Databases and Information Systems Group (AG5)
Max-Planck-Institute for Computer Science
Saarbrücken, Germany

Automatic ontology extraction for document classification

by

Natalia Kozlova

Supervised by

Prof. Dr.-Ing. Gerhard Weikum

Dipl.-Inform. Martin Theobald

A thesis submitted in conformity with the requirements
for the degree of Master of Science

Computer Science Department
Saarland University

February, 2005

Abstract

The amount of information in the world is enormous. Millions of documents in electronic libraries, thousands of them on each personal computer waiting for the expert to organize this information, to be assigned to appropriate categories. Automatic classification can help. However, synonymy, polysemy and word usage patterns problems usually arise. Modern knowledge representation mechanisms such as ontologies can be used as a solution to these issues. Ontology-driven classification is a powerful technique which combines the advantages of modern classification methods with semantic specificity of the ontologies.

One of the key issues here is the cost and difficulty of the ontology building process, especially if we do not want to stick to any specific field. Creating a generally applicable but simple ontology is a challenging task. Even manually compiled thesauri such as WordNet can be overcrowded and noisy.

We propose a flexible framework for efficient ontology extraction in document classification purposes. In this work we developed a set of ontology extraction rules. Our framework was tested on the manually created corpus of Wikipedia, the free encyclopedia. We present a software tool, developed with regard to the claimed principles. Its architecture is open for embedding new features in.

The ontology-driven document classification experiments were performed on the Reuters collection. We study the behavior of different classifiers on different ontologies, varying our experimental setup. Experiments show that the performance of our system is better, in comparison to other approaches. In this work we observe and state the potential of automatic ontology extraction techniques and highlight directions for the further investigation.

I hereby declare that this thesis is entirely my own work and that I have not used any other media than the ones mentioned in the thesis.

Saarbrücken, March 9, 2005

Natalia Kozlova

Acknowledgment

I would like to thank my supervisors Gerhard Weikum and Martin Theobald for their guidance and support during this thesis. Gerhard was extremely motivating. Each time he pointed to the important aspects of my problem drawing a big picture, helping to work on it. Martin has always been near and helped with ongoing problems; his gentle guidance and valuable advise helped me a lot. Friendly atmosphere and democratic relationships between members of AG 5 encourage and help.

I would like to thank Kerstin Meyer Ross, IMPRS coordinator, for the great help in solving problems of very different kinds.

I thank my family for their love and support. I thank my friends for being near, even if some of them were very far.

Contents

List of Figures

Chapter 1

Introduction

Formal ontology deals with the interconnections of ***things****, with objects and properties, parts and wholes, relations and collectives.*

Edmund Husserl (1859-1938)

1.1 Motivation

What is the most valuable thing in today's world? What can be priceless or can cost a lot, or can be useless when received not in time? One can answer: it is Information. In its most general sense Information, Knowledge has been a subject of the great importance since the beginning of the human history. The amount of available knowledge grows from day to day and the requirements for the simplicity and speed of requests grow in conference. How should we keep the data *accessible*? An ontology is one of the ways to put things in order.

The word "Ontology" comes from the Greek *ontos* for being and *logos* for word. The term "Ontology" came from Philosophy and operates with objects and their ties, things and their categories. About fifteen years ago this term was adopted by scientists who dealt with Knowledge Engineering. Since 90'th ontologies become more and more demanded in the tasks of information organization and management. So what is an ontology? There are plenty of definitions, let us sketch the most popular of them:

An ontology is an explicit specification of a conceptualization

or, closer to engineering world

Ontologies as a way of specifying content-specific agreements for the sharing and reuse of knowledge among software entities

(Gruber, 1993 [1]).

An ontology refers to an engineering artifact, constituted by a specific vocabulary used to describe a certain reality, plus a set of explicit assumptions regarding the intended meaning of the vocabulary words.

(Guarino, 1998 [8]).

During the last decade we observe an increasing interest to ontologies. The area of applicability for ontologies is wide: information retrieval and extraction, information systems design and enterprise integration, natural language processing, database design, conceptual modeling.

Concerning ontologies subdivision, ontologies are usually classified along different aspects of their application and structure. One of the possible criteria is the amount and type of structure of the conceptualization [9], [11]. According to this parameter there are three ontology types: Terminological ontologies (i.e. lexicons), Information ontologies (i.e. database schemata), Knowledge modeling ontologies (specify conceptualizations of the knowledge).

In this work we concentrate our attention on the problem of Terminological ontologies application in Information Retrieval, particularly in the text classification task. We will return to the issue of classification in more details later in this chapter.

1.2 Problem Definition

As we stated, there is a big interest to the research in ontology engineering. The development of ontological systems is a very complicated and expensive process. Even now there is no existing framework that requires no human work. Methodologies, suggested by some of the research groups,

help to develop big complicated system. However, there is still a lack of approaches, aimed to the fast ontology extraction. Another issue is the level of formality. For classification purposes we do not need an elaborated ontology, which includes metaphysical concepts and has a strict reasoning mechanism. We need to have a possibility to capture a reasonable number of concepts from the document, to find their place inside the ontology and to ask for the environment of the concept inside the concepts hierarchy. This desired environment should be expressive enough to allow us to solve ambiguity problems as much as possible and compact enough to fight synonymy/polysemy problems. This way we state that we interested more in the word's context than in its place in thehierarchy or certain affiliation detection for it. Thus, we narrow the area of ontological studies that can be applied to our problem.

There are not so many lexical ontologies, naming the most specific, we can mention WordNet, SENSUS [12] and GUM (Generalized Upper Model) [16]. All of them were developed manually. They will be described in details later in chapter 2. Between these three, only WordNet can be considered as strictly lexical ontology, the others are hybrid.

Back to the question of the suitable ontology, we could say that for the purpose of classification each of the mentioned ontologies has one similar problem. Every unit is overloaded with meaning and relations. Plenty of different meanings (polysemy) for each of the words in WordNet make disambiguation difficult. The second problem concerns the fact that these ontologies have been created manually. In this way the process of creation consumes a lot of resources and takes a lot of time.

Hence, there is a lack of tools, which are aimed to the end-user data structuring and automatic ontology extraction. We think that development of a general-purpose or conversely, specific ontology from the user-chosen semi-structured document collection is a challenging task. We pose a problem of construction the framework for automatic ontology extraction from existing collections. The second point of interest in this thesis is a problem of the ontology-driven classification.

1.3 Contribution

In according to the fact that, naturally, humans can judge better than machines, we decide to exploit the precious knowledge, which at the same time is highly available. We took Wikipedia, a broad collection, created by many authors. In fact any human-developed text corpus with internal structure and cross-references in systematic format can be used. We developed a tool that creates a structured representation of the collection in the form of ontology. Varying policies, applied to the ontology extraction process, we can produce ontologies of different sizes. We extracted four sample ontologies and used them as a support for the ongoing classification.

The thesis is organized as follows: the problems and challenges of ontologies creation and extraction are described in Chapter 2 as well as the basics of classification. The discussion of related work is also given there. The ontology extraction framework itself is described in Chapter 3, where we present developed heuristics and the methodology, suggested. Chapter 4 is devoted to implementation issues and describes the system architecture. Chapter 5 presents experiments. The conclusion about the work and the possible directions of the future work are in Chapter 6.

Chapter 2

The State of the Art

2.1 Ontology creation problem

> *Ideally, an ontology builder should first have a clear idea of why the ontology is wanted, what it will be used for, and possible mechanisms for use* [5].

The origins of the current state of the problem were defined by ARPA Knowledge Sharing Effort [15] and later by T. Gruber in 1993 [1], one of the authors. He identified a basic design criteria for the ontology creation task. Since that time, a number of researchers worked on this problem. In 1996 M. Uschold described several approaches for building ontologies. He suggested to differentiate ontologies by the purpose of usage. With regard to this principle, he proposed a unified methodology for ontology creation. His work, in its turn, was partially based on the earlier work of N. Guarino. He discussed the philosophical and epistemological aspects of ontologies [7] as well as the formal approach to describe ontologies, conceptualization and ontological levels. In his further works N. Guarino defended the systematic introduction of formal ontological principles in knowledge engineering, currently known as "Formal ontology" [6]. Since that time, lots of researchers worked on this problem. Additional information and references can be found in [1],[2],[3],[5],[8],[11],[4],[22],[15],[10].

2.1.1 Ontology definition and classification

The subject of the ontology is the study of categories of things that exist or may exist in some domain [22]. Ontology describes the subject matter using the notions of *concepts, instances, relations, functions and axioms.* Concepts in the ontology can be hierarchically organized in taxonomies to allow inheritance. There is no global accordance in definitions, so we will try to adopt existing ones for our task. The elements of ontologies, according to [1], [11] are:

- *Concepts.* Each single concept **c** is a *meaningful unit.* A concept can be anything about which something is said and, therefore, could also be the description of a task, function, action, strategy, reasoning process, etc[11].

- *Relations.* Relations represent interaction between concepts of the domain. They are formally defined as any subset of possible relations between concepts, that is: $\mathbf{R} \subseteq C$ xC.

- *Axioms.* Base rules of the ontology.

- *Domain.* Domain **D** defines what can be represented using this *conceptualization.*

Following these notion, we can formally define the ontology: Let **C** define a set of known concepts, each concept $\mathbf{c} =< u >$ contains one or more meaningful units. Let **R** define a relations between existing concepts. Each relation $\mathbf{r} \in R$. Let $\mathbf{A} =< a >$ be the set of axioms that defines concepts and relations superposition, their structure and type.
Then **ontology O** $=<$ **C, A, R** $>$ given current domain **D**.

As far as we defined components, we can implement the ontology. Ontologies can be classified by the method (language) of implementation. Uschold [5] used three dimensions to classify an ontology.

1. **Formality**
   Highly informal; Expressed loosely in natural language; Structured informal: expressed in a restricted and structured form of natural language; Semi-formal: expressed in an artificial formally defined language;

Rigorously formal: meticulously defined terms with formal semantics, theorems and proofs.

2. **Purpose**
   Communication; Inter-Operability; Systems Engineering Benefits: Re-Usability, Knowledge Acquisition, Reliability, Specification.

3. **Subject Matter**
   Subject considered separately from the problems or tasks that may arise relevant to the subject; The subject matter of problem solving; The subject matter of knowledge representation languages;

Using another well-known classification by Guarino [9], ontologies are subdivided along two dimensions and inside each dimension there are subtypes:

1. **The amount and type of structure of the conceptualization:**
   Terminological ontologies (i.e. lexicons); Information ontologies (i.e. database schemata); Knowledge modeling ontologies (specify conceptualizations of the knowledge)

2. **Subject of the conceptualization:**
   Application ontologies; Domain ontologies; Generic/Common ontologies; Representation ontologies.

One can identify interconncetions in these two classifications: A level of *genericity* is related to **Purpose**. An ontology in the first category of **Subject Matter** is frequently called a domain ontology. An ontology of the second category is usually called a *task-*, *method-* or *problem-solving* ontology. The terms "representation ontology" or "meta-ontology" are used to refer to ontologies of the third category.

Application ontologies are used for modeling a particular application, collect information about the task and means. They are almost not reusable. Domain ontologies capture information about the concepts within a domain; their relationships. They try to model principles, governing that domain. This type of ontologies is reusable w.r.t. the given domain. Generic/common ontologies have a vocabulary, related to general behavior and nature of things, events, time, space, causality, function and so on. Can be reused across

domains due to their generality. Representation ontologies contain the representation primitives used to formalize knowledge in given representation paradigms.
In this work we will deal with Terminological Generic Ontology.

2.1.2 Existing Approaches for Ontology Design

Gruber was the first, who discussed ontologies as design artifacts, and outlined a set of design criteria to guide the ontology development [1].
These criteria are:

1. **Clarity**: High level of objectivity and indepedency of context in definitions, formalisms.
2. **Coherence**: Axioms and inferences should be logically consistent.
3. **Extendibility**: Transparent architecture and easy-of-use
4. **Minimal encoding bias**: Encoding bias occurs when a choice of representation is made simply for the convenience of notation. The conceptualization should be specified at he knowledge level without depending on a particular symbol-level encoding.
5. **Minimal ontological commitment**: As few claims as possible about the world being modeled. Can be minimized by specifying the weakest theory (allowing the most models) and defining only those terms that are essential to the communication of knowledge consistent with that theory.

With regard to these principles, the process of ontology design will require to make tradeoffs among these criteria. They are not inherently at odds to each other. However, emphasizing Clarity (formal model) we restrict the possible interpretations of terms. Thus Extendibility is sacrificed, and so on. The process of ontology creation, based on this principles, requires a careful study of the possible costs of making decision.

In 1996 Uschold proposed a so-called "Unified Approach" for ontology creation. It leans on his ontology classification, described earlier and consists

of identifying the specified criteria. These criteria are Purpose, Formality and Subject Matter.

At the beginning the Purpose needs to be very clear. One should identify *Who* will be the users - range, skills, competency level and so on. Then we need to find out which area of {*Communication, Inter-Operability, Systems Engineering Benefits*} is closer to the subject. In other words, the general requirements to ontology should be specified at this step. The choice for Level of Formality follows from Purpose and from the area of applicability analysis.

The last parameter is Subject Matter. The output of this step is a set of concepts and terms covering the predestination area. It can be obtained by developing various working scenarios or brainstorming and trimming the result.

Finally the ontology should be implemented, using one of the existing tools.

Another methodology is proposed by Guarino. It uses theory of parts, theory of wholes, theory of identity, theory of dependence and theory of universals. He summarized the basic design principles that require to (1) be clear about the domain; (2) take identity seriously; (3) isolate a basic taxonomic structure; and (4) identify roles explicitly. For more details see [8].

As we could see, these two most representative scenarios (Uschold, Guarino) and the others existing (every ontology engineer introduces his own details) are common in one thing: they are greatly dependent on human guesses and couldn't be processed automatically. However a number of ontologies were developed meantime and in the section 2.2 we will present a brief sketch of the most interesting ones. Interested reader can find additional information and references in [26], [11].

2.1.3 RDF and OIL

There is a new stream in the area of ontologies development. The problem of distributed Internet services brings new requirements and poses new problems. The intensive research in this area began with the populariza-

tion of idea of "Semantic Web". Let us briefly describe RDF and OIL, two basic technologies for maintaining ontologies over the Internet and provide ontological services.

RDF. The Resource Description Framework (RDF) is a language for representing information about resources in the World Wide Web [32]. It is particularly intended for representing metadata about Web resources, such as the title, author, and modification date of a Web page, copyright and licensing information about a Web document, or the availability schedule for some shared resource. RDF can also be used to represent information about on-line shopping facilities - specifications, prices, and availability of items. It can also be used in the description of a Web user's preferences for information delivery.

The main point here is that RDF is intended for situations where this information needs should be processed by applications. RDF provides a common framework for expressing this information so it can be exchanged between applications without loss of meaning. The ability to exchange information between different applications means increased availability and reusability of knowledge.

RDF is based on the idea of identifying things using Web identifiers (called Uniform Resource Identifiers, or URIs), and describing resources in terms of simple properties and property values. This enables RDF to represent simple statements about resources as a graph of nodes and arcs representing the resources, and their properties and values. RDF also provides an XML-based syntax (called RDF/XML) for recording and exchanging these graphs.

OIL. OIL initiative is also claim that seamless exchange of information become the key issue for web deployments. [33]. It introduces the notion of ontologies, which provide a way of capturing a shared understanding of terms that can be used by humans and programs to aid in web information exchange. The ontology needs to be specified in some language. OIL (ontology inference layer) is a proposal for a layered approach to a standard ontology language. It includes modelling constructs found in many widely knowledge representation languages, is compatible with RDFS, and includes

a precise semantics for describing term meanings (and thus also for describing implied information). OIL combines the widely used modelling primitives from frame-based languages with the formal semantics and reasoning services provided by description logics. Its layers are shown on Figure 2.1.

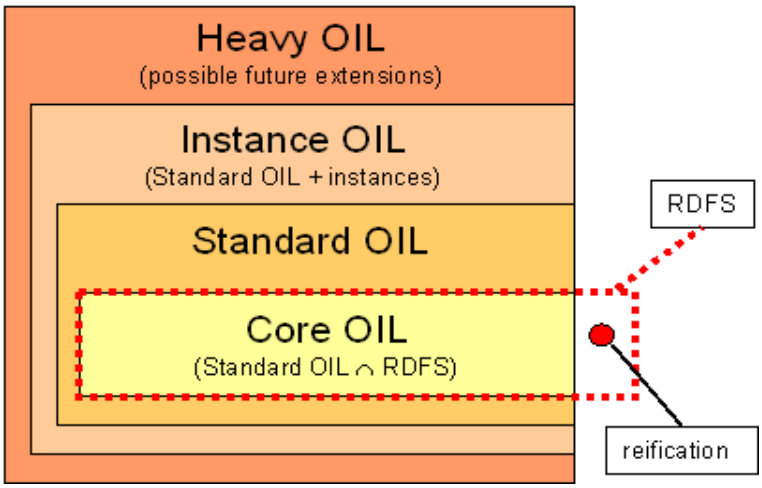


Figure 2.1: ConUtils class

Core OIL coincides largely with RDF Schema. This means that even simple RDF Schema agents are able to process the OIL ontologies, and pick up as much of their meaning as possible with their limited capabilities.

Standard OIL is a language intended to capture the necessary mainstream modelling primitives that both provide adequate expressive power and are well understood thereby allowing the semantics to be precisely specified and complete inference to be viable.

Instance OIL includes a thorough individual integration. While the previous layer - Standard OIL - included modelling constructs that allow individual fillers to be specified in term definitions, Instance OIL includes a full-fledged database capability. It has the same schema as Standard-OIL; the instances are described directly in RDF.

Heavy OIL may include additional representational (and reasoning) capabilities. Its syntax and RDF Schema are not yet defined.

2.2 Related work

2.2.1 Methodologies and existing systems

Cyc Cyc [17] is a large ontology that provides a vast amount of common sense knowledge. Cyc grew up from one of the important parts of the CYC project, Cyc Knowledge Base. It based on the micro theories, each of which captures the knowledge and reasoning required for different domains from different viewpoints such as space, time, causality. CYC Ontologies are implemented in CycL, a formal language whose syntax derives from first-order predicate calculus. The vocabulary of CycL consists of terms, which are combined into meaningful CycL expressions, ultimately forming meaningful closed CycL sentences. A set of such sentences forms a knowledge base. Cyc is Common ontology.

TOVE TOVE (Toronto Virtual Enterprise) [18] approach compose an integrated enterprise model with reasoning support, based on core ontologies. Core ontologies can be product ontology, service ontology, transportation ontology, inventory ontology, product design ontology, goals ontology and many others. TOVE provides a formalized method to combine them into one system. The methodology includes the following steps: define a set of Motivating Scenarios and a set of Informal Competency Questions that the ontology must answers due to these scenarios, define the Terminology of the ontology with First-Order Logic, define the semantics and constraints on the terminology. It creates a reusable domain ontology.

Enterprise Ontology The Enterprise Ontology [19] maintains a vocabulary, relevant to business processes and enterprises. It was developed in the Enterprise Project by Stanford University's Knowledge Systems Lab based on the methodology, described earlier in subsection 2.1.2 . The terms are grouped under dimensions of: Meta-Ontology and Time (e.g. Entity, Relationship, Role); Processes and Planning; Structure of organizations; High level planning for an enterprise; Marketing and Selling. It creates a reusable domain ontology.

On-To-Knowledge On-To-Knowledge [20] is a project in the European Commission Information Society Technologies (IST) Program. The methodology provides guidelines for introducing knowledge management concepts and tools into enterprises. It includes the identification of goals that should be achieved and based on the analysis of business processes and the different roles knowledge workers play in organizations. OIL (Ontology Interchange Language) is a standard language proposed by OnToKnowledge project. It used three paradigms: frame-based modeling with semantics based on description logic and syntax based on web standards such as XML schema and RDF schema.

2.2.2 Closely related existing systems

WordNet. The work on WordNet began about 20 years ago in Cognitive Science Lab of Princeton and the system is still evolving. WordNet is the lexical database based on psycholinguistic principles [13]. Originally it was developed for English, but nowadays one can find a significant number of WordNets for many languages from Russian to Old Persian. The basic information unit called "synset", which is the set of synonyms that are interchangeable in the particular context. Synonyms represent different meanings. WordNet contains about 114648 nouns and 79689 synsets. Various kinds of semantic relations are maintained among synsets.

WordNet defines the vocabulary of a language as a set **W** of pairs *(f,s)*, where f is a word or utterance and a sense s is an element from a given set of meanings. A word that has more than one sense is polysemous; two words that share at least one sense in common are said to be synonymous. WordNet uses sets of synonyms (synsets) to represent word senses.

The morphology is defined in terms of a set M of relations between word forms. The semantics is defined in terms of a set **S** of relations between word senses. The semantic relations into which a word enters determine the definition of that word. WordNet includes the following upper-level semantic relations:

- Synonymy is a symmetric relation between word forms. It is WordNets

basic relation.

- Antonymy (opposing-name) is also a symmetric semantic relation between word forms
- Hyponymy (sub-name) and its inverse, hypernymy (super-name), are transitive relations between synsets. Because there is usually only one hypernym, this semantic relation organizes the meanings of nouns into a hierarchical structure.
- Meronymy (part-name) and its inverse, holonymy (whole-name), are complex semantic relations.
- Troponymy (manner-name) is for verbs what hyponymy is for nouns, although the resulting hierarchies are much shallower. Entailment is also present.

The terms, contained in the synsets, are tagged with part-of-speech information. WordNet also contains so-called glosses. Gloss is a sequence of words, a sentence or a number of them, describing the particular term.

SENSUS. SENSUS is a natural language based ontology which goal is to provide a broad conceptual structure for work in machine translation. It contains a simple taxonomic structure (no meaningful axioms) and around 50000 concepts. SENSUS was developed by merging and extracting information from existing electronic resources, including WordNet and English Dictionary. One can create a domain-specific ontology by enriching the existing structure. The primary source for the SENSUS body was the semantic database of WordNet. To construct the main body of the ontology, they automatically connected WordNet concepts and English lexical items by discovering pairs of corresponding senses. In addition to housing the symbols to represent semantic meaning, the Ontology contains pointers from each symbol to appropriate lexical items in various languages. It contains a simple taxonomic structure and no axioms.

The topmost levels of the SENSUS, the Ontology Base (OB), consist of approx. 400 terms. The OB contains nodes that represent generalized distinctions required for the processing of the parsers, analyzers, and the gener-

ator. While the processing requirements of each lexeme are stored either in

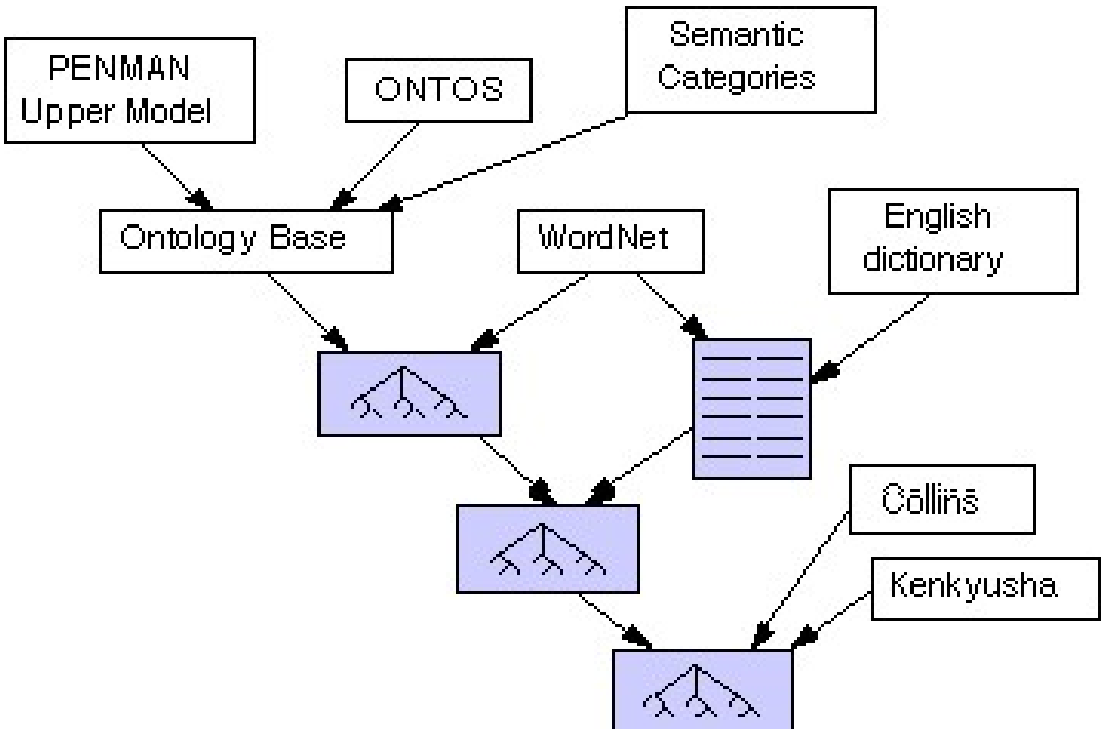


Figure 2.2: SENSUS merging scheme

the lexicon (for morphological and syntactic information) or in the ontology body (for semantic information), general semantic and syntactic patterns are captured as nodes in the OB. The OB is a merge of the Penman Upper Model (based on Systemic-Functional Linguistics), the top-level ONTOS ontology (a semantic network), and, for nouns, the LDOCE semantic categories. The hierarchy is shown on Figure 2.2. For details see [12].

2.3 Text categorization problem

There are a lot of approaches to systematization, beginning from the Dewey decimal system to modern librarian catalogs. Topic directories play the same role in the Internet. Nowadays Yahoo! Directory and Open Directory Project became the most elaborated ones. These directories contain the

most interesting (authoritative) web pages in each of the categories. However, web pages comprising the content of these categories are still manually chosen. The problem of automatic classification arises here and in related tasks.

The discipline of Statistical Learning deals with such a kind of problems. In general, there are two possible scenarios for classification: supervised and unsupervised. In the unsupervised scenario the learner (classifier) works without any guidelines. One of the examples is hierarchical clustering, when the rules for organizing data come from the data itself. The supervised classifier builds the statistical model based on the training set. In this set the proper classes are already marked on items. After the training phase, the model is applied to the set of interest, the test set. The supervised classification was intensively studied and successfully applied in many domains such as AI, pattern recognition, data warehousing and mining. Assigning topic labels to documents is one of the main tasks of supervised learning for text.

There are many approaches to the supervised automatic classification, from such a simple like *Nearest Neighbor* to the complicated like SVM (Support Vector Machines). In this work, two classifiers were used: the Naive Bayes classifier and SVM classifier. Bayesian approach was used as a baseline in our experiments. SVM functionality, implemented in the SVM-Light package is what we use for the ontology-driven classification in our system. For more information about SVM, see [28], [29]. The rationale behind the choice is following. The performance of Bayesian classification is usually used as a baseline due to its simplicity and speed of training. On the other hand, SVM produces the most accurate text classification among statistical methods.

Naive Bayes classifier. As it was mentioned, this approach is the basic probabilistic approach. Naive here refers to the assumption of probabilistic independence between terms. The set of training documents is given by $D = \{d_1, d_2, ...d_n\}$ and each document has a class label from the set $C = \{c_1, c_2...c_k\}$. Given a term vocabulary $\mathbf{T}$, a document model is a vector with 0/1 slot for each term in $\mathbf{T}$. The slot for the term t is 1 with probability p_t and 0 with probability $1 - p_t$. The set of all the p_t's is P_T, the parameter

set for the model. Given P_T the probability to generate document d is:

$$Prob(d|P_T) = \prod_{t \in d} p_{c,t} \prod_{t \in T, t \notin d} 1 - p_{c,t} \tag{2.1}$$

With this definition of p_t, the probability that a document, which contains term t at least once, belongs to class c is given by:

$$Prob(d|c) = \prod_{t \in d} p_{c,t} \prod_{t \in T, t \notin d} 1 - p_{c,t} \tag{2.2}$$

For classification we calculate these probabilities of belonging to class $c_i \in C$ for each of the documents $dtest_i$ from the test collection. The classification process assigns to each of the documents the label of its most probable class.

To avoid difficulties in the calculation of $\prod_{t \in T, t \notin d} 1 - p_{c,t}$ for each of the test documents, the equation 2.2 can be rewritten as:

$$Prob(d|c) = \prod_{t \in d} \frac{p_{c,t}}{1 - p_{c,t}} \prod_{t \in T} 1 - p_{c,t} \tag{2.3}$$

The part $\prod_{t \in T} 1 - p_{c,t}$ can be precomputed and stored for all the classes c. This description is the most basic one of the Naive Bayes classifier, but in this thesis the main focus is on the ontology extraction process. Interested readers can find more details on frequency-based feature vectors, feature selection, smoothing, etc. in [28].

Support Vector Machines. The idea of the method is based on the intuitive definition of a *separation hyperplane*, which separates two classes if they are linearly separable. The best separation lies on the plane which is as far as possible from the elements of the classes. SVM generalizes this approach. It works in cases when classes can overlap. SVM produces nonlinear decision boundaries by constructing linear boundaries in a higher-dimensional feature space, obtained by transformation of the original one.

Figure 2.3 (borrowed from [29]) demonstrates a hyperplane, which creates the largest margin C between training points. We need to maximize the margin C. The solution can be found by solving the optimization problem:

$$\begin{aligned} & min \frac{1}{\|\beta^2\|} \\ \text{subject to } & c_i(\beta \cdot x_i + \beta_0) \geq 1 \quad \forall i = 1..n \end{aligned} \tag{2.4}$$

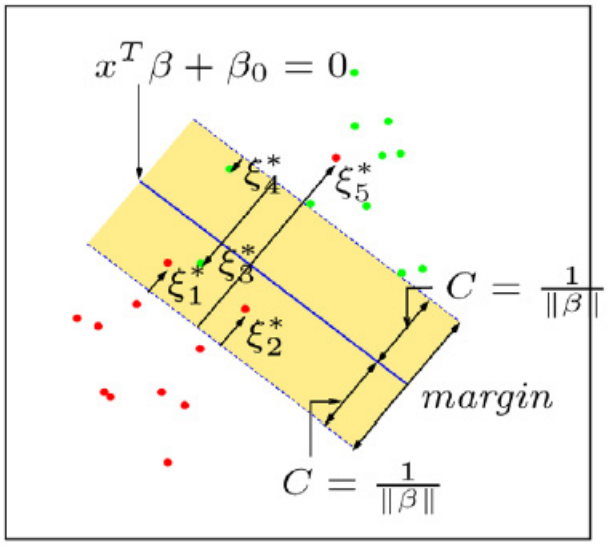


Figure 2.3: SVM optimal separating hyperplane

Here $x_1, ...x_n$ are training document vectors and $c_1, ...c_k$ are their corresponding classes. The distance from each of the training points to the optimized hyperplane is at least $\frac{1}{\|\beta^2\|}$.

This is sufficient when classes are separable. In real life, however, data can overlap. To handle this situation, it is possible to continue maximizing C while allowing some points to be on a wrong side of margin. To take this fact into account we introduce slack variables $\xi = \xi_1, ...\xi_n$. Thus, we can rewrite the previous equation as

$$min \frac{1}{\|\beta^2\|} + C \sum_i \xi_i \tag{2.5}$$
$$\text{subject to } c_i(\beta \cdot x_i + \beta_0) \geq C(1 - \xi_i) \ \forall i = 1..n$$
$$\text{and } \xi_i \geq 0 \ \forall i = 1..n$$

The meaning of the value ξ_i is the proportional amount, which defines the measure by which the prediction $f(x_i)$ is on the wrong side of the hyperplane. In other words, it is the extent till which the value of $c_i(\beta \cdot x_i + \beta_0)$ is on the wrong side of its margin. If the sum $\sum \xi_i$ is bounded, it bounds the total prediction incorrectness. Here misclassification occurs when $\xi_i > 1$, hence we can vary the total number of misclassifications by bounding the sum $\sum \xi_i$ to some constant value. For more details see [29], [30].

Chapter 3

Framework

3.1 Formalities and definitions

Looking for formality to base our system on, we discover that among lexical ontologies WordNet has the structure, which suits our purposes best of all. It has a clear structure and can be formally described. The description is simple and extensible, it is based on the formal concept analysis. For more details see [13], [14]. To define a basis for our system, we adopted a set of definitions from WordNet. In WordNet: *System vocabulary is a set V of pairs (w, s), where a form (word form) f is a string over a finite alphabet, and a sense s is an element from a given set of meanings. Each form with a sense in a language is called a word in that language. A dictionary is an alphabetical list of words.*

In our system we have senses information only about a limited number of terms in vocabulary, the sense usually comes from the context of the document and can be applied not to single word but to concept. Hence, we need the following definition:

Definition 1. ***System vocabulary*** *is a set $V = \{t_1, ..., t_n\}$ of terms $t_i = \{w_1, ..., w_k\}$, where a word w_j is a string over a finite alphabet. Each term consists of one or more words. A dictionary is an ordered list of terms.*

A term that has more than one sense is polysemous; two terms that share at least one sense in common are said to be synonyms. However, we have no possibility to assign the sense to all of the terms. The sense of the term is

defined in the context. In our framework the context of usage is the document in which the term is used. Context of reference is the document, which this term is actually mean or refer to. Here the notion of *Concept* arises.

Definition 2. ***Concept*** *c is the set of n pairs* (t, x)*, where each term t shares the sense with the others in the current context x;* $\{t\} \subset V$*.*

We can define relations between terms in the concept as a weak synonymy, even if does not always have a strict linguistic notion. Concepts can organize hierarchical structure. In fact, WordNet is directed acyclic graph, DAG.

Definition 3. ***Concept hierarchy*** $H_C \subseteq C\ x\ C$ *is an acyclic relation called concept hierarchy, if* $(c1\ R\ c2) \in H_C$ *then c1 is a subconcept of c2, c2 is a superconcept of c1.*

As we said before, WordNet uses sets of synonyms to represent word senses. The semantics is defined by relations between word senses. WordNet includes several semantic relations, here are relations we are interesting in: Synonymy, Hyponymy (sub-name) and its transitive Hypernymy (super-name). The latter two organizes hierarchy of meanings. Meronymy (part-name) is irreflexive, antisymmetric, and acyclic.
Relations represent affairs between concepts, $\mathbf{R} : C$ x C. The ontology itself is a lexical context $O = (C, S, R)$, where C is the set of concepts, S is the set of possible types of semantic relations and R is the set of relations. S can be explained using the following definition:

Definition 4.

Synonymy*:* $t_1\ SYN\ t_2 \Leftrightarrow syn(t_1) = syn(t_2)$*, i.e. words are called synonyms if they denote the same concept.*

Hyponymy*:* $t_1\ HYP\ t_2 \Leftrightarrow c_1(t_1) \leq c_2(t_2)$ *i.e. term* t_1 *is a hyponym of another term* t_2 *if the concept* c_1 *it denotes is a subconcept of the concept* c_2 *the other word denotes in the concepts hierarchy* H_C*.*

Meronymy*:* $t_1\ MER\ t_2 \Leftrightarrow dnt(t_1) R^m_{\forall w_i \in c1 \forall w_i \in c2} dnt(t_2)$*, where m is a meronymy relation among denotations of concepts* c_1 *and* c_2*.*

Hence, words are in the meronymy relation if their denotative word concepts are in meronymyc relation R^m*.*

Summarizing and simplifying the notion, described above, we can say that ontology is a graph $G = (V, E)$. Each concept (a set of synonyms) is a node, $c_i \in V$, and each relation is an edge, $r(c_i, c_j) \in E$. The edge can be of type $\{HYP, MER, UNSPEC\}$, where $UNSPEC$ is an unspecified relation. It will be introduced later in the text.

3.2 Corpora

3.2.1 Corpora selection process

The task of corpus selection is important for the future results of the ontology extraction. When choosing a corpus, we should consider the following criteria: the size of the corpus, corpus content and internal markup. There should be a proportion between the size of the average document in the collection and the size of the collection itself. A very big collection of very short documents or a small collection of long documents are equally undesirable. The content quality can be evaluated by the variety of topics this collection covers. We suggest that on-line encyclopedias or selected personal user collections are usually of high quality. The presence of special formatting elements like titles, sections and hyperlinks with their anchors allows to infer more meaning from the data.

Wikipedia is a well-known on-line encyclopedia. It contains about 350 000 articles, which are devoted to the different areas and sides of the human life, science and culture. Due to its wide range and availability, we based our work on the Wikipedia document collection.

3.2.2 Data format

There are several possibilities to obtain the document collection from Wikipedia. The most convenient way for us was to get the whole dataset as a database dump. The Wikipedia engine works in the following way: a user queries Wiki and the system selects suitable documents from its database. *php* scripts of the Wiki web interface (MediaWiki) format the output to

HTML. The result is displayed in the browser. The engine operates with database records, where documents are stored in the special LOB fields. The article consists of the article text in the standard Unicode and references to outer pages. The text inside uses various Wikipedia's own formatting tags. For further information about Wikipedia see [27]. All of the necessary details are described below.

Wikipedia.org provides opportunities for people, interested in the analysis of their document collection. We used the MediaWiki set of scripts from the local installation of Wiki's engine and the database dump for the MySql database. Wiki team posts dump updates at the web site, so it is possible to have a fresh copy of the dataset.

Field name	Purpose	Type
cur_id	Unique ID autoincrement	int(8)
cur_namespace	Determines whether this document is a real article or internal technical one	int(2)
cur_title	The title of the document, the actual identifier of the document	varchar(255) binary
cur_text	Document body	text/blob
cur_is_redirect	Determines whether this documents is a real entity or just a link to the real document	int(1)
cur_timestamp	The date of the last modification	datetime

Table 3.1: Wikipedia main table schema (trimmed)

The problem with such a big collection of general-purpose texts is a lack of strong hierarchy inside the collection. A number of possible classification schemes presented in articles in the collection. However such articles are usually incomplete and the data itself is unstructured. Wikipedia database records store only documents' titles and texts. In fact, Wiki schema has only one meaningful table. Its major fields are shown in Table 3.1.

Few words about namespaces: as far as wikipedia is free and can be freely edited, it is necessary to keep track of the changes. The namespace, which keeps the up-to-date documents has number 0. Wikipedia authors can dis-

cuss their articles and this information will be saved in namespace 9. Help documents, images and sounds are also kept in different namespaces. For details see [27].

3.2.3 Markup language

Element	Description
Underscore "_"	Used instead of spaces in titles and links inside the document.
== New section ==	The nesting level of the section determined by the number of "=" signs. Two "==" mean the first level.
[[Doc_name]]	Usual in-text link to the article named "Doc_name".
[[link anchor to appear in text | Doc_name]]	Allows to substitute the real article name in the link with more convenient word (link anchor) for this text.
♯ REDIRECT [[Doc_name]].	Tells that the current document contains no text and refers to the document Doc_name, which contain the actual text.
* *Lists*	A star as the first character in the line allows to itemize data. The number of stars defines the item's level in a list.
♯ *Numbered lists*	Allows to enumerate data. The number of "sharps" defines the item's level in a list.
♯ Mixed lists.	Allow to mix not numbered items with numbered. Number of stars defines the item's level in a list.
$< table >$ or {|	Begins a table. Wikipedia allows both the HTML and customized tags for tables.
$< tr >< /tr >$ or |−	Defines a row in a table. Wiki tags should be placed at the beginning of the line and do not require a closing tag.
$< td >< /td >$ or |	Defines a column in a table. Wiki tags should be placed at the beginning of the line and do not require a closing tag.
$< /table >$ or |}	Closes a table.

Table 3.2: Elements of Wikipedia markup language

The formatting language of Wikipedia has many different markup elements. We took into account constructions, listed in the table 3.2. A num-

ber of naming and formatting conventions exist in Wikipedia. The document name should begin from the capital letter in general. The rest of the title has to be in small letters. More capitals one by one mean abbreviation. If the document describes a thing, which has a definite name, i.e. a band name or planet name, each of the words can be written from capitals. Special symbols are avoided when it is not necessary for precise writing of the title. In our work we did not use links to images and sounds as well as links to pages in national alphabets. Nevertheless, if a word in the text is written with special characters, it is allowed.

3.3 Concept extraction

3.3.1 Document processing

The whole process of ontology creation begins from the corpus. Originally it can be in the form of files collection, but for our purposes we need to store all the documents in the database. It allows to have a fast and uniform access to each of the documents. All the data is stored at one place. After we finish the conversion of the corpus to the database representation, the second preprocessing step should be performed. We search outgoing links in each of the documents. When such a verbose link is found, we look for the corresponding document ID in the database. Hence, we create a document graph. Now we can begin the concept extraction process. The details for the links processing will be presented later in this section. For each of the valid documents in the collection we should maintain the following information:

- Document unique identifier
- Document title
- Document text

Now we can go to the concepts extraction. Concepts are defined in section 3.1. Our system distinguishes four major types of terms, which form concepts.

- S-Terms. The source of these elements are document titles. We consider them as the most confident terms inside the concepts. They are the primary building blocks of the system vocabulary.

- A-Terms. These terms are related to S- ones and share the sense with S-terms. For a given S-term, A-terms are extracted from anchors of the links in documents, that refer to S-term.

- NT-Terms. They appear in the document text as links, but these links have no target documents yet. In the best case such terms can be considered as meaningful phrases, related to the current context.

- E-Terms. Emphasized terms. The additional source for meaningful terms are words and phrases, emphasized with the special markup tags. It can be cods, apostrophes or parenthesis as well as paragraph or section names.

Each concept is the combination of these possible types of terms.

The first step in the ontology creation process is to extract concepts. To accomplish this task, all the documents titles are retrieved. Then we process each of the titles in according to the policy, which includes the following:

- Lowercasing strategy: do not make the case lower, make the case lower for all the terms except named entities (i.e. "Paris") or to make the case lower for all the terms.

- Which characters should be considered as special and removed. Usually we remove symbols which are not defined in the alphabet of our system (i.e. $\sharp, @, ', \%, -, *$) and so on.

- How to stem concepts: stem everything, do not stem, maintain both the forms.

- What to do with digits: remove digits from the term, disregard the whole term or keep it as it is.

- How to deal with stop words inside a concept: remove, keep or maintain both the forms.

- How to process E-terms inside: include or disregard.

Applying the chosen policy for all the titles we create a concepts core. Then we need to add all possible information to this skeleton. We perform a new iteration for A-terms to find out, whether we can add something to our existing concepts. We iterate through all documents and study their link structure. The purpose here is to collect all possible synonyms or context-related alliterations. For A-terms we apply the same policy, enhancing it with two rules.

- How to add synonyms: add all, forbid lexical forms of the same word, add only distinct terms to some threshold of lexical closure.
- How to deal with NT-terms: create artificial concepts for each of the distinct terms or ignore.

At this step we also calculate citing frequencies of every concepts. It is the number of their inlinks. The creation of the concepts set is completed.

3.3.2 Link processing

As we said before, we need to keep information about links between the documents, which are stored in the database. For each of the valid links in a document we should maintain the following information:

- Unique identifier of the source document (where this link was found).
- Unique identifier of the target document (on which this link points to).
- Title of the target document.
- Anchor text of this link, if it exists.
- Type of the link: this link is of the redirect type, this link is in the special section like “See also” or “Related links”. The list of the names for these sections is provided by the user.
- Position, where the link was found: in which section or subsection, whether it is the element of any list.

3.4 Relation extraction

3.4.1 Well-formed relations

When we have extracted our ontological concepts, the most difficult thing is to find out, what kind of relationships exist between them. The first method for obtaining relationships is to get as much as possible from concepts and their usage in the text. This has another rationale - for instance, the Wikipedia community has several "good fashion" guidelines about the structure of the document, the way to introduce links and the way to name documents. We use this information to extract higher-confident relations. Bad links with the broken markup are disregarded.

Synonym extraction

The patterns of the word usage differs from one author to another. The word, used to denote an entity in the text also depends of the context. When different words used to denote one subject, we refer to this phenomena as synonymy. Link and anchors analysis gives us a possibility to find out, which words were used to denote the current document concept across all the documents in the collection. Consider some examples of the links: [[America | United States]] and [[USA | United States]]; [[Capital of France | Paris]] and [[Paris]]. It is easy and confident to infer synonymy relations from such constructions. Another possibility here is to take a special structure of "redirect" pages into account. Consider the document with the title "Ada programming language" has a special tag "♯REDIRECT [[AdA]]" inside. If one will type the first phrase in browser she will be redirected to the page "Ada". This denotes that the first term means the same that the second one and both the terms have one explanatory article. From these regularities we infer rules for the synonyms extraction.

Synonymy (SYN):

$$\begin{aligned} [[t_1 \mid t_2]] &\rightarrow t_1 \text{ SYN } t_2 \\ t_1 \text{ REDIRECT } [[t_2]] &\rightarrow t_1 \text{ SYN } t_2 \end{aligned} \tag{3.1}$$

This information is extracted from the document body. A concept and its synonyms form a synset in the terminology of WordNet.

Hypernym/Hyponym extraction

Anchors. During our analysis we observed an interesting effect, which is worth to take into account within our framework. It related to the previous case where we look for citations of the current document. Recall the equation 3.1. It can happen that the document mention a specific instance of something, however the link inside the points to the general notion of it. Consider the following examples from Wikipedia: [[English language | Language]], [[Indian diamonds | Diamonds]], [[Terrestrial planet | Planet]]. We can easily say that the such an overlap in the term usage between anchors and titles denotes a hierarchical relationship between these concepts. Anchors here denotes more specific things, while actual titles - more general. We define a rule for the "in-anchor hypernymy" as follows.

In-Anchor Hypernymy (A-HYP):

$$[[t_1 t_2 | t_2]] \rightarrow t_1 \text{ HYP } t_1 t_2 \tag{3.2}$$

Titles. A person usually thinks on each of the words she uses when she writes something. The few words, on which the author thinks most of all, are the words in the title of the document. Consider four Wikipedia documents, named "Paris", "Paris, Tennessee","Paris, Texas", "Paris (god)". We have reasons to think that these documents discuss different things. Another example is "Saturn (planet)", "Saturn (rocket)", "Saturn (roman god)". Depending on our policy, described in section 3.3.1, we can disregard the part, surrounded by special characters (a parenthesis in this case). A excerpt of the typical Wikipedia reference page, which illustrates the example above is given on Figure 3.1 However, we'll be wrong if we will consider the rest (three Saturns) as one. We claim that words, emphasized by parenthesis and commas usually denote more general concept. This type of relation-

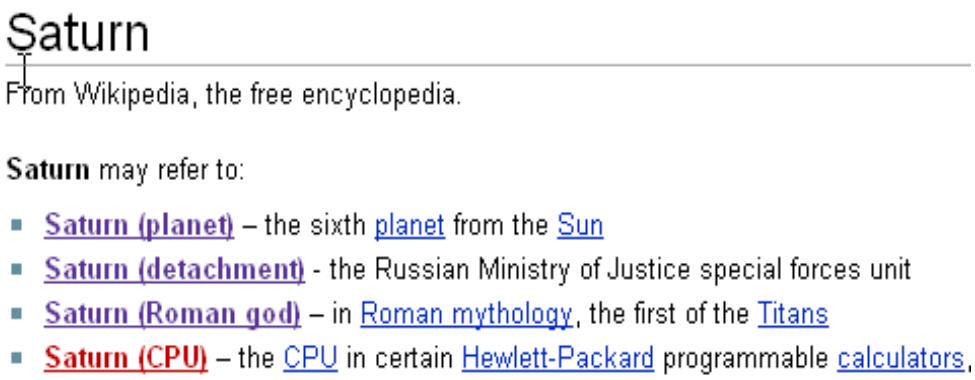


Figure 3.1: An explicit relationship example

ships will be denoted as "Explicit Hypernymy". We extract these relations from document titles. This regularity can be written down as a following rule:

Explicit Hypernymy (Ex-HYP):

$$\begin{aligned} t_1[,]^* \ (t_2) &\rightarrow t_1 \text{ HYP } t_2 \\ t_1, \ t_2 &\rightarrow t_1 \text{ HYP } t_2 \end{aligned} \tag{3.3}$$

where t_1 and t_2 are terms as it given by definition 1. This equation and all the equations below use java-style when presenting regular expressions.

Lists. Another possibility to obtain information about the hierarchy of concepts, is to analyze explicit lists and enumerations inside the documents (see markup description in 3.2.3). Consider the document, named "Canidae", which has inside a following species list (example from Wiki):

- Genus Canis (level 1)
  - Wolf, Canis lupus (level 2)
    - Domestic Dog, Canis lupus familiaris (level 3)
    - Dingo, Canis lupus dingo
    - ...many other subspecies...
  - Red Wolf, Canis rufus (level 2)
    - Coyote, Canis latrans (level 3)

    - Golden Jackal, Canis aureus
    - ...many others...
- ...many others...

The analysis of the list implies that the document concept "Canidae" has the highest level of generality, the concept "Wolf" subsumes the concept "Canidae" and "Dingo" subsumes "Wolf". The concept "Dingo" transitively subsumes "Canidae" then. We extract this relations from document bodies.

The notion of subsumption makes us to take into account one additional parameter for each of the concepts. This parameter is the nesting level (NLevel) of the concept. In the example above we explicitly marked the level of concepts in the list. It means that we memorize this information for each of the concepts, which were met inside the list tags (see table 3.2 for markup details). Each of the nested concepts in the list will have bigger nesting levels. The document title becomes an implicit parent concept for the concept of every level and an explicit parent of first-level concepts. We denote it as a "document concept". In the given example the document concept is "Canidae". The regularities, described above, gives raise to the following rules:

Subsumtion Hypernymy (S-HYP):

$$NLevel(t_1) \geq NLevel(t_2) \rightarrow t_1 \text{ HYP } t_2 \tag{3.4}$$

$$C_d \text{ HYP } \forall(t_i) \in T_1, \quad NLevel(t_i \in T_1) = 1 \tag{3.5}$$

where t_1 and t_2 are terms, C_d defines the document concept and $NLevel$ is the nested level of concept. Remark that the notion of term t allows it to consist of several words.

Lists in special sections. A special case of the previous scenario appears when a special "classifying section" exists. In the process of parsing documents we search for sections, which contain words like "classification", "types of", "kinds of" (the list of such words is provided by user). If we found an enumeration or table structure inside such a section, we consider its elements

as hierarchically related to the document concept. We will refer to this pattern as "special section".

Special Section Hypernymy (SS-HYP):

$$t_1 \in SSection \in C_d \rightarrow C_d \text{ HYP } t_1 \tag{3.6}$$

where t_1 is the term, C_d defines the document concept and $SSection$ is the special section, which name matches a phrase from the user-provided list of possible names for the special section. A typical example (trimmed) of such a construction is shown on Figure 3.2.

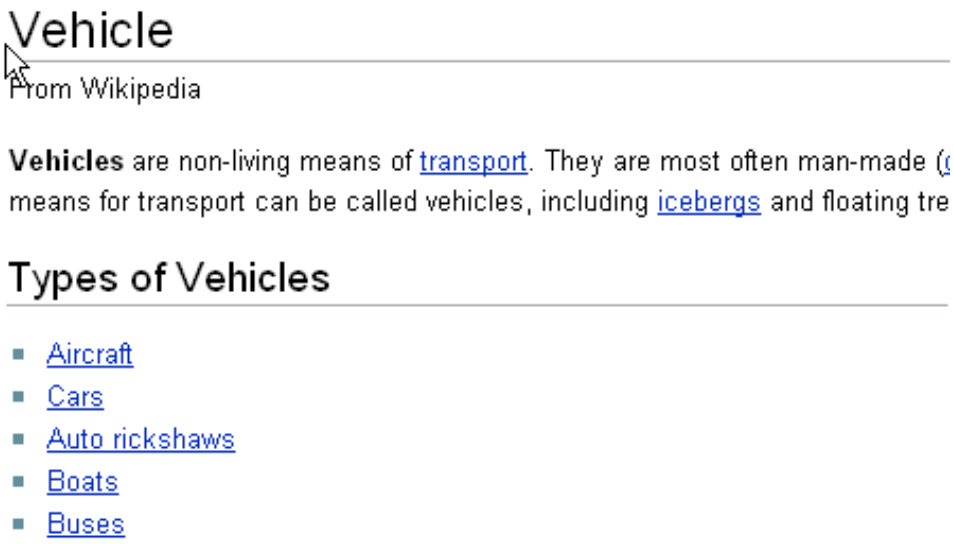


Figure 3.2: A classifying section example

Lexical relations. Another issue in the relations extraction process is to consider so-called explanatory articles. Such an article describes a particular object. They often use constructions like "[[*Jupiter*]] is the biggest [[*planet*]] in [[*Solar system*]]". Performing very simple lexical analysis, we can find concepts in the environment of the main concept of the document. From the example given we can infer a relationship between the concepts "planet" and "Jupiter" and "Jupiter" and "Solar system".

Lexical Hypernymy (L-HYP):

$$t_1 \text{ IS(one of |)A } t_2 \rightarrow t_1 \text{ HYP } t_2 \tag{3.7}$$

where

IS =(is|are|was|were|ha[s|d|ve] been)

A = (a|the|an)

3.4.2 Other relations

At this point we introduce into our scheme, defined in section 3.1, a generic relation UNSPC for the "unspecified" relationship between concepts. The term "unspecified" here means that for the moment this relation was extracted, we had no information about it certain type. However, we can say that concepts are somehow related.

Similarity relations Additional analysis of the documents' structure gives us another good source for establishing relationships between concepts. We take into account such structural elements as sections. The name of the section itself is usually meaningful in the context of the document. If a document is linked under one of the special sections like "see also", "similar topics" it indicates, that this document has something to do with the topic. From the other hand, documents, which do not belong to the corpus, are usually linked in the section "External links". The list of such section names of interest ($SNAMES$) is provided by the user. The rule for this case is follows.

Similar-to relationship (SIMTO):

$$C_d \text{ SIMTO } t_i \quad \forall(t) \in SSection, \quad SSection \in SNAMES \tag{3.8}$$

where C_d denotes a document concept and $SSection$ is one of the sections we consider to contain related terms.

Like in WordNet, which has SEE_ALSO and SIMILAR_TO relations, we fill these sets with relations, extracted by the described heuristic.

Links The interesting issue is to take into account links, which can be met in the text and do not fall in any of the mentioned categories. However, the approach to extract this specific type of relationship is different. We cannot exactly determine what kind of relation is it. A link on the page means that the author estimates highly the importance of this concept with regard to the topic of the document. However, there is a possibility, that the author makes a reference on the existing document in the database just because the document with this title exists. For instance, when the concept "mental diseases" used in the document "Autism" it has a perfect correlation with the topic. When the concept "apple" used in the document about Newton - it appears to represent another case. We propose a scheme of taking into account the information provided by these links. It is described in section 3.6. We will refer to this type as L-UNSPEC (in-list-unspecified)

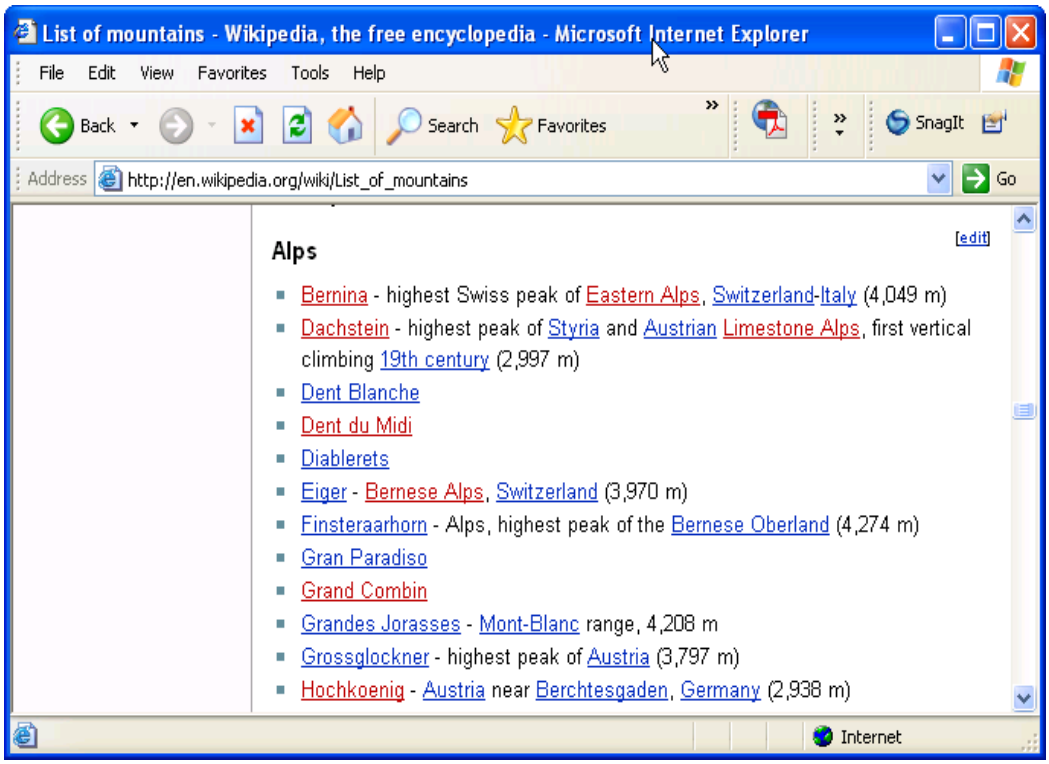


Figure 3.3: List-of-the-topics article example

Topic lists. The special case of the semi-specified relationship is shown on Figure 3.3. There are a lot of special pages, which contain only links. These

pages are specially constructed to collect references to articles, devoted to the certain topic. Hence, we can find a lot of similar to the document concept concepts. For instance, “List of Astronomy topics” or “List of USA topics”. The convention to name such documents is given by:

$$Document\_name := \text{ (List[s |of])* } C_d \text{ (topic[s])*}$$

where C_d denotes a document concept.
For the example given we will relate Usually such kind of pages are structured via sections. It is sometimes risky to extract the full hierarchy (i.e. document_concept → section_concept → subsection_concept → in-text_concept), however we obtain a good source of concepts, directly related to the topic (i.e. document_concept → in-text_concept). We will refer to this type as T-UNSPEC (topic list-unspecified). **List unspecified (T-UNSPEC):**

$$C_d \text{ T-UNSPEC } t_i \quad \forall(t) \in D \tag{3.9}$$

where D denotes a document and t is any document term.

3.5 Document representation and Relation weighting

As far as we described our heuristics to the relation extraction, we should specify the way to weight relations between pairs of concepts. In its turn, this problem is related to the way of documents representation. One of the most widely used methods for document representation is a vector space model. In this model, each document is represented as a vector in the term space. The term space is multidimensional and each dimension corresponds to the one term. The document vector contains the information about terms occurrences across all the dimensions, i.e. all the terms in the corpus. The word order in the document is not considered. In the vector space model a term weight in a document is usually determined by the term frequency tf and inverse document frequency idf. It gives rise to the popular term ranking formula, $tf \cdot idf$. A number of ways to compute these values exist. Various

coefficients and normalization parameters are used in different formulas. Our system does not stick to a fixed weighting scheme. Due to this fact for each of the terms in the document we keep only raw term frequencies tf and its idf values. It allows to choose a desired formula. For idf we took the most popular modification by Robertson and Spark Jones.

- $tf_{t,d}$ = a number of occurrences of the term t in the document d
- $idf_t = log(\frac{N+0.5}{docfreq_t})/log(N+1)$ where N is the collection size and $dcfreq_t$ is the number of documents, which use term t.

These values are accessible for every document and term in the collection. The $tf \cdot idf$ weight or the importance of the term is computed at the run-time. In our system there are two types of weights defined: weight of the term in a document, which we had discussed above and weight of the relation or the similarity between two concepts in the relation. To calculate these values, we parse the documents collection with regard to the set of existing concepts, the system vocabulary. During this process we calculate term frequencies for each of the terms in the given document and across the corpus. After obtaining term frequencies across the document collection, we calculate idf values. The next step is to calculate relations weights.

A lot of similarity measures are defined in literature. To quantify the relationship between two concepts we implemented two measures: Dice coefficient and Probability of co-occurrence. For two concepts A and B we define:

$$Dice = A \cap B / A \cup B$$
$$P(B|A) = P(A \cap B)/P(A) \tag{3.10}$$

The first measure is symmetric, it will assign the equal weight to both the related concepts. The probability of co-occurrence is non-symmetric. It captures different closures between i.e. hypernyms and hyponyms and vice versa. To take into account this difference in the similarity between two hierarchically related concepts, we used the probability of co-occurrence in our system.

3.6 Ontology extraction

When we extracted all the possible relations between concepts, we can create ontologies of different sizes. The question that arises here is how to choose relations for the resulting ontology. We suppose that at this point we have all the possible statistics about concepts and terms: frequencies, inverted document frequencies, similarities between concepts. The simplest way is to limit the size of the resulting ontology. It is possible to take into account relations, which were found by certain heuristics and thus, have the certain type.

Strategy 1: choose relations by type. The available types are: SYN, Ex-HYP, S-HYP, SS-HYP, L-HYP, SIMTO, UNSPEC, L-UNSPEC (see the previous section).

$$R_{c1 \to c2} = \begin{cases} sim, & \text{, if R} \in \text{RELATIONS} \\ \text{not exists} & \text{, otherwise} \end{cases} \tag{3.11}$$

where R is the possible relation between concepts $c1$ and $c2$, sim is the similarity from 3.10 and $RELATIONS$ is the selected set of relation types. The second possibility is to discriminate concepts inside relations by their IDF values across corpora. Varying the value of *idf* for suitable concept, we can influence on the size of the relation set considerably.

Strategy 2: choose relations by IDF values.

$$R_{c1 \to c2} = \begin{cases} sim, & \text{, if IDF(c2)} \geq \text{C} \\ \text{not exists} & \text{, otherwise} \end{cases} \tag{3.12}$$

where R is the possible relation between concepts $c1$ and $c2$ and C is the threshold value.

Another way to accomplish this task or another step after the methods, described above, is to apply thresholding approach to the set of unspecifies relations. If we have a link graph and weights between connected nodes, we can compute the importance of the concept c, linked inside document to the document concept C_d.

Strategy 3: choose more important relations.

$$Imp_{c \to C_d} = \alpha \cdot IO(c, C_d) + \beta \cdot OO(c, C_d) + \gamma \cdot OI(c, C_d) + \sigma \cdot sim(c, C_d)$$

where IO is a measure of the overlap between inlinks of d and c or co-citation index. OO is a measure of similarity between outlinks of d and c or co-reference measure. OI defines the overlap between outlinks of c and inlinks of d, in other words it is a measure of subsumption. The $sim(c, C_d)$ is calculated using the system-wide notion of similarity (the probability of co-occurrence in our case).

The choice of coefficients depends of the point we want to emphasize. To find closer concepts we may increase parameter α. If we want to find out possible hierarchical relationships, we increase γ. Varying these parameters and adding to the resulting ontology relations of certain type, we can evaluate ontologies of different sizes and content. To summarize the material, described in this chapter, the overall process of ontology extraction is shown on Figure 3.4.

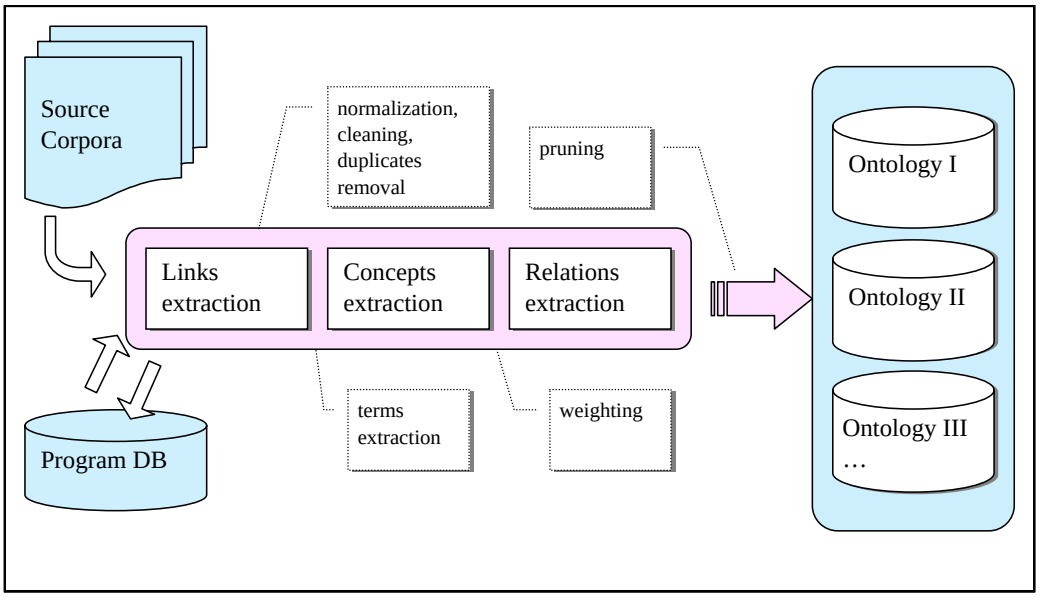


Figure 3.4: Ontology extraction process

Chapter 4

Implementation

4.1 Class hierarchy and Data structures

In the previous chapter we described the ontology extraction framework, which was developed in this thesis. We implemented the corresponding functionality in the OntoWiki package. The package is implemented in Java. It uses OntoServer packages and adds a few modules to the BINGO! code. OntoWiki keeps its data and stores the resulting ontologies in the different database schemes. In according to the generic process of ontology extraction, given in the previous chapter, a simplified class hierarchy, which implements the program logic, is shown on Figure 4.1. The basic objects for OntoWiki are Term, Link, Document, Concept and Relation. These objects are stateless. They intended to store data about each of the instances of the appropriate type.

Manager classes maintain collections of basic objects and provide functionality to iterate through their collections. These collections can be very large, so manager classes encapsulate a seamless access to the database storage. Modification of collection elements is also performed by Managers classes. Each of these classes implements a specific interface. It allows to exchange underlying collections and database schemes easily. The interface hierarchy and class hierarchies are depicted on figures in Table **??**.

The program makes intensive use of the database. All the program's data is stored in database tables. We used Oracle as the Database Management System due to its stability and performance characteristics. OntoWiki uses

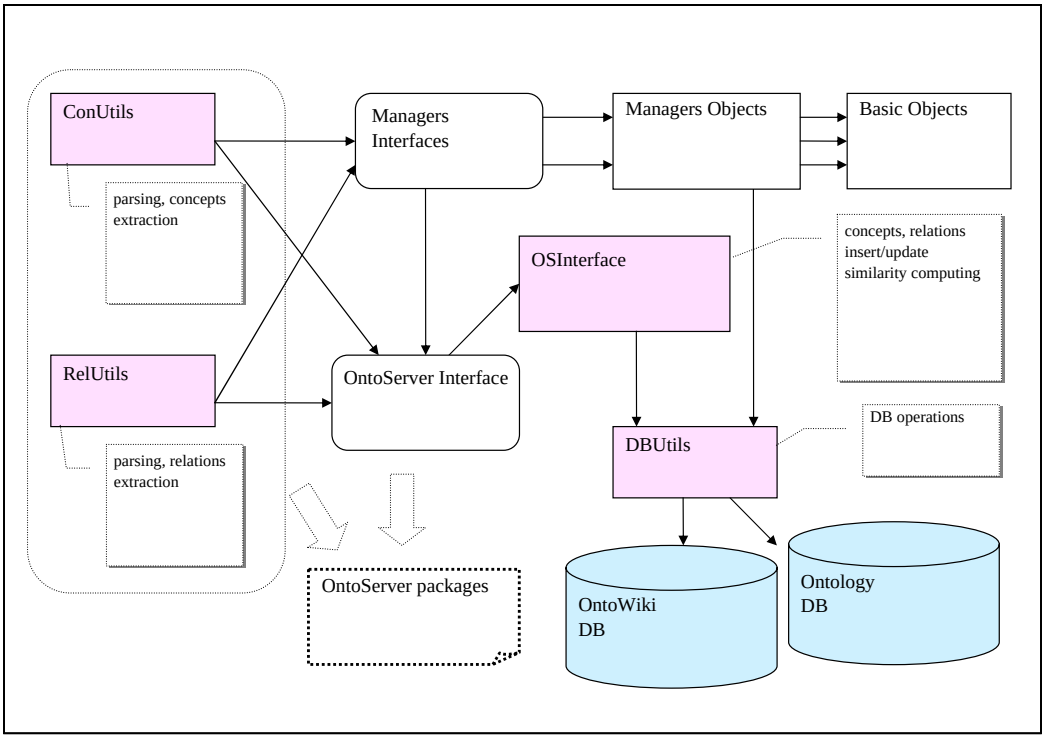
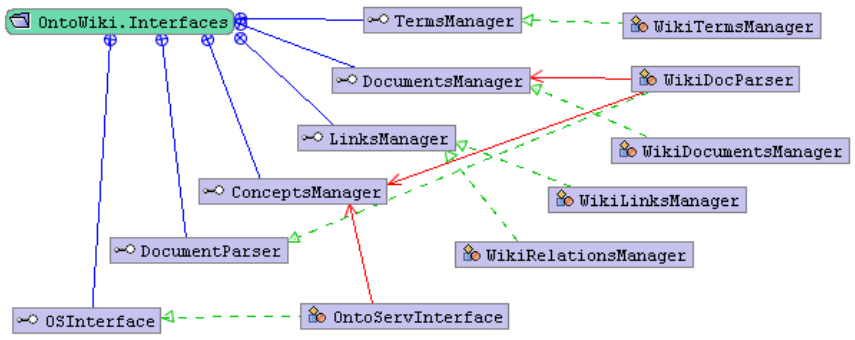


Figure 4.1: OntoWiki class hierarchy

two database schemes (just for its own purpose). These database schemes are presented in Table 4.2. One schema is intended to store corpus documents, parsed links, extracted concepts and all the preliminary information OntoWiki handles. The second schema is intended to store the local ontology. At the beginning all the possible concepts and relations, which had been extracted, stored there. Each of the resulting ontologies resides in a different schema, similar to the local ontology. Result is produced by varying the number and type of concepts and relations to be selected from the local ontology. Such ontology schemes can be created automatically, using functionality provided by the OntoServer package. The `DatabaseBackedOntology` class has a static method `createSchema` to create the database schema for the new ontology.

OntoWiki Interfaces scheme



OntoWiki Manager objects scheme

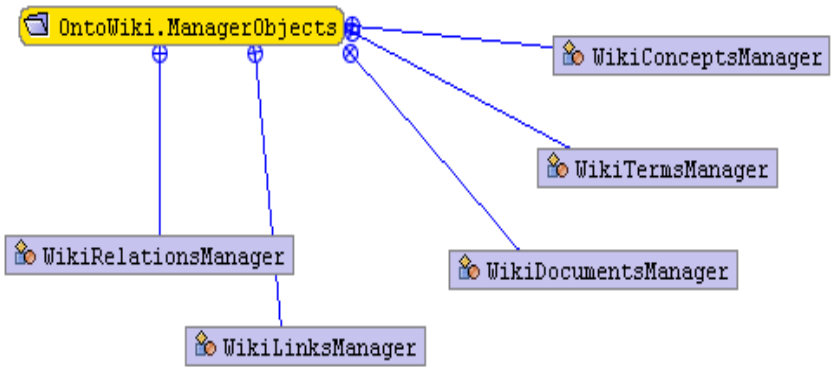


Table 4.1: OntoWiki classes and interfaces

When OntoWiki finishes the corpora processing, the local ontology schema is filled with the data. The resulting ontology can be produced using `OntoWiki.ExtractOntology` static method.

Back to the database schemes, let us present the details of OntoWiki own scheme here. The "DOC_TEXTS" table is filled first. Each of the articles is stored along with its title. The texts are stored in the LOB fields. "doc_id" is an auto-created unique identifier for each of the documents. The "DOC_LINKS" table stores in-document links, which have been found during the parsing process. For each of the documents in "DOC_TEXTS", "DOC_LINKS" contains a set of records with a "source_doc_id" field set to "doc_id" of the current document and "target_doc_id" set to the "doc_id" of the linked document. The anchor text, titles of the source and target, the link type and its position in the document (line number, section number and nesting level in the list, if applicable) are also stored there. The "link_type" field can take values of 0,1,4 (0 - usual link, 1 - redirect, 4 - a section name). The "CONCEPTS" table stores concepts in the form of synsets. It means that all the synonyms, comprising a synset have the same "concept_id". The "concept_id" field of each of the synsets has the same value as the "doc_id" of the corresponding document. Both the stemmed and unstemmed versions of the concept are stored. The "gtf" field stores the number of references to this concept across the collection. The "c_type" field determines, which heuristic has found this concept. It accepts values of 'n',1,3,4,6,7,8,9. ('n' is the standard title, 1 is the redirect, 3 and 4 are parts of the title with brackets, 7 and 8 are parts of the title with comma, 9 is the concept from the "topic list" document, if there is no concept exists for it). The "DOC_TERMS" table stores a set of distinct terms for each of the documents. Each term has its own frequency in the given document. It is stored in the "tf" field. If the unique concept can be found for the current term, the id of such a concept (CONCEPTS.concept_id) is stored in the "concept_id" field. The "IDF_VALUES" table keeps the *idf* (see 3.5) values for each of the distinct terms. The table "RELATED_CONCEPTS" serves as temporary storage for relations, discovered by each of the heuristics from 3.4.

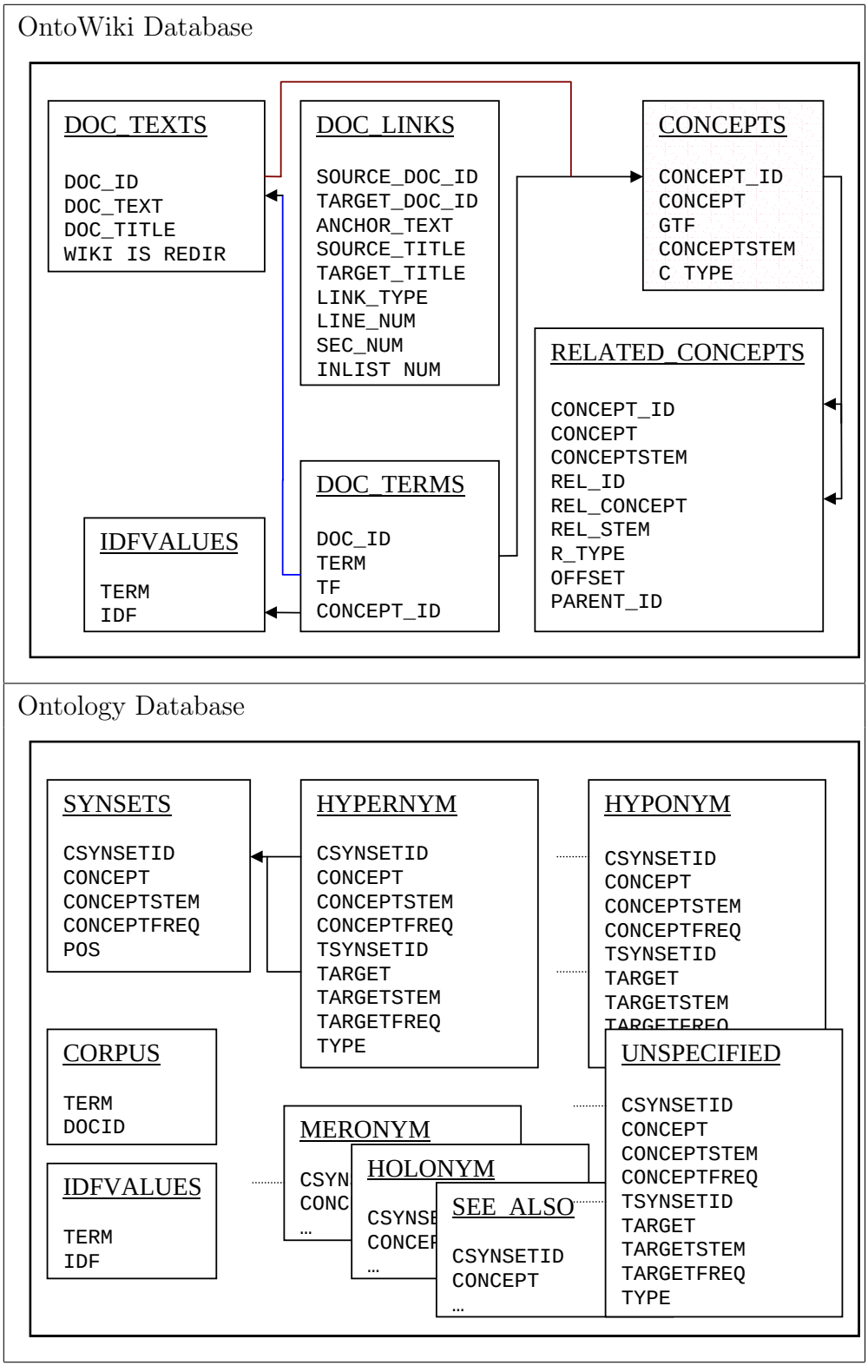


Table 4.2: Database schemes: OntoWiki DB and Ontology DB

4.2 Important classes

Basic objects classes. As we mentioned before, these classes are intended to keep the information about each particular object of the "basic" type.

`OWTerm`
`- Fields: {int doc_id; String term; String stem; int tf;}`
`- Methods: {void setTF(int num); void addTF()}`

`OWLink`
`- Fields: {String source_title; String target_title; int source_id; int target_id; int link_type; int line_number ; int inlist_level; int section_id;}`
`- Methods: {void addTF()}`

`OWRelation`
`- Fields: {int doc_id; String concept; String con_stem; int rel_id; String related; String rel_stem; int type;int parent_id; double sim; int origin;}`
`- Methods: {OWRelation getReverse()}`

`OWConcept`
`- Fields: {int concept_id; String concept; String stem; int type; int gtf; ArrayList related}`
`- Methods: {void addRelated(ArrayList o); OWConcept getRelated(int i); boolean valid(); int getIdxOfhMaxTF(); int getIdxOfType(int type)}`

`OWDocument`
`- Fields: {int doc_id; String doc_title; String rawtext; int redir; ArrayList doc_links;LinkedHashMap terms;}`
`- Methods: {void resetTermsMap(int capacity)}`

Table 4.3: Basic objects: fields and methods description

The most simple object is `OWTerm`, it stores information about the word and its frequency in a document. `OWLink` represents the link inside a document. It has information about the target document "doc_id" and the frequency of this link in the source document as well as the link text. `OWRelation`

class is quite similar to the `OWLink` class. The `OWConcept` object represents a concept and can store a set of related to this concept concepts. In the different situations it aims to different purposes. For instance, concepts can be grouped by the same concept_id (synonyms) or by the terms, which represent a concept (polysems). It also has a number of member functions to search the related concepts set. `OWDocument` is the heaviest object here. It represents a document from the database and maintains its full text as well as its links and its terms set. Each of the basic classes is given in details in the table 4.3.

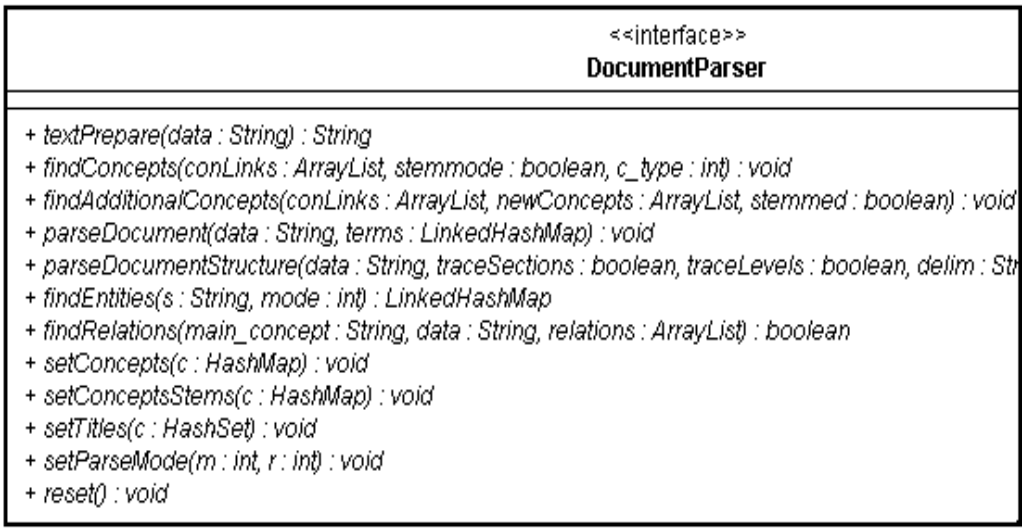


Figure 4.2: DocumentParser interface

DocumentParser. The `WikiDocParser` class implements concept extraction heuristics, described in section 3.3.1. The policy for the concept extraction is represented by regular expressions and boolean flags which user can set. This set of regular expressions is stored in the static fields `Config.delimiter` and `Config.parserRegExps` of the configuration class `Config`. It can be easily modified by providing another set of extraction rules. The interface of the document parser is shown on Figure 4.2. The rules for processing the document structure (inner links, tables, lists and other markup elements) are corpora-specific, that's why this set is maintained by `WikiDocParser` class itself. It is not included into the interface.

The function `parseDocumentStructure` is intended to extract structural elements like sections, tables and lists. This information is used to infer concepts hierarchy in the end. A method `parseDocument` implements a window parsing algorithm with lookups into the ontology concepts set. Functionality for the concept extraction is described in the next section.

OSInterface The `OntoServInterface` class implements functionality for interaction with the local ontology database. It contains functions for the basic DB operations, such as select and insert procedures for sets of concepts, insert and update procedures for concepts and relations. `OntoServInterface` provides the concepts sets for other classes. It also provides a function for resolving concept IDs for terms, found during the parsing process. This class is an unique connector with the local ontology database for all other classes. The interface of this class is shown on Figure 4.3.

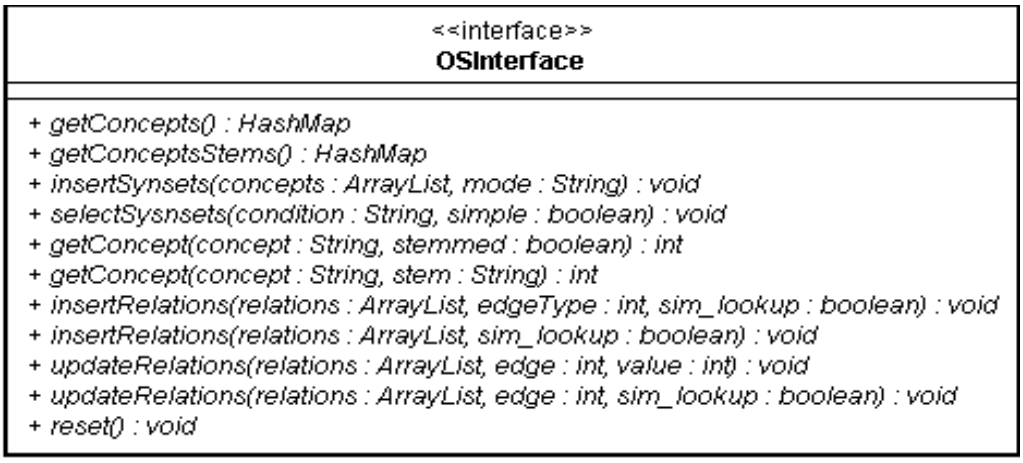


Figure 4.3: OSInterface interface

BINGO! code modifications The `BINGO.svm.NaiveBayesClassifier` class implements a multinomial Naive Bayes classifier. It was integrated in the system and can be called from the menu in group with the other BINGO! methods. The `BINGO.xml.XTwigBuilder` class was modified to allow window parsing with regard to the ontological concept set. If the static parameter `XTwigBuilder.externalParser` is set to "true", `OtoWiki.DocumentParser`

is used instead of the usual stemmer. The "unspecified" type of relations was added to the OntoServer package.

4.3 Concept extraction

`ConUtils` is the utility class in the OntoWiki package, which handles the whole process of concepts extraction. Its fields and methods are shown on Figure 4.4. If we don't provide explicit references later in this section, we discuss the functions of this class.

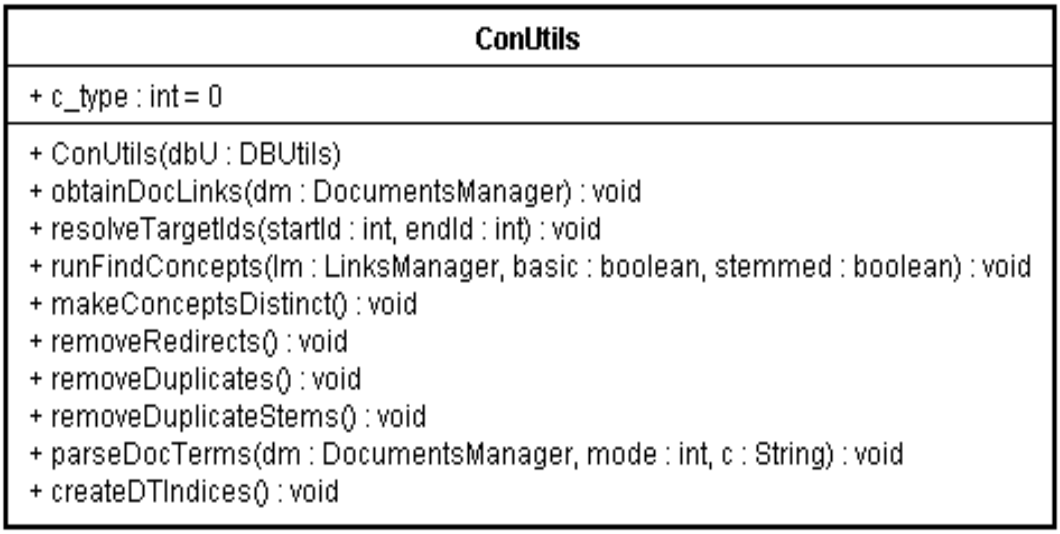


Figure 4.4: ConUtils class

Links extraction. The first step in the concepts extraction process is to extract known links from all of the documents. This operation is performed by the `obtainDocLinks` method. The input parameter is the `DocumentsManager` object which has the information about the range of documents to process. Inside the method, two objects are created: `WikiDocParser` and `LinksManager`.

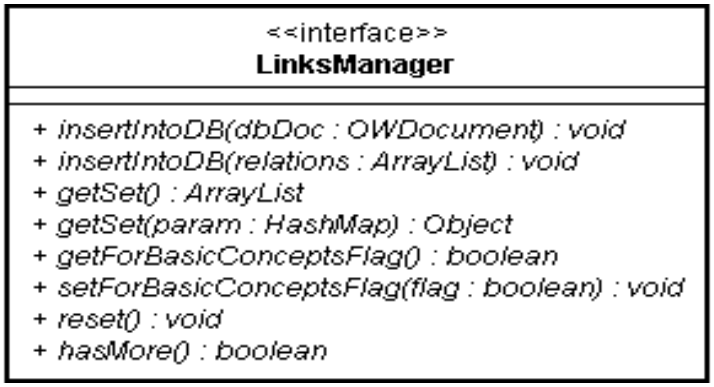


Figure 4.5: LinkManager interface

`DocumentsManager` iterates through corpus documents in the database (table DOC_TEXTS) and extracts them portion by portion into the queue of `OWDocument` objects. Then `WikiDocParser` processes each of the documents and delegates the result to `LinksManager`. The latter maintains the set of links and inserts them into the database (table DOC_LINKS) when necessary. At this moment we know only target documents' titles. When this work is finished, the default value of the field "DOC_LINKS.target_doc_id" should be substituted by the real "DOC_TEXTS.doc_id" of the target document for each of the links in the table DOC_LINKS. It is handled by the function `resolveTargetIds()`.

Concepts extraction. When all the links have been extracted we go to the concepts extraction step. In the terminology of the third chapter, we extract S-terms first. This task is accomplished by the `runFindConcepts` function. Its parameters denote the range of documents, we want to extract concepts from, and the stemming policy. Inside the method, two important objects are used: the `ConceptsManager` object and the already mentioned `WikiDocParser` object. `LinksManager` iterates through the links in the DOC_LINKS table. `ConUtils` feeds `WikiDocParser` with documents set by set. The latter parses each of them with regard to the policy. `ConceptsManager` collects extracted concepts and inserts them into the table CONCEPTS. To extract A-terms we call this function another time with the parameter `basic` set to false. The rest of the process is similar.

After these procedure is finished, we need to clean our concept space. As far as the "unnecessary" symbols were removed, concepts could have been stemmed and redirects were resolved, we need to clean our set of concepts. There are three functions, called `makeConceptsDistinct()`, `removeRedirects()` and `removeDuplicates()`, which handle this task. We call them consequently.

Parsing with concepts. After we have formed the concept space, we perform another document parsing to extract meaningful terms with regard to the ontological concepts set. We need the information about occurrences of concepts later on, at disambiguation and relations weighting steps.

The functionality for parsing with concepts is implemented in the `WikiDocParser` class. It implements a generic DocumentParser interface. Again, the document set is maintained by the `DocumentsManager` object. For each of the documents, the following operations are performed. The first step in this process is to extract concept phrases, which are already hyperlinked or somehow emphasized inside a document body. At this step we simply look for every link that the author made. To extract concepts, which are not hyperlinked, `WikiDocParser` scans the document text with the "window parser". It takes a word sequence of the *window* size and looks for a match in concepts. I there is no match was found, it removes words one by one from the end and repeats the lookup. This process continues until the match is found or a single word is left. The size of the window can be set in the configuration class, a `Config.window_size` parameter. We used the window of the size 5. The set of concepts is kept in the main memory, so the whole process is quite fast.

`WikiTermsManager` keeps track of the terms extracted and inserts them into the database table DOC_TERMS. Only distinct terms are inserted. Each of the terms has its own count of appearances in the current document. For each of the terms the list of positions where it appears in the document is also maintained.

4.4 Relation extraction

`RelUtils` is another utility class in the OntoWiki package. It handles the process of relation extraction. Its fields and methods are shown on Figure 4.6. If we do not provide explicit references later in this section, then we discuss functions of this class. The `RelUtils` object maintains two manager

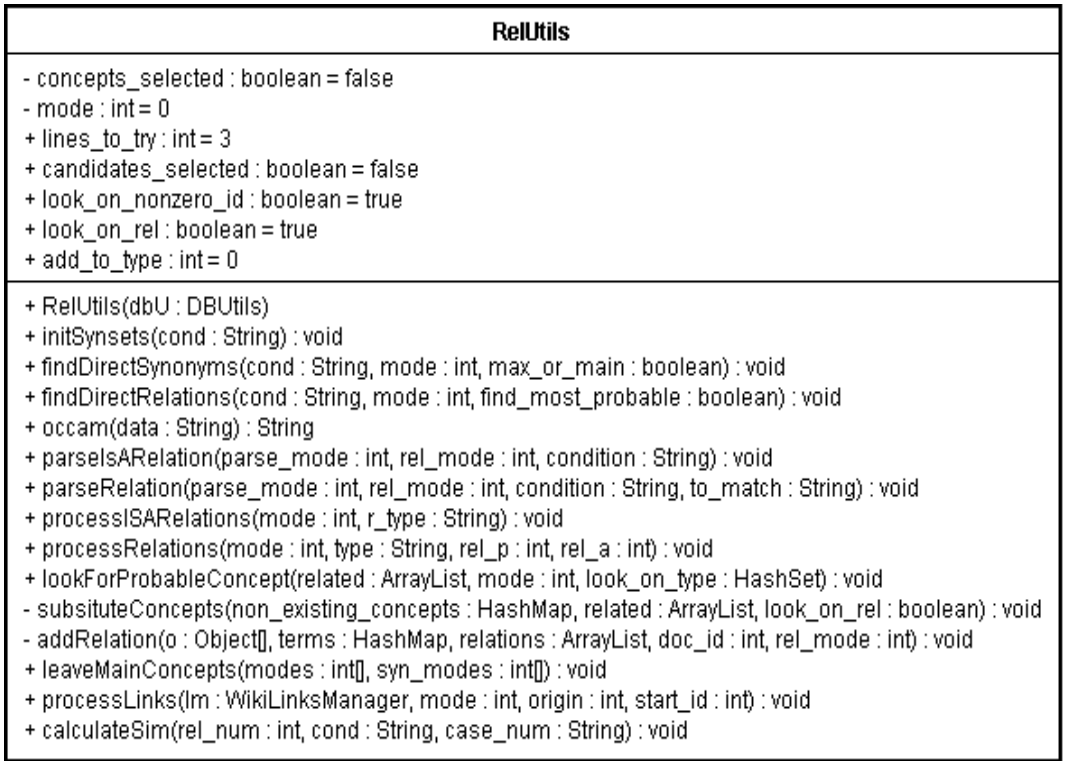


Figure 4.6: RelUtils class

objects: `ConceptsManager` and `RelationsManager`. The second one implements the already familiar `LinksManager` interface. `RelUtils` works with both the databases: OntoWiki DB and Ontology DB. For this purpose it also maintains the `OntoServInterface` object. The class implements the functionality, described in section 3.4.

Functions `parseIsARelation` and `parseRelation` implement rules, given by expressions 3.1 - 3.7. The first function parses documents and looks for known concepts in the environment of the current concept. The `lines_to_try` parameter determines the number of sentences around the concept where we will look for related concepts. The second one iteratively process doc-

uments, which titles or inside structure elements satisfy conditions given by 3.3. These functions insert relations, discovered, into the table RELATED_CONCEPTS. A function `processRelations` inserts the sets of discovered relations into local ontology tables. It also computes the similarity between related concepts using the function `calculateSim`. The function `processLinks` iterates through all the links in the given set of documents and looks for candidates, which suit to represent UNSPEC relations, given by 3.8 - 3.9.

The next two functions of the module `RelUtils` are intended to process the result of the function `parseRelation` and transfer it into the local ontology DB. The task of the synonyms detection is performed by the function `findDirectSynonyms`. It implements heuristics, given by expressions 3.1-3.2. This function also tries to find a match in the concepts set for the hypernym of the current concept. For instance, the result of parsing the title "Paris, Tennessee" will say that "Tennessee" is something more general for Paris (in the current context). We can not allow "Tennessee" to have the same ID as this "Paris". Thus, we leave "Paris" as a document concept and begin looking for the more general concept match for "Tennessee". It will be the document concept "Tennessee". Its "doc_id" will be set as the hypernym ID in the original relation. This action is performed by the `lookForProbableConcept` function. This feature works only if there is no ambiguity between the possible hypernym concepts. In other words, when such a concept, named "Tennessee" is unique. The utility function `leaveMainConcepts` removes less frequent synonyms from each of the concepts in relation. It means that the relation will be comprised from the most used synonyms of both the concepts. Then this new relation is added to the internal storage, `ArrayList relations`. The function `OntoServInterface.insertRelations` inserts relations into the local ontology database. The similarity between related concepts can be computed on this stage or later, using the function `calculateSim`.

Pruning the relation set. After we had extracted all the possible relations, it can appear that their amount is too large. It especially concerns relations of the type UNSPEC. To avoid the ontology overload we create our resulting ontologies using different pruning conditions. Another point

here is to avoid less important or noisy relations. The approaches for thresholding are described in more details in section 3.6. First, we can simply prohibit some particular types of relations. Second, we can apply a thresholding scheme and cut relations, which contain not important terms. It is possible to set that the target term should appear in the corpora not less than n times. To make it more strict, we can also ask for the certain level of similarity between concepts in the relation. When we narrowed the relations set, we can run a thresholding algorithm. This algorithm is implemented in the class `DBScanner`. It accepts coefficients for in-, out-, in-out and similarity level as parameters. Then the algorithm iterates through the specified relations set and marks suited relations. After this procedure finishes, the pruned set of relations can be inserted into the final ontology.

4.5 Classification with BINGO!

To simplify the task of classification we used BINGO! It includes modules for processing input documents, implements variants of SVM (provided by SVM Light package). It also provides tools for disambiguation and ontological concepts mapping in the classification purpose.

We will not go into deep details describing BINGO!. The input training and test data should be parsed by BINGO! first. BINGO! creates an internal representation of both the sets of documents. Each document is represented as a vector with term frequencies counted and concepts disambiguated. It uses the following formulas:

$$tf_{t,d} = tf/max(tf_d)$$

$$idf_t = log(1 + N/df_t)$$

where N is the size of the corpus; tf_d is the maximum term frequency in the document d; df_t is a number of document, which contain term t. For the feature selection BINGO! uses the Mutual Information criteria, which measures the discriminative power of the feature F_i in the topic C_k

$$MI(f_i, C_k) = \sum_{F \in \{F_i, \overline{F_i}, F\}, C \in \{C_k, \overline{C_k}\}} P(F \wedge C) log \frac{P(F \wedge C)}{P(F)P(C)} \qquad (4.1)$$

BINGO! asks a user to provide the training directory or file and the test directory or file. Then it asks for a number of features for the mutual information calculation. The classification can be performed by four different methods: Naive Bayes classification with Laplace smoothing, SVM classification, SVM classification with ontology concepts mapping and disambiguation and SVM with disambiguation plus incremental mapping. All four methods were used in experiments.

4.5.1 BINGO! disambiguation method

In the process of disambiguation we examine possible meanings for each of the terms with regard to their environment. For each of the terms we take the paragraph where it was found, or in the case of XML document - the term's subtree. We put this piece of document terms into the sparse vector, let us name it $paragraphVector$.

Each of the terms in its turn stores a set of ontological concepts IDs, where it appears as a part of synset. We call this set *related concepts*. For each of the related concepts we also create a sparse vector, so we end up with a set $\{concept1Vector, concept2Vector...conceptNVector\}$. These vectors are created from the ontological environment of the related concept and so-called gloss, which represents a short textual description of the term in WordNet. The environment is comprised from concepts which are in hyperymy, meronymy and similarity relations with the current concept. In other words, these concept vectors represent available senses for the current term in the ontology.

Now we can calculate the cosine similarity between the $paragraphVector$ and each of the concept vectors. Larger cosines represent larger overlap in the words usage between the term in the document and in the ontology. The most similar sense is chosen for the current term and the process is repeated for each of the terms in the document.

4.5.2 BINGO! incremental mapping method

The incremental mapping procedure affects the classification process. Suppose in the test document we found a term, which does not appear in the feature space of our model. Statistical classifiers can not distinguish, how useful or useless this term is in the context or current document when it does not appear in the model. Incremental mapping allows to find an unknown term in the ontology and map it to the feature in the model. This process maps unknown terms to the most close term features in sense of the ontological relatedness and measure of similarity.
The whole process holds as follows. When creating sparse vectors for test documents, the special flag determines whether the incremental merge should be performed. If it is true, we look for the *best ontology match* for each of the unknown terms. Like in the previous case we try to find all the concepts in which the current term occurs. When such a set of related concepts is found, we calculate the similarity between each of the related concepts and the current term with regard to its context (like in the previous case). When such a match is found, we put a new feature into the document sparse vector. This process repeats for each of the unknown terms in each of the test documents.

4.5.3 Classification with phrases detection

The `DocumentParser` class in the `OntoWiki` package was embedded in the BINGO! code as external parser. It functionality is described in details in paragraph 4.2. It allows to parse documents with regard to the ontological concepts set. During this process the document vector is filled not only with the single terms, but with the phrasal concepts also.

Chapter 5

Experimental results

5.1 General experimental setup

It is a difficult question, how to evaluate the quality of the ontology. We decided to make comparative experiments to evaluate the performance of the ontology-driven classification on our ontologies versus WordNet. We tried to check extracted ontologies on various combinations of the training documents sets sizes, documents length and style they written. In this chapter we provide evidences that our approach allows to create ontologies, that can outperform WordNet in the tasks of classification.

In these purposes we created four ontologies. All of these ontologies were extracted with the same policy, applied to the concepts extraction process (see 3.3.1). They differ in the rules, used for the relations extraction (see 3.4). We took only S-terms and A-terms for concepts. Our policy was to lowercase everything, to keep both the stemmed and unstemmed terms, to remove all special characters and to keep numbers in terms, but not in relations. We added synonyms to the concept only if they had no lexical overlap with the other terms of this concept. All the possible well-formed and unspecified relations were extracted into the local ontology database first ($LO2$). Probabilities of co-occurrence were calculated for each of the relations.

We compared our ontologies with the WordNet ontology. The statistics for WN is given in Table 5.1.

For the first ontology $LO1$ we took all the well-formed relations. L-UNSPEC and T-UNSPEC relations were thresholded at the level of simi-

Number of synsets	109684
Number of distinct terms	191350
Number of hypernymy relations	310319
Number of similar_to relations	49748
Number of see_also relations	5011

Table 5.1: WordNet statistics

larity of 0.075. This ontology has the most variety of heuristics between all of them. The statistics for LO1 is given in Table 5.2.

Number of synsets	250407
Number of distinct terms	318415
Number of hypernymy relations	88603
Number of similar_to relations	11900
Number of see_also relations	41503

Table 5.2: LO1 statistics

The second ontology, *LO*3 contains the same set of concepts except a small amount, which came from processing L-UNSPEC-relations. Only the hierarchical well-formed relations, generated by the first two rules from equation 3.1 have been included. This ontology is the smallest one in our setup. The statistics for LO3 is given in table 5.3.

Number of synsets	250394
Number of distinct terms	306916
Number of hypernymy relations	28825

Table 5.3: LO3 statistics

The third ontology, *LO*4 contains the same set of concepts as the second one. All the hierarchical well-formed relations have been used. This ontology is the second smallest one. The statistics for LO4 is given in table 5.4.

The fourth ontology, *LO*5 contains the same set of concepts with the first one. Only the relations, which came from links have been used. The set

Number of synsets	250394
Number of distinct terms	306916
Number of hypernymy relations	88603

Table 5.4: LO4 statistics

of relations for this ontology was extracted by thresholding the whole set in *LO*2. We took document with the frequency for a target term not less than 20. The statistics for LO4 is given in Table 5.5.

Number of synsets	250407
Number of distinct terms	318415
Number of hypernymy relations	581042

Table 5.5: LO5 statistics

The pivot table 5.6 presents details about types of relations, which exist in each of our ontologies.

Rule	LO1	LO3	LO4	LO5
G-HYP	14255	14255	14255	0
Ex-HYP	60507	14324	60507	0
S-HYP	8874	0	8874	0
SS-HYP	4613	0	4613	0
T-UNSPEC	0	0	0	254492
L-UNSPEC	0	0	0	326442
SIMTO	0	0	0	0
UNSPEC	124372	0	0	0
TOPLIST	55302	0	0	0

Table 5.6: Number of relations obtained by different heuristics across ontologies

For the main series of experiments, we took the Reuters collection. It has a convenient topic structure and training/testing parts separation. We used "ModApte". For this scenario we took two rich topics, *Acq* and *Earn*.

We fixed the number of test documents on 150. The training set was chosen separately; we split it randomly on different-sized samples. The size of these samples goes from ten to two hundreds. We did not intentionally select larger documents to comprise our samples. A usual document in this collection is quite small, about 2-3k. The style they written is official and concise, a document usually describe some event. The task of classification was to assign a proper topic to each of the training documents. We should mention here that experiments were quite time consuming. That's why in our experiments we did not perform multiple sampling for training data and results averaging over all samples. We created our training sets only once. In the further investigation it will be more accurate to provide such an analysis, however in this thesis the task was both the framework development and performance evaluation. This fact has restricted us in time.

We analyze the following classifiers:

- NB - A terms-only Naive Bayes classifier.
- SVM - A multiclass SVM classifier.
- $SVM + D$ - SVM with the ontology-driven terms disambiguation (see 4.5.1).
- $SVM + D + I$ - SVM with the ontology-driven terms disambiguation and incremental mapping (see 4.5.2).
- $NB + P$ - A Naive Bayes classifier with the ontology-driven phrases extraction (see 4.5.3).
- $SVM + P + D$ - SVM with the ontology-driven phrases extraction and terms disambiguation.
- and $SVM+P+D+I$ - SVM with the ontology-driven phrases extraction and disambiguation plus incremental mapping.

All the SVM variants are implemented in BINGO! They use a mutual information criteria 4.1 for the feature selection process. The number of features

was set to 500.

To evaluate the results, we used standard well-known measures such as Precision, Recall and F1 variant of the popular F-measure. We calculated statistics per topic as well as aggregated statistics. Precision and Recall per topic are given by standard formulas:

$$Precision = \frac{N \text{ of docs, correctly classified to the topic}}{N \text{ of the docs classified to this topic}}$$

;

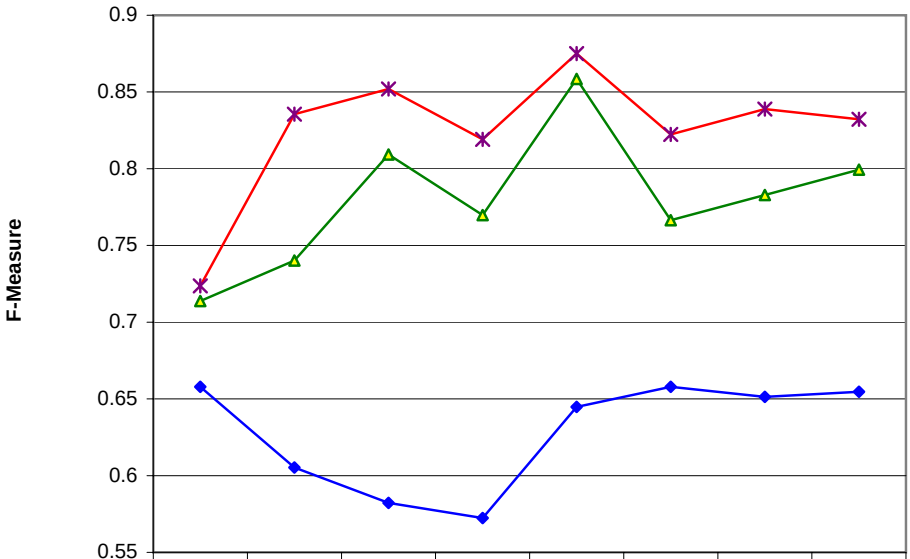
$$Recall = \frac{N \text{ of docs, correctly classified to the topic}}{N \text{ of documents, which belong to the topic}}$$

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

We also calculated a microaverage of these values. In our evaluation we used the microaveraged F1 measure.

5.2 Baseline experiment

As a baseline for further experiments we took the performance of the Naive Bayes and SVM classifiers. Documents were parsed with the standard parser, terms were stemmed. Further we compare the performance of every classifier on every ontology (WN, LO1,...) with regard to the given baseline.

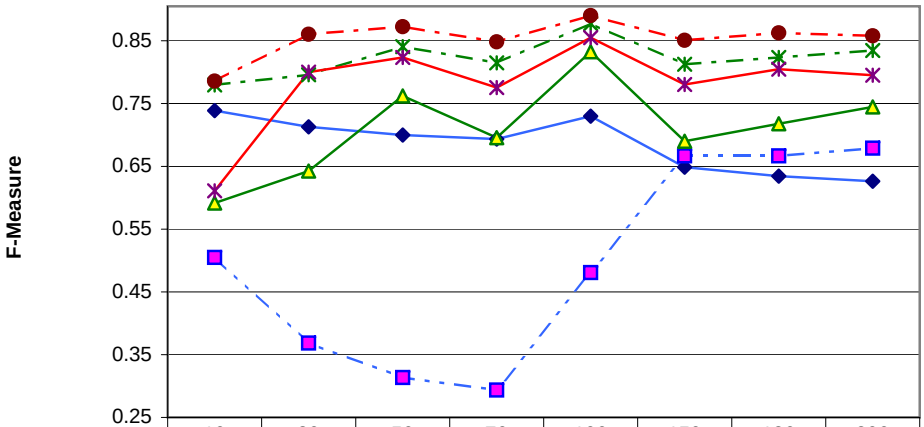


	10	30	50	70	100	150	180	200
Baseline NB	0.657895	0.605263	0.582237	0.572368	0.644737	0.657895	0.651316	0.654605
Baseline SVM	0.713816	0.740132	0.809211	0.769737	0.858553	0.766447	0.782895	0.799342
Baseline SVM+D+I	0.723684	0.835526	0.851974	0.819079	0.875	0.822368	0.838816	0.832237

Figure 5.1: Microaveraged F1 as a function of the training set size for NB and SVM

Figure 5.1 shows the overall performance of NB, simple SVM and $SVM + D + I$ with disambiguation and incremental mapping. First two methods do not use any ontology. A baseline for the ontology-driven classification is the performance of LO3 on the same input file. LO3 was chosen for this task because it is a smallest ontology in our setup. The ontology lookup depth was set to 1.

The performance of NB classification is the worst, like expected. SVM result has reasonable accuracy, while SVM with the incremental mapping outperforms both.



	10	30	50	70	100	150	180	200
Acq NB	0.738693	0.712919	0.699764	0.693396	0.73	0.648649	0.634483	0.626335
Earn NB	0.504762	0.368421	0.313514	0.293478	0.480769	0.666667	0.666667	0.678899
Acq SVM	0.591549	0.642534	0.762295	0.695652	0.832685	0.689956	0.717949	0.74477
Earn SVM	0.779747	0.795866	0.840659	0.814815	0.877493	0.812665	0.823529	0.834688
Acq SVM+D+I	0.611111	0.8	0.823529	0.77551	0.854962	0.780488	0.804781	0.795181
Earn SVM+D+I	0.785714	0.860335	0.872521	0.848485	0.890173	0.850829	0.862745	0.857939

Training set size

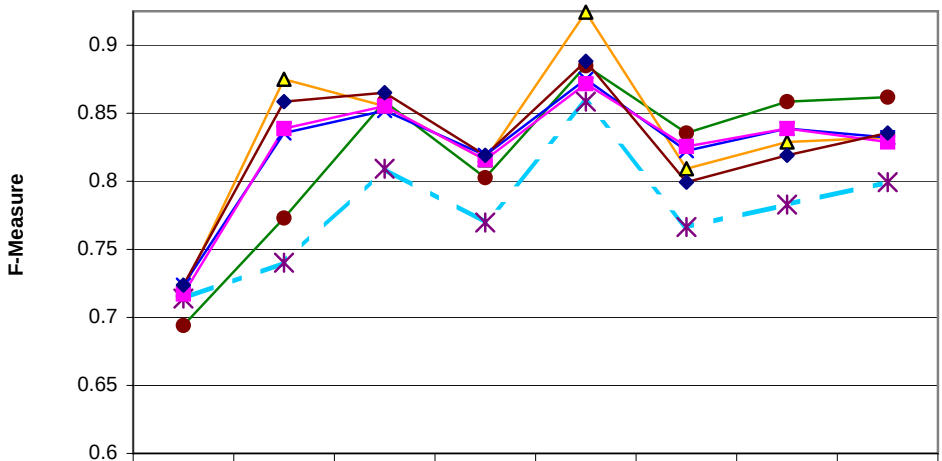
Figure 5.2: Microaveraged F1-Measure for two classes

We consider that the use of ontology makes the classification results better. Another observation is that ontological lookups make the results smoother. We can see that the corpora-conditioned performance falls at the level of 70 and 150 documents are more expressed in the performance of SVM. $SVM + D + I$ performs smoother in these cases. These two curves, the NB performance and SVM performance will appear as baselines on the charts later in this chapter.

Figure 5.2 presents the classification behavior for both the classes, *Acq* and *Earn*. Naive Bayes results are in blue lines, SVM results are in green and $SVM + D + I$ are in red. The SVMs show quite similar behavior, while NB shows a big drop down in the classification accuracy for the class *Earn*.

5.3 SVM with ontology-driven terms disambiguation

In this section we study the behavior of the SVM classifier with the ontology-driven terms disambiguation. In this setup the input documents were parsed with the standard BINGO! parser. The disambiguation was performed during the process of the document vectors creation. To accomplish this task the term environment from the document in the form of the sparse vector is extracted. For detail about the disambiguation process see section 4.5.1. The environment depth parameter was set to 1. We tested WordNet vs each of our ontologies. The baseline for SVM is shown in blue.



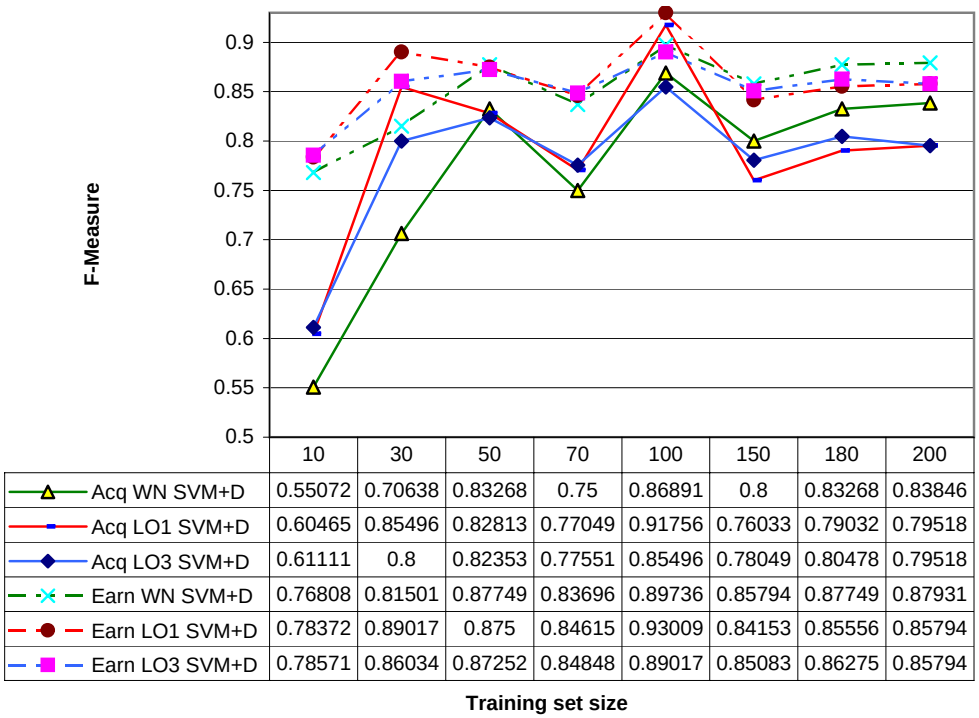
	10	30	50	70	100	150	180	200
Baseline SVM	0.713816	0.740132	0.809211	0.769737	0.858553	0.766447	0.782895	0.799342
WN SVM+D	0.694079	0.773026	0.858553	0.802632	0.884868	0.835526	0.858553	0.861842
LO1 SVM+D	0.720395	0.875	0.855263	0.815789	0.924342	0.809211	0.828947	0.832237
LO3 SVM+D	0.723684	0.835526	0.851974	0.819079	0.875	0.822368	0.838816	0.832237
LO4 SVM+D	0.717105	0.838816	0.855263	0.815789	0.871711	0.825658	0.838816	0.828947
LO5 SVM+D	0.723684	0.858553	0.865132	0.819079	0.888158	0.799342	0.819079	0.835526

Figure 5.3: Microaveraged F1 as a function of training set size for SVM+D

Figure 5.3 presents the $SVM + D$ classification results. It shows the dependency of the F1 measure on the number of training documents. Due to the fact that all the four of our ontologies have mainly coinciding concepts set, the results are very close for all of them. All the ontologies are above

baseline. On the smaller samples we can see that our ontologies outperform WordNet. When the number of the trainings in the experiment grows, the gap becomes insignificant. The biggest samples shows the minor benefit of WordNet. We explain this phenomena as follows. At the beginning our ontologies perform better, because their concepts sets are more comprehensive. Increasing the size of the training sample, we involve more noise in the model. The relations hierarchy of WordNet is more strict, thats why it can perform the disambiguation better, cutting out more noise. However, this setup does not take into account the discriminative power of concept phrases. Consequent experiments study this problem.

Figure 5.6 presents the classes behavior for the $SVM + D$ method on three ontologies LO1, LO3, WN.

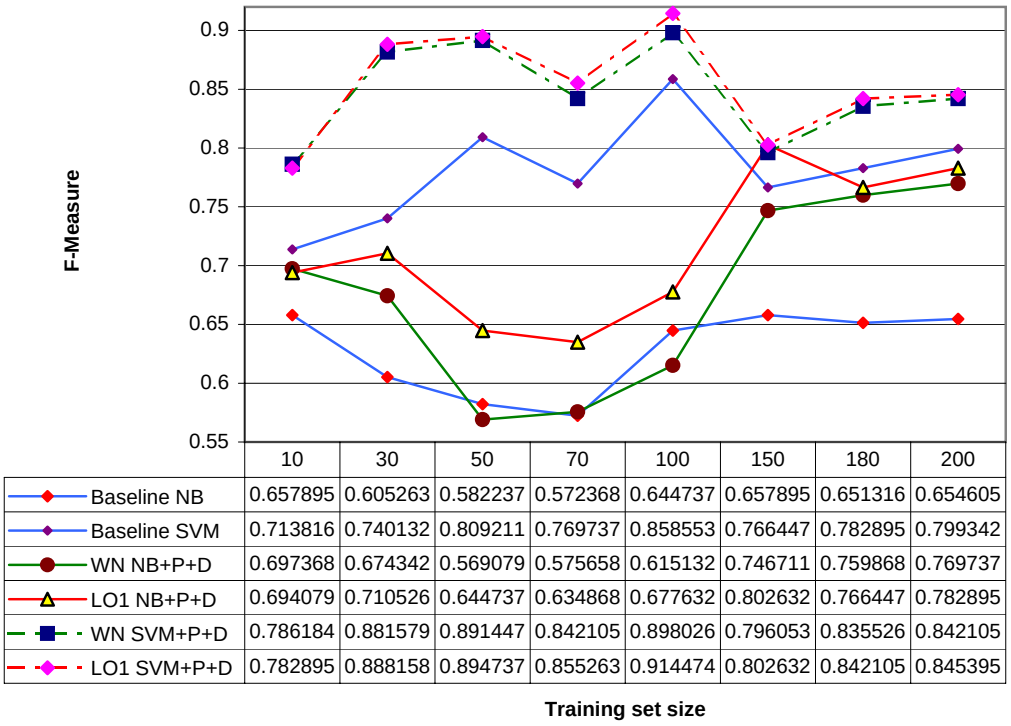


	10	30	50	70	100	150	180	200
Acq WN SVM+D	0.55072	0.70638	0.83268	0.75	0.86891	0.8	0.83268	0.83846
Acq LO1 SVM+D	0.60465	0.85496	0.82813	0.77049	0.91756	0.76033	0.79032	0.79518
Acq LO3 SVM+D	0.61111	0.8	0.82353	0.77551	0.85496	0.78049	0.80478	0.79518
Earn WN SVM+D	0.76808	0.81501	0.87749	0.83696	0.89736	0.85794	0.87749	0.87931
Earn LO1 SVM+D	0.78372	0.89017	0.875	0.84615	0.93009	0.84153	0.85556	0.85794
Earn LO3 SVM+D	0.78571	0.86034	0.87252	0.84848	0.89017	0.85083	0.86275	0.85794

Figure 5.4: Microaveraged F1-Measure for two classes for SVM+D

5.4 NB and SVM with ontology-driven phrases extraction

In this section we study the behavior of the Naive Bayes and SVM classifiers in the case of phrases extraction. In this setup the input documents were processed with regard to the ontological concept set. The feature space consists of the mixture of simple terms as well as of the concept phrases, detected. For details see section 4.5.3. As far as all the four our ontologies have a similar concept set, we can choose one. We tested the performance of LO1 vs WordNet. The baselines are shown in blue lines.



	10	30	50	70	100	150	180	200
Baseline NB	0.657895	0.605263	0.582237	0.572368	0.644737	0.657895	0.651316	0.654605
Baseline SVM	0.713816	0.740132	0.809211	0.769737	0.858553	0.766447	0.782895	0.799342
WN NB+P+D	0.697368	0.674342	0.569079	0.575658	0.615132	0.746711	0.759868	0.769737
LO1 NB+P+D	0.694079	0.710526	0.644737	0.634868	0.677632	0.802632	0.766447	0.782895
WN SVM+P+D	0.786184	0.881579	0.891447	0.842105	0.898026	0.796053	0.835526	0.842105
LO1 SVM+P+D	0.782895	0.888158	0.894737	0.855263	0.914474	0.802632	0.842105	0.845395

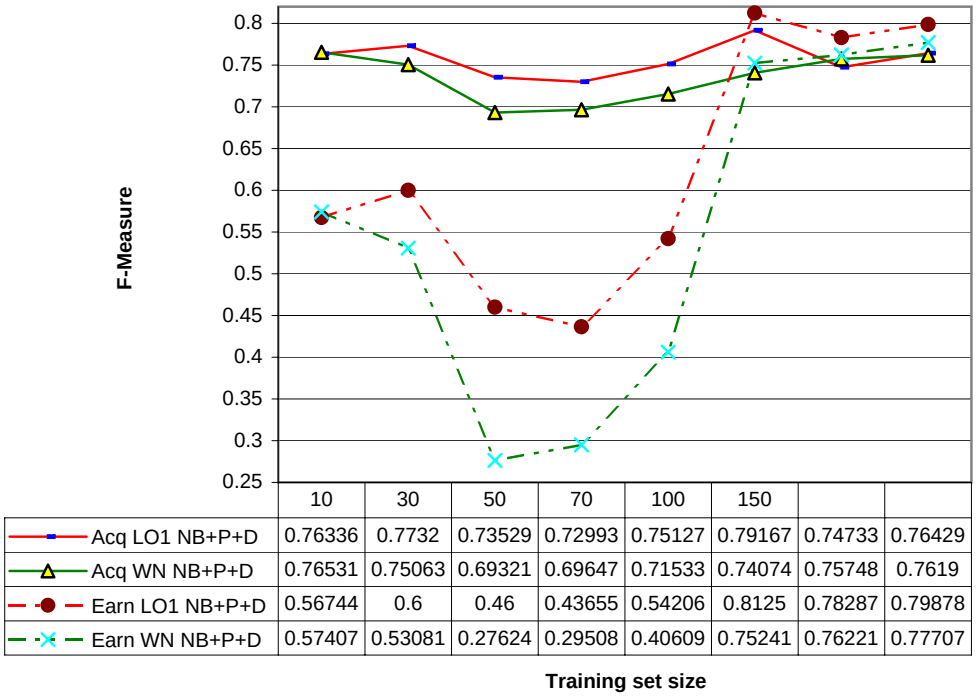
Figure 5.5: Microaveraged F1 as a function of training set size for NB+P and SVM+P

Figure 5.5 shows the dependency of the F1 measure on the number of training documents for $NB + P$ and $SVM + P$. The performance of the $NB + P$ classification is better than the baseline NB. The $NB + P$ results for both $LO1$ and $WordNet$ are similar in the behavior, however $LO1$ is visibly better. It can be explained with the fact that $LO1$ concept space

is more saturated and more flexible in the words usage, due to the natural properties of the corpus, we extracted our ontologies from.

The classification accuracy shows the same trends that in the baseline NB case, however the most significant distinction reveals with the grows of the size of the training sample. As the standard NB seems to reach the point of saturation at the level of 100 documents, the ontology-enriched $NB + P$ continue to benefit from the grows of the feature space. We consider that taking phrases into account allows to capture more semantics and make classification more precise.

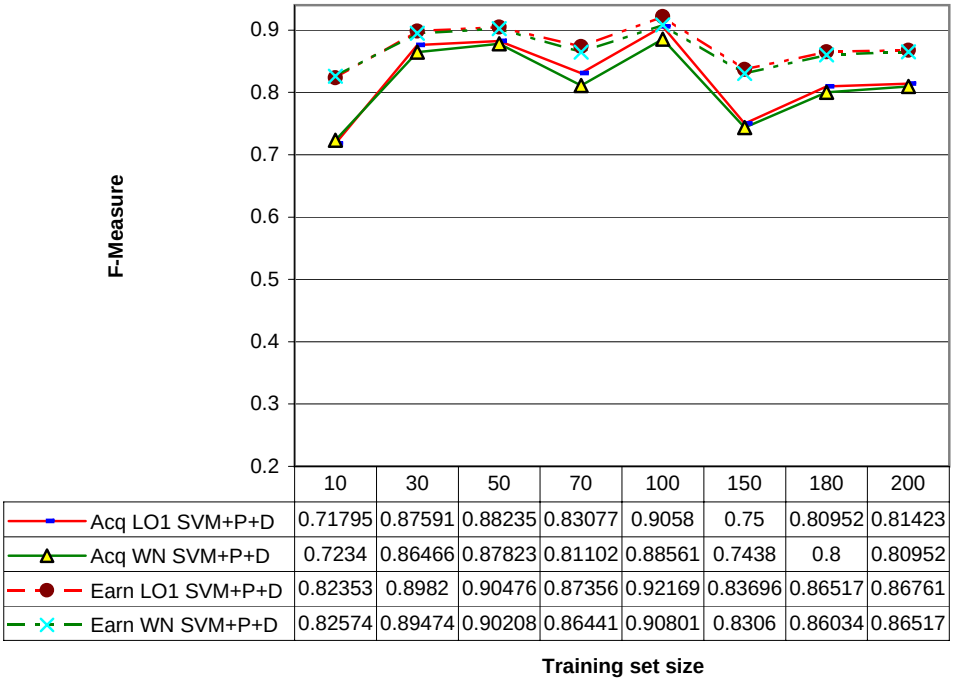
Figure 5.6 presents the classes behavior for the $NB + P$ method. The results for $LO1$ are shown in red lines and WN results are in green.



	10	30	50	70	100	150		
Acq LO1 NB+P+D	0.76336	0.7732	0.73529	0.72993	0.75127	0.79167	0.74733	0.76429
Acq WN NB+P+D	0.76531	0.75063	0.69321	0.69647	0.71533	0.74074	0.75748	0.7619
Earn LO1 NB+P+D	0.56744	0.6	0.46	0.43655	0.54206	0.8125	0.78287	0.79878
Earn WN NB+P+D	0.57407	0.53081	0.27624	0.29508	0.40609	0.75241	0.76221	0.77707

Figure 5.6: Microaveraged F1-Measure for two classes

Figure 5.7 compares the classes behavior for two ontologies, for the $SVM + P$ method. The results for $LO1$ are shown in red lines and WN results are in green.

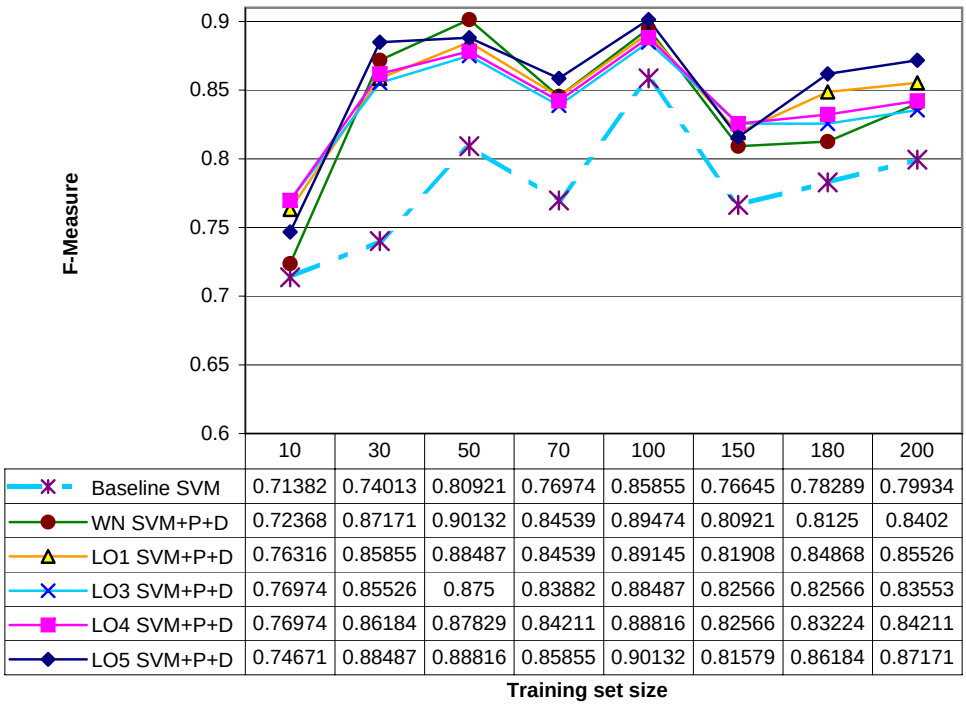


	10	30	50	70	100	150	180	200
Acq LO1 SVM+P+D	0.71795	0.87591	0.88235	0.83077	0.9058	0.75	0.80952	0.81423
Acq WN SVM+P+D	0.7234	0.86466	0.87823	0.81102	0.88561	0.7438	0.8	0.80952
Earn LO1 SVM+P+D	0.82353	0.8982	0.90476	0.87356	0.92169	0.83696	0.86517	0.86761
Earn WN SVM+P+D	0.82574	0.89474	0.90208	0.86441	0.90801	0.8306	0.86034	0.86517

Figure 5.7: Microaveraged F1-Measure for two classes

5.5 SVM with ontology-driven terms disambiguation and phrases detection

In this section we study the behavior of the SVM classifier in the case of terms disambiguation and phrases detection. In this setup the input documents are processed with regard to the ontological concept set. The feature space consists of the mixture of simple terms as well as of the concept phrases, detected. We tested the performance of $LO1$ vs $WordNet$. The baselines is shown in blue lines. As we could predict, a powerful SVM classifier enriched with ontological semantical mapping boosted the accuracy of classification in comparison with the baseline SVM. The performance of all the ontologies is very similar, however two bigger ontologies, $LO1$ and $LO5$ outperform WN in cases of larger training sets. The chart is shown on Figure 5.8.

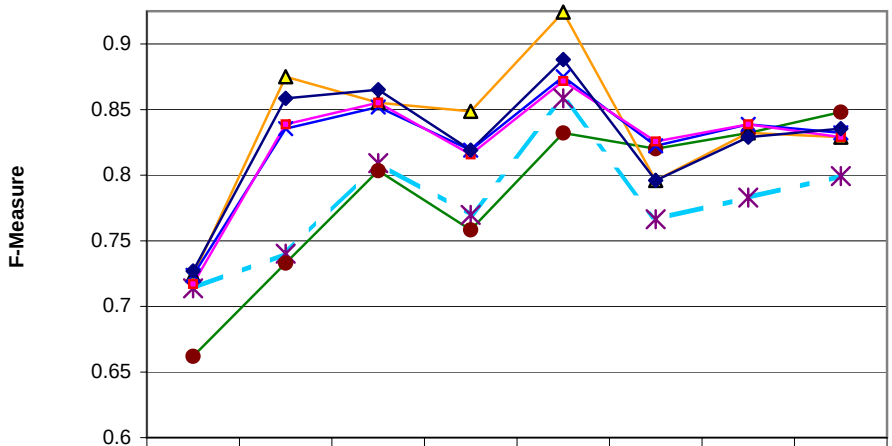


	10	30	50	70	100	150	180	200
Baseline SVM	0.71382	0.74013	0.80921	0.76974	0.85855	0.76645	0.78289	0.79934
WN SVM+P+D	0.72368	0.87171	0.90132	0.84539	0.89474	0.80921	0.8125	0.8402
LO1 SVM+P+D	0.76316	0.85855	0.88487	0.84539	0.89145	0.81908	0.84868	0.85526
LO3 SVM+P+D	0.76974	0.85526	0.875	0.83882	0.88487	0.82566	0.82566	0.83553
LO4 SVM+P+D	0.76974	0.86184	0.87829	0.84211	0.88816	0.82566	0.83224	0.84211
LO5 SVM+P+D	0.74671	0.88487	0.88816	0.85855	0.90132	0.81579	0.86184	0.87171

Figure 5.8: Microaveraged F1 as a function of training set size SVM+D+P

5.6 SVM with ontology-driven terms disambiguation and incremental mapping

The particular point of interest is to find matches for the concepts, that are not in our model, but exists in ontology. In this section we study the performance of the SVM classifier with disambiguation and incremental merge for unknown concepts. In this setup the input documents were parsed with the standard BINGO! parser. At the classification step BINGO! finds matches in the ontology for the terms from the test set. In this experiment, we tested all of our ontologies VS WordNet. The classification results are shown on Figure 5.9 We can see that, in comparison with the $SVM + D$ experiment shown of Figure 5.3, the gap between the accuracy of WordNet-driven classification and our ontologies became more significant.



	10	30	50	70	100	150	180	200
Baseline SVM	0.713816	0.740132	0.809211	0.769737	0.858553	0.766447	0.782895	0.799342
WN SVM+D+I	0.661972	0.732984	0.803478	0.758261	0.832168	0.820168	0.832168	0.84808
LO1 SVM+D+I	0.723684	0.875	0.855263	0.848684	0.924342	0.796053	0.832237	0.828947
LO3 SVM+D+I	0.723684	0.835526	0.851974	0.819079	0.875	0.822368	0.838816	0.832237
LO4 SVM+D+I	0.717105	0.838816	0.855263	0.815789	0.871711	0.825658	0.838816	0.828947
LO5 SVM+D+I	0.726974	0.858553	0.865132	0.819079	0.888158	0.796053	0.828947	0.835526

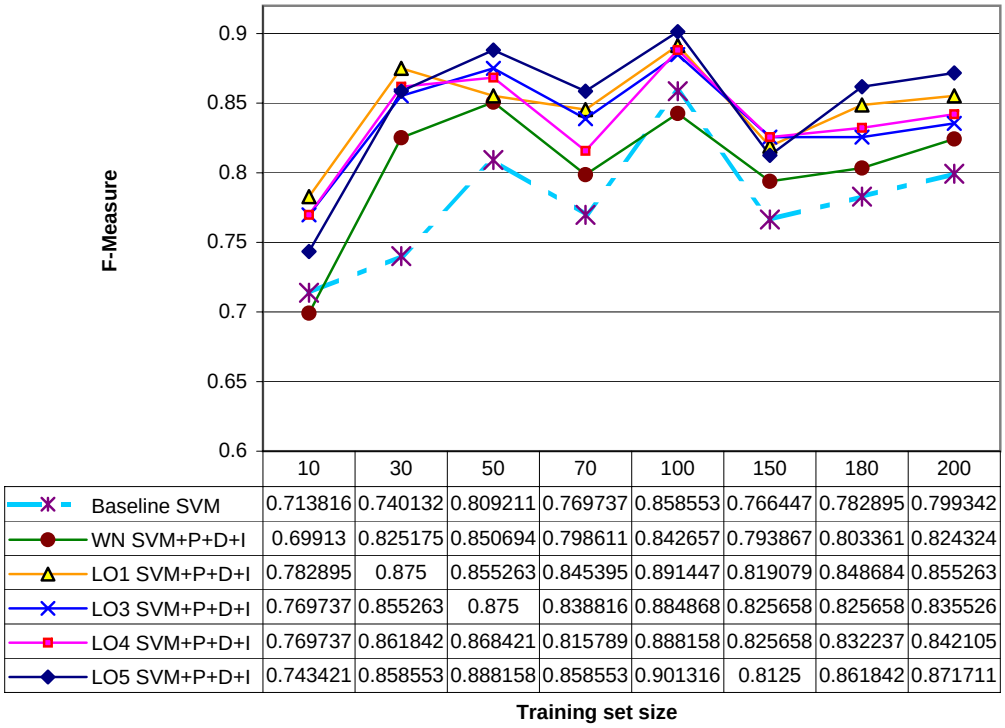
Training set size

Figure 5.9: Microaveraged F1 as a function of training set size SVM+D+I

5.7 SVM with ontology-driven terms disambiguation, phrases detection and incremental mapping

This experiment presents the performance of the SVM classifier with disambiguation, phrases detection and incremental merge for unknown concepts. For details see section 4.5.3. In this setup the input documents were parsed with regard to the ontological concept set. Extracted terms were disambiguated. At the classification step BINGO! finds matches in ontology for the terms from the test set. In this experiment, we tested all of our ontologies VS WordNet. The classification results are shown on Figure 5.10.

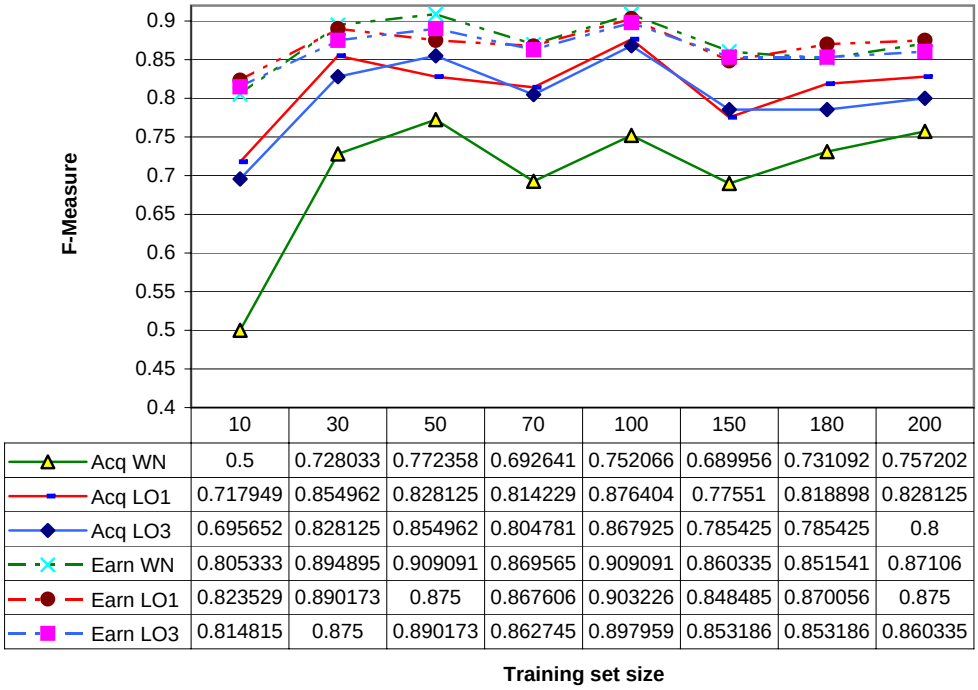
The performance differs more significantly, than in the case without phrases detection. All of our ontologies perform better than WN, however between themself two larger ontologies $LO1$ and $LO5$ perform best of all. The accuracy for the classes for $LO1$, $LO3$ and WN ontologies is shown on Figure



	10	30	50	70	100	150	180	200
Baseline SVM	0.713816	0.740132	0.809211	0.769737	0.858553	0.766447	0.782895	0.799342
WN SVM+P+D+I	0.69913	0.825175	0.850694	0.798611	0.842657	0.793867	0.803361	0.824324
LO1 SVM+P+D+I	0.782895	0.875	0.855263	0.845395	0.891447	0.819079	0.848684	0.855263
LO3 SVM+P+D+I	0.769737	0.855263	0.875	0.838816	0.884868	0.825658	0.825658	0.835526
LO4 SVM+P+D+I	0.769737	0.861842	0.868421	0.815789	0.888158	0.825658	0.832237	0.842105
LO5 SVM+P+D+I	0.743421	0.858553	0.888158	0.858553	0.901316	0.8125	0.861842	0.871711

Figure 5.10: Microaveraged F1 as a function of training set size SVM+D+P+I

5.11. We can see that $LO1$ and $LO3$ capture the word usage for both the classes mostly balanced, whether WN shows a significant dropdown, capturing the class Acq.



	10	30	50	70	100	150	180	200
Acq WN	0.5	0.728033	0.772358	0.692641	0.752066	0.689956	0.731092	0.757202
Acq LO1	0.717949	0.854962	0.828125	0.814229	0.876404	0.77551	0.818898	0.828125
Acq LO3	0.695652	0.828125	0.854962	0.804781	0.867925	0.785425	0.785425	0.8
Earn WN	0.805333	0.894895	0.909091	0.869565	0.909091	0.860335	0.851541	0.87106
Earn LO1	0.823529	0.890173	0.875	0.867606	0.903226	0.848485	0.870056	0.875
Earn LO3	0.814815	0.875	0.890173	0.862745	0.897959	0.853186	0.853186	0.860335

Figure 5.11: Microaveraged F1-Measure for two classes

Chapter 6

Conclusion and Future Work

Ontologies become a useful methodology nowadays, however the process of ontology development is time-consuming and requires human work. In this thesis we presented a new approach to the problem of automatic ontologies extraction from a given corpus.

The main idea behind this work is that we can use the valuable resources, namely human-developed text collections, to learn concepts and automatically extract hierarchical relations between them. The fact that concepts can be not only terms, but phrases leads to the additional benefit. The work on this thesis has shown a big potential in the exploration of continuously growing information sources in natural language, such as Wikipedia. We proposed a flexible framework to extract related entities from the tagged text. It allows to easily incorporate new features and reflect changes in the underlying data source as well as to be enriched with the new functionality.

We present the empirical study of the proposed methodology. We extracted several ontologies and performed comparison between them and WordNet. We used ontologies for the word sense disambiguation and unknown concepts mapping in the ontology-driven text classification process.
During the experimental study, we revealed that the natural properties of our ontologies, extracted from the real-life text collection can boost results of the ontology-driven classification. Among these properties we emphasize the size and diversity of the concept set, ability to capture phrasal concepts and the amount of relations of different types.

We understood that such natural knowledge sources can be noisy. To

fight this problem, we proposed a simple method to prune out some noise. This is a question of great importance. Therefor, one of the main directions of further studies should be the development of a methodology for ontology pruning. This issue is related to the problem of taxonomies extraction. Ontologies of smaller size or problem-specific ontologies can be applied in the tasks of information search and classification personalization. Another exciting issue in the ontology extraction process is to find out more elaborated ways to inferring relationships between the concepts. The main goal is to study and understand the strengths and weaknesses of our approach and find a new ways to improve it.

Bibliography

[1] Gruber, T.
A translation approach to portable ontology specifications
Knowledge acquisition, 5(2), 199-220, 1993

[2] Gruber, T. R.
A translation approach to portable ontology specifications.
Knowledge Acquisition, 5:199220, 1993.

[3] Gruber, T.,Olsen, R.
An ontology for engineering mathematics. Technical report, Knowledge Systems Laboratory, Stanford University, CA, 1994.
http://ksl-web.stanford.edu/knowledge-sharing/papers/engmath.html

[4] Ying, D., Foo, S.
Ontology Research and Development: Part 1 - A Review of Ontology Generation.
Journal of Information Science 28(2), 2001.

[5] Uschold, M.
Building Ontologies: Towards a Unified Methodology.
Proceedings of Expert Systems 96, the 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems, 1996.

[6] Guarino, N. and Poli, R. (eds.)
Formal Ontology in Conceptual Analysis and Knowledge Representation.
Special issue of the International Journal of Human and Computer Studies, vol. 43 n. 5/6, Academic Press, 1995.

[7] N. Guarino and P. Giaretta.
Ontologies and knowledge bases: Towards a terminological clarification.
In N.J.I. Mars, editor, Towards Very Large Know ledge Bases. IOS Press, 1995.

[8] Guarino, N.
Formal Ontology and Information Systems. In N. Guarino, editor, Proceedings of the 1st International Conference on
Formal Ontologies in Information Systems, FOIS'98, Trento, Italy, pages 3– 15. IOS Press, June 1998.

[9] Guarino, N.
Understanding, building and using ontologies.
Intl. J. of Human and Computer Studies 46(2/3), 1997, 293-310

[10] Bateman, J.
The Theoretical Status of Ontologies in Natural Language Processing.
In S. Preuss and B. Schmitz, eds, 'Text Representation and Domain Modelling - ideas from linguistics and AI', KIT-Report 97
Technische Universitaet Berlin, pp. 50 - 99.

[11] Benjamins, R., Gomez Perez,A.
Knowledge Management through Ontologies.
In: PAKM-98, pp. 5.1-5.12, 1998.
`citeseer.ist.psu.edu/article/benjamins98knowledge.html`

[12] B. Swartout, R. Patil, K. Knight, and T. Russ
Toward Distributed Use of Large-Scale Ontologies
In Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW-96), Banff, Alberta, Canada, November 9-14, 1996.

[13] Fellbaum, C. (edt)
WordNet: An Electronical Lexical Database
The MIT Press, Cambridge, Massachusetts, 1998.

[14] Priss, U.
The Formalization of WordNet by Methods of Relational Concept Analysis.
In: Fellbaum, Christiane (ed.), WordNet: An Electronic Lexical Database and Some of its Applications, MIT press, 1998, p. 179-196.

[15] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, W. R. Swarton
Enabling technology for knowledge sharing
AI Magazine 12 (3) (1991) 36–56.

[16] **Generalized Upper Model**
Online document. `http://www.darmstadt.gmd.de/publish/komet/gen-um/newUM.html`

[17] **Cyc ontology**
Online document. `www.cyc.com`

[18] Gruñinger, M., Fox, M.
Methodology for the design and evaluation of ontologies.
In Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing held in conjunction with IJCAI-95, Montreal, Canada, 1995. `http://www.eil.utoronto.ca/tove/toveont.html`

[19] Uschold, M., Gruñinger, M.
The Enterprise Ontology
The Knowledge Engineering Review , Vol. 13, Special Issue on Putting Ontologies to Use
`http://www.aiai.ed.ac.uk/project/pub/documents/1998/98-ker-ent-ontology.ps`

[20] Fensel D., van Harmelen, F., Akkermans, H. et al.
Ontoknowledge: Ontology-based tools for knowledge management.
In Proceedings of the eBusiness and eWork 2000 Conferencee (EMMSEC 2000), Madrid, Spain, October 2000.

[21] **The web-site of Ontoknowledge Project**
Online document. http://www.ontoknowledge.org/

[22] Sowa, J.F.
Knowledge Representation: Logical, Philosophical, and Computational Foundations
Brooks Cole Publishing Co., Pacific Grove, CA, 2000. Actual publication date, 16 August 1999

[23] AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy
Learning to Map between Ontologies on the Semantic Web
In The Eleventh International WWW Conference, Hawaii, US, 2002

[24] Halevy A., Madhavan J.
Corpus-Based Knowledge Representation
Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI-03, 2003

[25] Ding, Y.
Ontology: The enabler for the Semantic Web
Online document. http://homepage.uibk.ac.at/~c703205/download/ontology.pdf

[26] Fernandez Lopez, M.
Overview Of Methodologies For Building Ontologies
Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5) Stockholm, Sweden, August 2, 1999

[27] Wikipedia online resources
Wikipedia Guide
Online document. http://meta.wikimedia.org/wiki/Help:Contents

[28] Chakrabarti, S.
Mining the web
Morgan Kaufmann Publishers, 2003.

[29] Hastie T., Tibshirani R. and Friedman, J.
The Elements of Statistical Learning: Data Mining, Inference, and Prediction
Springer-Verlag, 1st ed. 2001. Corr. 3rd printing, 2003, XVI, 533 p.

[30] Burges, C.J.C.
A tutorial on support vector machines for pattern recognition.
Data Mining and Knowledge Discovery, 2(2):955-974, 1998. `http://citeseer.ist.psu.edu/burges98tutorial.html`

[31] Robertson, S. E. and K. Sparck Jones. **Relevance Weighting Of Search Terms** Journal of the American Society for Information Science, vol. 27, 1977. `http://www.soi.city.ac.uk/~ser/pubs.html`

[32] W3C (http://www.w3.org/) **RDF Primer** Online document. `http://www.w3.org/TR/2004/REC-rdf-primer-20040210`

[33] OIL **Description of OIL** Online document. `http://www.ontoknowledge.org/oil/index.shtml`