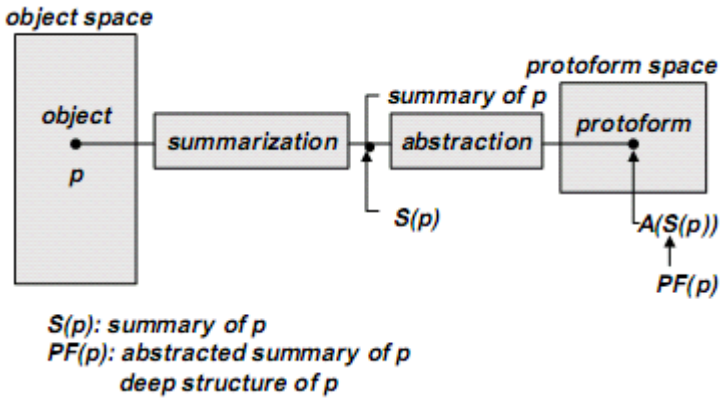


Figure 14 Steps for protoform definition

Abstraction has levels, just as summarization does. For this reason, an object may have a multiplicity of protoforms (see Figure 15). Conversely, many objects may have the same protoform. Such objects are said to be protoform-equivalent, or PF-equivalent, for short.

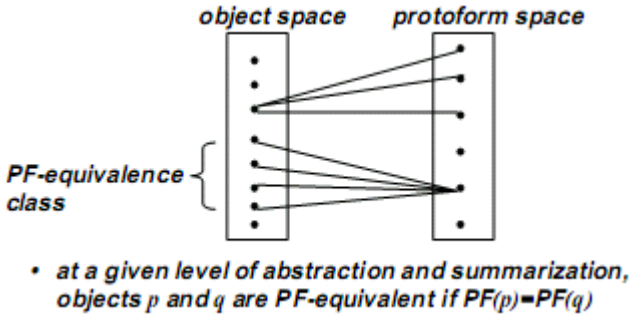


Figure 15 Correspondence between objects and protoforms

The set of protoforms of all precisiable propositions in NL, together with rules which govern propagation of generalized constraints, constitute what is called the Protoform Language (PFL). Examples of propositions in NL:

Proposition	Monika	is	young
Instantiation	Age(Monika)	is	young
Protoform	A(M)	is	Y

Where A: abstraction of Age; M: abstraction of Monika; Y: abstraction of young.

Proposition	Monika is much younger than Pat		
Instantiation	C(Age(Monika), Age(Pat))	is	much younger
Protoform	C(A(M), A(P))	is	MY

Where A: abstraction of Age; M: abstraction of Monika; P: abstraction of Pat; C: abstraction of comparison relation; MY: abstraction of much younger.

Proposition	distance between New York and Boston is about 200 miles		
Instantiation	Distance(New York, Boston)	is	about 200 miles
Protoform	D(N,B)	is	R

Where D: abstraction of Distance; N: abstraction of New York; B: abstraction of Boston; R: abstraction of about 200 miles.

Proposition	usually Robert returns from work at about 6pm		
Instantiation	Prob{ Time(Robert.returns.from.work) is about 6 pm }	is	usually
Protoform	Prob{ T is A }	is	R

Where T: abstraction of Time(Robert.returns.from.work); A: abstraction of about 6 pm; R: abstraction of usually.

Proposition	Carol lives in a small city near San Francisco		
Instantiation	Location(Residence(Carol))	is	(city near SF, small city)
Protoform	L(R(C)	is	(A and B)

Where L: abstraction of Location; R: abstraction of Residence; C: abstraction of Carol; A: abstraction of city near SF; B: abstraction of small city.

Proposition	most Swedes are tall		
Instantiation	1/n ΣCount(Swedes[Height] is tall)	is	most
Protoform	1/n ΣCount(S[H] is T)	is	M

Where S: abstraction of Swedes; H: abstraction of Height; T: abstraction of tall; M: abstraction of most

Two propositions, *p* and *q*, are *protoform equivalent* if they have identical protoforms. This concept provides a basis for a mode of knowledge organization, deduction and search in which what matters is the deep semantic structure rather than the surface structure and domain. Thus, in this kind of organization all protoform equivalent propositions are grouped together in a protoform module. Each module comprises a collection of protoformal rules of deduction.

Abstraction of elements of Generalized Constraint Language gives rise to what is referred to as Protoform Language, PFL. A consequence of the concept of protoform equivalence is that the cardinality of PFL is orders of magnitude smaller than the cardinality of GCL, or, equivalently, the set of precisiable propositions in NL.

The small cardinality of PFL plays an essential role in deduction. Protoform-based deduction may be viewed as a generalization of deduction in classical symbolic logic. The principal difference is that in protoformal deduction each rule is associated with a computational part, and rules are large in number and are drawn from a wide variety of fields and methodologies.

The rules of deduction in GTU are, basically, the rules which govern constraint propagation. In GTU, such rules reside in the Deduction Database (DDB), which comprises a collection of modules as:

- possibility module,
- probability module,
- fuzzy arithmetic module,
- fuzzy logic module,
- search module,
- extension principle module.

These modules contain rules drawn from various fields and various modalities of generalized constraints. A typical rule has a symbolic part, which is expressed in terms of protoforms; and a computational part which defines the computation that has to be carried out to arrive at a conclusion. In [258] some of the basic rules are listed as well as a number of other rules without describing their computational parts. The motivation for doing so, he says, is to point to the necessity of developing a set of rules which is much more complete than the few rules which are used as examples in this section.

Following, some protoforms available in Approximate Reasoning are identified [233] Premises are identified by *Pj* and consequent is denoted by *Q*. Projection Protoform, PF-1, provides a basic protoform for inferring information about a component variable from a joint relation.

- PF-1: Projection Protoform

$P1$: $(V1, V2, \ldots, Vq)$ *is* H

Q: $V1$ *is* D where $D = \text{Proj}_{V1}((V1, V2, \ldots, Vq) \text{ is } H)$

We recall

$D(x1) = \text{Max}_y [H(x1, y)]$

where

$y = (x2, \ldots, xq) \in Y$ and $Y = X2 * X3 * \ldots * Xq$

The Conjunction/Projection Protoform, PF-2, is a fundamental protoform. It provides the basic protoform for making inferences from a joint relationship and

companion propositions about its components. It essentially involves a conjunction of the premises followed by a projection onto the relevant variable.

- PF-2 Conjunction/Projection Protoform

$P1$: $V1$ *is* $B1$

$P2$: $V2$ *is* $B2$

Pq: Vq is Bq

$Pq+1$: $(V1, V2, \ldots, Vq)$ *is* H

Q: Vj *is* D ($D = \text{Proj}_{Vj}((V1, V2, \ldots, Vq)$ *is* E)

Here

$$E(x1, \ldots, xq) = \text{Min}[B1(x1) \wedge B2(x2) \wedge \ldots \wedge Bq(xn) \wedge H(x1, \ldots, xq)].$$

Note if any of the $P1, \ldots, Pq$ are missing, this still holds and we can replace Bi by Xi. There are also other protoforms which can be seen as special cases of the preceding.

Let's show an example. Suppose that we have two NL propositions, p: *Mary is young* and q: *Tom is a few years older than Mary*. They can be easily transformed in two generalized constraints, $GC(p)$: $Age(Mary)$ is *young* and $GC(q)$: $Age(Tom)$ is $Age(Mary) + few$, which can be expressed as the protoforms $PF(p)$: X is A and $PF(q)$: Y is $(X + B)$. Then, by the deduction rule:

X is A

Y is $(X + B)$

Y is $A + B$

we obtain that $Age(Tom)$ is $(young + few)$ where $\mu_{A+B}(v) = \sup_u(\mu_A(u) \wedge \mu_B(v\text{-}u))$

4.3.1. Introducing protoforms to manage with noun phrases, copular sentences, and comparative sentences.

The concept of a protoform is related with the concepts of semantic network, conceptual graph and ontology. The main difference is that the concept of a protoform is formulated within the conceptual structure of fuzzy logic.

Taking into account the notation we adopted before to represent the constraints and that, in English, the expression age of Monika is equivalent to Monika's age, we would prefer to represent the prototypical form of attributes like age, height, length, etc, using the dot notation, i.e. Monika.age, in spite of the one with parenthesis, i.e. age(Monika). Therefore we will write the protoform of the proposition *Monika is young* as *S.A* is *V*, where *S* is an abstraction of the subject *Monika*, *A* is an abstraction of the attribute *age*, and *V* is an abstraction of the value *young*, assigned to that attribute. If we have the proposition *the rose is red*, *S* is an abstraction of the subject *the rose*, *A* is an abstraction of the attribute *color*, and *V* is an abstraction of the value *red.* The same prototypical form could be applied to the sentences *Monika is very intelligent* where *very intelligent* is the value of the Monika's attribute, *intelligence*.

If we have a different kind of copular sentence like *Monika is outside*, where the complement is an adverb, we can still use the same kind of protoform *S.A* is *V*. In this case, *S* is an abstraction of the subject *Monika*, *A* is an abstraction of the attribute *place* or *situation* of the subject, and *V* is an abstraction of the value *outside*, which specifies the value assigned to that attribute. If *Monika is in the classroom*, then "*in the classroom*" will represent the value of the attribute *place* for *Monika.*

Based on the previous examples, we consider that the protoform *S.A* is *V* could be applied to any noun phrase where the intrinsic semantic structure of that expression corresponds to specifying or describing the value of a subject attribute, including an adverb modifying the adjective. Therefore, those types of proposition are *protoform equivalent* because all of them have identical protoforms and can be grouped together in a protoform module comprising the same collection of deduction rules.

Let's analyze another kind of copular sentence: *John is a doctor*. In this case, it says that *John* belongs to a certain category, *doctor*, which means that John has attributes that characterize him as an element of this category. Then, in this case, we are not talking about one special attribute but about a whole set of them. Therefore, we will

introduce another type of protoform in order to represent this situation: *S* is *C*, where *S* is the subject and *C* is the category to which belongs the subject. In the sentence, *The Raiders is the winning team*, the protoform is still the same, *S* is *C*, where *The Raiders* is the subject which has the attributes to be considered as belonging to the category *the winning team* that is *C*. A short resume of the 2 main types of protypical forms that represent copulative sentences is shown in Table 31

Table 31 Prototypical forms of copulative sentences

Protoform	**Example**
S.A is *V* *S* subject, *A* attribute, *V* value	*Monika is young* *Monika is very intelligent* *Monika is outside* *Monika is in the classroom*
S is *C* *S* subject *C* category of the subject	*John is a doctor*

By other hand, the comparative sentences like *Mary is younger than Tom*, will correspond to another type of protoform. As we explained before, there we have a comparison relation between two subjects by a certain attribute. The prototypical form defined for this case is: $C(S_1.A, S_2.A)$ is *V*, where *C* is the abstraction of the comparison relation, S_1 and S_2 are the two subjects, *Mary* and *Tom* respectively, *A* is the abstraction of the attribute *age* and *V* is the abstraction of the value *younger* (or *more young*). The same protoform could be applied in case of sentences like: *Mary is more intelligent than Tom*, and *Tom is less intelligent than Mary*, just changing the value (*A*: *intelligence*; *V*: *more intelligent* and *V*: *less intelligent*, respectively).

Let's analyze another kind of comparison like in *Mary works harder than Tom*. There the comparison is about an action, not about an attribute, but we can still use the same type of protoform: $C(S_1.D, S_2.D)$ is *V* just changing the interpretation of the complement of the subject, *D* which now represents the abstraction of the action, *works*; all the other symbols remain with the same meaning as before; in this case, *harder* will be the value represented by the abstraction *V*. The same protoform could be applied to situations like *Mary travels more frequently than Tom* and *Tom travels less frequently than Mary*, just changing the value (*V*: *more frequently* and *V*: *less frequently*, respectively).

If the comparison involve a property as in *Eloise has more children than Chantal* then a similar protoform could be introduced: $C(S_1.P, S_2.P)$ is V where P is the abstraction which indicates property (*has*) and V is the abstraction of the value *more children*. Th same protoform could be applied in case of sentences like: *Eloise has more money than Chantal* (V: *more money*), *Chantal has less money than Eloise* (V: *less money*), and *Chantal has fewer children than Eloise* (V: *fewer children*).

All the three previous protoform types: $C(S_1.A, S_2.A)$ is V, $C(S_1.D, S_2.D)$ is V, and $C(S_1.P, S_2.P)$ is V could be integrated into a more general category $C(X_1, X_2)$ is V, in case that we do not need to differentiate between attributes, actions or properties. In this case, X_1, X_2 are abstractions of two entities or things that we are just comparing.

Let's analyze another kind of comparison sentences: *Peter is as old as John*. In this case, we consider that we can apply the protoform type $C(S_1.A, S_2.A)$ is V, considering that A is an abstraction of the attribute *age*, and the value V is an abstraction of *as old*, in spite of *older* or *more intelligent* or *less intelligent* as in the previous examples. If the sentence is *Peter is as intelligent as John*, then the value V would be an abstraction of *as intelligent*.

If the comparison is about an action like in *Peter runs as fast as John*, following the same reasoning we could represent it by the protoform $C(S_1.D, S_2.D)$ is V, which is the one related with actions. As the previous example, the value V will represent the abstraction of *as fast*, and D the action *runs*. In sentences like *John eats as much food as Peter*, the value could be considered "*as much food"*.

If the comparison is about a property like in *They have as many children as us*, we could represent it by the protoform $C(S_1.P, S_2.P)$ is V, which is the one related with properties. As the previous example, the value V will represent the abstraction of *as many children*, and P is the abstraction which indicates property (*have*). The same type of protoform could be applied to sentences like *Tom has as few books as Jane*, and *Jim has as little food as Sam*. In those cases, the value will be *as few books* and *as little food* respectively.

Another type of situation occurs when we have a negative comparison as *Mont Blanc is not as high as Mount Everest*. There, the indicated protoform type will be $C(S_1.A, S_2.A)$ is V, where the value included the negative operator *not as high*. Other

interpretation could be that the protoform type is $C(S_1.A, S_2.A)$ is not V and the value is *as high.*

A similar situation will be for comparison sentences which include an action verb: John does not work as hard as Tom. There, the prototype form will be $C(S_1.D, S_2.D)$ is V or $C(S_1.D, S_2.D)$ is not V. If the comparison is about a property like in *They do not have as many children as us*, we could represent it by the protoform $C(S_1.P, S_2.P)$ is V, or $C(S_1.P, S_2.P)$ is not V. A short resume of the 2 main types of protypical forms that represent superlative sentences is shown in Table 32

Table 32 Prototypical forms of comparative sentences

Protoform	**Example**
$C(S_1.A, S_2.A)$ is V C comparison relation, S_1, S_2 subjects A attribute V value	*Mary is younger than Tom* *Mary is more intelligent than Tom* *Tom is less intelligent than Mary* *Peter is as old as John* *Mont Blanc is not as high as Mount Everest*
$C(S_1.D, S_2.D)$ is V C comparison relation, S_1, S_2 subjects D action V value	*Mary works harder than Tom* *Mary travels more frequently than Tom* *Tom travels less frequently than Mary* *Peter runs as fast as John* *John eats as much food as Peter* John does not work as hard as Peter
$C(S_1.P, S_2.P)$ is V C comparison relation, S_1, S_2 subjects P property V value	*Eloise has more children than Chantal* *Eloise has more money than Chantal* *Chantal has less money than Eloise* *Chantal has fewer children than Eloise* *Chantal has as many children as Eloise* *Chantal has as few books as Eloise* *Chantal has as little food as Eloise*

For superlative sentences like *Mary is the youngest child in the family*, we define another type of protoforms: $S(E.A, C.A)$ is V, where S is the abstraction of the superlative relation, E is the abstraction of the distinguishing entity, *Mary*, being compared, A is an abstraction of the attribute *age*, considered in the comparison, C represents the collection or set of entities (*child in the* family) involved in the comparison, and V is the comparison value (*youngest*). The same type could be applied to sentences like *Mary is the most intelligent child in the family* (V: *most intelligent*), and *John is the least intelligent in the office* (V: *least intelligent*, C: *in the office*).

If the superlative sentence involves an action verb like in *Mary works the hardest of all the students*, then the type of protoform will be: $S(E.D, C.D)$ is V, where S is the

abstraction of the superlative relation, *E* is the abstraction of the distinguishing entity, *Mary*, being compared, *D* is an abstraction of the action verb, *works*, considered in the comparison, *C* represents the collection or set of entities (*of all the students*) involved in the comparison, and *V* is the comparison value (*hardest*). It could be also applied in cases like *Mary travels the most frequently in the classroom* (*V*: *most frequently*), and *John travels the least frequently of all his friends* (*V*: *least frequently*). A short resume of the 2 main types of protypical forms that represent superlative sentences is shown in Table 33

Table 33 Prototypical forms of superlative sentences

Protoform	**Example**
S(*E.A*, *C.A*) is *V* *S* superlative relation, *E* entity, *A* attribute *C* collection of entities *V* value	*Mary is the youngest child in the family* *Mary is the most intelligent child in the family* *John is the least intelligent in the office*
S(*E.D*, *C.D*) is *V* *S* superlative relation, *E* entity, *D* verb, *C* collection of entities *V* value	*Mary works the hardest of all the students* *Mary travels the most frequently in the classroom* *John travels the least frequently of all his friends*

4.4. Summary

In the previous chapter, we analyzed the idea proposed by Zadeh [249] to introduce Natural Language Computation (Computation with information described in Natural Language): given a description of a perception (i.e. a proposition *p*) in NL, to translate it into a generalized constraint *GC*(*p*), a precisiation of its meaning. Then the generalized constraint *GC*(*p*) is transformed into a protoform *PF*(*p*), an abstraction of *GC*(*p*). After that, based on protoformal deduction rules, it is possible to deduce relevant information.

In order to apply Zadeh's ideas, we consider that it is necessary to identify propositions in NL and to transform them into GC. Our approach is to do this by recognizing noun phrases in a NL document, considering that noun phrases act as constraints, specifying and describing nouns. It is easy to realize that adjectives and adverbs constraint the meaning of nouns while describing them. Therefore, we consider that those noun phrases could be transformed into GC, which could be used later to

deduce new pieces of information. We also consider that copular sentences and comparative sentences define relations that constraint the subject of those sentences. Over the previous chapter, this idea was analyzed in detail and formal constraining relations between nouns and adjectives and adverbs were proposed. Other formal relations were defined to express the constraints expressed by copular and comparative sentences.

Once we have formalized those constraints present in noun phrases, copular sentences and comparative sentences, we analyzed their deep semantic structure and summarize them via prototypical forms. In this way, we consider that we have contributed to move forward a little bit more into this new field of NL-Computation.

Chapter 5

Implementation aspects

5. Implementation aspects

In the following part, general implementation aspects about the fuzzy models developed for Information Retrieval (IR) will be introduced. One of the models implemented allows measuring the presence of concepts in documents based on the concepts of synonymy and polysemy. This approach infers concept references by weighing the concurrence of synonymous terms in a document. The software can measure the similarity between documents based on the presence of concepts on those documents.

Another fuzzy model for IR that will be presented here is based on measuring the presence of the words included in glossary definitions in order to infer the presence of the corresponding concept or meaning. This model includes the formulas 39 to 40. The software, as well as the previous one, can also measure the similarity between documents based on the presence of concepts on those documents.

Some other programs that will be explained in this part are also associated with the semantic interpretation of the information contained in natural language documents. Those programs are related with detecting the presence of certain natural languages constraints in the documents and transforming them into generalized constraints and protoforms.

5.1. Implementation aspects for Information Retrieval Fuzzy Models.

Web search requires methods which take into account semantic aspects of the information contained in documents and web pages. In this chapter, a set of programs called FASPIR (Fuzzy Approach for Synonymy and Polysemy for Information Retrieval) will be introduced. FASPIR has been developed using SWI Prolog [226] in order to process document collections in natural languages and to identify the most important concepts referenced into those documents. It include a set of programs developed in order to calculate the *concept frequency* of a meaning in some document based on equations previously introduced from 24 to 33 and to calculate the similarity between documents using equations 34 to 38. Those formulas take into account the relations of synonymy and polysemy existing between terms which appear in the documents.

FASPIR software system has been developed using the XPCE's kernel of SWI Prolog. This kernel is an object-oriented engine that allows for the definition of applications as XPCE-classes with their methods defined in Prolog. Therefore FASPIR is structured in a hierarchy of classes and objects which allow access using a graphic interface with the user.

FASPIR programs use WordNet dictionary in order to recognize the terms included in the documents as well as the synonymy sets (WordNet synsets) in order to determine which terms are synonyms. Based on those synsets is also calculated the number of different meanings of a term and its polysemy degree.

Currently there exist different document collections in natural language such as TREC[82] (Text REtrieval Conference), CLEF[83] (Cross-Language Evaluation Forum), MultiSemCor[84]. Those collections provide an infrastructure for the testing, tuning and evaluation of information retrieval systems and test-suites of reusable data which can be employed by system developers for benchmarking purposes. They allow comparing different software programs and are a reference point for everyone currently working in this area.

FASPIR was proved using CLEF 2006 collection, which documents are in SGML format. CLEF 2006 offered a 1.35 millions of documents in different languages with a volume of 3.6 gigabytes of text. The English language collection includes about 23,924 documents, which correspond to news appeared in LA Times in 1994 and Glasgow Herald in 1995. LA Times collection has a volume of 150 megabytes distributed over 311 files whereas Glasgow Herald collection has a volume of 420 megabytes distributed over 365 files. Each file corresponds to the news published in a day.

FASPIR system has been organized as a sequence of successive steps (see Figure 16) in order to maximize the utilization of memory space in each step, because of the WordNet dictionary occupies too much memory. The program steps are:

a) File input

b) Filtering source terms

c) Counting term occurrences

[82] http://trec.nist.gov/
[83] http://www.clef-campaign.org/
[84] http://multisemcor.itc.it/

d) Collecting meanings

e) Calculating concept frequency (*Cf*)

Taking into account that FASPIR should be capable to process collections in different formats, FASPIR first step includes an interface which transforms the format of the document into text format. This way, the rest of the FASPIR programs are not involved with the original format of the document. It was decided to use plain text as the internal representation because the documents are not supposed to have labels or some other information about their content.

The interface program receives as input two directory paths: one of those directories includes the collection of CLEF files in SGML format, which has to be processed and the other is the output directory i.e. the directory which stores the text files produced by the first step. It should be noted that each SGML file usually contains several documents, therefore they are split into several text files, each one including just one document. The program includes a lexical analyzer which allows recognizing the different SGML labels and structures and identifies the various documents included in the source file. Each document, according to the CLEF format, contains an alphanumeric identifier, a title and a text, between some other fields. The program stores the title and the text into the file, using the identifier to generate the file name.

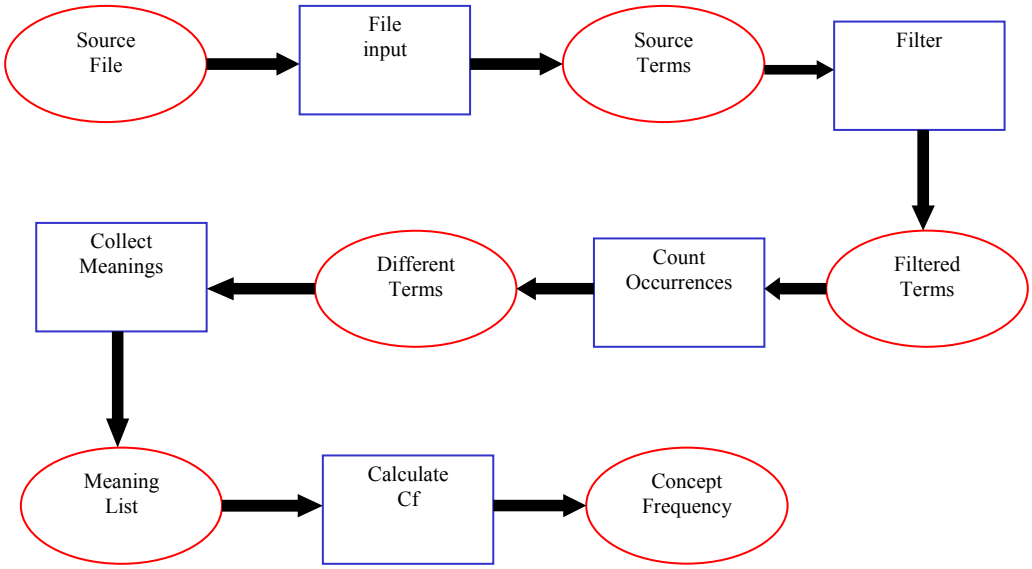


Figure 16 sequential organization of FASPIR software

Once the documents are stored in the same directory in text format, they are processed independently, identifying each one of the terms, deleting punctuation signs

and character strings which do not belong to WordNet vocabulary because they do not have an associated meaning. Numbers, articles, some adverbs, plural forms, ing endings, ed endings, etc are also eliminated from the original document obtaining a filtered set of terms, such that each term do exist in WordNet vocabulary.

WordNet is a large lexical database of English [121, 122] developed at Princeton University[85] from 1985. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. WordNet 3.0 vocabulary contains 155287 unique strings (terms) and 117659 synsets, for a total of 206941 word-sense pairs with around 12 megabytes in compressed format.

Two kinds of building blocks could be distinguished in the WordNet source files: word forms and word meanings. Word forms are represented in their familiar orthography; word meanings are represented by synonym sets (synsets) – lists of synonymous word forms that are interchangeable in some context. Two kinds of relations are recognized: lexical and semantic. Lexical relations hold between word forms; semantic relations hold between word meanings.

The filtering program uses ET [38], an efficient tokenizer written in ISO standard Prolog and fully compatible with SWI-Prolog in order to break the document up into words and/or other significant units. Each token is tagged as word, number, or special character.

The stemming process (i.e. the process for reducing inflected words to their stem, base or root form) is accomplished by using the `library porter_stem` which implements the algorithm described by Porter [161] and implemented for SWI Prolog by J. Wielemaker[86].

Once the terms have been filtered, the program proceed to calculate the number of times that each term appears and its frequency with respect to the total number of terms. After that, some statistics related with the number of occurrences of the terms as mean, variance, minimum and maximum are calculated.

[85] http://wordnet.princeton.edu/
[86] http://www.swi-prolog.org/packages/nlp.html

Then the meanings associated with each one of the different terms are recollected by using the WordNet synsets. And the frequency of use of each meaning is calculated according to equation 33.

After the previous step, a file is produced for each document processed containing all the concept frequencies related with the terms contained in the document. These files are used subsequently to calculate the degree of similarity between the different documents contained in the collection, producing a similarity matrix corresponding to all the documents. This final step is executed using a set of procedures developed using MATLAB 5.3

In order to process the documents, several classes were developed using XPCE toolkit [227, 228] to represent documents (class C_Document), terms (class C_Term) and meanings (class C_Meaning). Each one of these classes contains methods for:

- Initializing class objects
- Showing object attributes by the computer monitor
- Saving object information in files
- Destroying objects

A graphical user interface was developed to facilitate the interaction with the software. The interface includes interactive menus specific for this program. XPCE features for managing collections of objects were used to implement document collections, term collections and meaning collections. For example, the method **show** of class C_Document displays information about the document. It collects the name of the document, as well as the number of source terms, filtered terms, and different terms and show them via a dialog window which include several buttons and titles, one for each one of the available options (see figure)

```
show(DocPt):->
    get(DocPt, name, Name),                    %gets document name
    get(DocPt, n_source_term, N_source_term), %gets the number of source terms
    get(DocPt, n_term, N_term),                %gets the number of filtered terms
    get(DocPt, ndif_term, Ndif_term),          %gets the number of different terms
    new(Dialog, dialog('Document Information')), %opens a dialog window
    send_list(Dialog, append,                  %with several buttons and titles
    [
     new(_, text_item(name, Name)),
     new(_, text_item(n_source_term, N_source_term)),
     new(_, text_item(n_term, N_term)),
     new(_, text_item(ndif_term, Ndif_term)),
     button(source_terms,message(@prolog,send,DocPt,show_source_term)),
     button(terms,message(@prolog,send,DocPt,show_terms)),
     button(different_terms,message(@prolog,send,DocPt,show_dif_terms)),
     button(statistics,message(@prolog,send,DocPt,show_stat)),
     button(object_terms,message(@prolog,send,DocPt,show_obj_terms)),
     button(object_meanings,message(@prolog,send,DocPt,show_obj_means)),
button(quit, message(Dialog, destroy))
    ]),
    send(Dialog, default_button, quit),
    send(Dialog, open).
```

The class C_Document is the most important and complex of all the developed classes, because it keeps control of the whole document processing. C_Document contains methods for

- Reading the document file and obtaining a list of the document terms,
- Filtering the document term list and keeping just the terms recognized by WordNet,
- Calculating the number of occurrences and the frequency of each term in the list,
- Calculating statistics related with term occurrences,
- Collecting the meanings associated with the each one of the terms using WordNet synsets,
- Calculating the concept frequency for each meaning referred in the document

It should be emphasize that the methods initialize, show, save and destroy, which are common for the classes C_Document, C_Term and C_Meaning are polymorphic, that

is, the same message (i.e. show) could be send to two or more objects from different classes and each object will interpret the message in right way, the way implemented by its class. Polymorphism allows a sending object to communicate with other different objects in a consistent manner without worrying about the different implementations of the same message.

For example, if the message **save** is sent to an object of class C_Document

```
send(DocPt, save).
```

the corresponding method **save** of class C_Document will be triggered. This method writes in a Stream the information related with the terms and meanings contained in a document. In order to do that, it opens a File with a name associated with the document's name, associates a Stream to it and writes down the information in the File

```
save(DocPt):->
    write('Saving document information about its terms and meanings.....'), nl,
    get(DocPt, name, DocName),          %gets the document's name
    obj_file(DocName, File_Name),       %gets the file name
    get(DocPt, dest_file_path, Path),   %gets the destination file path
    get(DocPt, n_term, N_term),         %gets the number of terms in the document
    get(DocPt, ndif_term, Ndif_term),   %gets the number of different terms
    get(DocPt, n_means, N_means),       %gets the number of meanings
    path_file(Path, File_Name, PathFile), %joins the file path with the file name
    open(PathFile, write, Str),         %opens a file stream
    save_doc(DocPt, Str, N_term, Ndif_term, N_means), %saves general info.
    get(DocPt, obj_term_list, OTList),  %gets the term list
    save_objs(DocPt, Str, OTList),      %saves the objects in the list
    get(DocPt, obj_means_list, OMList), %gets the meaning list
    save_objs(DocPt, Str, OMList),      %saves the objects in the list
    close(Str).                         %closes the file stream
```

The predicate **save_objs**, just sends the **save** message to each object in the list.

```
save_objs(DocPt,Str,[PtObj|RObjs]):-
    send(PtObj, save(DocPt, prolog(Str)) ),
    save_objs(DocPt,Str,RObjs).
save_objs(_,_,[]).
```

This way, if the object pointer corresponds to a term, the **save** method of class C_Term is fired. This method writes in a Stream the Id of the Term, its number of

occurrences and its term frequency. A similar method is implemented inside the class C_Meaning.

```
save(PtTerm, DocPt,Str):->
    get(PtTerm, term, Term),            %gets the term identifier
    get(PtTerm, term_occur, TO),        %gets the number of occurrences
    get(PtTerm, term_freq, TF),         %gets the term frequency
    objId_docId(DocPt, DocId),          %gets the document identifier
    objId_docId(PtTerm, TermId),        %gets the term identifier
    writeq(Str, doc_term(DocId, TermId, Term, TO, TF) ), %writes it into the file
    write(Str,'.'), nl(Str).
```

The FASPIR system is being used for experimentation by the SMILe group, having obtained satisfactory results which will be shown later in this thesis. This way by using FASPIR approach, although a term does not appear in a document, if another synonym term does appear in the document, the underlying meaning could be detected. The similarity measures between documents improve the results obtained by just measuring term frequency.

5.2. Implementation aspects. Transforming Constraints in Natural Language to Generalized Constraints

The basic idea proposed by Zadeh [249] is: given a description of a perception (i.e. a proposition *p*) in NL, to translate it into a generalized constraint *GC*(*p*) as a precisiation of its meaning. Then the generalized constraint *GC*(*p*) is transformed into a protoform *PF*(*p*), an abstraction of *GC*(*p*) (see Table 34) After that, based on protoformal deduction rules, it is possible to deduce relevant information.

Table 34 Zadeh's basic idea

Natural Language	**Generalized Constraint Language**	**Protoform Language**
p → GC(*p*) → PF(*p*)		

In order to apply Zadeh's ideas, it is necessary to identify propositions in NL and to transform them into GC. Our approach is to do this by recognizing existent constraints in NL and transforming them into GC. As it was shown before noun phrases as well as copular and comparative sentences could be considered constraints that do exist in

natural language. Therefore, in this thesis a software system, NL2GC, is introduced as a first approximation to this goal.

The software NL2GC is divided in several successive independent programs which correspond to

- Analyzing the source document and obtaining a collection of parse trees corresponding with the difference sentences contained into the document,
- Transforming the parse trees into object oriented models reflecting the structure of the different parts of the sentences: clauses, phrases, words.
- Interpreting and extracting the information contained on each sentence

The software is oriented to recognize noun phrases and other types of phrases in a NL document. Then, those phrases are transformed into GC, which could be used later to deduce new pieces of information. The software could also recognize and transform copular and comparative sentences.

5.2.1. Transforming phrases and sentences into parse trees

Definite Clause Grammars (DCGs) have proven useful for describing natural and formal languages, and can be conveniently expressed and executed in Prolog. This programming language was born of a project aimed at processing natural languages, while Colmerauer [34, 36, 35] was working on natural language understanding, using logic to represent semantics and using resolution for question-answering [156]. Different authors [218, 219, 221, 69] have reported using parsing models for natural languages based on the DCG concept.

Difference lists are powerful tools for parsing applications, in which the input and the output stream are represented by lists. DCG provides syntax for writing more readable grammar parsing rules, without including the linked difference lists, and looks like normal Prolog.

In [176], Russell and Norvig presented a formal grammar for a small fragment of English that was taken as a first approximation to develop the grammar that will be explained in the following parahraphs. Based on this grammar, a program which was denominated Sentence Identifier Program (SIP) was developed. This program is capable to read and process documents in text format, to recognize and extract the sentences included on them, as well as the noun phrases included in those sentences. It is

important to say that this program assumes that the documents are well written, that is, it does not verifies if there is agreement between the subject and the verb in a sentence, nor verifies the punctuation, and other probable written mistakes. It is also not conceived to generate sentences based on the grammar.

The lexicon or list of allowable words used in SIP comes from WordNet. The words were grouped into categories (see Table 35) like common and proper nouns, verbs, adjectives and adverbs, articles, contractions and prepositions.

Table 35 Dictionary entries

Word class	Entries
Common nouns	85525
Proper nouns	33681
Verbs	11550
Adjectives	22681
Adverbs	4772
Contractions	128
Prepositions	141

Furthermore, there were introduced rules in order to recognize:

- Multiword names, including abbreviations for social titles (Mr., Mrs., Mme., etc.), professional titles (Dr., Prof., Atty., etc.), political titles (Pres., Sen., Gov., etc.) among others.
- Plural form of nouns, including regular form, special cases for nouns ending with -ch, -sh, -x, -s,–y or –f like taxes, bushes, flies, lives and irregular plurals like roofs, potatoes, parentheses, children, deer.
- Different types of pronouns (see Table 36) and adjectives (see Table 37).

Table 36 Pronoun rules

Pronoun	Distinctions	Cases
Personal	person, gender, form, subjective case, objective case	16
Possessive	person, number gender	8
Reflexive	person, number gender	8
Demonstrative		4
Interrogative		9
Relative		8
Indefinite		19

Table 37 Adjective rules

Adjective	Cases
Possessive	7
Demonstrative	4
Interrogative	9
Indefinite	19
Comparative	Regular + 21 (irregular)
Superlative	Regular + 21 (irregular)

The SIP software just recognizes simple sentences according to the classification given by Kolln and Funk [87], where the sentence is composed by a subject, generally represented by a noun phrase, and a predicate which can be composed by 0, 1 or 2 objects and 0 or more complements.

sentence → noun_phrase, verb_phrase;

Phrases are categorized in five types: noun phrase (NP), verb phrase (VP), adjective phrase (AdjP), adverb phrase (AdvP) and prepositional phrase (PreP). All phrases correspond to a basic three-part structure: the Head of the phrase, the pre-Head and the post-Head strings. The type of the head gives name to the phrase. Therefore, the head of a noun phrase is made up by a noun or a pronoun or several nouns.

```
noun_phrase → pers_pronoun.
noun_phrase → proper_noun.
noun_phrase → common_noun.
noun_phrase → nominals.
noun_phrase → number.
noun_phrase → noun_phrase_preH nominals noun_phrase_postH.
nominals → nominal, nominals; nominal.
nominal → common_noun; proper_noun.
proper_noun → [‘Mary’]; [‘Tom’]; …
common_noun → [children]; [book]; [bus];….
```

The pre head of a noun phrase could include one or more determiners as well as an adjective phrase. The post head of a noun phrase could include prepositional phrases and non finite clauses and relative clauses.

```
noun_phrase_preH → determiners adj_phrase; adj_phrase.
noun_phrase_postH → prep_phrase; non_finite_clause; relative_clause.
determiners → pre_determiner central_determiner post_determiner.
pre_determiner → [all]; [both]; mult_exp; fraction.
mult_exp → [twice]; [double]; …./* twice times, double times, ten times*/
fraction → [half]; …../* a half, one quarter, a third, one third*/
central_determiner → def_article; indef_article; possessive_det; demonstrative_det.
def_article → [the].
indef_article → [a]; [an].
possessive_det → [my]; [your]; [his]; [her]; [its]; [our]; [their].
demonstrative_det → [this]; [that]; [these]; [those].
post_determiner → cardinal; ordinal; gral_ordinal; non_spec_quantity.
cardinal → [one], [two], [three], …..
ordinal → [first]; [second]; [third]; ….
gral_ordinal → [next]; [previous]; [last]; [first]; [subsequent]; ….
non_spec_quantity →
    [many]; [several]; [few]; [all]; [any]; [some]; [both]; [many]; [each]; [every];
    [enough];…
pronoun(subjective) → [I]; [you]; [he]; [she]; [it]…
pronoun(objective) → [me]; [you]; [him]; [her]; [it];…
```

The head of an adjective phrase is an adjective while the head .of an adverb phrase is an adverb.

```
adj_phrase → adv_phrase, adjectives; adjectives.
adjectives → adjective, adjectives; adjective.
adjective → [young]; [old]; [tall]; [short]; [high]; [low]; ……
adv_phrase → adverbs.
adverbs →adverb, adverbs; adverb
adverb →very; extremely;
```

The head of a verb phrase is a verb or a verb chain followed by zero or more objects or complements.

```
verb_phrase → copula, adjective.
verb_phrase → copula, adverb.
verb_phrase → copula, noun_phrase.
verb_phrase → verb.
verb_phrase → verb, complements.
complements → complement, complements; complement.
complement →
    adj_phrase; adv_phrase; prep_phrase; noun_phrase;
    subord_word_clause; non_finite_clause.
prep_phrase → preposition, noun_phrase(objective).
```

Clauses should contain at least a verb phrase. Clauses are either finite or nonfinite, depending on the Verbs they contain. Finite verb phrases carry tense (present tense or past tense), and the clauses containing them are Finite Clauses. Nonfinite verb phrases and clauses do not carry tense. Their main verb is either to-infinitive, bare infinitive, -ed form or -ing form. Subordinate clauses may be finite or nonfinite. Many subordinate clauses are named after the form of the verb which they contain: To-Infinitive Clause, Bare Infinitive Clause, -ing Participle Clause, -ed Participle Clause.

```
non_finite_clause →
    ing_form_clause; ed_form_clause; to_inf_form_clause.
ing_form_clause → ing_form, noun_phrase.
ed_form_clause → ed_form, noun_phrase.
to_inf_form_clause → to_inf_form, , noun_phrase.
relative_clause →
    relative_pronoun, verb_phrase;
    relative_pronoun, noun_phrase, verb_phrase
that_clause → [that], subordw_clause.
if_clause → [if], subordw_clause.
subord_word_clause → subord_word, subordw_clause.
subordw_clause → noun_phrase, verb_phrase.
```

A verb or compound verb is perhaps the most important part of the sentence. A compound verb is constructed out of an auxiliary verb and another verb. The most common auxiliary verbs are "be," "do," and "have". All tenses, aspects and moods except the simple present and the simple past are formed with auxiliary verbs and modals. In English grammar, tense refers to any conjugated form expressing time, aspect or mood. Compound verbs are identified here as verb chains.

Table 38,

Table 39, and Table 40 are detailed lists of the tenses considered by this program. All the chains referred include negation and contractions.

```
verb → auxiliar_verb; simple_present; simple_past; verb_chain.
auxiliar_verb → am, is, are, have, has
simple_present → /* simple present form */
simple_past → /* simple past form */
verb_chain →
    will_chain; modal_chain; bed_chain; had_chain; have_chain; do_chain;
    did_chain; be_chain.
will_chain → /* future with will*/
modal_chain → /* modal form*/
bed_chain → /* was/were chain */
had_chain → /* had chain*/
have_chain → /* have/has chain */
do_chain → /* do/does chain */
be_chain → /* am/is/are chain */
did_chain → /* did chain */
```

Table 38 present and past tense

was, were	ing			past continuous	I **was studying** English when you called this morning.
				Simple past	Two years ago, I **was** a student.
	not	ing		neg. past cont.	He **wasn't (was not) working** when she arrived.
				Neg. simp. past	Two years ago, I **was not** a student.
had	been	ing		Past perfect continuous.	I **had been studying** English for two years before I moved to Canada
	ed			past perfect	I **had studied** English before I moved to Canada.
	not	been	ing	Neg. past perf. cont.	I **hadn't (had not) been sleeping** for long when I heard the doorbell ring.
		ed		Neg. past perf.	She **hadn't (had not) been** to Rome before that trip.
have/has	been	ing		Present perf. cont.	I **have been studying** English for two years.
	ed			Pres. perf.	I **have studied** English in several Canadian cities.
	not	been	ing	neg. pres. perf. cont.	They **haven't (have not) been studying** for long.
		ed		neg. pres. perf.	She **hasn't (has not) been** to New York.

Table 39 future tense and modal forms

will	have	been	ing		Future perfect continuous 1	I **will have been studying** English for one hour by the time you arrive.
		ed			Fut. perf. 1	I **will have studied** all the verb tenses by the end of today.
	be	ing			fut.cont.1	I **will be studying** English when you arrive today.
	verb				Simple fut.1	I **will help** you study English tomorrow.
	not	have	been	ing	neg. fut.perf.cont.1	She **won't (will not) have been working** for long by 5 o'clock.
			ed		neg. fut. perf. 1	She **won't (will not) have finished** her homework by the time we arrive.
		be	ing		neg. fut.cont.1	They **won't (will not) be living** in Paris this time next year.
		verb			neg. simp.fut.1	He **won't (will not) be** able to come.
modal	have	been	ing		Modal perf. cont.	I **could have been swimming** at the beach instead of working in the office.
		ed			mod. perf.	I **could have swum** at the beach yesterday
	be	ing			mod. cont.	I **could be swimming** at the beach right now.
	verb				mod. simp.	I **could swim** at the beach.
	not	have	been	ing	neg. mod. perf. cont.	I **could not (couldn't) have been swimming** at the beach instead of working in the office.
			ed		neg. mod. perf.	I **could not (couldn't) have swum** at the beach yesterday.
		be	ing		neg. mod.cont.	I **could not (couldn't) be swimming** at the beach right now.
		verb			neg. mod. simp.	I **could not (couldn't) swim** at the beach.

Table 40 going to future

am/is/are	going to	have	been	ing		Future perf. cont. 2	We **are going to have been studying** for three hours.
			ed			Fut. perf. 2	We **are going to have studied** all the chapters by five o'clock.
		be	ing			fut. cont. 2	We **are going to be studying** English next year in Canada.
		verb				Simple fut. 2	I **am going to study** English next year in Canada.
	ing					pres. cont. 2	I **am studying** English now.
						simp. pres.	I **am** a student
	not	going to	have	been	ing	neg. fut. perf. cont. 2	We **are not going to have been studying** for three hours.
				ed		neg. fut. perf. 2	We **are not going to have studied** all the chapters by five o'clock.
			be	ing		neg. fut. cont. 2	We **are not going to be studying** English next year in Canada.
			verb			neg. simp. fut. 2	They're **(are) not going** to invite the Browns.
		ing				neg. pres. cont. 2	They **aren't (are not) coming** this evening.
						neg. simp. pres.	Monika **is** not young
ed						simp. past	Two years ago, I **studied** English in Canada.
						simp. pres.	I **study** English everyday
did	not	verb				neg. simp. past	They **didn't (did not) drive** to work.
do/does	not	verb				neg. simp. pres.	She **doesn't (does not) read** the lesson
do/does	verb					pres. intensive	I **do write**
did	verb					past intensive	I **did write**

The SIP program is organized in three parts (see Figure 17):

- The main part including the program nucleus as well as the DCG rules
- Several independent files which contain rules to obtain noun plural forms, abbreviation and contraction rules, etc.
- Several files containing the WordNet lexicon.

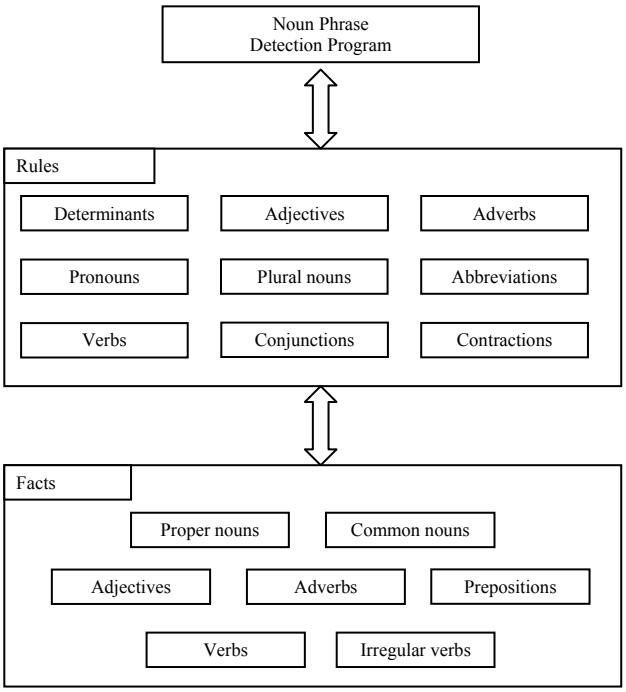


Figure 17General structure of the SIP program

The program was divided into several independent and sequential phases or steps. In order to avoid unnecessary backtracking, every time a token or a substring is analyzed, the results are stored so it is not necessary to re-analyze it later. These kinds of algorithms are called chart parsers [176].

- File to token: read a text file and produce a list with the tokens (words, numbers and punctuation signs) included in it.
- Remove contractions: look for contractions (Let's, aren't, you're, she'll, Who're, there's, you'd, he's among others) and substitute them with the full form (are not, there is, etc).
- Remove abbreviations like Dr., Mr., and Prof. and substitute them with the full form (Doctor, Mister, etc).
- Identify sentences: sentences are delimited by the signs '.' o '?' o '!' corresponding to the main sentence types: declarative, interrogative and exclamative.
- Identify first words: sentences should begin with capital letter, but words in the database usually are in lower case, so first words have to be converted to match.

- Identify tokens: determines if a token, i.e. a word, is a common noun or an adjective, etc. according to WordNet database. Usually the same word could be classified in 2-3 different categories; so, the whole classification is kept for the subsequent phases.
- Identify unknowns: sometimes the ending of a sentence is absent, for example, in titles and other phrases. Therefore, the program does not recognize a sentence ending. In these cases, we apply the same procedure as for identifying first words.
- Identify verb chains: allowed to recognize different verb chains produced by the different verb tenses and modal forms, including negative forms as will have studied, have been studying, could not swim, etc.
- Identify sentences: recognize the different kind of phrase items and collect those forming clauses.
- Save list: save the list of clauses generated from processing the document.

Unfortunately, there are English sentences that are rejected, mostly because unknown words or misinterpretation.

Once a noun phrase is detected, the adverb, adjective and noun can be used to determine the characteristic associated to that noun which is being described by them.

Following there are some examples of noun phrases and sentences analyzed by the program:

His surviving works

noun_phrase([dets(['His']), adjs([surviving]), c_noun([works])])

the lyrical Richard II

noun_phrase([dets([the]), adjs([lyrical]), p_noun(['Richard', 'II'])])

The literary critic A.C.Bradley

noun_phrase([dets(['The']), c_noun([literary, critic]), p_noun(['A.C.', 'Bradley'])])

The psychoanalyst Sigmund Freud

noun_phrase([dets(['The']), c_noun([psychoanalyst]), p_noun(['Sigmund', 'Freud'])])

Sir Arthur Conan Doyle

noun_phrase([p_noun(['Sir', 'Arthur', 'Conan', 'Doyle'])])

The Great Boer War

noun_phrase([dets(['The']), adjs(['Great']), p_noun(['Boer', 'War'])])

An estimated four billion copies

noun_phrase([dets(['An']), adjs([estimated, four, billion]), c_noun([copies])])

I'll be there at nine if I catch the early train.

```
sentence(
        noun_phrase( [pron('I', personal)] ),
        verb_phrase( [ verb( simp_fut( [will, be] ) ),
                        adv_phrase( [there] ),
                        prep_phrase( [ prep( [at] ),
                                noun_phrase( [ nouns( [ c_noun( [nine] ) ] ) ] )
                                ] ),
                        subordw_clause( [ subord_conj( [if] ),
                                noun_phrase( [ pron('I', personal) ] ),
                                verb_phrase( [ verb( [catch] ),
                                        noun_phrase( [
                                                determiners( [the] ),
                                                adj_phrase( [adjectives([ early ]) ]),
                                                nouns( [ c_noun( [train] ) ] )
                                                        ] )
                                                ] )
                                         ] )
                        ] )
                ).
```

The poor woman had no money because she had lost her job.

```
sentence(
        noun_phrase( [
                determiners( ['The'] ),
                adj_phrase( [ adjectives([poor]) ] ),
                nouns( [ c_noun( [woman] ) ] )
                        ] ),
        verb_phrase( [ verb( [had] ),
                adv_phrase( [no] ),
                noun_phrase( [ nouns( [ c_noun( [money] ) ] ) ] ),
```

```
                subordw_clause( [ subord_conj( [because] ),
                        noun_phrase( [ pron( she, personal ) ] ),
                        verb_phrase( [ verb( past_perf( [ had, lost ] ) ),
                                adj_phrase( [ adjectives( [ her ] ) ] ),
                                noun_phrase( [ nouns( [ c_noun( [ job ] ) ] ) ] )
                                        ] )
                                ] )
                        ] )
        ).
```

He is much younger than his brother

```
    sentence(
            noun_phrase( [ pron('He', personal ) ] ),
            verb_phrase( [ verb( simp_pres( [ is ] ) ),
                    adj_phrase( [
                            adv_phrase( [ much ] ),
                            adjectives( [ younger ] )
                                    ] ),
                    prep_phrase( [ prep( [ than ] ),
                            noun_phrase( [
                                    determiners( [ his ] ),
                                    nouns( [ c_noun( [ brother ] ) ] )
                                            ] )
                                    ] )
                    ] )
            ).
```

I know that Mary bought the dog.

```
    sentence(
            noun_phrase( [ pron( 'I', personal ) ] ),
            verb_phrase( [ verb( [ know ] ),
                    subordw_clause( [ subord_conj( [ that ] ),
                            noun_phrase( [ p_noun( [ 'Mary' ] ) ] ),
                            verb_phrase( [ verb( [bought]),
                                    noun_phrase( [
                                            determiners( [the]),
                                            nouns( [ c_noun( [dog] ) ] )
                                            ] )
                                    ] )
                            ] )
                    ] )
            ).
```

The fact that you are here is reassuring to me.

```
sentence(
        noun_phrase( [
                determiners( ['The']),
                nouns( [c_noun( [fact])] ),
                rel_clause( [ rel_pron(that),
                        noun_phrase([pron(you, personal)]),
                        verb_phrase( [ verb(simp_pres( [are])),
                                adv_phrase( [here])
                                ] )
                        ] )
                ] ),
        verb_phrase( [ verb(pres_cont([is, reassuring])),
                prep_phrase( [ prep( [to]),
                        noun_phrase( [pron(me, personal)])
                        ] )
                ] )
        ).
```

Her father died when she was eleven years old.

```
sentence(
        noun_phrase( [
                determiners( [ det('Her',cent) ] ),
                nouns( [ c_noun(father) ] )
                ] ),
        verb_phrase( [ verb(died),
        subordw_clause( [ subord_conj(when),
                noun_phrase( [ pron(she, personal) ] ),
                verb_phrase( [ verb( simp_past(was) ),
                        noun_phrase(
                                adj_phrase( [ adj(eleven, general) ] ),
                                [ nouns( [ c_noun(years), c_noun(old) ] )
                                ] )
                        ] )
                ] )
        ] )
        ).
```

5.3.2. Transforming the parse trees into object oriented structures

A parse tree expressed in linear form can be easily transformed into an equivalent structure based on objects and classes. This way, having an object oriented structure (O-O S) representing the parse tree makes easier to process it. The original document is in this way, transformed into a sequential list of O-O S, one of them per each sentence.

Each O-O sentence structure includes other O-O structures corresponding to phrases, clauses, and words.

Therefore, a hierarchy of classes associated with the different parts (clauses, phrases, words) of a sentence was defined.

The class hierarchy includes a class C_Document with the corresponding list of sentences that compose the document and some other information about the document as the corresponding file identification and other general information. To transform the document sentences from the linear form format to the object format, a file with the sentences is conveniently supplied to an object of class C_Document. This object, for each one of the sentences in the file, creates an object of class C_Sentence and transfers the sentence parse tree to the object.

The class C_Sentence includes the list of the sentence parts, the syntactical type of the sentence (simple, complex or compound) and their classification according to their use (declarative, interrogative, imperative and exclamative). Once an object of class C_Sentence receives a sentence parse tree, it proceeds to recognize its different parts and to create new objects corresponding to each part class and to transfer them a section of the parse tree according to this part. The corresponding object will proceed to recognize the parse tree section and to create new objects when ever it is necessary, transferring them the corresponding subsections. This process will continue recursively until the whole sentence is recognized and transformed.

The class C_Clause includes the clause type (subordinate, non-finite, *to*-infinitive clause, -*ing* participle clause, -*ed* participle clause, etc) and the clause itself.

Phrases constitute a hierarchy of classes with a general class C_Phrase which defines three attributes in common for all the subclasses. Those attributes correspond to the general structure of a phrase: a head, a pre-head and a post head strings. The subclasses C_NounPhrase, C_VerbPhrase, C_AdvPhrase, C_AdjPhrase, C_PrepPhrase inherit this general structure.

Words also constitute a hierarchy of classes with a general class C_WordType with several subclasses C_Noun, C_ProperNoun, C_CommonNoun, C_Verb, C_Adjective, C_Adverb, C_Preposition.

All of these classes include a common type attribute and methods for:

- Initializing an object from the corresponding parse tree segment
- Showing the content of the object in different formats by the screen or saving it into a file
- Interpreting the object

5.3.3. Interpreting sentences

As a result of the previous step, the original document is transformed into a sequential list of O-O structures (LOOS) corresponding to the original sentences. Each one of this O-O S is also composed by others O-O S corresponding to phrases, clauses, and words.

The LOOS is processed by a program that is called the interpreter, because it interprets each sentence, recognizes the entities that are mentioned in the sentences and represents them in a virtual model, called the **context** (**CL**). If a sentence, a phrase or a clause mentions a person for the first time, the interpreter creates an object of type person. Then, as the sentences of the document describe that person, the interpreter associates attributes and characteristics to the object. If the entity is an animal or a thing, the interpreter creates an object of that type and associates attributes to it in the same way.

The virtual entities as the real ones exist in relation with time and places. Therefore, time and places are also objects that do exist in the context in asscoaition with the other entities. The context is just a container as a set to keep the entities together; there is no order between them inside the context.

In a processing task like that, world knowledge and common knowledge are essentials, therefore several knowledge bases as ConceptNet, OpenCyc, WordNet and YAGO support the interpreter task. If some information about the document indicates to use some special known context, then it could be incorporated. In any case, all the information contained into the document should be interpreted just as the opinion of the document author as well as it truthfulness.

The interpreter uses two other auxiliary structures to facilitate its job:

1. A sorted list, denominated **time line** (**TL**), which keeps pointers to the dates already mentioned in the document.

2. A stack which keeps control of the order in which the entities are mentioned in the document in order to identify them when a pronoun is used. The pronoun and the entity should agree in person, number and gender. The selected entity will be the one nearest to the stack top. The stack is called **references** (**RS**).

The idea of the interpreter is to create a virtual world as a representation of the document, where all the entities of the document could coexist. After creating this virtual world, the entities could be processed and interrogated in order to obtain information about them. In this way, new information could be obtained which were not in the document at the beginning.

The original document represented by the LOOS is processed sequentially, sentence by sentence, clause by clause, phrase by phrase. As they are processed new entities are created, putting them into the context world and the RS. As new information about the entities appear in the LOOS, new attributes and characteristics are defined about the entities mentioned.

Logically the way to process a noun phrase (NP) that appears in the subject is not the same as the way to process a NP that appears as a complement or into a prepositional phrase (PP) or a clause. In order to take into account these differences a different implementation of the interpreter method is defined for each one of those classes: the implementation of the interpreter method is different for a NP, a PP, a clause, etc.

During the interpretation process different relations between entites as described by the sentences are introduced by the interpreter, representing them by facts and clauses also in the CL.

The interpreter processes the sentences one by one, in order according to the LOOS. The interpretation process for each sentence consists on:

1) Traverse the parse tree down to the NPs, which describe the entities involved.

2) If it is the first time that the entity is referenced, then create a new object of this type, and put a pointer to it into the context model and into the stack of references.

3) If the entity was referenced before, then get a pointer to the existing object which represents it.

4) If there are adjectives and adverbs in the NP, transform them as attributes and characteristics of the entity.

5) If there are clauses included into the NP, they should be processed independently. After being processed, a new relation should be created between the entity mentioned in the head of the NP and the other entities mentioned into the clause. The same process should be applied if there is a phrase in spite of a clause.

6) If there are time references into the phrases or clauses, new pointers should be incorporated into the TL

Once the interpreter finishes processing the LOOS, all the information about the entities mentioned in the document should appear into the different auxiliary structures as the CL, the TL, and the RS. Then the information kept into those structures could be used to deduce new information and to answer queries about the entities.

Chapter 6

Experimental results

6. Experimental results

In this chapter, the results obtained using the fuzzy models for Information Retrieval introduced in Chapter 3 will be presented. The mentioned models were integrated with clustering algorithms previously developed by other members of the SMILe group.

The model used by the clustering algorithms for document representation is very important to obtain good results during the clustering process. Traditionally, a document is considered as a bag of words, and each document is represented as a vector of term frequency values. The main problem of these approaches is that they only consider lexicographical aspects and do not consider the semantic relations between words [9]. By means of fuzzy models like FASPIR, a conceptual representation of documents could be used in spite of just a lexicographical one. The results show that the improvement was positive.

In English, noun phrases can be treated as single grammatical units that act as the subject or object of a verb. Therefore, noun phrases could be considered references to entities which play an important role in the sentence. Furthermore, those noun phrases describe the already mentioned entities and constraint its meaning. Then recognizing the noun phrases in a document could be very much helpful in order to recognize which concepts are being referenced on it. In this chapter, the results obtained while extracting constraints from natural language documents are also presented.

6.1. General aspects about Clustering

Clustering could be considered as the unsupervised learning process of organizing objects into groups whose members are similar in some way, such that unobvious relations and structures can be revealed. A loose definition of clustering could be “the process of organizing objects into groups whose members are similar in some way”. A cluster is therefore a collection of objects which are “similar” between them and are “dissimilar” to the objects belonging to other clusters.

Another kind of clustering is conceptual clustering: two or more objects belong to the same cluster if this one defines a concept common to all that objects. In other words,

objects are grouped according to their fit to descriptive concepts, not according to simple similarity measures.

The main objective of the clustering methods is to organise data into meaningful structures [74]. Data clustering is a common technique for statistical data analysis, which is used in many fields, including machine learning, data mining, pattern recognition, image analysis and bioinformatics. Data clustering is usually regarded as a form of unsupervised learning which can be considered one of the most important problems of this kind. Besides the term data clustering (or just clustering), there are a number of terms with similar meanings, like cluster analysis, automatic classification and numerical taxonomy.

Data clustering deals with finding a structure in a collection of unlabeled data. Clustering is the process of dividing data elements into classes or clusters so that items in the same class are as similar as possible, and items in different classes are as dissimilar as possible. Depending on the nature of the data and the purpose for which clustering is being used, different measures of similarity may be used to place items into classes, where the similarity measure controls how the clusters are formed. Some examples of measures that can be used as in clustering include proximity according to some defined distance measure, connectivity, and intensity.

Data clustering could be classified in *hard* (*exclusive*) and *fuzzy* (*overlapping*) clustering. In hard clustering, data is divided into distinct clusters, where each data element belongs to exactly one cluster. In fuzzy clustering, data elements can belong to more than one cluster, and associated with each element is a set of membership levels. These indicate the strength of the association between that data element and a particular cluster. Fuzzy clustering is a process of assigning these membership levels, and then using them to assign data elements to one or more clusters.

Clustering algorithms may also be classified as *hierarchical* vs. *partitioning*. Hierarchical algorithms find successive clusters using previously established clusters. Hierarchical algorithms can be *agglomerative* (*bottom-up*) or *divisive* (*top-down*). Agglomerative algorithms begin with each element as a separate cluster and merge them in successively larger clusters. Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.

By the other hand, partitioning algorithms determine all clusters at once, classifying data through a number of clusters fixed a priori. For example, the K-means partitioning algorithm randomly place K centers and assigns points to the nearest cluster i.e. nearest center.

Clustering algorithms can also be categorized into two classes based on how the clusters are described: Monothetic algorithms are those in which a document cluster is defined based on a single word, where as polythetic algorithms, each document cluster is described by several words.

Several clustering approaches assume that the appropriate number of groups is known. However, if we want to divide into clusters a set of documents which are obtained as query result, the number of groups can change for each set of documents that result from an interaction with the engine.

Common clustering techniques have the disadvantage that they do not provide good descriptions of the clusters. It is essential build concept hierarchies that expose the different concepts present in the document collection.

Document (or text) clustering is a subset of the larger field of data clustering, which borrows concepts from the fields of information retrieval (IR), natural language processing (NLP), and machine learning (ML), among others. Document Clustering is the process of finding natural groupings in documents. Document clustering will hereafter be simply referred to as clustering.

Clustering involves dividing a set of documents into a specified number of clusters, so that documents within a cluster have high similarity in comparison to one another, but are very dissimilar to documents in other clusters. The cluster hypothesis [222] explains that the relevant documents tend to be more similar to each other and therefore tend to appear in the same clusters. Document clustering methods have been widely applied in Information Retrieval, IR, supported by the hypothesis that documents relevant to a given query should be more similar to each other than to irrelevant documents, so they would be clustered together [268].

The process of clustering aims to discover natural groupings, and thus present an overview of the classes (topics) in a collection of documents. In the field of artificial intelligence, this is known as unsupervised machine learning. In a clustering problem,

neither the number, nor properties, nor membership (composition) of classes is known in advance.

In document clustering the usability of the classical approaches is limited by several shortcomings. Partitioning algorithms are famous for their simplicity and efficiency in large data sets clustering. However, these algorithms have one shortcoming: the clustering result is heavily dependent on the user-defined variants [210], i.e., the selection of the initial centroid seeds and the number of clusters (k). These clustering approaches assume that the appropriate value of the number of clusters is known. However, when the distribution of the data set is unknown, the optimal k is difficult to obtain.

Hierarchical clustering algorithms provide a representation with different levels of granularity, good for visualizing and browsing through document collections. The main shortcoming of these algorithms is their high complexity that increases with the number of documents [269]

Document representation model is very important to obtain good results in the clustering process. Traditionally, a document is considered as a bag of words, once the stop words have been removed. Term frequency is a statistical measure often used in IR to evaluate the relative importance of a word in a document. Term frequency is the number of times the word appears in a document divided by the total number of words in it. Each document is represented as a vector of term frequency values. The main problem of these approaches is that they only consider lexicographical aspects and do not consider the semantic relations between words [9].

6.2. Using a Hybrid Model for Document Clustering

In [170] a new model to document clustering based on a conceptual representation of documents was introduced. To solve the several shortcomings of classical clustering algorithm a soft approach to hybrid model is used [171]. The model employs a hybrid clustering algorithm to obtain good quality results without loss of effectiveness. In that occasion, they used the FIS-CRM model [141] for generating the conceptual representation of documents. FIS-CRM uses the fuzzy synonymy and the fuzzy generality relationships between words to obtain a conceptual representation of the document.

The clustering procedure uses two connected and tailored algorithms with the aim to build a fuzzy-hierarchical structure. A fuzzy hierarchical clustering algorithm is used to determine an initial clustering. Then, the process is completed using an improved soft clustering algorithm. Finally, a relevant terms selection process to create a meaningful polythetic representation for the clusters is used. Experiments show that by using this model, clustering tends to perform better than the classical approach.

With the aim of detecting the initial relationships between the documents to be able to later ascertain several conceptual document clusters, an agglomerative hierarchical clustering process is carried out. Every document is initially assigned to its own cluster and then pairs of clusters are repeatedly merged until the whole tree is formed using the similarity function. A modified Repertory Grids like technique [45] is used with different aggregation, density and normalization functions to choose the cluster to merge in the agglomerative process.

The process of merging finishes when the documents are clustered into sets that correspond to different concepts. Therefore, the termination criterion is a threshold on cohesion of the common meaning in the clusters, i.e. a threshold on the similarity measure between documents in a cluster.

In order to improve the quality of the hierarchical results, a specific method for hierarchy fuzzyfication is used. In this way, the cluster cardinality is corrected. The method described in [2] is used, with several changes to fit the method for finding hierarchical features.

This initial result is always the same and the number of initial clusters is optimal, but the obtained organization is not concluding. For the next clustering step a fast algorithm is used, which is able to deal with large datasets, and provide a reasonable accuracy. The result is refined using the SISC [71] clustering algorithm improved in FISS meta-searcher structure [141].

This modified algorithm is characterized by using the obtained clusters on the previous step, followed by an iterative process that moves each document into the clusters whose average similarity is greater than the similarity threshold, which is automatically calculated using the same density and normalization functions defined in the agglomerative step.

The algorithm also considers merging clusters and removing documents for clusters when their average similarities decrease under the threshold. In order to get a hierarchical structure, big clusters and the bag cluster (formed by the less similar documents) are reprocessed with the same method.

All these processes dynamically produce a fuzzy hierarchical structure of groups of "conceptually related" documents. The resulting organization is hierarchical, so, from a large repository of documents a tree folder organization is obtained. The resulting clusters can be considered as fuzzy sets, so each one of the retrieved documents has a membership degree (obtained from an average similarity degree) to each one of these clusters.

The clustering algorithms normally do not use data labels for classification. However, labels can be used for evaluating the performance of partition result. Therefore, the analysis of clustering results and the representation of document clusters is the last step. Then the problem of summarizing the cluster content arises, i.e., a textual description or a set of terms representing the cluster topics must be identified. Good clusters must have concise descriptions.

Each document cluster is represented using its relevant concept set. The relevant concept set is a collection of meanings with an important presence in all the cluster documents. Therefore, meanings should exceed a percentage threshold to be considered important enough to be included in that set. Then, they will be extracted with the algorithm modification explained in [15]. An example of cluster description using the most significant word of a concept explanation is shown in the Table 41. The cluster description is conformed to the most relevant meanings. Each meaning has a most significant word.

Table 41 Example of cluster descriptions.

Tag	Description
trade	dollar (13481061), market (01082610), inflation (13325078), policy (06567622), exchange (01078424)
money-supply	money-supply: loan (13226412), interest (13147070),cash (13214226)
income	price (05084251), economy (08252295), money (13212169) , account(06430339)

6.2.1. Experimental results based on the FASPIR model

In this thesis a new version of the previously mentioned algorithm for hybrid clustering was used in combination with the FASPIR (Fuzzy Approach of Synonymy and Polysemy for Information Retrieval) model [196, 171] explained on Chapter 3. The FASPIR model is introduced in order to manage with conceptual aspects of documents. Those fuzzy formulas were introduced in this thesis in order to calculate the degree of synonymy and polysemy between terms, based on WordNet [121, 122] synonym sets.

To measure the presence degree of a concept in a document (or even in a document collection), a concept frequency formula is used. The concept frequency is calculated as a statistical measure of the relative importance of a concept or meaning in a document. According to concept frequency the similarity degree between documents is estimated [196].

Using the clustering algorithm previously described, the collection of documents is split up in a reduced number of groups made up of documents with enough conceptual similarity. Each group contains one or more relevant meanings that make it different from the rest.

Documents are represented using an extended vector-space model using FASPIR formula $Cf_j(m)$, which estimates the frequency of use of a meaning m in a document j, based on the number of times (n_{ij}) that a term (t_i) associated to this meaning appears in the document.

$$Cf_j(m) = \frac{\sum_{t_i \in T(m)} \left(n_{ij} \cdot \left(1 - I_p(t_i) \right) \right)}{n_{*j}} \quad . \tag{47}$$

By calculating $Cf_1(m)$ and $Cf_2(m)$ for all meaning m, two vectors Cf_1^M and Cf_2^M are obtained. Based on them, it is easy to measure the degree of similarity between both documents by equation 48, which provides a fuzzy way to calculate it. The similarity relation between both documents is defined using the well known cosine similarity coefficient [120]:

$$similar^{M}(D_1, D_2) = \frac{\sum_{m \in M} (Cf_1(m) * Cf_2(m))}{\sqrt{\sum_{m \in M} Cf_1(m)^2 * \sum_{m \in M} Cf_2(m)^2}} \quad (48)$$

Evaluation of the hierarchy quality generated by a particular algorithm is an important and non-trivial task. Following the data sets used for performing the practical experiments are described. The experimental results are also presented. Documents were clustered using the Hybrid Clustering algorithm explained before. After the clustering process, documents are labelled with their relevant terms, identifying in this way the document classes. Then comparing the actual classes with the found classes, the quality measures can be obtained.

Some of the most relevant evaluation measures [234] were used to compare and analyse the performance of clustering methods.

- Measures of the document representation method:
  - Mean Similarity (MS): Average of similarity of each element with the rest of the set.
  - Number of Outliers (NO): An outlier is an object that is quite different from the majority of the objects in a collection.
- Measures of the clustering results:
  - Internal quality measures that depend on the representation
    - Cluster Self Similarity (CSS): the average similarity between the documents in a cluster.
    - Size of Noise Cluster (SNC): number of elements unclassified in the hierarchical structure.
  - External quality measures based on a known categorization.
    - F Measure [94]: combines the precision (p) and recall (r) values from IR [217, 86].

The F measure of cluster *j* and class *i* is given by:

$$F(i,j) = \frac{2 * r_{ij} * p_{ij}}{r_{ij} + p_{ij}}. \qquad 49$$

For an entire cluster hierarchy the F measure of any class is the maximum value obtained at any node in the tree. An overall value for the F measure is computed by taking the weighted average of all values for the F measure as follows, where n is the number of documents and the maximum is calculated over all clusters at all levels:

$$F = \sum \frac{n_i}{n} * \max\{F(i,j)\}. \qquad 50$$

A pre-classified set of documents is necessary in order to evaluate the effectiveness of the method. SMART[87] and Reuters[88] collections which are widely used in other publications and reflect the conditions in a broad range of real life applications will be used here. Experimental details are shown in Table 42:

- The SMART collections contains
  - 1400 CRANFIELD documents from aeronautical systems papers,
  - 1033 MEDLINE documents from medical journals and
  - 1460 CISI documents from information retrieval papers
- The Reuters text categorization test collection Distribution 1.0 [LEW00] consists of 21578 articles from the Reuters news service in the year 87. All documents from the specified categories according to the "ModApte" split [APT94] were used

Table 42 Technical Note of the Experiment

Property	SMART	REUTERS
Number of documents	3893	8654
Number of meanings	5068	12594
Number of ranked meanings	1062	1254

[87] ftp://ftp.cs.cornell.edu/pub/smart
[88] Lewis, D.: Reuters-21578 text categorization text collection 1.0. http://www.research.att.com/~lewis.

The results obtained by this model are compared with those obtained by the classical methods, such as the tf-idf representation method [181] and the fuzzy c-means clustering algorithm [152].

The experimental results are shown in the Table 43, expressed in percent. In the first part of the table are grouped the results corresponding to metrics of type "higher is better". Then, in the second part, are grouped the results corresponding to metrics of type "lower is better". Figure 18 shows a graphic with the results.

Table 43 Experimental Results.

	TF-IDF & FCM	Hybrid Model	TF-IDF & FCM	Hybrid Model
Metric	**SMART**	**SMART**	**REUTER**	**REUTER**
MS	37	49	29	45
CSS	24	55	22	43
F-measure	43	63	45	54
NO	22	10	25	15
SNC	15	8	28	10

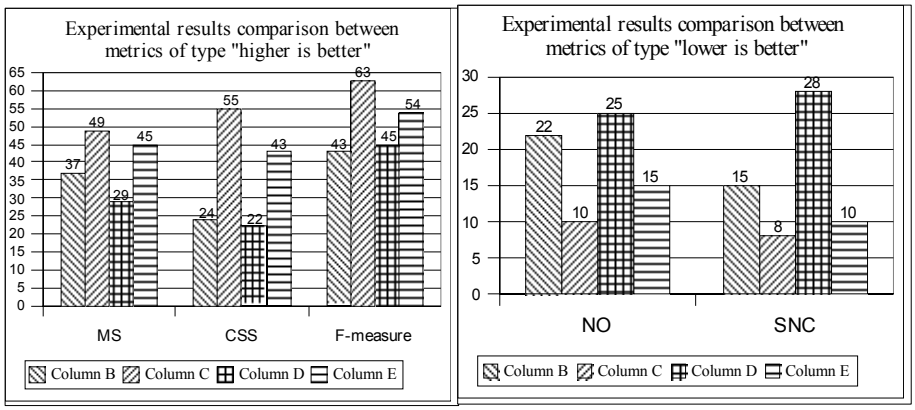


Figure 18 Experimental results comparison

6.2.2. Experimental results based on Concept Measuring in Documents

Another version of the Hybrid Clustering algorithm explained before was used in order to manage with conceptual aspects. To measure the presence degree of a concept in a document, a concept frequency formula is used. This formula is based on

measuring the presence in the document of the terms included in the concept definition and was explained on Chapter 3 [197]

Usually the meaning of a word is explained by a descriptive definition, a statement which captures the use, the function and the essence of a term or a concept. This approach assumes that terms included in the definition of a meaning constitute a set of keywords associated with the meaning essence. For example, the set {motor-vehicle, four-wheels; internal-combustion, engine} describe the essence of an auto. Therefore, measuring the presence of those keywords in a document is a way to measure the presence of "auto". Words meaning is used here to calculate a fuzzy similarity degree according with the representation method.

For our clustering algorithm documents are represented using an extended vector-space model using equation 51.

$$Def_j(m) = \frac{\sum_{t_i \in B(m)} n_{i,j}}{n_{*,j}} \qquad 51$$

Where $Def_j(m)$ is a function over the interval [0, 1], which could be interpreted as the membership function for the fuzzy set of meanings which are present in a document; $B(m)$ the set of terms that describe the essence of a meaning m, n_{ij} is the number of occurrences of term t_i and n_{*j} is the total number of occurrences of terms in the document D_j.

By calculating $Def_j(m)$ for all te meanings m, two vectors Def_1^M and Def_2^M are obtained. Given those vectors, it is possible to employ once more the cosine similarity to calculate the similarity between both documents.

Evaluation of the hierarchy quality generated by a particular algorithm is an important and non-trivial task. The data set used for performing the experiments and to evaluate the effectiveness of the method was the Reuters data set[89].

Clustering accuracy is used to determine the external quality of the results, which is given by:

[89] D. Lewis Reuters-21578 text categorization text collection 1.0. http://www.research.att.com/~lewis.

$$Ac = N_c / |D| \quad (52)$$

Where:

- Ac: Accuracy degree.
- $|D|$: Total of documents.
- N_c: Documents correctly classified.

The results obtained by this model were compared with those obtained by the classical methods [12], such as the tf-idf representation method [181] and the fuzzy c-means clustering (FCM) algorithm [152]. Experimental results are shown in Table 44. Experiments seem to indicate that use of richer features such as the hybrid model with concept measuring tends to perform better than the classical approaches.

Table 44 Comparative Results

Metric	TF-IDF & FCM	Hybrid Model with Concept Measuring
Accuracy	43%	65%

6.3. Fuzzy Optimized Self-Organizing Maps for Document Clustering

Self-organizing maps (SOMs) are one of the most popular neural network models. Introduced by Kohonen [82], SOM represents the input space using the topological structure of a grid to store neighborhood relations. In contrast with most of the neural networks methods, the main feature of SOM is the usage of unsupervised learning for clustering tasks. This neural network has proven very useful in a wide range of problems, and it is considered especially suitable for clustering large high dimensional data set like images, documents or financial data [3]. In document retrieval, text classification using self-organizing maps is widely used. The most popular project in this scope is WebSOM [92], which is based on a document automatic organization tool with the aim to provide an interactive exploration of document collections. The document collection is analyzed by means of the SOM leading to a word category map. Besides, a document map based on document similarity is obtained. The synergy of

these two components produces an efficient structure for dealing with new incoming documents.

The goal of the learning process algorithm is to adapt iteratively the weights, so the final neurons represent a cluster of data vectors. The basic learning process algorithm is shown in Figure 19 below.

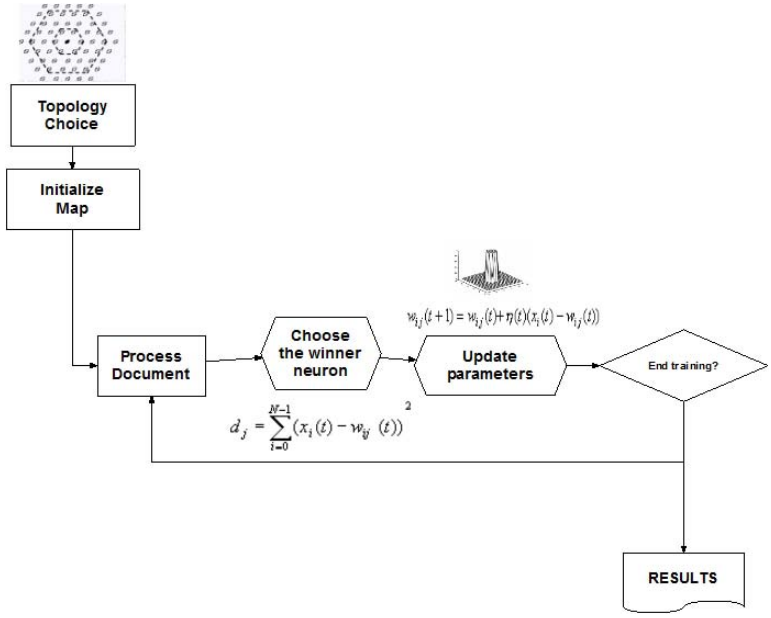


Figure 19 Basic Learning process

Self-organizing major drawbacks are the huge amount of training time, the great volume of resources required for training the document map and the necessity to repeat the process if the number of document increases. Thus, these drawbacks can make difficult the application of SOM to organize large documents collections. Another shortcoming is that the map topology has to be defined by the user. In [137], an approach that tries to adapt the map to the distribution of the underlying data by a growing process is developed.

In [174], some fuzzy logic techniques are used in order to solve the drawbacks presented by the classic SOM algorithm. The aim has been, at first, to reduce the training time (iterations and epochs) and at last, to improve the quality results within the clustering documental scope. Therefore, several steps of the learning process have been modified (Figure 20).

- Using a semantic indexing process, based on FIS-CRM [141] in order to build the input vectors according to the document contents.

- Choosing the number of neurons and the 2D lattice of the output layer that correspond to the collection features.
- Heuristic initialization of the output layer with the purpose of obtaining significant enough prototype groups to reduce the learning stages.
- Some improvements in the learning process: the use of a fuzzy similarity measure and changes in the neighborhood functions and learning rate.
- Group knowledge representation. An advanced representation of each neuron using keywords extractions and fuzzy sets is used.

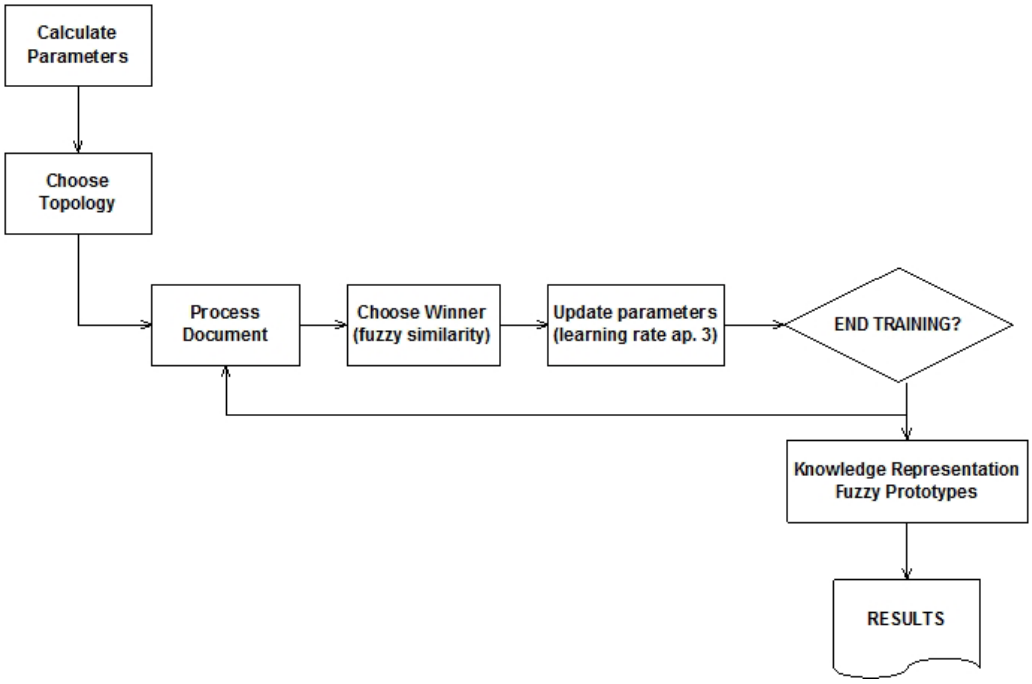


Figure 20 Improved Learning process

6.3.1. Experimental results using Fuzzy Optimized Self-Organizing Maps

The FASPIR (Fuzzy Approach of Synonymy and Polysemy for Information Retrieval) model [196, 171] previously introduced in Chapter 3 has been combined in this thesis with the fuzzy optimized self-organizing maps in order to integrate the improvements of both models.

FASPIR model as was explained before allows managing with conceptual aspects of documents via the concept frequency measure *Cf* (see equation 53), based on the degree of polisemy previously defined:

$$Cf_j(m) = \sum_{t_i \in T(m)} \left(n_{ij} \cdot \left(1 - I_p(t_i)\right)\right) = \sum_{t_i \in T(m)} \left(\frac{n_{ij}}{N_m(t_i)} \right) . \quad 53$$

where n_{ij} is the number of occurrences of term *ti* in the document *dj*.

That way, it is easy to compare the relative importance of different meanings in a document *dj*. This measure is equivalent to term frequency which is one of the most referenced in IR, but for meanings which include some semantic interpretation of the terms.

The relative concept frequency measure *Cfr* of a meaning with respect to a document will be calculated by Equation 54. This formula will be used to estimate the similarity between documents after the disambiguation process.

$$. \; Cfr_j(m) = \frac{Cf_j(m)}{\max_{m \in D}\left(Cf_j(m)\right)} \quad 54$$

Throughout the training process, for each step, the similarity between a document and the neuron map has to be evaluated in order to choice the winning neuron.

In such a vector-space model, the similarity between two text documents corresponds to the distance between their vector representations [64]. The cosine similarity is one of the most commonly used methods to estimate document similarity.

During document clustering, the application of fuzzy similarity functions [30] allows to obtain better results than applying classical functions. In this work a fuzzy similarity function between documents and neurons (see Eq. 55) has been defined taking into account the conceptual interpretation of the vectors and the indications proposed in [224]:

$$. \; sim(d, n_j) = \frac{\sum_{m \in d} Cfr(m, n_j) \otimes Cfr_d(m)}{\sum_{m \in d} Cfr(m, n_j) \oplus Cfr_d(m)} \quad 55$$

where Cfr(m) is the weight of the meaning *m* in the neuron *j*, *Cfrd*(*m*) is the relative concept frequency value of meaning *m* in document *d,* and $\otimes$ and $\oplus$ denote the fuzzy conjunction (t-norm) and disjunction operators (t-conorm), respectively. Using the algebraic product as t-norm and the algebraic sum as t-conorm, the similarity is defined using Equation 56:

$$sim(d, n_j) = \frac{\sum_{m \in d} Cfr(m, n_j) \cdot Cfr_d(m)}{\sum_{m \in d} \left(\left(Cfr(m, n_j) + Cfr_d(m) \right) - \left(Cfr(m, n_j) \cdot Cfr_d(m) \right) \right)} \quad 56$$

This function is used to estimate the similarity between a neuron and a document in order to select the winner neuron. For each document the winner neuron n_{win} is the one with the maximum similarity value.

A pre-classified set of documents is necessary in order to evaluate the effectiveness of the fuzzy optimized self-organized maps in document clustering problems. In this work, the Reuters collection [99] has been used. The training data set with 6015 articles without errors is used to train the self-organized map. And the test data set with 2339 articles is used to compare the obtained results with the theoretical results. The distribution could be observed in Table 45

Table 45 Reuter's collections by categories

Category	Training Docs	Test Docs
earn	2700	1040
acq	1474	641
money-fx	425	128
grain	375	126
trade	317	107
crude	305	305
interest	175	75
ship	141	58
coffee	103	24
Total	6015	2339

In Table 46 the maximum precision, maximum recall and maximum F measure, previously explained, obtained for each group after the experiment are shown.

Table 46 Reuters experimental results.

Neuron	% Elements	Max. Precision	Max. Recall	Max F.
1	13,47	0,46	0,01	0,61
2	0,09	0,50	0,01	0,01
3	12,31	0,45	0,93	0,61
4	0,56	0,62	0,06	0,11
5	0,00	0,00	0,00	0,00
6	3,63	1,00	0,08	0,15
7	29,07	0,84	0,89	0,87
8	0,47	0,91	0,01	0,02
9	40,40	0,90	0,82	0,86

The corresponding aggregated metrics are shown in Table 47.

Table 47 Aggregated results from Reuters experiment.

Max. Precision	Max. F-measure
0,77	0,76

Resulting values are good enough to remark the system performance. The existence of a neuron without any stimulus should be reported although the preliminary study to determine the kind of topology to be used anticipated this situation. By comparing with other SOM approximations, it could be observed that the proposed modifications improved the quality of the results.

Table 48 and Figure 21 show the results obtained after applying different variations of SOM algorithm:

- Basic SOM: based on TF-IDF
- SOM+: Basic SOM applied to the vectors obtained by the proposed model.

- SOM++: SOM+ with the improvements already mentioned (FASPIR model, learning factor, similarity function, etc)

Table 48 Results obtained by different SOM algorithms.

	Basic SOM	SOM+	SOM++
F-measure	0,46	0,53	0,76

Following, the performance of SOM++ algorithm is compared with BBK and CFWMS algorithms:

- BBK: (Bisecting k-means using background knowledge) [70]. This clustering algorithm uses the vector space model and enhances the text representation by adding synonyms and up to five levels of hypernyms for each noun based on the document context using WordNet as ontology.
- CFWMS (Clustering based on Frequent Word Meaning Sequences) [100]. This algorithm uses frequent word meaning sequences to measure the similarity between documents.

Applied to the Reuters test collection, SOM++ presents a better performance than those other two algorithms (see Table 49 and Figure 21).

Table 49 Comparing results.

	CFWS	BBK	SOM++
F-measure	0,57	0,46	0,76

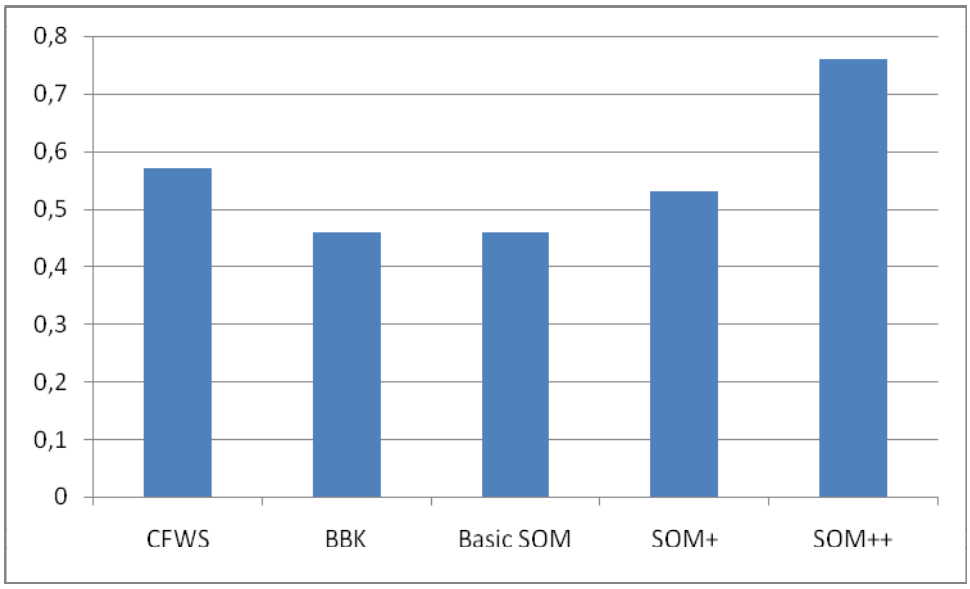


Figure 21 Comparison between different clustering algorithms.

6.4. Experimental Results Extracting Constraints from Natural Languages Texts

The Noun Phrase Detection (NPD) program has been proved with several files in order to check it correctness. At the beginning, the files were built up by collecting sentence examples from English grammar courses corresponding to several universities. Later, looking for more complex and real situations, the files were selected as pages from Wikipedia.com. At that time, we decided to choose Web pages from important writers as Agatha Christie (AC), Arthur Conan Doyle (ACD), Edgar Allan Poe (EAP), H. P. Lovecraft (HPL), and William Shakespeare (WS). Pages were translated to text (.TXT) format, and photos, tables and references were removed.

In Table 50 some basic measures about processing those files are shown. Files were ordered by their size in Kbytes.

Table 50: Basic measures

File	File Size Kbytes	Execution Time sec	Number of Sentences detected	Number of Noun Phrases recognized	Number of unknowns detected
AC	10	24,30	82	513	91
ACD	16	37,61	188	849	108
EAP	29	67,67	291	1616	97
WS	40	95,06	362	2130	193
HPL	61	150,59	464	3148	506

Table 51 shows the rate between the different parameters and the file size. It also shows the arithmetic mean (μ), variance (σ2), standard deviation (σ) and coefficient of variation (COV = σ/μ) obtained from the calculated rates.

Table 51: Analysing file size influence

File	Time / Size sec/Kbytes	Sentences / Size #/Kbytes	Noun Phrases / Size #/Kbytes	Unknowns / Size #/Kbytes
AC	2,43	8,20	51,30	9,10
ACD	2,35	11,75	53,06	6,75
EAP	2,33	10,03	55,72	3,34

WS	2,38	9,05	53,25	4,83
HPL	2,47	7,61	51,61	8,30
μ	2,39	9,33	52,99	6,46
σ^2	0,00	2,67	3,08	5,70
σ	0,06	1,63	1,75	2,39
COV	0,02	0,18	0,03	0,37

Analysing both tables, it could be observed that the rate between file processing time and file size could be considered around 2.4 seconds per Kbytes. The rate between the number of noun phrases detected and the file size could be considered around 53 noun phrases per Kbytes. In both cases, the coefficient of variation is low. For the other two rates, the coefficient of variation, although is still low, indicates a higher variation (over ten times), which may suggest that it depends more on file content.

Taking into account the previous results, we decided to analyze the influence of the number of sentences with respect to the other parameters (see Table 52). In this table, we can observe that the processing time for a sentence is around 0.26 sec per sentence. The number of noun phrases detected per sentence is relatively high, around 5.8. By other hand, the number of unknowns per sentence is near one per sentence, which implies that in almost every sentence there is some grammatical structure that it is not possible to identify. In all those cases, the coefficient of variation is over 10% which may suggest that there is some relation with the file content, maybe because of the redaction style of the Web page author.

Table 52: Analysing the influence of the number of sentence detected

File	Time / Sentences (sec/#)	Noun Phrases / Sentences (#NP/#S)	unknowns / Sentences (#U/#S)
AC	0,296	6,26	1,11
ACD	0,200	4,52	0,57
EAP	0,233	5,55	0,33
WS	0,263	5,88	0,53
HPL	0,325	6,78	1,09
μ	0,263	5,80	0,73
σ^2	0,002	0,72	0,12
σ	0,049	0,85	0,35
COV	0,188	0,15	0,48

After this preliminary study, we decided to verify the success in noun phrases identification of the NPD program. As this verification should be done by hand, we decided to apply statistical sampling techniques in order to reduce the effort required.

A common goal of survey research is to collect data representative of a population. In applying statistics to a problem, it is necessary to define the population to be studied. For practical reasons, rather than compiling data about an entire population, one usually studies a chosen subset of the population, called a sample. Sampling is that part of statistical practice concerned with the selection of individual observations intended to yield some knowledge about a population of concern, especially for the purposes of statistical inference.

Data are collected about the sample in an observational or experimental setting. The data are then subjected to statistical analysis, which serves two related purposes: description and inference. Each observation measures one or more properties (weight, location, etc.) of an observable entity enumerated to distinguish objects or individuals.

Descriptive statistics can be used to summarize the data, either numerically or graphically, to describe the sample by the mean and standard deviation. Inferential statistics is used to model patterns in the data, accounting for randomness and drawing inferences about the larger population. These inferences may take the form of answers to yes/no questions (hypothesis testing), estimates of numerical characteristics (estimation), etc.

The very basic steps for an experiment are:

- to design the experiment, concentrating on the system model and the interaction of independent and dependent variables,
- to use descriptive statistics summarizing the observations collected to feature their commonality by suppressing details

The sampling process consists of 7 simple stages:

1) Definition of the population of concern
2) Specification of a sampling frame, a set of items or events that it is possible to measure
3) Specification of sampling method for selecting items or events from the frame

4) Determine the sample size
5) Implement the sampling plan
6) Sampling and data collecting
7) Review of sampling process

The first step in a sampling process is to define the population which is going to be analyzed. Typically, we seek to take action on some population. A statistical population is a set of entities concerning which statistical inferences are to be drawn, often based on a random sample taken from the population. In our experiments, the population selected was a document file each time. That is, each file from the previous tables was analyzed independently from the others, taking into account that they have different redaction styles, different vocabularies, etc. Therefore, in each experiment, we have a population of N sentences, the total number of sentences detected in the document.

Once defined the population, the sampling frame should be specified. In the most straightforward case, it is possible to identify and measure every single item in the population and to include any one of them in our sample. As a remedy, we seek a sampling frame which must be representative of the population and must have the property that we can identify every single element and include any in our sample. Taking into account that each sentence in our case has one or more noun phrases included, we decided to choose on each sentence just the first noun phrase from the beginning of the sentence i.e. from left to right. These noun phrases usually correspond with the sentence subject, thus they could be considered especially interesting in order to extract information about the subject. Therefore, for each experiment, we have up to N noun phrases to analyze, where N corresponds with the total number of sentences in the document.

A variety of sampling methods can be employed as quota sampling, stratified sampling and so on. Taking into account that in our case there are not categories of sentences inside a document, we decided to use simple random sampling. In this sampling method, all such subsets of the frame are given an equal probability. Each element of the frame thus has an equal probability of selection. The frame is not subdivided or partitioned. The main benefit of simple random sampling is that it guarantees that the sample chosen is representative of the population. This ensures that the statistical conclusions will be valid.

Therefore, we decided to use simple random sampling, so the same probability of selection was assigned to each one of the sentences included in each document D and also to the noun phrases chosen from it. Each sentence and its corresponding noun phrase were identified by its sequential position into the document. After that, a random number generator was used to obtain the sequence of object identifiers to be considered for the sample.

Perhaps the most frequently asked question concerning sampling is, "What size sample do I need?" The sample size of a statistical sample is the number of observations that constitute it. It is typically denoted *n*, and is a non-negative integer.

The answer to the previous question is influenced by a number of factors, including the study purpose, the population size, the sampling error and the risk of selecting a bad sample. In addition to the purpose of the study and population size, three criteria usually need to be specified to determine the appropriate sample size: the level of precision, the level of confidence or risk, and the degree of variability in the attributes being measured.

Sampling is that part of statistical practice concerned with the selection of individual observations intended to yield some knowledge about a population of concern, especially for the purposes of statistical inference. Each observation measures one or more properties (weight, location, etc.) of an observable entity enumerated to distinguish objects or individuals.

Typically, different sample sizes lead to different precision of measurement, as can be seen in the law of large numbers and the central limit theorem, where a larger sample size *n* leads to increased precision in estimates of various properties of the population.

The central limit theorem states that given a distribution with a mean μ and variance σ^2, the sampling distribution of the mean approaches a normal distribution with a mean (μ) and a variance σ^2/N as N, the sample size, increases. The amazing and counter-intuitive thing about the central limit theorem is that no matter what the shape of the original distribution, the sampling distribution of the mean approaches a normal distribution.

Central Limit Theorem: Let X1, X2,… be independent, identically distributed random variables having mean μ and finite nonzero variance σ^2. Let Sn = X1+…+Xn. Then

$$\lim_{n\to\infty} P\left(\frac{S_n - n\mu}{\sigma\sqrt{n}} \le x\right) = \phi(x) \qquad 57$$

where Φ(x) is the probability that a standard normal random variable is less than x.

A typical statistical aim is to demonstrate with 95% certainty that the true value of a parameter is within a distance B of the estimate. Here, B is an error range that decreases with increasing sample size (n). The value of B generated is referred to as the 95% confidence interval.

For example, a simple situation is estimating a proportion in a population as it is our case. To do so, a statistician will estimate the bounds of a 95% confidence interval for an unknown proportion. A practical principle for a conservative B for a proportion derives from the fact the estimator of a proportion, p = X/n, (where X is the number of 'positive' observations) has a binomial distribution and is also a form of sample mean from a Bernoulli distribution [0,1] which has a maximum variance of 0.25 for parameter p = 0.5. So, the sample mean X/n has maximum variance 0.25/n. For sufficiently large n, this distribution will be closely approximated by a normal distribution with the same mean and variance.

The Bernoulli distribution[90] is a discrete distribution with two possible outcomes, corresponding to “success” or “failure”. The successful outcome has an associated probability denoted by p, while the probability for the failure outcome is denoted by q = 1 – p, where 0 < p < 1. A random variable n has a Bernoulli distribution with parameter 0 < p < 1 if

$$P(n) = \begin{cases} 1-p & \text{for } n = 0 \\ p & \text{for } n = 1, \end{cases} \qquad 58$$

where P(n) is the probability of outcome n. The parameter p is often called the "probability of success". The expected value of a Bernoulli random variable X is μ = p, its variance is $\sigma^2 = p*q = p*(1-p)$.

[90] E. W. Weisstein, Bernoulli Distribution, MathWorld A Wolfram Web Resource, 2003
http://mathworld.wolfram.com/BernoulliDistribution.html

The Bernoulli distribution plays a fundamental role in probability theory and statistics. It is a good model for any random experiment with two possible outcomes, for example, yes/no answer (of a respondent in an opinion poll), died/survived (in a drug trial) etc. For example, a single toss of a coin has a Bernoulli distribution with p=0.5 (where 0 = "head" and 1 = "tail"). In our case, we considered a success if the noun phrase selected is completely right and a failure if there is anything wrong with it.

Let consider a Bernoulli trials process with probability *p* for success on each trial. Let *Xi* = 1 or 0 according as the *i*th outcome is a success or failure, and let *Sn* = *X*1 +*X*2 +…+*Xn*. Then *Sn* is the number of successes in *n* trials, which has binomial distribution.

We note that the maximum values of the distributions appeared near the expected value *np*. By subtracting the expected number of successes *np* from *Sn*, the new random variable *Sn – np is obtained*. Now the maximum values of the distributions will always be near 0. Then, we can normalize *Sn - np* to have variance 1 by dividing by its standard deviation *√npq*. The *standardized sum* of *Sn* is given by Equation 59

$$s_n^* = \frac{S_n - np}{\sqrt{npq}} \qquad 59$$

S^*_n always has expected value 0 and variance 1.

Cochran's formula [33] for error estimation uses two key factors:

the error the researcher is willing to accept in the study, commonly called the margin of error, and

the level of acceptable risk the researcher is willing to accept that the true margin of error exceeds the acceptable margin of error; i.e., the probability that differences revealed by statistical analyses really do not exist; also known as Type I error or the alpha level.

There is also another type of error that will not be addressed further here, namely, Type II error, also known as **beta error**. Type II error occurs when statistical procedures result in a judgment of no significant differences when these differences do indeed exist.

The **alpha level** used in determining sample size in most studies is either .05 or .01 In Cochran's formula, the alpha level is incorporated into the formula by utilizing the t-value for the alpha level selected (e.g., t-value for alpha level of .05 is 1.96 for sample sizes above 120). In general, an alpha level of .05 is acceptable for most research. An alpha level of .10 or lower may be used if the researcher is more interested in identifying marginal relationships, differences or other statistical phenomena as a precursor to further studies. An alpha level of .01 may be used in those cases where decisions based on the research are critical and errors may cause substantial financial or personal harm, e.g., major programmatic changes.

The general rule relative to **acceptable margins of error** is that, for categorical data, 5% margin of error is acceptable. For example, a 3% margin of error would result in the researcher being confident that the true mean of a seven point scale is within ±.21 (.03 times seven points on the scale) of the mean calculated from the research sample. For a dichotomous variable, a 5% margin of error would result in the researcher being confident that the proportion of respondents who were male was within ±5% of the proportion calculated from the research sample. Researchers may increase these values when a higher margin of error is acceptable or may decrease these values when a higher degree of precision is needed.

Let X_i , i = 1, 2, ..., n be independent observations taken from a normal distribution with mean μ and variance σ^2 . Let us consider two hypotheses, a null hypothesis H0 and an alternative hypothesis H1:

$$H_0 : \mu = 0$$
$$H_1 : \mu = \mu^* \qquad 60$$

for some small significative difference $\mu^* > 0$, that is, the minimum value for which μ will be considered different to zero (0). Now, if we wish to

- reject H_0 with a probability of at least 1-β when H_1 is true, and
- reject H_0 with probability α when H_0 is true,

Note, this is a 2-tailed test.

Then we need that if z_α is the upper α percentage point of the standard normal distribution, then

$$\Pr\left(\bar{x} > \frac{z_\alpha \sigma}{\sqrt{n}} \middle| H_0 true\right) = \alpha \qquad 61$$

Then, to reject H0 if our sample average is greater than $\left(z_\alpha \sigma / \sqrt{n}\right)$ is a decision rule which satisfies (2).

Now we wish for this to happen with a probability at least 1-β when H1 is true. In this case, our sample average will come from a Normal distribution with mean μ*. Therefore we require that:

$$\Pr\left(\bar{x} > \frac{z_\alpha \sigma}{\sqrt{n}} \middle| H_1 true\right) \geq 1 - \beta \qquad 62$$

Through careful manipulation, this can be shown to happen when

$$n \geq \left(\frac{\Phi^{-1}(1-\beta) + z_\alpha}{\mu / \sigma}\right)^3 \qquad 63$$

where Φ is the normal cumulative distribution function.

There are two types of random variables: categorical and numerical. Categorical random variables yield responses such as 'yes' or 'no' and 'Which day of the week are you most likely to wash clothes?' Numerical random variables yield numerical responses, such as your height in centimetres. In our case, the data type is categorical: each observation (noun phrase) just could be accepted or rejected.

Let consider two groups [41]:

- **Group 1**: our sample, with sample size N1 and number of noun phrases accepted R1.
- **Group 2**: the whole population, with population size N2 and number of noun phrases accepted R2.

Let define Pi = Ri/Ni for both groups. The investigator's hypothesis is that P2 is different from P1. This hypothesis can be stated as a null hypothesis, H0 (i.e., there is no difference between the two proportions), and a statistical test is devised to test that hypothesis. If the null hypothesis is rejected, then the investigator can conclude, at significance level α, that there is a difference between the two proportions. If the null hypothesis is not rejected, then the alternative hypothesis is rejected with the probability that a false-negative of β has occurred. These hypotheses can be stated as follows:

$$H_0 : (P_1 - P_2) = 0$$
$$H_1 : (P_1 - P_2) \neq 0 \qquad 64$$

The formula for determining sample size is derived from a common statistical test for H0. Usually the investigator knows or can estimate the proportion of the group 1, which will have the outcome being observed, and can state a difference between the group 1 and the group 2 that he/she wishes to detect. The smaller this difference, the more observations will be needed.

For populations that are large, Cochran [33, 13, 73] developed equation 65 to yield a representative sample for proportions, where n_0 is the sample size, Z^2 is the abscissa of the normal curve that cuts off an area α at the tails (1 - α equals the desired confidence level, e.g., 95%), e is the desired level of precision, p is the estimated proportion of an attribute that is present in the population, and q = (1 - p).

Suppose that the NPD program will be evaluated. Assume that a 95% confidence level and ±5% precision is desired. Assume also that population is large but that the proportion variability is not known, then maximum variability will be assumed (i.e. p =.5).

$$n_0 = \frac{Z^2 pq}{e^2} = \frac{(1.96)^2 (0.5)(0.5)}{(0.05)^2} = 385 \qquad 65$$

If the population is small then the sample size can be reduced slightly. This is because a given sample size provides proportionately more information for a small population than for a large population. The sample size (n0) can be adjusted using Equation 66. In the example, the number of sentenced detected in the Agatha Christie's Web page was used.

$$n_1 = \frac{n_0}{\left(1 + \frac{n_0 - 1}{N}\right)} = \frac{385}{\left(1 + \frac{385 - 1}{82}\right)} = 68 \qquad 66$$

After some preliminary experiments, we realize that:

The proportion variability of NPD program was not so high, thus maximum variability need not be assumed, and

Repeating the experiment with several document files will improve the results accuracy, while applying it to different samples and populations.

Therefore we decided to use probability P = 0.6, considering that NPD program could obtain more than 60% of successful observations. We also decided to set the alpha level to 0.10, which is acceptable for initial studies.

Then, we proceeded to run the program with the 5 documents (i.e. AC, ACD, EAP, WS, HPL) and to collect the corresponding sample for each one of them. The procedure which generates the random numbers and selects the sample was incorporated to NPD program. It produces an output file with the random sequence and the sample.

After that, each one of the samples was verified by hand carefully. Every observation (i.e. noun phrase selected) that was not truly correct was rejected. The results are shown in Table 53.

Table 53 Experimental results

	Population size N	Sample size n0	Sample Size adjusted n1	# Noun Phrases rejected	# Noun Phrases accepted	% Noun Phrases accepted
AC	82	92,20	43,65	9	36	0,80
ACD	188	80,67	56,66	16	41	0,72
EAP	291	92,20	70,20	17	54	0,76
WS	362	92,20	73,65	18	56	0,76
HPL	464	92,20	77,05	19	59	0,76

As can be seen in Table 53, the number of accepted noun phrases is over 70% and sometimes even over 80%, which means that NPD program has an acceptable performance.

6.5. Summary

The results obtained while using the fuzzy models introduced in Chapter 3 integrated with clustering algorithms has been presented in this chapter. By means of these models, a conceptual representation of documents could be used in spite of just a lexicographical one. The experiments demonstrate that the quality and effectiveness of clustering using this document representation is better than the usual tf-idf representation.

Noun phrases could be considered as single grammatical units that reference the subject or objects of a sentence that is entities which play a very important role. Noun phrases describe the characteristics of those entities constraining them. Therefore, recognizing the noun phrases in a document could be very much helpful in order to recognize which concepts are being used. In this chapter, the results obtained while extracting constraints from natural language documents has been presented with results over 70%.

Chapter 7
Conclusions

7. Conclusions

Nowadays, search engines are able to retrieve efficiently millions of page references in less than a second, but unfortunately, with a low level of efficacy because users receive millions of useless documents, irrelevant for their query. The low level of efficacy strongly depend on the fact that most crawlers just look for words or terms in the documents without considering their meaning

The generalized-constraint-based computational approach to NL-Computation opens the door to a wide ranging enlargement of the role of natural languages in scientific theories. Development of new methods for Web Information Retrieval based on conceptual characteristics of the information is vital to reduce the quantity of unimportant documents retrieved by today search engines.

The SMILe group is deeply involved in the development of IR methods for WWW based on conceptual characteristics of the information contained in documents. This thesis is enclosed into the research lines of this group. It is concerned with considering the conceptual aspects of the information contained in documents in order to improve search engine results. The approaches proposed into the thesis are more oriented to the meaning of the terms and their semantic in spite of the lexicographic aspects as the number of times that they appear in a document.

At the same time, the thesis is concerned with retrieving information from documents which can be used to characterize user preferences and, therefore, to construct a user profile which can be helpful in query expansion to retrieve more relevant information. The thesis is also concerned with retrieving information from natural language documents which can be used to deduce new pieces of information not previously contained into them.

Following we will go through the thesis goals making some comments about their achievement. The first two goals could be summarized as:

- To take into account the semantic of the words in order to obtain a higher level of relevance during searching, retrieval and management of documents in natural language.

- To develop mathematical models based on Fuzzy Logic, which reflect the previous aspects.

Two models which are logical complements of FIS-CRM model have been introduced in this thesis. The first model is oriented to measure the presence of concepts in documents by using fuzzy interpretations of synonymy and polysemy relations.

The other model introduced here is based on measuring the meaning presence in documents by using the bag of words employed in the meaning description. Usually, the meaning of a word is explained by a descriptive definition, a statement which captures the use, the function and the essence of a term or a concept. Here, a new approach to measure the presence in a document of a meaning or concept is proposed, based on the terms included in its definition. This approach assumes that terms included in the definition of a meaning constitute a set of keywords associated with the meaning essence. For example, the set {motor-vehicle, four-wheels; internal-combustion, engine} describe the essence of an auto. Therefore, measuring the presence of those keywords in a document is a way to measure the presence of "auto".

A third model which is a logical consequence of the previous one has also been introduced here. This model is also based on measuring the meaning presence in documents by using the bag of words employed in the meaning description. But the difference is that this method measures the presence of combinations of terms which appears in the definition of a meaning in spite of measuring the terms alone. Moreover, those combinations are noun phrases which appear in the definition and in the document, which give them a semantic connotation.

The following goals could be summarized as:

- To develop computational methods and algorithms which allow implementing the previously mentioned models.

The next goals were

- to develop programs able:
    - To process the information contained in documents collection like SMART and Reuters, taking into account the semantic aspects already mentioned in the first goal.

  - To elaborate indexes based on those semantic aspects which allowed an efficient retrieval of the information contained into the documents.

- To carry out experiments with document collections such as SMART and Reuters, in order to evaluate the developed techniques.

Programs using previously mentioned model were developed to perform a clustering algorithm with the aim to divide a set of documents into a specified number of clusters, so that documents within a cluster have high conceptual similarity. The experiments demonstrate that the quality and effectiveness of clustering using both methods is better than the usual tf-idf representation. Details about the results obtained with those programs are given later in this chapter.

The next goal was

- To identify patterns in sentences and phrases which allow us to represent them by formal relations.

An extensive research was developed about English grammar and the syntax and semantic of sentences, phrases and clauses in English. A high quantity of rules, interpretations, and heuristics was collected and organized. Using all this information, several patterns concerning sentences, phrases and clauses were identified and represented via formal relations.

Knowledge bases as ConceptNet and YAGO has been obtained and incorporated into the environment which support the programs that we are developing.

- To explore the possibility to extract information from documents expressed in natural language (NL) and to deduce new information, information which was not present in the original document.

- To use those formal relations and knowledge bases to deduce new pieces of information in simple situations.

Several situations obtained from web pages has been analyzed and represented by the formal relations defined by us. Later, new information, which was not present before, was deduced applying the rules which govern constraint propagation.

- To investigate the application of Generalized Constraints and Protoforms as defined by Zadeh to represent the previously mentioned relations and to operate with them in order to infer new pieces of information.

Generalized Constraints and Protoforms were applied to represent the previously mentioned formal relations as main ideas in order to define and formalize the relations and to operate with them. The rules which govern constraint propagation were applied in order to deduce new pieces of information.

- To build a bridge between natural languages and mathematics by trying to precisiate the meaning of natural language propositions via constraints.

We consider that the formal relations introduced in this thesis as well as the rules, interpretations, and heuristics collected and organized about English sentences, phrases and clauses contribute to build a bridge between natural languages and mathematics by trying to precisiate the meaning of natural language propositions via constraints. We are clear that: It will be a long, long time before translating any natural language to any version of logic, including controlled NLs, can be fully automated (paraphrasing John Sowa[91]).

CONTRIBUTIONS

In pursuit of the previously mentioned goals, this thesis makes the following contributions:

1. A Fuzzy Approach of Synonymy and Polysemy for Information Retrieval (FASPIR) was defined, which allowed to measure the presence of concepts in documents by using fuzzy interpretations of synonymy and polysemy. Based on the defined formulas, even though a certain term does not appear in a document, it is possible to estimate its presence according to the degree of synonymy shared with terms that do appear in the document.

FASPIR is a logical extension and complement of the FIS-CRM (Fuzzy Interrelations and Synonymy Conceptual Representation Model) [50, 141] model, which has been very successful. FIS-CRM uses a Spanish dictionary, which include about 27 thousands words and several thematic ontologies. Our approach uses an English dictionary, WordNet, which contains about 150,000 words organized in over 115,000 synonymy sets for a total of 207,000 word-sense pairs.

A concept in FIS-CRM is not an absolute concept that has a meaning itself, i.e. there is not any kind of concept definition set or concept index. In FIS-CRM a concept is

[91] J.Sowa, To: cg@conceptualgraphs.org Sat, 14 Jul 2007 09:02:13 -0400

dynamically managed by means of the semantic areas of different words. In our approach, a concept or meaning is the definition of a term that appears in a dictionary, in this case, WordNet. Those meanings define the synsets of WordNet and are used by our approach to manage with the weak words. A polysemy index was defined, which help to share the term occurrences between the different sense.

The main handicap of the sharing process in FIS-CRM is managing with weak words (words with several meanings). Three situations are distinguished depending on the implication of weak or strong words. In the approach presented in this thesis, the introduction of the polysemy index extremely simplifies the management of weak and strong words, incorporating all the three cases mentioned in only one formula.

With the concept frequency coefficient, it is possible to measure how similar are two or more documents depending on their use of some concept. In this approach, this coefficient could also be used to order a document collection in relation with the use made by the different documents of some concept.

2. Another index to measure the presence of concepts in documents was also defined. This coefficient is based on measuring the presence of the definiens (i.e. the terms used in its definition) of certain meaning in one or more documents; thus, it was called DEF coefficient. Based on DEF, even though a certain term does not appear in a document, it is possible to estimate its presence according to the degree of presence of its definiens in the document.

This model is based on measuring the meaning presence in documents by using the bag of words employed in the meaning description. Usually, the meaning of a word is explained by a descriptive definition, a statement which captures the use, the function and the essence of a term or a concept. Here, a new approach to measure the presence in a document of a meaning or concept is proposed, based on the terms included in its definition. This approach assumes that terms included in the definition of a meaning constitute a set of keywords associated with the meaning essence. For example, the set {motor-vehicle, four-wheels; internal-combustion, engine} describe the essence of an auto. Therefore, measuring the presence of those keywords in a document is a way to measure the presence of “auto”.

3. By using FASPIR model and DEF coefficient, it is possible to measure how similar are two or more documents depending on their use of some meaning or

concept. These coefficients could also be used to order a document collection in relation with the use that different documents make of some concept.

4. Programs using both FASPIR model and DEF coefficient were developed to perform clustering algorithms with the aim to divide a set of documents into a specified number of clusters, so that documents within a cluster have high conceptual similarity. The experiments demonstrate that the quality and effectiveness of clustering using both methods is better than the usual tf-idf representation [208, 181].

Using those models a weighted formula is introduced in order to measure the meaning presence in documents. These formulas are later used for a hybrid fuzzy clustering algorithm, which has also obtained good results.

5. In this thesis, the possibility to extract information from documents expressed in natural language (NL) and to deduce new information, information which was not present in the original document is explored.
6. To build a bridge between natural languages and mathematics by trying to precisiate the meaning of natural language propositions via constraints is explored in the thesis.
7. Although natural languages are intrinsically imprecise, we try to identify patterns in sentences and phrases which allow us to represent them by formal relations.
8. We consider that adjectives and adverbs constraint the meaning of nouns while describing them. Therefore, noun phrases could also be considered noun constraining relations.
9. A schematic resume of the structure of noun phrases and the different grammatical components that could appear in each part were obtained. A model to represent noun pharses by several formal relations is proposed in the thesis, once they are identified in a natural language document.
10. Copular sentences are also considered here as meaning constraints of those entities referred to by the subject (i.e. a noun phrase) and the complements (i.e. adjective phrases, adverbial phrases, noun phrases).

11. Copular sentences are the most frequent in English. Half of the basic sentence patterns in English syntax correspond to this kind of sentences. A schematic resume of their structre were obtained. A model to represent those patterns by by several formal relations is proposed in the thesis, once they are identified in a natural language document.
12. Furthermore, comparative sentences are considered a conjunction of two clauses joined together by a comparative formula, which constraints the meaning of those entities referred to by both subjects (i.e. the subjects of the two clauses).
13. A schematic resume of 18 different comparison situations were obtained. A model to represent those patterns by a formal relation with different parameters is proposed in the thesis, once they are identified in a natural language document.
14. Superlative sentences are used to compare more than two things, constraining the meaning of those entities involved in the sentence.
15. A schematic resume of 6 different superlative sentence schemes were obtained. A model to represent those patterns by a formal relation is proposed in the thesis, once they are identified in a natural language document.
16. Using the mathematical relations obtained from noun phrases, copular sentences and comparative sentences, procedures to deduce new pieces of information are described.
17. The role of those formal relations as fuzzy relations and generalized constraints is described in this thesis. The rules which govern constraint propagation are applied to deduce new pieces of information.
18. Protoforms are used here as abstractions of generalized constraints in order to summarize the deep semantic structure of the object to which they are applyied and to represent local reasoning patterns based on the structure of the component propositions.
19. The most frequent in English are the copular sentences. Analyzing the the deep semantic structure of copular sentences we have defined 2 general symbolic expressions which represent an abstract summary of the basic structure of English copular sentences.

20. A schematic resume of 18 different comparison sentences were previously mentioned as a result. According to it, we have defined 3 general symbolic expressions which represent an abstract summary of the basic structure of English comparative sentences.
21. A schematic resume of 6 different superlative sentence schemes were previously mentioned as a result. According to it, we have defined 2 general symbolic expressions which represent an abstract summary of the basic structure of English superlative sentences.
22. The possibility to infer new information from a knowledge base with propositions whose structure matches the premise of a protoform is analyzed in the thesis. The rules which govern constraint propagation are applied to deduce new pieces of information as a generalization of deduction in classical symbolic logic.
23. A Noun Phrase Detection (NPD) program using Definite Clause Grammar (DCG) was developed to recognize noun phrases in documents. The program has been applied to several biographical documents. The program generates the parse tree for those sentences recognized. The experimental results show that the program has detected over 70% of noun phrases with a 95% confidence level and 10% precision.
24. An extension of NPD program for copular sentences and comparative sentences has been also developed. The program has been tested with several examples taken from different web pages with satisfactory results.
25. A program to interpret natural language sentences expressed in parse tree format has also been developed. The program generates several objects which model the entities referred to by the sentences.

OPEN QUESTIONS FOR FUTURE WORK.

The SMILe group is developing a common software platform oriented to integrate the different research results mentioned so far, FIS-CRM, FzMail, GUMSE and so on. The FASPIR and DEF models are being included also in this environment that could be used for information retrieval purposes. Up to the moment these models have not been used for searching information queried by a user, thus it will be a good opportunity to do so. We are planning also to apply these models to store information about user

preferences, clustering it and then creating and maintaining user profiles to be used while disambiguating user queries. It is important also to develop methods which allow combining and integrating the different models.

The generalized-constraint-based computational approach to NL-Computation opens the door to a wideranging enlargement of the role of natural languages in scientific theories [261]. Although it will be a long time before translating any natural language to any version of logic can be fully automated, we think that the approach proposed in this thesis is promising. Many different studies about parsing natural language texts are being developed with promising results. The use of those tools in combination with algorithms to express and represent the knowledge extracted from the texts as formal relations, which could be used later to deduce new information is very much promising. New patterns should be identified and represented by formal relations. Then new deductive procedures concerning those relations should be developed and be integrated with the previous ones. Software tools for representing and operating with these relations should be developed and/or improved. Several ontologies as OpenCyc, ConceptNet, YAGO, and many others more are being developed to model human knowledge (common knowledge or world knowledge). They should be incorporated into the already mentioned tools. Those tools could be used then to support successful question answering systems.

References

References

1. Abberley, D., Kirby, D., Renals, S., & Robinson, T. (1999). The THISL broadcast news retrieval system. In Proc. ESCA ETRW Workshop Acessing Information in Spoken Audio, (Cambridge), 14-19.
2. Akrivas, G., Wallace, M., Andreou, G., Stamou, G., & Kollias, S. (2002). Context - Sensitive Semantic Query Expansion, Proceedings of the IEEE International Conference on Artificial Intelligence Systems (ICAIS), Divnomorskoe, Russia, September 2002.
3. Allinson, N., Yin, H., Allinson, L., Slack, J. (Eds.) (2001). Advances in Self-Organizing Maps, In Proceedings of the Third Workshop on Self-Organizing Maps (WSOM), Springer.
4. Álvez, J., Atserias, J., Carrera, J., Climent, S., Laparra, E., Oliver, A., & Rigau, G. (2008). Complete and Consistent Annotation of WordNet Using the Top Concept Ontology. In Proceedings of the 6th Language Resources and Evaluation Conference (LREC 2008) Marrakech. Morocco.
5. Arms, W. (2000). Digital Libraries, M.I.T. Press, 287 pages.
6. Arnold, D.J., Balkan, L., Meijer, S., Humphreys, R.L., & Sadler, L. (1994). Machine Translation: an Introductory Guide, Blackwells-NCC, London, 200 pages.
7. Baeza-Yates, R. (2000). Desenredando la Madeja, NOVATICA, Special Edition 25th anniversary, may.-jun, Asociación de Técnicos de Informática, 72-77.
8. Baeza-Yates, R. (2003). Information retrieval in the Web: beyond current search engines, International Journal of Approximate Reasoning Vol. 34, Nº 2, Elsevier Inc, 97-104.
9. Baeza-Yates, R., & Ribeiro Neto, B. (1999). Modern Information Retrieval, Addison-Wesley-Longman Publishers, ACM Press, New York, 544 pages.
10. Bainbridge, R.I. (1985). Montagovian Definite Clause Grammar, Proceedings of the Second conference on European Chapter of the Association for Computational Linguistics (ACL), Geneva, Switzerland, 25-34.
11. Barker, C. (2004). Continuations in natural language. In Hayo Thielecke, editor, Proceedings of the fourth ACM SIGPLAN workshop on continuations, 55-64.

12. Barrett, R., & Selker, T. (1995). AIM: A new approach for meeting information needs. Technical report, IBM Research.

13. Bartlett II, J.E., Kotrlik, J.W., & Higgins, C. (2001). Organizational research: Determining appropriate sample size for survey research. Information Technology, Learning, and Performance Journal, 19 (1) 43-50.

14. Baskervill, W.M., & Sewell, J.W. (1896). An English Grammar, Tenn. USA.

15. Beil, F., Ester, M., & Xu, X. (2002). Frequent Term-Based Clustering, Proceedings of the SIGKDD'02, Edmonton, Canada.

16. Bezdec, J.C. (1981). Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, New York.

17. Bird, S. (2006). NLTK: the natural language toolkit, Annual Meeting of the ACL, Proceedings of the COLING/ACL on Interactive presentation sessions, Sydney, Australia, 69-72.

18. Blecua, J.M. (1997). Diccionario Avanzado de Sinónimos y Antónimos de la Lengua Española, Diccionarios de lengua española Vox, Barcelona, 647 pages.

19. Brachman, R.J., & Schmolze, J. (1985). An Overview of the KL-ONE Knowledge Representation System in Cognitive Sci 9(2).

20. Brackenbury, I., & Ravin, Y. (2002). Machine intelligence and the Turing Test, Technical Forum 2002 IBM Systems Journal, VOL 41, NO 3, 524-529.

21. Buscaldi, D., Rosso, P., & Sanchis, E. (2005). Using the WordNet ontology in the GeoCLEF geographical information retrieval task. In: Accessing Multilingual Information Repositories, Revised Selected Papers of CLEF-2005, Springer-Verlag, LNCS (4022), 939-946.

22. Buscaldi, D., Rosso, P., & Sanchis, E. (2006). WordNet as a geographical information resource. In: Proc. 3rd Global WordNet Int. Conf., GWN-2006, Cheju, S.Korea, January 22-26, 37-42.

23. Bush, V. (1945). As We May Think. Atlantic Monthly, 176, 101-108, July 1945.

24. Cao, T.H. (1995). Fuzzy Conceptual Graph Programs, Master's Thesis, Asian Institute of Technology.

25. Cao, T.H. (1999). Foundations of Order-Sorted Fuzzy Set Logic Programming in Predicate Logic and Conceptual Graphs, PhD Thesis, University of Queensland, 244 pages.

26. Cao, T.H. (2001). Fuzzy Conceptual Graphs for the Semantic Web, BISC International Workshop on Fuzzy Logic and the Internet FLINT 2001.

27. Carpenter, B. (1991). Linguistic Knowledge Representation and Reasoning with Hybrid Constraints. In Proceedings of the AAAI Fall Symposium on Hybrid Reasoning, Asilomar.

28. Carpenter, B. (1992). The Logic of Typed Feature Structures. Cambridge Tracts in Theoretical Computer Science 32. Cambridge University Press.

29. Carpenter, B., & Penn G. (1996). Compiling typed attribute-value logic grammars. In H. Bunt and M. Tomita, editors, Recent Advances in Parsing Technologies, Kluwer, 145-168.

30. Chakraborty, C., & Chakraborty, D. (2007). A fuzzy clustering methodology for linguistic opinions in group decision making. Applied Soft Computing, Vol. 7, Issue 3, 858-869.

31. Chomsky, N. (1956). Three models for the description of language. Information Theory, IEEE Transactions Vol. 2 No. 3, 113-124.

32. Cleverdon, C.W. (1967). The Cranfield tests on index language devices. Aslib Proceedings, 19, 173-192.

33. Cochran, W.G. (1977). Sampling techniques, 3rd ed., New York: John Wiley & Sons, 444 pages.

34. Colmerauer, A. (1970). Les Systèmes-Q ou un Formalism por Analyser et Synthétiser des Phrases sur Ordinateur. Internal Publication 43, Département d'Informatique, Université de Montreal, Canada.

35. Colmerauer, A., Kanoui, H., Pasero, R., & Roussel, P. (1973). Un Système de Communication Homme-Machine en Français. Rapport, Groupe d'Intelligence Artificielle, Université d'Aix-Marseille II.

36. Colmerauer, A., & Roussel, P. (1993). The birth of Prolog, The second ACM SIGPLAN conference on History of programming languages Cambridge, Massachusetts, United States, 37-52.

37. Cooley, R., Mobashe, B., & Srivastaba, J. (1997). Grouping web page references into transactions for mining World Wide Web browsing patterns, Technical report TR 97-021, University of Minnesota, Minneapolis.

38. Covington, M.A. (2003). ET: an Efficient Tokenizer in ISO Prolog, The University of Georgia, Athens, U.S.A.

39. De la Mata, J., Olivas, J.A., & Serrano-Guerrero, J. (2004). Overview of an Agent Based Search Engine Architecture, Proceedings of the International Conference on Artificial Intelligence, ICAI-04, Volume I, Las Vegas, Nevada, USA, 62-67.

40. Delgado, M., Martin-Bautista, M.J., Sanchez, D., Serrano, J.M., & Vila, M.A. (2003). Association rules and fuzzy associations rules to find new query terms, Proc. of the Third Conference of the EUSFLAT, 49-53.

41. Dell, R.B., Holleran, S., & Ramakrishnan, R. (2002). Sample Size Determination, ILAR Journal, 43 (4) 207-213.

42. Dixon, R.M.W. (1977). Where Have all the Adjectives Gone? Studies in Language, 1, 19-80.

43. Dixon, R.W. (1982). Where have all the adjectives gone? Mouton Publishers.

44. Du, M.W., Chang, S.C., & Chow, A. (1989). A2QDT: A Syntax-Directed Prolog Dialect Translator, Proceedings of the 13th Annual International Computer Software and Applications Conference, COMPSAC 89, Orlando, FL, USA, 786-787.

45. El-Hamdouchi, A., Willet, P. (1989). Comparison of Hierarchic Agglomerative Clustering Methods for Document Retrieval, The Computer Journal, Vol. 32, No. 3.

46. Elsna, H. (1996). The Lonely Dreamer, Health Research Books, 187 pages.

47. Fasolo, M., Garbuio, L., Malanima, A., & Guarino, N. (1992). Robust parsing of natural language descriptions expressed in telegraphic style, Proceedings of the Third Conference on Applied Natural Language Processing, Trento, Italy, 229-230.

48. Fernandez, S. (2001). Una contribución al procesamiento automático de la sinonimia utilizando Prolog, Ph.D. thesis, Santiago de Compostela University, Spain, 358 pages.

49. Fernandez, S., Grana, J., & Sobrino, A. (2002). A Spanish e-dictionary of synonyms as a fuzzy tool for information retrieval. In Actas de las I Jornadas de Tratamiento y Recuperación de Información (JOTRI 2002), León, Spain.

50. Garcés, P.J., Olivas, J.A., & Romero, F.P. (2002). FIS-CRM: A Representation Model Based on Fuzzy Interrelations for Internet Search. ICAI-02, 219-224.

51. Garcés, P.J., Olivas, J.A., & Romero, F.P. (2006). Concept-matching IR systems versus Word-matching IR systems: Considering fuzzy interrelations for indexing web pages. Journal of the American Society for Information Science and Technology JASIST, 57 (4), 564-576.

52. Gasser, M. (2006). How Language Works, Edition 3.0, Indiana University.

53. George, H.V. (1997). Essays in Informational English Grammar with reference to English language teaching, Language Centre, La Trobe University, Bundoora, Victoria, Australia.

54. Ghosh, J. (2003). Scalable Clustering in The Handbook of Data Mining, Nong Ye (Ed), Lawrence Erlbaum Assoc., Chapter 10, 247-278.

55. Gomez, F. (2007). Semantic Interpretation and the WordNet Upper-Level Ontology, Journal of Intelligent Systems, Freund Publishing House Ltd. Copyright Freund Publishing House Ltd.

56. Gomez, F. (2008). The acquisition of common sense knowledge by being told: an application of NLP to itself, 13th International Conference on Applications of Natural Language to Information Systems, NLDB 2008.

57. Gonzalo, J., Verdejo, F., Chugur, I., & Cigarran, J. (1998). Indexing with WordNet synsets can improve retrieval, Proc. of the COLING/ACL Work. on usage of WordNet in natural language processing systems.

58. Grinberg, D., Lafferty, J., & Sleator, D. (1995). A robust parsing algorithm for link grammars. Carnegie Mellon University Computer Science technical report CMU-CS-95-125 and Proceedings of the Fourth International Workshop on Parsing Technologies, Prague.

59. Gruber, T.R. (1993). A translation approach to portable ontologies. Knowledge Acquisition, 5(2), 199-220.

60. Gruber, T.R. (1995). Toward principles for the design of ontologies used for knowledge sharing. Presented at the Padua workshop on Formal Ontology, March 1993, later published in International Journal of Human-Computer Studies, Vol. 43, Issues 4-5, 907-928.

61. Gruber, T. (2008). Ontology, to appear in the Encyclopedia of Database Systems, Ling Liu and M. Tamer Özsu (Eds.), Springer-Verlag.

62. Grune, D, & Jacobs, C.J.H. (1990). Parsing Techniques - A Practical Guide, Ellis Horwood, Chichester, England.

63. Hamdi, M.S. (2006). MASACAD: A multi-agent approach to information customization for the purpose of academic advising of students, Applied Soft Computing Article in Press, Elsevier B.V. Science Direct.

64. Han, J., & Kamber, M. (2001). Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers, New York.

65. Harman, D.K. (1993). Overview of the first Text REtrieval Conference (TREC-1) In Proceedings of the First Text REtrieval Conference (TREC-1), NIST Special Publication 500-207, 1-20.

66. Henry, P., & Bassac, C. (2007). A toolkit for a generative lexicon, Fourth International Workshop on Generative Approaches to the Lexicon, Paris, France.

67. Herrera-Viedma, E., Pasi, G. (2003). Fuzzy approaches to access information on the Web: recent developments and research trends, Proceedings of the Third Conference of the EUSFLAT, 25-31.

68. Hirsch, Jr. E.D., Kett, J.F., & Trefil, J. (Ed.) (2002). The New Dictionary of Cultural Literacy, Third Edition, Houghton Mifflin Company.

69. Hoefler, S. (2004). The Syntax of Attempto Controlled English: An Abstract Grammar for ACE 4.0, Technical Report ifi-2004.03, Department of Informatics, University of Zurich, Switzerland.

70. Hotho, A., Staab, S., & Stumme, G. (2003). Ontologies improve text document clustering, in Proceedings of the 3rd IEEE International Conference on Data Mining, 541-544.

71. Huang, L. (2000). A survey on web information retrieval technologies. Tech. rep., ECSL, 1-33.

72. ISO (2001). Conceptual Graph, ISO working document: ISO/JTC1/SC 32/WG2 N 000.

73. Israel, G.D. (2003). Determining Sample Size, PEOD6, Agricultural Education and Communication Department, Institute of Food and Agricultural Science, University of Florida, 1-5.

74. Jain, A.K., Murty, M.N., & Flynn, P.J. (1999). Data clustering: A review, ACM Comput. Surv. 31 (3) 264-323.

75. Jurafsky, D., & Martin, J.H. (2008). Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, Second Edition, Pearson Prentice Hall.

76. Karlsson, F. (1990). Constraint Grammar as a Framework for Parsing Running Text, Proceedings of the 13th International Conference on Computational Linguistics (COLING-90), vol. III, Helsinki, 168-173.

77. Karlsson, F., Voutilainen, A., Heikkilä, J., & Anttila, A. (Ed.) (1995). Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text. Natural Language Processing, No 4. Mouton de Gruyter, Berlin and New York.

78. Kaufman, L., & Rousseeuw, P.J. (1990). Finding Groups in Data: An Introduction to Cluster Analysis, John Wiley & Sons.

79. King-ip, L., & Ravikumar, K. (2001). A similarity-based soft clustering algorithm for documents. Proc. of the Seventh Int. Conf. on Database Sys. for Advanced Applications 2001.

80. Kiryakov, A.K., & Simov, K.I. (1999). Ontologically supported semantic matching, Proceedings of NODALIDA'99: Nordic Conference on Computational Linguistics, Trondheim.

81. Kitchenham, B. (2004). Procedures for Performing Systematic Reviews, TR/SE-0401, Keele University, 33 pages.

82. Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. Biological Cybernetics, Vol. 43, 59-69.

83. Kohonen, T. (1995). Self-organizing Maps, Series in Information Sciences, vol. 30, Springer.

84. Koster, C.H.A. (1991). Affix Grammars for natural languages. In: Attribute Grammars, Applications and Systems, International Summer School SAGA, Prague, Czechoslovakia, June 1991. Lecture Notes in Computer Science, volume 545. Springer-Verlag.

85. Koster, C.H.A. (2003). Extracting Head/Modifier Frames from English Text, Computing Science Institute, University of Nijmegen, The Netherlands, Working document.

86. Kowalski, G. (1997). Information Retrieval Systems – Theory and Implementation, Kluwer Academic Publishers.

87. Kolln, M., & Funk, R. (1997). Understanding English Grammar, Prentice Hall; 5 edition, 496 pages.

88. Kreidler, C.W. (1998). Introducing English semantics, Routledge, 332 pages.

89. Lafferty, J., Sleator, D., & Temperley, D. (1992). Grammatical Trigrams: A Probabilistic Model of Link Grammar. Proceedings of the AAAI Conference on Probabilistic Approaches to Natural Language.

90. Lafourcade, M. (2003). Conceptual vectors and fuzzy templates for discriminating hyperonymy (is-a) and meronymy (part-of) relations, Proceeding of the OOIS 2003 Workshop MASPEGUI, P. Valtchev, M.Huchard, H. Astudillo (eds.), Montreal, 19-29.

91. Lafourcade, M., & Prince, V. (2001). Relative Synonymy and conceptual vectors, Proceedings the Sixth Natural Language Processing Pacific Rim Symposium, Japan, (202), 127-134.

92. Lagus, K., Honkela, T., Kaski, S., & Kohonen, T. (1999). WEBSOM for textual data mining. Artificial Intelligence, 16-22.

93. Lancaster, F.W. (1968). Information Retrieval Systems: Characteristics, Testing and Evaluation, Wiley, New York.

94. Larsen, B., & Aone, C. (1999). Fast and Effective Text Mining Using Linear-time Document Clustering, KDD-99, San Diego, California.

95. Leacock, C., & Chodorow, M. (1998). Combining local context and Wordnet similarity for word sense disambiguation, In WordNet, an Electronic Lexical Database, MIT Press, Cambridge Ma, 285-303.

96. Lehnert, W.G. (1980). Narrative Text Summarization, Proceedings of the First Annual National Conference on Art i ficia l Intelligence. Palo Alto, California.

97. Lehnert, W.G. (1981). Plot Units and Narrative Summarization. Cognitive Science. vol. 5, No. 4.

98. Levine, R.D., & Meurers, D.W. (2006). Head-Driven Phrase Structure Grammar: Linguistic Approach, Formal Foundations, and Computational Realization. Encyclopedia of Language and Linguistics (second edition) Ed. Keith Brown. Oxford: Elsevier.

99. Lewis, D. (1999). Reuters-21578 text categorization test collection 1.0. http://www.research.att.com/~lewis.

100. Li, Y., Chung, S.M., & Holt, J.D. (2008). Text document clustering based on frequent word meaning sequences. Data Knowledge and Engineering, vol 64, 381-404.

101. Liddy, E., & McCracken, N. (2005). Hands-on NLP for an interdisciplinary audience. In Proc 2nd ACL Workshop on Effective Tools and Methodologies for Teaching NLP and CL, 62-68.

102. Hsi-Ching, L., Li-Hui, W., & Shyi-Ming C., (2006). Query expansion for document retrieval based on fuzzy rules and user relevance feedback techniques Expert Systems with Applications, Volume 31, Issue 2, 397-405.

103. Liu, H., & Singh, P. (2004). Commonsense reasoning in and over natural language. Proceedings of the 8th International Conference on Knowledge-Based Intelligent Information & Engineering Systems (KES-2004).

104. Liu, H., & Singh, P. (2004). ConceptNet: A Practical Commonsense Reasoning Toolkit. BT Technology Journal, Volume 22, Issue 4, October 2004. Kluwer Academic Publishers, 211-226.

105. Loper, E. (2004). NLTK: Building a Pedagogical Toolkit in Python, In PyCon DC 2004. Python Software Foundation, Washington DC.

106. Loper, E., & Bird, S. (2002). NLTK: The Natural Language Toolkit. In Proc ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics, 62-69.

107. Loupy, C., & El-Bèze, M. (2002). Managing synonymy and polysemy in a document retrieval system using WordNet, Proceedings of the LREC2002: Workshop on Linguistic Knowledge Acquisition and Representation.

108. Luhn, H.P. (1957). A statistical approach to mechanized encoding and searching of literary information. IBM Journal of Research and Development.

109. Lynn Muehleisen, V. (1997). Antonymy and Semantic Range in English, PhD dissertation, Northwestern University, Evanston, Illinois.

110. Martin-Bautista, M.J., Vila, M.A., Kraft, D.H., Chen, J., & Cruz, J. (2002). User profiles and fuzzy logic for Web retrieval, Journal of Soft Computing, 6, 5, 365-372.

111. MacKay, D.J.C. (2003). Information Theory, Inference, and Learning Algorithms, Cambridge University Press.

112. Mann, G.S. (2002). Fine-Grained Proper Noun Ontologies for Question Answering, International Conference on Computational Linguistics, COLING-02 on SEMANET: building and using semantic networks, Vol. 11 1-7.

113. Maron, M.E., & Kuhns, J.L. (1960) On relevance, probabilistic indexing and information retrieval. Journal of the ACM, 7 216-244.

114. Matsumoto, Y. (1986). A Parallel Parsing System for Natural Language Analysis, Proc. of 3rd International Conference on Logic Programming.

115. Matsumoto, Y., & Sugimura, R. (1987). A Parsing System Based on Logic Programming, In Proceedings of IJCAI 87.

116. Matuszek, C., Witbrock, M., Kahlert, R., Cabral, J., Schneider, D., Shah, P., & Lenat, D.B. (2005). Searching for Common Sense: Populating Cyc from the Web In

Proceedings of the Twentieth National Conference on Artificial Intelligence, Pittsburgh, Pennsylvania.

117. Mazón, J.N., Trujillo, J., Serrano, M., & Piattini, M. (2005). Using WordNet Ontology to automatically enrich dimension hierarchies in a data warehouse, VLDB Workshop on Ontologies Based Techniques for Databases and Information Systems, Trondheim.

118. McCarthy, J., Minsky, M., Sloman, A., Gong, L., Lau, T., Morgenstern, L., Mueller, E.T., Riecken, D., Singh, M., & Singh, P. (2002). An architecture of diversity for commonsense reasoning, Technical Forum 2002 IBM Systems Journal, Vol. 41, NO 3, 530-539.

119. Metzler, D., & Croft, W. (2004). Combining the language model and inference network approaches to retrieval. Information Processing and Management 40(5), 735-750.

120. Mendes, M.E.S., & Sacks, L. (2003). Evaluating fuzzy clustering for relevance-based information access. In Proc. of the IEEE International Conference on Fuzzy Systems, FUZZ-IEEE'03.

121. Miller, G.A. (1995). WordNet: A lexical database for English, Communications of the ACM 11, 39-41.

122. Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K.J. (1990). WordNet: An on-line lexical database, International Journal of Lexicography, 3:4, 235-312.

123. Minsky, M. (1970). Form and Content in Computer Science, Journal of the ACM (JACM), Volume 17, Issue 2, April 1970, 197-215.

124. Minsky, M. (1974). A Framework for Representing Knowledge, MIT-AI Laboratory Memo 306, June, 1974. Reprinted in The Psychology of Computer Vision, P. Winston (Ed.), McGraw-Hill, 1975. Shorter versions in J. Haugeland, Ed., Mind Design, MIT Press, 1981, and in Cognitive Science, Collins, Allan and Edward E. Smith (eds.) Morgan-Kaufmann, 1992.

125. Montague, R. (1960). On the nature of certain philosophical entities, originally published in The Monist 53 (1960), revised version in Montague (1974) 148-187.

126. Montague, R. (1970). English as a formal language, reprinted in Montague (1974), 188-221.

127. Montague, R. (1970). Universal grammar, reprinted in Montague (1974), 222-246.

128. Montague, R. (1970). The proper treatment of quantification in ordinary English, reprinted in Montague (1974), 247-270.

129. Montague, R. (1973). The proper treatment of quantification in ordinary English. In Hintikka, K.J.J., Moravcsik, J.M.E., & Suppes, P. (eds.) Approaches to Natural Language. Dordrecht: Reidel, 221-242. Reprinted in Montague (1974), 247-270; Reprinted in Portner and Partee, (eds.), 17-34.

130. Montague, R. (1974). Formal Philosophy, Yale University Press, New Haven.

131. Morton, S.K. (1978). Conceptual graphs and fuzziness in artificial intelligence, PhD Thesis, University of Bristol.

132. Nederhof, M.J., & Koster, C.H.A. (1992). A customized grammar workbench, In: J. Arts, P. de Haan and N. Oostdijk, English Language Corpora: Design, Analysis and Exploitation, Papers from the thirteenth International Conference on English Language Research on Computerized Corpora, 163-179.

133. Niles, I., & Pease, A. (2001). Origins of the Standard Upper Merged Ontology: A Proposal for the IEEE Standard Upper Ontology. In Working Notes of the IJCAI-2001 Workshop on the IEEE Standard Upper Ontology, Seattle, Washington.

134. Niles, I., & Pease, A. (2001). Towards a Standard Upper Ontology. In Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001), Chris Welty and Barry Smith, eds, Ogunquit, Maine.

135. NIST (2001). Issues, Tasks and Program Structures to Roadmap Research in Question & Answering (Q&A), Q&A Roadmap Committee, National Institute of Standards and Technology (NIST), 35 pages.

136. Numi, Y., & Kobayashi, Y. (1990). A Speech Interface to an Information Retrieval System, Studia Phonologica XXIV.

137. Nurnberger, A., & Detyniecky, M. (2006). Externally growing self-organizing maps and its application to e-mail database visualization and exploration, Applied Soft Computing 6, 357-371.

138. Oakman, R.L. (1996). The Computer Triangle: Hardware, Software, People, John Wiley & Sons, Inc, 1996, 360 pages.

139. Olivas, J.A. (2006). Las Técnicas de Soft-Computing en la Recuperación de Información, PhD course notes, 76 pages.

140. Olivas, J.A., Rios, S. (2006). Three-Dimensional Representation of Conceptual Fuzzy Relations, Lecture Notes in Computer Science, Numb 4027, Springer-Verlag, Germany, 681-690.

141. Olivas, J.A., Garces, P.J., & Romero, F.P. (2003). An application of the FIS-CRM model to the FISS metasearcher: Using fuzzy synonymy and fuzzy generality for representing concepts in documents. Int. J. Approx. Reasoning 34, (2-3), 201-219.

142. Olivas, J. A., de la Mata, J. & Serrano-Guerrero, J. (2004). Ontology Constructor Agent for improving Web Search with GUMSe In Proceedings of 10th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems IPMU 2004, Perugia, Italia, 1341-1348.

143. Olivas, J.A., de la Mata, J., Serrano-Guerrero, J., Garcés, P.J., & Romero, F.P. (2006). Desarrollo de motores inteligentes de búsqueda en Internet en el marco del grupo de investigación SMILe-ORETO. En Olivas, J. A., Sobrino, A. (eds.): Recuperación de información textual, Text Information Retrieval. Santiago de Compostela University, 89-102.

144. Olivas, J.A., Garcés, P.J., de la Mata, J., Romero, F.P., & Serrano-Guerrero, J. (2007). Conceptual Soft-Computing based Web search: FISCRM, FISS Metasearcher and GUMSe Architecture. In M. Nikravesh, J. Kacprzyk and L. A. Zadeh (eds.), Forging the New Frontiers: Fuzzy Pioneers II, Studies in Fuzziness and Soft Computing, Springer Verlag, 126-150.

145. Ortiz, J.L., & Valle, M. (2002). Fuzzy DCG Syntactic Parser for Command Language Recognition under Special Conditions, Proceeding (374) Software Engineering and Applications – SEA 2002, Cambridge, USA.

146. Paradis, C. (2001). Adjectives and boundedness. Cognitive Linguistics 12.1, 47-65

147. Partee, B.H. (2001). Montague grammar. In International Encyclopedia of the Social and Behavioral Sciences, ed. Neil J. Smelser and Paul B. Baltes.Oxford: Pergamon/ Elsevier Science.

148. Partee, B.H. (2006). Richard Montague (1930-1971), in Brown, Keith, ed., Encyclopedia of Language and Linguistics, Vol. 8, 2nd ed. Oxford: Elsevier: 255-257.

149. Partee, B,H,, Meulen, A.T., & Wall, R.E. (1990). Computational Linguistics Volume 18, Number 1, Mathematical Methods in Linguistics, Dordrecht: Kluwer Academic Publishers, Studies in Linguistics and Philosophy 30, 663 pages.

150. Pasi, G. (2002). Flexible information retrieval: some research trends, Mathware and Soft Computing 9, 107-121.

151. Pease, A., Murray, W. (2003). An English to logic translator for ontology-based knowledge representation languages, Natural Language Processing and Knowledge Engineering, 2003. Proceedings. 2003 International Conference on Volume, Issue , 26-29, 777-783.

152. Pedrycz, W. (1996). Conditional Fuzzy C-Means, Pattern Recognition Letters, Vol.17, 625-631.

153. Pedrycz, W. (2005). Knowledge-Based Clustering, John Wiley & Sons, Inc.

154. Peirce, C.S. (1868). On a New List of Categories, Proceedings of the American Academy of Arts and Sciences vol. 7, 287-298.

155. Penn, G., & Carpenter, R. (1999). Ale for Speech: A Translation Prototype, In Proceedings, Sixth European Conference on Speech Communication and Technology (EUROSPEECH'99), Budapest, Hungary.

156. Pereira, F.C.N., & Shieber, S.M. (1987). Prolog and Natural-Language Analysis, Center for the Study of Language and Information Publication Lecture Notes, Lecture Notes 10, 70-90.

157. Perkovitz, M., & Etzioni, O. (2000). Towards adaptive web sites: Conceptual framework and case study, Artificial Intelligence 118, 245-275.

158. Pfeiffer, H.D., Chavez Jr, N.R., & Pfeiffer Jr, J.J. (2007). CPE Design Considering Interoperability, Proceedings 2nd Conceptual Structures Tool Interoperability Workshop, CS-TIW 2007, Sheffield Hallam University, Sheffield, UK, Research Press International, UK, 71-76.

159. Pollard, C., Sag, I.A. (1987). Information-based Syntax and Semantics. Volume 1: Fundamentals. Stanford: CSLI Publications.

160. Pollard, C., Sag, I.A. (1994). Head-driven phrase structure grammar. Chicago: University of Chicago Press.

161. Porter, M. (1980). An algorithm for suffix stripping, Program, Vol. 14, no. 3, 130-137.

162. Pustejovsky, J. (1991). The Generative Lexicon, in Computational Linguistics, 17.4.

163. Qvortrup, L. (2007). The public library: from information access to knowledge management: a theory of knowledge and knowledge categories Information Research, 12(4) paper colis17.

164. Reed, S., & Lenat, D.B. (2002). Mapping Ontologies into Cyc. In AAAI 2002 Conference Workshop on Ontologies For The Semantic Web, Edmonton, Canada.

165. Ricarte, I., Gomide, F. (2001). A reference model for intelligent information search. Proceedings of the BISC Int. Workshop on Fuzzy Logic and the Internet, 80-85.

166. Robertson, S.E. (1977). The probabilistic ranking principle in IR. Journal of Documentation, 33, 294-304.

167. Robertson, S.E. (2004). Understanding inverse document frequency: on theoretical arguments for IDF, Journal of Documentation 60, 503-520.

168. Robertson, S.E., & Sparck Jones, K. (1994). Simple, proven approaches to text retrieval. University of Cambridge Computer Laboratory Technical Report no. 356, 1994 Updated. 2006.

169. Romero, F.P., Olivas, J.A., Garces, P.J., & Jimenez, L. (2003). fzMail: A Fuzzy Tool for Organizing E-Mail, The 2003 International Conference on Artificial Intelligence, ICAI-03, Las Vegas, USA.

170. Romero, F.P., Olivas, J.A., & Garcés, P. (2006). A soft Approach to Hybrid Models for Document Clustering", Proceedings of the Information Processing and Management of Uncertainty in Knowledge-based Systems IPMU'06, Paris Les Cordeliers, France, Vol I, 1040-1045.

171. Romero, F.P., Soto, A., & Olivas, J.A. (2007). A Hybrid Model for Document Clustering based on a Fuzzy Approach of Synonymy and Polysemy, International Fuzzy Systems Association IFSA 2007 World Congress, Cancún, Mexico.

172. Romero, F.P., Soto, A., & Olivas, J.A. (2007). A Hybrid Model for Document Clustering based on a Fuzzy Approach of Synonymy and Polysemy, Theoretical Advances and Applications of Fuzzy Logic and Soft Computing, Advances in Soft Computing, Springer Verlag, Ed. O. Castillo et al., Vol. 42, 171-179.

173. Romero, F.P., Soto, A., Olivas, J.A. (2007). Fuzzy Clustering based on Concept Measuring in Documents, EUROFUSE workshop New Trends in Fuzzy Preference Modelling, Jaén, Spain.

174. Romero, F.P., Peralta, A., Olivas, J.A., & Serrano-Guerrero, J. (2007). Clustering documental basado en mapas de Kohonen optimizados mediante técnicas de lógica borrosa, Actas del II Simposio sobre Lógica Fuzzy y Soft Computing (LFSC 2007), 97-105.

175. Rubens, N.O. (2006). The Application of Fuzzy Logic to the Construction of the Ranking Function of Information Retrieval Systems, Computer Modelling and New Technologies, Vol.10, No.1, 20-27.

176. Russell, S., & Norvig, P. (2002). Artificial Intelligence: A Modern Approach, Second Edition Prentice Hall Series in Artificial Intelligence, 1081 pages.

177. Sag, I.A., Wasow, T., & Bender, E. (2003). Syntactic theory: a formal introduction. 2nd ed. Chicago: University of Chicago Press.

178. Sætre, R., Tveit, A., Steigedal, T.S., & Lægreid, A. (2005). Semantic annotation of biomedical literature using Google. In Data Mining and Bioinformatics Workshop, volume 3482 of Lecture Notes in Computer Science. Springer.

179. Salton, G. (Ed.) (1971). The SMART Retrieval System—Experiments in Automatic Document Retrieval. Prentice Hall Inc., Englewood Cliffs, NJ, USA.

180. Salton, G., Wong, A., & Yang, C.S. (1975). A vector space model for automatic indexing. Communications of the ACM, Vol. 18, No. 11, 613-620.

181. Salton, G., & McGill, M.J. (1983). Introduction to Modern Information Retrieval. McGraw-Hill.

182. Schmolze, J.G., & Israel, D.J. (1983). KL-ONE: Semantics and classification, Research in Knowledge Representation and Natural Language. BBN Tech. Rep. N.5421. Bolt, Beranek and Newman, Cambridge, Mass.

183. Sedona (2007). The Sedona Conference Best Practices Commentary on the Use of Search & Information Retrieval Methods in E-Discovery, The Sedona Conference Journal, Vol. 8, 189-223.

184. Sehitoglu, O.T. (1996). A Sign-Based Phrase Structure Grammar for Turkish, Master of Science Thesis, arXiv:cmp-lg/9608016 v2 28.

185. Shapiro, S.C. (2003). Knowledge Representation. In Lynn Nadel, Ed. Encyclopedia of Cognitive Science, Vol. 2, Macmillan Publishers Ltd., 671-680.

186. Siegel, N., Goolsbey, K., Kahlert, R., & Matthews, G. (2004). The Cyc® System: Notes on Architecture, Cycorp Inc.

187. Singh, P. (2002). The Public Acquisition of Commonsense Knowledge, American Association for Artificial Intelligence.

188. Singh, P., Lin, T., Mueller, E.T., Lim, G., Perkins, T., & Li Zhu, W. (2002). Open Mind Common Sense: Knowledge Acquisition from the General Public, Lecture Notes In Computer Science; Vol. 2519, On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE 2002 Confederated International Conferences DOA, CoopIS and ODBASE 2002, 1223-1237.

189. Singh, P., Barry, B., & Liu, H. (2004). Teaching machines about everyday life. BT Technology Journal, 22(4), 227-240.

190. Singhal, A. (2001). Modern Information Retrieval: A Brief Overview, Bulletin of the IEEE Computer Society Technical Committee on Data Engineering 24 (4), 35-43.

191. Sleator, D., & Temperley, D. (1991). Parsing English with a Link Grammar. Carnegie Mellon University Computer Science technical report CMU-CS-91-196.

192. Sleator, D., & Temperley, D. (1993). Parsing English with a Link Grammar, Third International Workshop on Parsing Technologies.

193. Smith, B. (2003). Ontology, in L. Floridi (ed.) Blackwell Guide to the Philosophy of Computing and Information, Oxford: Blackwell, 155-166.

194. Snasel, V., Moravec, P., & Pokorny, J. (2005). Using BFA with WordNet ontology based model for web retrieval. In Proceedings of SITIS'2005. 254-259.

195. Snasel, V., Moravec, P., & Pokorny, J. (2005). WordNet Ontology Based Model for Web Retrieval, Proceedings of the International Workshop on Challenges in Web Information Retrieval and Integration 220-225.

196. Soto, A., Olivas, J.A., & Prieto, M.E. (2006). Fuzzy Approach of Synonymy and Polysemy for Information Retrieval International Symposium on Fuzzy and Rough Sets, ISFUROS 2006, Santa Clara, Cuba.

197. Soto, A., Olivas, J.A., & Prieto, M.E. (2007). Fuzzy Measure of Meaning Presence in Documents, Eleventh International Conference on Computer Aided Systems Theory EUROCAST-07, Canary Islands, Spain, 117-119.

198. Soto, A., Olivas, J.A., & Prieto, M.E. (2007). Using Generalized Constraints and Protoforms to deal with adjectives, FUZZ-IEEE 2007 IEEE International Conference on Fuzzy Systems, Imperial College, London, UK.

199. Soto, A., Olivas, J.A., & Prieto, M.E. (2007). Using Generalized Constraints and Protoforms to deal with adverbs, 5th Conference of the European Society for Fuzzy Logic and Technology EUSFLAT 2007, Ostrava, Czech Republic.

200. Soto, A., Olivas, J.A., & Prieto, M.E. (2008). Fuzzy Approach of Synonymy and Polysemy for Information Retrieval in Granular Computing: At the Junction of Rough Sets and Fuzzy Sets, Springer Berlin / Heidelberg, Serie Studies in Fuzziness and Soft Computing, Vol. 224, 179-198.

201. Soto, A., Olivas, J.A., & Prieto, M.E. (2008). Interpretando Información en Lenguaje Natural como Restricciones Generalizadas, XIV Congreso Español sobre Tecnologías y Lógica Fuzzy ESTYLF 2008, Mieres-Langreo, 625-630.

202. Sowa, J.F. (1976). Conceptual graphs for a data base interface, IBM Journal of Research and Development 20:4, 336-357.

203. Sowa, J.F. (1984). Conceptual Structures - Information Processing in Mind and Machine, Addison-Wesley Publishing Company, Massachusetts.

204. Sowa, J.F. (Ed.). (1991). Principles of Semantic Networks: Explorations in the Representation of Knowledge, Morgan Kaufmann Publishers, San Mateo, CA.

205. Sowa, J.F. (1992). Semantic Networks, in Encyclopedia of Artificial Intelligence, edited by Stuart C. Shapiro, John Wiley & Sons, 1987, second edition, 1792 pages.

206. Sowa, J.F. (2000). Knowledge Representation: Logical, Philosophical, and Computational Foundations, Brooks Cole Publishing Co., Pacific Grove, CA, 594 + xiv pages.

207. Sowa, J.F. (1986). Way EC, Implementing a semantic interpreter using conceptual graphs, IBM Journal Res. Develop. Vol. 30, No. 1.

208. Sparck Jones, K. (2004). A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation 28, 11-21, 1972. Also Journal of Documentation 60, 493-502.

209. Sparck Jones, K., Walker, S., & Robertson, S.E. (2000). A probabilistic model of information retrieval: development and comparative experiments. Part 1. Information Processing and Management 36, 779-808.

210. Spath, H. (1980). Clustering Analysis Algorithms for Data Reduction and Classification of Objects, Ellis Horwood, Chichester.

211. Suchanek, F.M., Kasneci, G., & Weikum, G. (2007). YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia, Proceedings of the 16th international conference on World Wide Web, Banff, Alberta, Canada, 697-706.

212. Suchanek, F.M., Kasneci, G., & Weikum, G. (2008). YAGO: A Large Ontology from Wikipedia and WordNet, Elsevier Journal of Web Semantics.

213. Tang, Y., & Zhang, Y. (2001). Personalized library search agents using data mining techniques, Proceedings of the BISC Int. Workshop on Fuzzy Logic and the Internet, 119-124.

214. Turing, A.M. (1950). Computing machinery and intelligence. Mind, 59, 433-460.

215. Turtle, H., & Croft, W. (1991). Evaluation of an inference network-based retrieval model. ACM Transactions on Information Systems 9(3), 187-222.

216. Van Rijsbergen, C.J. (1979). Information Retrieval, Information Retrieval Group, University of Glasgow, Butterworths, London, UK.

217. Van Rijsbergen, C.J. (1989). Information Retrieval, Second edition, Buttersworth, London, UK.

218. Vilares, M., & Alonso, M.A. (1996). An LALR Extension for DCGs in Dynamic Programming, in P. Lucio, M. Martelli, M. Navarro (eds.), Proc. of APPIA-GULP-PRODE'96 Joint Conference on Declarative Programming, Donostia-San Sebastián, Spain, 79-88.

219. Vilares, M., Alonso, M.A., & Cabrero, D. (1996). An Experience on Natural Language Parsing, in C. Martín Vide (ed.), Congreso de Lenguajes Naturales y Lenguajes Formales XII, PPU, Barcelona, Spain, 555-562.

220. Vilares, M., Alonso, M.A., & Cabrero, D. (1996). Autómatas Lógicos y Lenguaje Natural, V Congreso Iberoamericano de Inteligencia Artificial, IBERAMIA'96, 341-351, Editorial Limusa, S.A. de C.V., Mexico D.F., Mexico.

221. Vilares, M., Alonso, M.A., Graña, J., & Cabrero, D. (1998). GALENA: Tabular DCG Parking for Natural Languages, Proc. of First Workshop on Tabulation in Parsing and Deduction (TAPD'98), Paris, France, 44-51.

222. Voorhees, E. (1985). The cluster hypothesis revisited, Annual ACM Conference on Research and Development in Information Retrieval, Proceedings of the 8th annual international ACM SIGIR conference on Research and development in information retrieval, Montreal, Quebec, Canada, 188-196.

223. Whaley, J.M. (1999). An application of word sense disambiguation to information retrieval, Dartmouth College Computer Science Technical Report PCS-TR99-352.

224. Widyantoro, D.H., & Yen, J. (2000). A fuzzy similarity approach in text classification task, Proceedings of Ninth IEEE International Conference on Fuzzy Systems, vol. 2, 653-658.

225. Widyantoro, D., & Yen, J. (2001). Incorporating fuzzy ontology of term relations in a search engine, Proceedings of the BISC Int. Workshop on Fuzzy Logic and the Internet, 155-160.

226. Wielemaker, J. (2003). An overview of the {SWI-Prolog} Programming Environment, Proceedings of the 13th International Workshop on Logic Programming Environments, editor Fred Mesnard and Alexander Serebenik, Katholieke Universiteit Leuven, Heverlee, Belgium, 1-16.

227. Wielemaker, J., & Anjewierden, A. (2001). Programming in XPCE/Prolog. SWI, University of Amsterdam, Roetersstraat 15, 1018 WB Amsterdam, The Netherlands.

228. Wielemaker, J., & Anjewierden, A. (2002). An Architecture for Making Object-Oriented Systems Available from Prolog. Alexandre Tessier, editor; WLPE 2002 Software Engineering 97-110.

229. Wilson, K.G. (1993). The Columbia Guide to Standard American English. Columbia University Press.

230. WordNet (1998). An Electronic Lexical Database. Christiane Fellbaum (editor). The MIT Press, Cambridge, MA.

231. W3C (2004). Wordnet in RDFS and OWL, WordNet Task Force, Semantic Web Best Practices and Deployment Working Group, W3C, http://www.w3.org/2001/sw/BestPractices/WNET/wordnet-sw-20040713.html.

232. Wulff, S. (2003). A multifactorial corpus analysis of adjective order in English, International Journal of Corpus Linguistics 8:2, John Benjamins Publishing Company, 245-282.

233. Yager, R.R. (2006). Knowledge trees and protoforms in question-answering systems, Journal of the American Society for Information Science and Technology, Volume 57, Issue 4, 550-563.

234. Yang, Y. (1999). An Evaluation of Statistical Approaches to Text Categorization, Journal of Information Retrieval, Vol 1, No. 1/2, 67-88.

235. Yao, Y.Y. (2002). Information retrieval support systems, Proceedings of FUZZ-IEEE'02, pp773-778.

236. Yao, J.T., Yao, Y.Y. (2003). Web-based information retrieval support systems: building research tools for scientists in the new information age Proceedings of the IEEE/WIC International Conference on Web Intelligence (WI'03), Halifax, Canada, 570-573.

237. Zadeh, L.A. (1965). Fuzzy sets. Information and Control, Vol. 8. 338-353.

238. Zadeh, L.A. (1973). Outline of a new approach to the analysis of complex system and decision processes. IEEE Trans. On Systems, Man, and Cybernetics, Vol. SMC–3, No. 1, 28-44.

239. Zadeh, L.A. (1976). A fuzzy-algorithmic approach to the definition of complex or imprecise concepts, Int. J. Man-Machine Studies 8, 249-291.

240. Zadeh, L.A. (1982). A Note on Prototype Theory & Fuzzy Sets, Cognition, Elsevier Sequoia, 12, 291-297.

241. Zadeh, L.A. (1997). Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. Fuzzy Sets and Systems 90, Elsevier, 111-127.

242. Zadeh, L.A. (2000). Toward the Concept of Generalized Definability. Lecture presented at the Rolf Nevanlinna Colloquium, University of Helsinki, Helsinki, Finland, 3 pages.

243. Zadeh, L.A. (2001). A New Direction in AI: Toward a Computational Theory of Perceptions, AI MAGAZINE, American Association for Artificial Intelligence, 0738-4602-2001, 73- 84.

244. Zadeh, L.A. (2001). A Prototype-Centered Approach to Adding Deduction Capability to Search Engines -- The Concept of Protoform, Berkeley Initiative in Soft Computing, Submitted to the BISC mailing list 19th Dec.

245. Zadeh, L.A. (2002). From Computing with Numbers to Computing With words. From Manipulation of Measurements to Manipulation of Perceptions. Int. J. Appl. Math. Comput. Sci., Vol.12, No.3, 307-324 Reprinted, with permission from IEEE Transactions on Circuits and Systems – I: Fundamental Theory and Applications, Vol. 45, No. 1, January 1999, 105-119.

246. Zadeh, L.A. (2002). Toward a perception-based theory of probabilistic reasoning with imprecise probabilities. Journal of Statistical Planning and Inference 105, Elsevier, 233-264.

247. Zadeh, L.A. (2003). Protoform Theory and Its Basic Role in Human Intelligence, Deduction, Definition and Search, PerMIS 2003 Performance Metrics for Intelligent Systems Workshop, Gaithersburg, MD.

248. Zadeh, L.A. (2003). Web intelligence and fuzzy logic—The concept of Web IIQ (WIQ). WI'03 and IAT'03 Keynote Talk, Halifax, Canada, 90 pages.

249. Zadeh, L.A. (2004). A note on web intelligence, world knowledge and fuzzy logic, Elsevier Science Publishers B. V.Amsterdam, The Netherlands, The Netherlands, Data & Knowledge Engineering Volume 50 , Issue 3, September 2004, 291-304.

250. Zadeh, L.A. (2004). Fuzzy Logic Systems: Origin, Concepts, and Trends, Distinguished Lecture at Hong Kong Baptist University, Hong Kong. November 10, 138 pages.

251. Zadeh, L.A. (2004). Web Intelligence, World Knowledge and Fuzzy Logic, Berkeley Initiative in Soft Computing (BISC), Fuzzy Logic and the Internet Initiative (FLINT), 18 pages.

252. Zadeh, L.A. (2004). Web Intelligence, World Knowledge and Fuzzy Logic, November 11, Santa Clara, 124 pages.

253. Zadeh, L.A. (2004). Web Intelligence, World Knowledge and Fuzzy Logic - The Concept of Web IQ (WIQ). KES 2004: 1-5 Mircea Gh. Negoita, Robert J. Howlett, Lakhmi C. Jain (Eds.): Knowledge-Based Intelligent Information and Engineering Systems, 8th International Conference, KES 2004, Wellington, New Zealand, September 20-25, 2004. Proceedings. Part I. Lecture Notes in Computer Science 3213 Springer.

254. Zadeh, L.A. (2005). From Search Engines to Question- Answering Systems—The Problems of World Knowledge, Relevance, Deduction and Precisiation, Keynote Speech, IRI 2005 IEEE International Conference on Information Reuse and Integration, August 15, Las Vegas, Nevada, USA, 150 pages.

255. Zadeh, L.A. (2005). From Search Engines to Question- Answering Systems—The role of fuzzy logic, Progress in informatics, 1-3.

256. Zadeh, L.A. (2005). Fuzzy Logic as a Basis for Theory of Precisiation of Meaning (TPM), Joint Fourth Conference of the European Society for Fuzzy Logic and Technology & 11th Logique Floue et ses Applications Conference (EUSFLAT-LF), 7-9 September, Barcelona, Spain.

257. Zadeh, L.A. (2005). Toward A Computational Theory Of Precisiation Of Meaning Based On Fuzzy Logic—The Concept Of Cointensive Precisiation, 11th World Congress of International Fuzzy Systems Association (IFSA), Beijing, China, 8 pages.

258. Zadeh, L.A. (2005). Toward a Generalized Theory of Uncertainty (GTU)—An Outline, Information Sciences—Informatics and Computer Science: An International Journal, Volume 172, Issue 1-2 (June 2005), 40 pages.

259. Zadeh, L.A. (2005). Web Intelligence, World Knowledge and Fuzzy Logic, 1-18 in Soft Computing for Information Processing and Analysis, Series: Studies in Fuzziness and Soft Computing, Nikravesh, Masoud; Zadeh, Lotfi A.; Kacprzyk, Janusz (Eds.), Vol. 164, X, 456 pages.

260. Zadeh, L.A. (2006). A New Frontier in Computation – Computation with Information Described in Natural Language, CIS (Center for Intelligent Systems) Seminar, UC Berkeley, 9 November 2006.

261. Zadeh, L.A. (2006). A New Frontier in Computation—Computation with Information Described in Natural Language IEEE IS'06, 3rd IEEE Conference On Intelligent Systems, University of Westminster, United Kingdom, 4-6 September 2006.

262. Zadeh, L.A. (2006). A New Frontier in Computation—Computation with Information Described in Natural Language International Conference on Computational Intelligence for Modelling, Control and Automation - CIMCA06 Jointly with International Conference on Intelligent Agents, Web Technologies and Internet Commerce - IAWTIC06 29 November to 1 December 2006 Sydney, Australia.

263. Zadeh, L.A. (2006). From Search Engines to Question Answering Systems: The Problems of World Knowledge, Relevance, Deduction and Precisiation. In Fuzzy Logic and the Semantic Web, edited by Elie Sanchez, Elsevier, 163-210.Zadeh, L.A. (2006). From Search Engines to Question- Answering Systems—The Problems of World Knowledge, Relevance, Deduction and Precisiation. Lecture at McDonell Douglas Engineering Auditorium, 1st-Mar- 2006. Mailing list of the Center for Pervasive Communications and Computing at the University of California, Irvine.Zadeh, L.A. (2006). Generalized theory of uncertainty (GTU)—principal concepts and ideas, Computational Statistics & Data Analysis, Elsevier, In Press, Corrected Proof, 32 pages.

266. Zadeh, L.A. (2008). Toward Human Level Machine Intelligence-Is It Achievable? The Need for a Paradigm Shift, IEEE Computational Intelligence Magazine, Vol. 3, No. 3, 11-22.

267. Zadeh, L.A., & Nikravesh, M. (2002). Perception-Based Intelligent Decision Systems, ONR Summer Program Review, University of California Los Angeles UCLA, USA. July 30-August 1, 65 pages.Zamir, O., & Etzioni, O. (1999). Grouper: A dynamic clustering interface to web search results, Proceedings of the WWW8.

269. Zhao, Y., & Karypis, G. (2002). Evaluation of hierarchical clustering algorithms for document datasets. In Proceedings of CIKM, ACM Press, 515-524.

Appendixes

Appendix 1: Glossary of fuzzy model definitions and formulas

V: vocabulary, a set of terms

M: set of meanings associated to the terms in V

NM: number of meanings in M

meaning: crisp relation such that

meaning (t, m) = 1 iff there ∃ term t ∈ V and meaning m ∈ M where m is one of the meanings of t.

$$meaning\, V\, x\, M \rightarrow [0,1]. \tag{67}$$

M(t): set of different meanings associated with a certain term *t*.

$N_m(t)$: number of meanings associated with term *t*, i.e. the number of elements of *M(t)*

$$M(t) = \{m \in M \,/\, meaning(t,m) = 1\} \tag{68}$$

$I_p(t)$ index that represent the polysemy degree of term *t*.

$$I_p : V \rightarrow [0,\ 1] \quad \text{where} \quad I_p(t) = 1 - \frac{1}{N_m(t)}. \tag{69}$$

$I_s(t)$ index that represent the strength degree of term *t*.

$$I_S(t) = \frac{1}{N_m(t)} = 1 - I_p(t)\ . \tag{70}$$

S: fuzzy relation between two terms $t_1, t_2 \in V$ such that $S(t_1,t_2)$ express the degree of synonymy between both terms:

$$S(t_1,t_2) = \frac{|M(t_1) \cap M(t_2)|}{|M(t_1)|}. \tag{71}$$

T(m): set of terms that share a meaning m:

$$T(m) = \{t \in V / meaning(t, m) = 1\}. \quad (72)$$

D: collection of documents $D = \{D_1, D_2, D_3, \ldots, D_{nd}\}$

tf: term frequency, a measure of the importance of a term $t_i \in V$ within a document $D_i \in D$

$$tf_{ij} = \frac{n_{ij}}{\sum_k n_{kj}} = \frac{n_{ij}}{n_{*j}} \quad (73)$$

n_{ij}: number of occurrences of term t_i in the document D_i

n_{*j}: number of terms in document D_j

R_m: a measure of the use of a meaning $m \in M$ in a document D_j based on the number of occurrences of the terms associated with that meaning.

$$R_j(m) = \sum_{t_i \in T(m)} \left(n_{ij.} \left(1 - I_p(t_i)\right)\right) = \sum_{t_i \in T(m)} \left(\frac{n_{ij}}{N_m(t_i)}\right) \quad (74)$$

$Cf_j(m)$: concept frequency of a meaning $m \in M$ in a document D_j

$$Cf_j(m) = \frac{R_j(m)}{\sum_k n_{kj}} = \frac{\sum_{t_i \in T(m)} \left(n_{ij} \cdot \left(1 - I_p(t_i)\right)\right)}{n_{*j}} = \frac{\sum_{t_i \in T(m)} \left(n_{ij} \cdot I_S(t_i)\right)}{n_{*j}} \quad (75)$$

Cosine similarity between two vectors X_a and X_b

$$sim(X_a, X_b) = \cos ine\left(\frac{X_a \cdot X_b}{\|X_a\| \cdot \|X_b\|}\right) \quad (76)$$

tf_{i*}: term frequency of $t_i \in V$ for a whole collection of documents D

$$tf_{i^*} = \frac{n_{i^*}}{\sum_k n_{k^*}} = \frac{n_{i^*}}{n_{**}} \tag{77}$$

RD(m): measure of the use of a meaning m ∈ M in the whole collection D

$$RD(m) = \sum_{t_i \in T(m)} \left(n_{i^*} \left(1 - I_p(t_i) \right) \right) \tag{78}$$

$$= \sum_{t_i \in T(m)} \left(\frac{n_{i^*}}{N_m(t_i)} \right)$$

Cfj(m): concept frequency coefficient for the whole collection D

$$CfD(m) = \frac{RD(m)}{\sum_k n_{k^*}} = \frac{\sum_{t_i \in T(m)} \left(n_{i^*} \cdot \left(1 - I_p(t_i) \right) \right)}{n_{**}} \tag{79}$$

maxm(t): meaning m ∈ M of a term t ∈ V with the maximum concept frequency coefficient CfD for collection D

$$\max_m(t) = \max_{m_i \in M(t)} (m_i), \tag{80}$$
$$m_i \geq m_j \Leftrightarrow CfD(m_i) \geq CfD(m_j)$$

Qe: query expansion of a user query Q such that for each term t ∈ Q, there will be included in Qe all the synonyms of t by a meaning m which has maximum concept frequency

$$Q_e = \bigcup_{t \in Q} T\left(\max_m(t) \right) \tag{81}$$

Defj(m) degree of presence of a meaning m in the document Dj

$$Def_j(m) = \frac{\sum_{t_i \in B(m)} n_{i,j}}{n_{*,j}}. \tag{82}$$

Appendix 2: Brontë daughters

Brontë From Wikipedia, the free encyclopedia

The Brontë sisters, painted by their brother Branwell, c. 1834. From left to right: Anne, Emily and Charlotte (there still remains a shadow of Branwell, which appeared after he painted himself out).

The Brontë sisters (pronounced /□brɒnti/), Charlotte (April 21, 1816 – March 31, 1855), Emily (July 30, 1818– December 19, 1848) and Anne (January 17, 1820 – May 28, 1849), were English writers of the 1840s and 1850s. Their novels caused a sensation when they were first published and were subsequently accepted into the canon of great English literature.

Three sisters emerge

The sisters grew up in in Haworth, near Keighley in West Yorkshire (the region has come to be known as Brontë Country), surviving their mother and two elder sisters into adulthood. In 1824 the four eldest Brontë daughters were enrolled as pupils at the Clergy Daughter's School atCowan Bridge. The following year Maria and Elizabeth, the two eldest daughters, became ill, left the school and died; Charlotte and Emily were brought home.

They had written compulsively from early childhood and were first published, at their own expense, in 1846 as poets under the pseudonymsCurrer, Ellis and Acton Bell. The book attracted little attention, selling only two copies. The sisters returned to prose, producing a novel each in the following year. Charlotte's *Jane Eyre*, Emily's *Wuthering Heights* and Anne's *Agnes Grey* were released in 1847 after their long search to secure publishers.

The novels attracted great critical attention and steadily became best-sellers, but the sisters' careers were shortened by ill-health. Emily died the following year before she could complete another novel, and Anne published her second novel, *The Tenant of Wildfell Hall*, in 1848, a year before her death. Upon publication *Jane Eyre* received the most critical and commercial success of all the Brontë works, continuing to this day. Charlotte's *Shirley* appeared in 1849 and was followed by *Villette* in 1853. Her first novel, *The Professor*, was published posthumously in 1857; her uncompleted fragment, *Emma*, was published in 1860; and some of her juvenile writings remained unpublished until the late twentieth century. Charlotte died at the age of 38 in 1855 after a short illness, possibly related to her pregnancy. She had married her father's curate, Arthur Bell Nicholls, less than a year earlier.

The first biography of Charlotte was written by her friend Elizabeth Gaskell and published in 1857. It helped create the myth of a doomed family living in romantic solitude.

Anne Brontë From Wikipedia, the free encyclopedia

Anne Brontë

Anne Brontë, by Charlotte Brontë, 1834

Born	January 17, 1820 Thornton, Yorkshire, England
Died	May 28, 1849 (aged 29) Scarborough, England
Occupation	Governess, novelist, poet

Anne Brontë (pronounced /□brɒnti/) (January 17, 1820 – May 28, 1849) was a British novelist and poet, the youngest member of the Brontë literary family.

The daughter of a poor Irish clergyman in the Church of England, Anne Brontë lived most of her life with her family at the remote village of Haworth on the Yorkshire moors. For a couple of years she went to a boarding school. At the age of nineteen, she left Haworth working as a governess between 1839 and 1845. After leaving her teaching position, she fulfilled her literary ambitions. She wrote a volume of poetry with her sisters (*Poems by Currer, Ellis, and Acton Bell*, 1846) and in short succession she wrote two novels: *Agnes Grey*, based upon her experiences as a governess, was published in 1847; her second and last novel, *The Tenant of Wildfell Hall*appeared in 1848. Anne's creative life was cut short with her death of pulmonary tuberculosiswhen she was only twenty-nine years old.

Anne Brontë is often overshadowed by her more famous sisters, Charlotte, author of four novels including *Jane Eyre*, and Emily, author of *Wuthering Heights*. Anne's two novels, written in a sharp and ironic style, are completely different from the romanticism followed by her sisters. She wrote in a realistic, rather than a romantic style. Her novels, like those of her sisters, have become classics of English literature.

Charlotte Brontë From Wikipedia, the free encyclopedia

Charlotte Brontë	
	
Born	April 21, 1816 Thornton, Yorkshire, England
Died	March 31, 1855 (aged 38) Haworth, Yorkshire, England
Pen name	Currer Bell
Occupation	Governess, Novelist, Poet
Genres	Novel
Influences	Robert Burns, Robert Southey, Walter Scott
Influenced	George Eliot, Thomas Hardy, Jean Rhys

Charlotte Brontë (pronounced /□brɒnti/) (April 21, 1816 – March 31, 1855) was a British novelist, the eldest of the three famous Brontë sisters whose novels have become standards of English literature. Charlotte Brontë, who used the pen name **Currer Bell**, is best known for *Jane Eyre*, one of the most famous of English novels.

Life

Charlotte Brontë was born in Thornton, Yorkshire, England, the third of six children, to Patrick Brontë (formerly "Patrick Brunty"), an Irish Anglican clergyman,

and his wife, Maria Branwell. In April 1821 the family moved a few miles to Haworth, where Patrick had been appointed Perpetual Curate. Maria Branwell Brontë died of cancer on 15 September 1821, leaving five daughters and a son to the care of her sister Elizabeth Branwell. In August 1824, Charlotte was sent with three of her sisters; Emily, Maria and Elizabeth, to the Clergy Daughters' School at Cowan Bridge inLancashire (which she would describe as Lowood School in *Jane Eyre*). Its poor conditions, Charlotte maintained, permanently affected her health and physical development and hastened the deaths of her two elder sisters, Maria (born 1814) and Elizabeth (born 1815), who died oftuberculosis in May of 1826 soon after they were removed from the school.

At home in Haworth Parsonage, Charlotte and the other surviving children — Branwell, Emily andAnne — began chronicling the lives and struggles of the inhabitants of their imaginary kingdoms. Charlotte and Branwell wrote stories about their country — Angria — and Emily and Anne wrote articles and poems about theirs — Gondal. The sagas were elaborate and convoluted (and still exist in part manuscripts) and provided them with an obsessive interest in childhood and early adolescence, which prepared them for their literary vocations in adulthood.

Charlotte continued her education at Roe Head, Mirfield, from 1831 to 1832, where she met her lifelong friends and correspondents, Ellen Nussey and Mary Taylor. During this period (1833), she wrote her novella *The Green Dwarf* under the name of Wellesley. Charlotte returned as a teacher from 1835 to 1838. In 1839 she took up the first of many positions as governess to various families in Yorkshire, a career she pursued until 1841. In 1842 she and Emily travelled to Brussels to enroll in a pensionnat run by Constantin Heger (1809 – 1896) and his wife Claire Zoé Parent Heger (1814 – 1891). In return for board and tuition, Charlotte taught English and Emily taught music. Their time at the pensionnat was cut short when Elizabeth Branwell, their aunt who joined the family after the death of their mother to look after the children, died of internal obstruction in October 1842. Charlotte returned alone to Brussels in January 1843 to take up a teaching post at the pensionnat. Her second stay at the pensionnat was not a happy one; she became lonely, homesick, and deeply attached to Constantin Heger. She finally returned to Haworth in January 1844 and later used her time at the pensionnat as the inspiration for some of *The Professor* and *Villette*.

In May 1846, Charlotte, Emily and Anne published a joint collection of poetry under the assumed names of Currer, Ellis and Acton Bell. Although the book failed to attract interest (only two copies were sold), the sisters decided to continue writing for publication and began work on their first novels. Charlotte continued to use the name 'Currer Bell' when she published her first two novels. Of this, Brontë later wrote:

> "Averse to personal publicity, we veiled our own names under those of Currer, Ellis and Acton Bell; the ambiguous choice being dictated by a sort of conscientious scruple at assuming Christian names positively masculine, while we did not like to declare ourselves women, because -- without at that time suspecting that our mode of writing and thinking was not what is called 'feminine' -- we had a vague impression that authoresses are liable to be looked on with prejudice; we had noticed how critics sometimes use for their chastisement the weapon of personality, and for their reward, a flattery, which is not true praise." [1]

Her novels were deemed coarse by the critics. Much speculation took place as to who Currer Bell really was, and whether Bell was a man or a woman.

Charlotte's brother, Branwell, the only son of the family, died of chronic bronchitis and marasmusexacerbated by heavy drinking in September 1848, although Charlotte believed his death was due totuberculosis. Emily and Anne both died of pulmonary tuberculosis in December 1848 and May 1849, respectively.

Charlotte and her father were now left alone. In view of the enormous success of *Jane Eyre*, she was persuaded by her publisher to visit London occasionally, where she revealed her true identity and began to move in a more exalted social circle, becoming friends with Harriet Martineau, Elizabeth Gaskell, William Makepeace Thackeray and G. H. Lewes. Her book had sparked a movement in regards to feminism in literature. The main character, Jane Eyre, in her novel Jane Eyre, was a parallel to herself, a woman who was strong. However, she never left Haworth for more than a few weeks at a time as she did not want to leave her aging father's side.

In June 1854, Charlotte married Arthur Bell Nicholls, her father's curate, and became pregnant very soon thereafter. Her health declined rapidly during this time, and according to Gaskell, her earliest biographer, she was attacked by "sensations of perpetual nausea and ever-recurring faintness."[2] Charlotte and her unborn child died March 31, 1855. Her death certificate gives the cause of death as phthisis (tuberculosis),

but many biographers suggest she may have died from dehydration and malnourishment, caused by excessive vomiting from severe morning sickness. There is also evidence to suggest that Charlotte died from typhus she may have caught from Tabitha Ackroyd, the Brontë household's oldest servant, who died shortly before her. Charlotte was interred in the family vault in The Church of St. Michael and All Angels, Haworth, West Yorkshire, England.

The Life of Charlotte Brontë, the posthumous biography of Charlotte Brontë by fellow novelist Elizabeth Gaskell, was the first of many biographies about Charlotte to be published. Though quite frank in places, Gaskell suppressed details of Charlotte's love for Heger, a married man, as being too much of an affront to contemporary morals and as a possible source of distress to Charlotte's still-living friends, father and husband (Lane 1853 178-183). Gaskell also provided doubtful and inaccurate information about Patrick Brontë, claiming, for example, that he did not allow his children to eat meat. This is refuted by one of Emily Brontë's diary papers, in which she describes the preparation of meat and potatoes for dinner at the parsonage, as Juliet Barker points out in her recent biography, The Brontës. It was discovered that Charlotte wrote 20 manuscript pages of a book but died before she could finish; however another author, Clare Boylan, took up the project and the novel was released under the title of Emma Brown: A Novel from the Unfinished Manuscript by Charlotte Bronte in 2003.

Emily Brontë From Wikipedia, the free encyclopedia

Emily Jane Brontë

Portrait by Branwell Brontë

Born	July 30, 1818 Thornton, Yorkshire, England
Died	December 19, 1848 (aged 30) Haworth, Yorkshire, England
Occupation	Novelist, poet

Emily Jane Brontë (pronounced /□brɒnti/); (July 30, 1818 – December 19, 1848) was a Britishnovelist and poet, now best remembered for her only novel *Wuthering Heights*, a classic of English literature. Emily was the second eldest of the three surviving Brontë sisters, being younger thanCharlotte and older than Anne. She published under the masculine pen name **Ellis Bell**.

Biography

Emily Brontë was born in Thornton, near Bradford in Yorkshire to Patrick Brontë and Maria Branwell. She was the younger sister of Charlotte Brontë and the fifth of six children. In 1824, the family moved to Haworth, where Emily's father was perpetual curate, and it was in these surroundings that their literary oddities flourished. In childhood, after the death of their mother, the three sisters and their brother Patrick

Branwell Brontë created imaginary lands, which were featured in stories they wrote. Little of Emily's work from this period survived, except for poems spoken by characters (*The Brontës' Web of Childhood*, Fannie Ratchford, 1941).

In 1842, Emily commenced work as a governess at Miss Patchett's Ladies Academy at Law Hill School, near Halifax, leaving after about six months due to homesickness. Later, with her sister Charlotte, she attended a private school in Brussels run by Constantin Heger and his wife, Claire Zoé Parent Heger. They later tried to open up a school at their home, but had no pupils.

It was the discovery of Emily's poetic talent by Charlotte that led her and her sisters to publish a joint collection of their poetry in 1846, *Poems by Currer, Ellis, and Acton Bell*. To evade contemporary prejudice against female writers, the Brontë sisters adopted androgynous first names. All three retained the first letter of their first names: Charlotte became Currer Bell, Anne became Acton Bell, and Emily became Ellis Bell. In 1847, she published her only novel, *Wuthering Heights*, as two volumes of a three volume set (the last volume being *Agnes Grey* by her sister Anne). Its innovative structure somewhat puzzled critics. Although it received mixed reviews when it first came out, the book subsequently became an English literary classic. In 1850, Charlotte edited and published *Wuthering Heights* as a stand-alone novel and under Emily's real name.

Emily's health, like her sisters', had been weakened by the harsh local climate at home and at school. She caught a cold during the funeral of her brother in September which led to tuberculosis. Consequently, having refused all medical help, she died on December 19, 1848 at about two in the afternoon. She was interred in the Church of St. Michael and All Angels family capsule, Haworth, West Yorkshire, England.

Appendix 3: Generalized Constraints and Protoform Notation

Appendix 3.1. General Constraints

Example	Expressed as	Notation
The rose is red	the_rose has_attrib red degree m	Entity has_attrib Property degree Degree
	the_rose has_chrc colour.	Entity has_chrc Characteristic.
The rose is red	the_rose has_chrc colour value red.	Entity has_chrc Characteristic value Property.
Mary is young	'Mary' has_chrc age value young degree m modality possibilistic	Entity has_chrc Characteristic value Property modality M.
Mary is 25.	'Mary' has_chrc age value 25 degree m modality equality	Entity has_chrc Characteristic value Property modality M.
Mary sings beautifully	Mary do sing manner beautifully	
John is a doctor	John isA doctor.	Entity isA Category.
John and Mary are doctors	doctors isASetOf doctor. John isMemberOf doctors. Mary isMemberOf doctors. [John, Mary] isASetOf doctor.	Set isASetOf ElementType Elem isMemberOf Set.
Doctors are educated	doctors isASetOf doctor. doctors has_attrib educated.	
The car is outside	the_car has_chrc place value outside.	Entity has_chrc place value outside.
The Raiders is the winning team.	winning_team subClassOf team. the_Raiders isA winning_team.	SubC subClassOf SupraC
Arthur Conan Doyle was a prolific writer.	Arthur_Conan_Doyle was a_prolific_writer. a_prolific_writer subClassOf writer. a_prolific_writer has_attrib prolific.	

Appendix 3.2. Comparative sentences

Example	**Expressed as**	**Notation**
Mary is younger than Tom.	compare(Mary, is, more, young, Tom, M)	compare(E1,is, more, Adj, E2, M)
Mary works harder than Tom	compare(Mary, works, more, hard, Tom, M).	compare(E1,do, more, Adj, E2, M)
Mary is more intelligent than Tom.	compare(Mary, is, more, intelligent, Tom, M).	compare(E1,is, more, Adj, E2, M)
Mary travels more frequently than Tom	compare(Mary, travels, more, frequently, Tom, M).	compare(E1,do, more, Adv, E2, M)
Eloise has more children than Chantal	compare(Eloise, has, more, children, Chantal, M).	compare(E1,has, more, Prop, E2, M)
Eloise has more money than Chantal.	compare(Eloise, has, more, money, Chantal, M).	compare(E1, has, more, Prop, E2, M)
Tom is less intelligent than Mary.	compare(Tom, is, less, intelligent, Mary, M).	compare(E1,is, less, Adj, E2, M)
Tom travels less *frequently* than Mary.	compare(Tom, travels, less, frequently, Mary, M).	compare(E1,do, less, Adv, E2, M)
Chantal has less money than Eloise.	compare(Chantal, has, less, money, Eloise, M).	compare(E1, has, less, Prop, E2, M)
Chantal has fewer children than Eloise	compare(Chantal, has, less, children, Eloise, M).	compare(E1, has, less, Prop, E2, M)
Peter is as old as John.	compare(Peter, is, as_as, old, John, M).	compare(E1, is, as_as, Adj, E2, M)
Peter runs as fast as John.	compare(Peter, runs, as_as, fast, John, M).	compare(E1, do, as_as, Adv, E2, M)
Mont Blanc is not as high as Mount Everest.	compare(Mont_Blanc, is, not_as, high, Mount_Everest, M).	compare(E1, is, not_as, Adj, E2, M)
John does not work as hard as Tom.	compare(John, work, not_as, hard, Tom, M).	compare(E1, do, not_as, Adv, E2, M)
They have as many children as us.	compare(they, have, as_as, children, us, M).	compare(E1, have, as_as, Prop, E2, M)
Tom has as few books as Jane.	compare(Tom, has, as_as, books, Jane, M).	compare(E1, have, as_as, Prop, E2, M)
John eats as much food as Peter.	compare(John, eats, as_as, food, Peter, M)	compare(E1, do, as_as, Prop, E2, M)
Jim has as little food as Sam.	compare(Jim, has, as_as, food, Sam, M)	compare(E1, has, as_as, Prop, E2, M)

Appendix 3.3. Superlative sentences

Example	Expressed as	Notation
Mary is the youngest child in the family.	superlative(Mary, is, most, young, child_in_the_family, M)	superlative(E1, is, most, Adj, Group, M)
Mary works the hardest of all the students.	superlative(Mary, works, most, hard, of_all_the_students, M)	superlative(E1, do, most, Adv, Group, M)
Mary is the most intelligent child in the family.	superlative(Mary, is, most, intelligent, child_in_the_family, M)	superlative(E1, is, most, Adj, Group, M)
Mary travels the most frequently in the classroom.	superlative(Mary, travels, most, frequently, in_the_classroom, M)	superlative(E1, do, most, Adv, Group, M)
John is the least intelligent in the office.	superlative(John, is, least, intelligent, in_the_office, M)	superlative(E1, is, least, Adj, Group, M)
John travels the least frequently of all his friends	superlative(John, travels, least, frequently, of_all_his_friends, M)	superlative(E1, do, least, Adv, Group, M)

Appendix 3.4. Protoforms

Protoform	**Abstraction**	**Example**
S.A is *V*	*S* subject *Monika*, *A* attribute *age*, *V* value *young*	*Monika is young*
	S subject *Monika*, *A* attribute *age*, *V* value *very intelligent*	*Monika is very intelligent*
	S subject *Monika*, *A* attribute *place*, *V* value *outside*	*Monika is outside*
	S subject *Monika*, *A* attribute *place*, *V* value *in the classroom*	*Monika is in the classroom*
S is *C*	*S* subject *John* *C* is the subject's category, *doctor*	*John is a doctor*
C($S_1.A$, $S_2.A$) is *V*	*C* comparison relation, S_1 subject *Mary* S_2 subject *Tom* *A* attribute *age* *V* value *younger* (or *more young*)	*Mary is younger than Tom*
	C comparison relation, S_1 subject *Mary* S_2 subject *Tom* *A* attribute intelligence *V*: *more intelligent*	*Mary is more intelligent than Tom*
	C comparison relation, S_1 subject *Tom* S_2 subject *Mary* *A* attribute intelligence *V*: *less intelligent*	*Tom is less intelligent than Mary*
	C comparison relation, S_1 subject *Peter* S_2 subject *John* *A* attribute age *V*: *as old*	*Peter is as old as John*
	C comparison relation, S_1 subject *Mont Blanc* S_2 subject *Mount Everest* *A* attribute height *V: not as high*	*Mont Blanc is not as high as Mount Everest*
C($S_1.D$, $S_2.D$) is *V*	*C* comparison relation, S_1 subject *Mary* S_2 subject *Tom* *D* action, *works* *V* value *harder or more hard*	*Mary works harder than Tom*
	C comparison relation, S_1 subject *Mary*	*Mary travels more frequently than Tom*

	S_2 subject *Tom* *D* action, *travels* *V* value *more frequently*	
	C comparison relation, S_1 subject *Tom* S_2 subject *Mary* *D* action, *travels* *V* value *more frequently*	*Tom travels less frequently than Mary*
	C comparison relation, S_1 subject *Peter* S_2 subject *John* *D* action, *runs* *V* value *as fast*	*Peter runs as fast as John*
	C comparison relation, S_1 subject *John* S_2 subject *Peter* *D* action, *eats* *V* value *as much food*	*John eats as much food as Peter*
	C comparison relation, S_1 subject *John* S_2 subject *Peter* *D* action, *work* *V* value *as hard*	John does not work as hard as Peter
C(S_1.*P*, S_2.*P*) is *V*	*C* comparison relation, S_1 subject *Eloise* S_2 subject *Chantal* P property, *has* *V* value *more children*	*Eloise has more children than Chantal*
	C comparison relation, S_1 subject *Eloise* S_2 subject *Chantal* P property, *has* *V* value *more money*	*Eloise has more money than Chantal*
	C comparison relation, S_1 subject *Chantal* S_2 subject *Eloise* P property, *has* *V* value *less money*	*Chantal has less money than Eloise*
	C comparison relation, S_1 subject *Chantal* S_2 subject *Eloise* P property, *has* *V* value *fewer children*	*Chantal has fewer children than Eloise*
	C comparison relation, S_1 subject *Chantal* S_2 subject *Eloise* P property, *has* *V* value *as many children*	*Chantal has as many children as Eloise*
	C comparison relation, S_1 subject *Chantal*	*Chantal has as few books as Eloise*

	S_2 subject *Eloise* P property, *has* *V* value *as few books*	
	C comparison relation, S_1 subject *Chantal* S_2 subject *Eloise* P property, *has* *V* value *as little food*	*Chantal has as little food as Eloise*

Appendix 3.5. Superlative

Protoform	**Abstraction**	**Example**
S(*E.A*, *C.A*) is *V*	*S* superlative relation, *E* entity, *Mary*, *A* attribute *age*, *C* collection of entities *child in the* family *V* value *youngest*	*Mary is the youngest child in the family*
	S superlative relation, *E* entity, *Mary*, *A* attribute *intelligence* *C* collection of entities *child in the* family *V* value *most intelligent*	*Mary is the most intelligent child in the family*
	S superlative relation, *E* entity, *John*, *A* attribute *intelligence* *C* collection of entities *in the office* *V* value *least intelligent*	*John is the least intelligent in the office*
S(*E.D*, *C.D*) is *V*	*S* superlative relation, *E* entity, *Mary*, *D* verb, *works* *C* collection of entities *all the students* *V* value *hardest*	*Mary works the hardest of all the students*
	S superlative relation, *E* entity, *Mary*, *D* verb, *travels* *C* collection of entities *in the classroom* *V* value *most frequently*	*Mary travels the most frequently in the classroom*
	S superlative relation, *E* entity, *John* *D* verb, *travels* *C* collection of entities *all his friends* *V* value *least frequently*	*John travels the least frequently of all his friends*