Measuring the Similarity of Concept Hierarchies and its Influence on the Evaluation of Learning Procedures

Diploma Thesis

Submitted by
Klaas Dellschaft
(klaasd@uni-koblenz.de)

UNIVERSITÄT KOBLENZ · LANDAU

Institute for Computer Science
Working Group ISWeb

Prof. Dr. Steffen Staab

December 2005

Hiermit erkläre ich, dass ich diese Diplomarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe.

Koblenz, den 21. Dezember 2005

Table of Contents

1 Introduction

1.1 Motivation

The information available in corporate intranets and in the Internet grows from day to day. Looking for a specific information often the question is how to find it. Therefore it is the aim of researchers to allow a more efficient access to large collections of information. Many of the developed algorithms are dependent on additional domain knowledge for improving the achieved results (see (Gonzalo et al., 1998) and (De Buenaga Rodríguez et al., 2000)).

The domain knowledge is often available in the form of ontologies. An ontology reflects the understanding of a domain, on which a community has agreed upon. An ontology consists of different parts like a set of concepts and their mutual relations. These concepts are organized in a hierarchy of sub- and superconcepts.

In order to actually improve the results of an application with the help of an ontology, it is crucial to accurately and exhaustively model the domain in question. Because this is a very complex and time consuming task it is a goal to extract an ontology at least semi-automatically. Such learning procedures use documents from the domain for extracting the necessary information. Often these documents are natural language texts like websites or dictionaries which contain domain knowledge (see (Kietz, Maedche and Volz, 2000) and (Cimiano, Hotho and Staab, 2004)).

The quality of an automatically learned ontology is basically influenced by two parameters: The actual learning procedure and the document corpus. There exist several alternative learning procedures. They are further differentiated by the types of documents which they can process, i.e. whether they can process unstructured, semi-structured or structured documents. Websites are an example for unstructured documents, while dictionary entries and encyclopedia articles are examples for semi-structured documents. Documents containing artificial languages like database schemes are finally classified as structured documents. It is often assumed that the availability of structural information leads to a better quality of the extracted ontology.

In order to enable a comparison of the different learning procedures, so that one can choose the best procedure for a certain purpose, they are often evaluated on an example corpus of documents. Subsequently it is tried to objectively measure the quality of the extracted ontology. Such an evaluation may also be used for fine tuning the parameters of a learning procedure, so that better results are achieved. One way of objectively evaluating a learning procedure is to measure the similarity between the learned ontology and a previously defined reference ontology. This similarity is then an equivalent for the quality. It is assumed that the learning procedure will always produce results with a comparable quality. This quality will only be influenced by the document corpus which must contain the correct informations.

1.2 Goals

The comparison of different learning procedures is often complicated because they are often evaluated with different measures. Therefore not only the learning procedures but also the applied measures have to be compared. The first goal of this diploma thesis is to provide a framework of common building blocks, which may be used for making different measures for the evaluation of concept hierarchies comparable with each other. This framework will be deduced from analyzing already existing evaluation measures like the taxonomic overlap described in (Maedche, 2002) or the precision and recall known from information retrieval. In a next step, this framework will be used for defining new measures and comparing the characteristics of all measures with each other. This results in a recommendation of the best measures.

Finally the recommended measures have to prove their usefulness in an actual comparison of learning procedures. Therefore three different learning procedures will be implemented and compared: The Hearst patterns described in (Hearst, 1992), the Head Finding procedure described in (Chodrow and Byrd, 1985) and finally the ArcRank algorithm described in (Jannink and Wiederhold, 1999). They will be applied on different corpora of unstructured and semi-structured documents. The recommended measures will then be used for optimizing those learning procedures and for providing a rank order. It will especially be interesting whether the measures lead to other optimization decisions and another rank order of the learning procedures. It is also expected that this evaluation will provide indices in how far the additional structural information in semi-structured documents may be useful for improving the results of learning procedures.

1.3 Structure of this Document

This diploma thesis starts in chapter 2 with giving a general overview of ontology engineering techniques. Especially the notion of an ontology and its parts will be given in section 2.1. It is followed by an overview of the general procedure for learning an ontology. Finally, TextToOnto will be presented as an example for an ontology engineering tool. It will also be used as the framework for the implementations done for this diploma thesis.

Chapter 3 starts with an overview of different approaches for evaluating an ontology, which may be divided into application neutral and application centered approaches. In section 3.1 the measures precision and recall will be introduced. These measures are known from information retrieval, where they are used for comparing a automatical retrieval with a reference retrieval. Subsequently, this idea of precision and recall will be adapted to the comparison of lexicons (section 3.2) and concept hierarchies (section 3.3). This last section of chapter 3 also contains the framework of building blocks, which will be used for making different evaluation measures comparable with each other. The framework is followed by definitions of new measures and the integration of already existing measures into the framework. These measures are then compared with each other, which will lead to a recommendation of three measures in section 3.4.

Chapter 4 explains the differerent types of documents which may be used as input for learning procedures. Section 4.1 starts with the characteristics of unstructured documents (e.g. websites or PDF documents), while section 4.2 concentrates on semi-structured documents, especially dictionaries and encyclopedias. Finally, section 4.3 contains a description of the document corpora, which will be used in the evaluation of learning procedures.

These learning procedures are then described in more detail in chapter 5. The actual description of the Hearst patterns, the Head Finding procedure and the ArcRank algorithm in section 5.1 is followed by a detailed evaluation of the achieved results in section 5.2. For this evaluation the recommended measures from chapter 3 will be used.

This diploma thesis is concluded with chapter 6. It contains a final evaluation of the recommended measures and their usefulness for assessing learned concept hierarchies. Also an outlook on further work will be given.

2 Ontology Engineering

This chapter gives an overview of ontology engineering techniques. Section 2.1 starts with defining the notion of an ontology. An ontology consists of different parts, namely concepts and taxonomic and non-taxonomic relations. Closely associated to an ontology are the lexicon and the knowledge base. Then in section 2.2 an overview of the general procedure for learning an ontology is given. This chapter is concluded by an introduction to TextToOnto, which is an example for an ontology engineering tool.

2.1 Ontologies

Originally the term ”Ontology” is the name of a philosophical discipline which originates in early Greece and which was introduced by Aristotle. One of the basic questions is ”What are the fundamental categories of being?”[1]. Such fundamental categories are *object* and *property*. Another question is then whether there might exist fundamental properties which are independent of an object or whether an object is just a collection of properties.

But in Computer Science the term ”ontology” has a different meaning, although it shares some of its terms with the philosophical discipline. In general in Computer Science an ontology is a formalized model of the world or of a specialized domain upon which a community has reached an agreement. The following definition and listing of parts of an ontology is taken from (Maedche, 2002). It represents a common understanding of ontologies and can be easily mapped onto existing ontology representation languages. Given this definition, an ontology O is a 5-tuple $O := (C, \mathcal{R}, \mathcal{H}^C, rel, \mathcal{A}^O)$ whose elements will be defined in the following.

Central to an ontology is the **set of concepts** C. Each concept in this set has a unique identifier and denotes a concept in the real world. Often a concept is equated with a single lexical term like “carriage”. But this would keep out the possible synonymy and homonymy of lexical terms. In the “carriage” example one has to concretize that the concept from the real world is meant, which may be denoted by “carriage”, “equipage” and “rig”. So the only requirement for a concept is its unique identifier, the connection with lexical terms is then done via the lexicon (see subsection 2.1.1).

Those concepts are then laid out in a **concept hierarchy** or **taxonomy** denoted by $\mathcal{H}^C \subseteq C \times C$, which is a set of directed taxonomic relations between two concepts. If c_1 is a subconcept of c_2, then it can be either expressed as $\mathcal{H}^C(c_1, c_2)$ or as $c_1 < c_2$. The taxonomy defines a strict partial order on the set of concepts, which means that it is irreflexive and transitive. Additionally it is defined that this concept hierarchy must have exactly one root element, which is also part of C.

1 Cf. the Wikipedia article about Ontology: http://en.wikipedia.org/wiki/Ontology (accessed on September 19th, 2005)

Besides the taxonomic relations, which form the concept hierarchy, there also exists a set of **non-taxonomic relations** between concepts. Each of those relations is denoted by one of the identifiers in $\mathcal{R}$. The identifiers are mapped to the actual relations with the help of the function $rel: \mathcal{R} \rightarrow C \times C$. Instead of $rel(R) = (c_{1,} c_2)$ one can also write $R(c_{1,} c_2)$, where c_1 is called domain and c_2 range of the relation R.

The last important part of an ontology are the **axioms** $\mathcal{A}^O$, which express further constraints on the concepts and relations. They are formulated in an appropriate logical language and they may be used for deducting new information from an ontology (cf. (Shamsfard and Barforoush, 2003)).

While the taxonomic relations in $\mathcal{H}^C$ correspond to the hypnoymy or isA-relation between two concepts, there exist different types of non-taxonomic relations. Often the meronymy or rather the partOf-relation is included in the set of non-taxonomic relations. But this involves certain difficulties, because different senses of the partOf-relation exist (see (Iris, Litowitz and Evens, 1988)). The first sense is being a functional component of a whole. An example for such a functional component is the hole in a colander. Without the holes, the colander would become a bowl instead. The second sense is being a segment of a whole. This can be demonstrated by a cake. If it is divided into pieces, each piece will still be a cake as well as the remaining cake. Further senses of the partOf-relation are the relation between a collection and its members and the relation between sets and subsets.

Those different senses differ in the property of transitivity. This is best made clear with the following two examples of a partOf-relation: “The sleeve has a cuff; the coat has a sleeve; the coat has a cuff” and “the door has a handle; the house has a door; the house has a handle” (cf. (Lyons, 1977), Vol. I: p. 313). While in the former example the partOf-relation between sleeve, coat and cuff is transitive, in the latter example the partOf-relation between house, door and handle isn't. So it has to be clear which of the senses are meant, because they differ in the property of transitiveness. This has an impact on the possible deductions which can be drawn with the help of an ontology, especially because logicians often mention the transitivity as one of the fundamental axioms of mereology. So the meronymy must not be equated with the mereology.

2.1.1 Lexicon

Often a lexicon is associated with an ontology, especially if the ontology is used in the area of natural language processing. According to (Maedche, 2002), the lexicon is a 4-tuple $\mathcal{L} := \{\mathcal{L}^C, \mathcal{L}^\mathcal{R}, \mathcal{F}, \mathcal{G}\}$. It consists of the sets lexical entries $\mathcal{L}^C$ and $\mathcal{L}^\mathcal{R}$, which are used for referencing concepts and non-taxonomic relations in an ontology. The relations $\mathcal{F} \subseteq \mathcal{L}^C \times C$ and $\mathcal{G} \subseteq \mathcal{L}^R \times \mathcal{R}$ provide the corresponding mapping between lexical entries and their associated concepts and non-taxonomic relations. The inverse relations $\mathcal{F}^{-1}$ and $\mathcal{G}^{-1}$ provide a mapping back from the ontology to the lexicon.

It is possible that multiple lexical entries are associated with a single concept in the ontology. Just as well it is possible that one lexical entry is associated with multiple concepts. This multiple to multiple relation between lexical entries and concepts is the general case, which reflects synonyms and homonyms in natural languages. But in practice it may also be assumed that the relation between lexical entries and concepts is a bijection, i.e. each lexical entry is associated with exactly one concept and vice versa. In this case, the possible existence of synonyms and homonyms is ignored.

2.1.2 Knowledge Base

While an ontology contains concepts and their relations, the knowledge base contains instances of these concepts and relations. So the ontology is quite stable over time, while the knowledge base will be regularly changed. For example, if an ontology contains the concept ”city”, the associated knowledge base might contain the knowledge that ”Berlin” and ”Koblenz” are instances of this concept. According to (Maedche, 2002), a knowledge base for an ontology O is a 4-tuple $\mathcal{KB} := \{O, \mathcal{I}, inst, instr\}$, consisting of the set of instances $\mathcal{I}$ and the functions $inst$ and $instr$, which provide a mapping between concepts or relations and their associated instances. Similarly to the ontology, also here a lexicon is associated with the knowledge base.

Although ontology, knowledge base and their associated lexicons are theoretically separated constructs, they aren't necessarily separated in practice. For example, if the relations between entries in the lexicon and entries in the ontology are bijections, then it is possible to merge the lexicon with the ontology. But those constructs found in the practice are only special cases of the general notions of ontology, lexicon and knowledge base, as they were given in the previous subsections.

2.2 Ontology Learning

The construction of an ontology and its associated parts like knowledge base and lexicon is a costly procedure. Therefore it is often striven for at least a semi-automatic construction of ontologies. This approach tries to combine the advantages of manual construction with its high precision and the automatic construction with its often higher recall.[2] For example, an ontology engineer might use automatically learned parts of an ontology for replenishing its own ontology or an automatically learned ontology might be pruned by an ontology engineer to the parts relevant for the final ontology.

This section starts with an overview of sources of prior knowledge, which might be transformed to a first version of the ontology with little effort. Then in subsection 2.2.2 different input sources are presented, from which learning proce-

2 For an introduction to precision and recall see subsection 3.1

dures try to extract relevant informations for the construction of an ontology. Often those input sources have to be preprocessed in order to find the relevant information with different extraction procedures. The preprocessing is described in subsection 2.2.3, while subsection 2.2.4 contains an overview of different types of learning procedures for extracting informations.

2.2.1 Prior Knowledge

Often the construction of a new ontology isn't done completely from scratch. There exists prior knowledge which can be easily used for generating a first version of the ontology. This first version may then be further refined by automatic learning procedures and by manual editing through an ontology engineer. For example, in (Kietz, Maedche and Volz, 2000) it is described how entries from a dictionary of corporate terms were used as good candidates for the set of concepts or the lexicon. But not only domain specific dictionaries may be used for extracting a first version of an ontology, but also general dictionaries like WordNet[3].

The most important criteria for selecting possible sources of prior knowledge is whether they can be easily assessed and whether the expected quality of the extracted information is good enough so that it doesn't need time-consuming manual editing by the ontology engineer. The sources of prior knowledge are dependent on the environment for which the ontology is constructed, so there doesn't exist a final list of such sources. Often the procedures for extracting the prior knowledge have to be manually adapted to the current source. Therefore the effort of adapting the procedures has to be weighed up against the expected quality of the extracted information. Also adaptation effort has be compared with the effort of manually constructing an ontology with the same information.

Such a first version of an ontology, either extracted from prior knowledge or manually constructed, might then be used for improving the results of subsequent learning steps. For example, if the first version of the ontology already contains a set of concepts, then procedures for extracting relations between the concepts might only propose those relations where at least one of the already known concepts participates. This might lead to a higher precision of the learned relations.

2.2.2 Input Data

Automatic procedures for learning parts of an ontology often try to extract information implicitly available in the input data. Examples for such input data might be text documents like the website of a company, forms or the collected correspondence with the customers of a company. Not only automatic procedures need this kind of input data but also the ontology engineer can use these documents for gathering information about the domain for which the ontology should be modeled. In general, there exist three different types of input data:

- The majority of input data is **unstructured**, which means that no or at least very less structural information may be used for improving the results of learning procedures. Typical representatives of this class of input data

3 WordNet will be explained in subsection 4.2.1.

are any text documents like web pages or PDF and Word documents. In section 4.1 the characteristics of unstructured documents are explained in more detail.

- Examples for **semi-structured** input data are dictionaries like WordNet or the Wiktionary, which are explained in section 4.2. Dictionaries are semi-structured because they are divided into entries and subentries, whose meanings are described by short definitional texts or glosses. But while the division into entries and subentries is a structural information, from which possible lexical entries and concepts might be extracted, the definitional texts are unstructured.
- The last category of input data is **structured**. Here only the structural information is used for extracting parts of an ontology. An example would be a database schema which may be used for extracting concepts and their relations. Additionally the content of the corresponding database might be used for extracting the knowledge base of the ontology.

If any structural information is used for extracting parts of an ontology, the quality of the results will normally be better than the results extracted from the unstructured part of the data. Also the information from structured data can be easily extracted by automatic procedures. So normally the structured part of input data can be used as a source of prior knowledge in the ontology engineering process, as it was described in the previous subsection.

2.2.3 Preprocessing

If the unstructured part of documents contains natural language text, then it is often preprocessed. During the preprocessing step, the text is annotated with structural information like part of speech tags. This additional information may then be used by learning procedures, in order to ease the extraction process or to improve the results.

Generally there exist shallow processing techniques and deep understanding techniques for preprocessing a text. An example for a shallow processing technique would be to add part of speech tags simply by looking a word up in large lists. If the currently processed word doesn't exist in one of the lists or if it exists in multiple lists, then additional heuristics might be applied for determining the correct part of speech of the word. In opposite, if a text is annotated with part of speech tags with a deep understanding technique, then the sentence will be parsed with the help of a grammar for the current language. This will help to improve the accuracy of the gathered information, as well as it allows to provide more additional information. For example, a deep understanding of a text might not only provide syntactical information like the part of speech, but also semantical information like disambiguated terms or to which part of the sentence the word belongs.

The information provided by deep understanding will normally be more accurate then the same information provided by shallow processing, besides that it is much more information. But this higher accuracy has its downside on the processing time needed for a document. Therefore in (Ahlswede and Evens, 1988) it was researched whether the higher accuracy and the additional information provided by deep understanding also significantly improves the results of text processing techniques, which are dependent on such preprocessing steps. They came to the conclusion that in their case the results achieved with deep understanding doesn't justify the higher processing time if they are compared to the results achieved with shallow processing. But it is in question, whether the processing time is still so important today. Nevertheless it might be used as an argument, why it is sufficient to use shallow processing techniques, if only a prototype should be created.

2.2.4 Extraction

Subsequently to the preprocessing step, the actual learning procedures are applied on the texts. There exist a huge number of procedures which are specialized on different input data types and on extracting different parts of an ontology. But independently from these differences, they can be classified as applying one of the following learning approaches (cf. (Shamsfard and Barforoush, 2003)):

- **Statistical** approaches try to extract the semantics of words from their distribution over different contexts. For example, it may be assumed that two terms are related to each other if they often cooccur with each other. The cooccurence of words may be checked for different units of informations, like sentences or documents. Examples for statistical learning approaches are the ArcRank algorithm and the formal concept analysis. ArcRank is explained in subsection 5.1.3. For an introduction to the formal concept analysis see (Cimiano, Hotho and Staab, 2004).
- **Logical** approaches can be further divided into inductive and deductive methods. While deductive methods generate new knowledge by applying rules on already existing knowledge, the inductive methods try to induce hypotheses from given examples and then synthesize new knowledge based on further experience.
- **Linguistic** approaches are often used if the input data consists of natural language texts. In a preprocessing step these texts are tagged with linguistic information, like the part of speech. These additional informations are then used for extracting the knowledge.
- **Pattern Based** approaches are a subclass of the linguistic approaches. They search the text for predefined keywords and patterns. The most known example for a pattern based approach will probably be the Hearst patterns which are explained in subsection 5.1.1. A less known example is the Head Finding procedure described in subsection 5.1.2.

Another criterion for distinguishing learning procedures is the amount of needed user interaction. While the automatic learning procedures work without any interaction, the semi-automatic learning procedures try to support an ontology engineer in his task of ontology building. For example, further concepts and relations might be proposed, which have to be confirmed by the engineer. Semi-automatic approaches try to combine high precision of manual ontology engineering and the high recall of automatic approaches.

2.3 TextToOnto

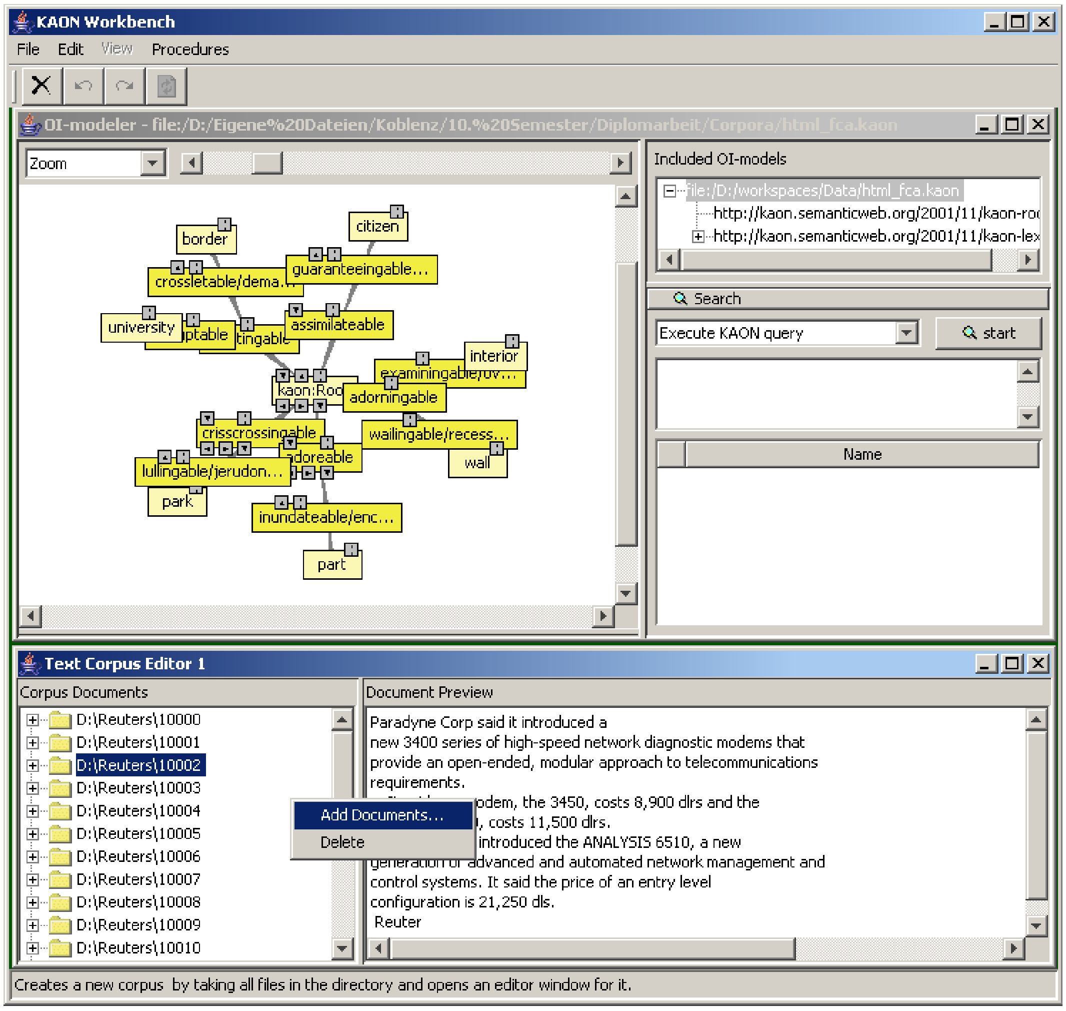
TextToOnto is a Java-based tool for extracting ontologies from natural language texts. It is part of the KAON framework which supports the ontology engineering task in general. The general architectures of TextToOnto and KAON are described in (Maedche and Staab, 2000) and in (Oberle et al., 2004). Both tools are available under an open-source license at http://kaon.semanticweb.org. TextToOnto provides the framework for implementing the learning procedures presented in section 5.1. A list of further ontology learning tools is contained in (Shamsfard and Barforoush, 2003), which includes tools like DOODLE II, SynDiKATe or Web→KB.

In TextToOnto there exist several modules which support the ontology engineer in creating a new ontology or refining an existing one. The most important modules are the corpus editor, which contains a list of natural language texts, and the actual OI model, which represents the current state of the ontology (see figure 1). The documents in the corpus editor may be used by other modules of TextToOnto for extracting information and adding new concepts and relations to the OI model. TextToOnto contains the following specialized modules for the extraction of different parts of an ontology:

- The **Term Extraction** module tries to identify candidate terms in the corpus, which may be used as concepts in the learned ontology. Therefore for each of these candidate terms different measures or their frequency in the corpus are computed. Based on these measures the ontology engineer can decide which of the candidate terms should actually be used as concepts.
- In the **Instance Extraction** module, possible instances for the previously extracted concepts are searched in the text corpus. Different statistical and linguistic procedures may be used. The found instances are then either automatically or semi-automatically integrated into the knowledge base of the ontology. In the semi-automatic mode the ontology engineer has to confirm those instances, where the confidence value computed by TextToOnto is below a certain threshold.
- There exist also a **Relation Learning** and a **Taxo Builder** module. They are used for extracting taxonomic and non-taxonomic relations from the text corpus. The relation learning module may also be operated in a automatic or semi-automatic mode, as it was the case for the instance extraction module. For extracting taxonomic relations the ontology engineer has to choose between different learning procedures like formal concept analysis (FCA), Hearst patterns or heuristics. The Hearst patterns are explained in more detail in section 5.1. For an introduction to FCA see (Cimiano, Hotho and Staab, 2004). It is possible to reduce the set of

extracted taxonomic relations to those, where both participating concepts already exist in the OI model.

- Besides these modules there also exist further modules, which may be used for pruning a given ontology or for computing the precision, recall and taxonomic overlap of two ontologies.



Paradyne Corp said it introduced a
new 3400 series of high-speed network diagnostic modems that
provide an open-ended, modular approach to telecommunications
requirements.
odem, the 3450, costs 8,900 dlrs and the
, costs 11,500 dlrs.
introduced the ANALYSIS 6510, a new
generation of advanced and automated network management and
control systems. It said the price of an entry level
configuration is 21,250 dls.
Reuter

Creates a new corpus by taking all files in the directory and opens an editor window for it.

Figure 1: Corpus Editor and OI Modeler of TextToOnto

TextToOnto will be used as a framework for the implementations in the context of this diploma thesis. Especially the corpus editor and the Taxo Builder module are useful, if it comes to implementing the learning procedures described in section 5.1. The modifications of TextToOnto are explained in more detail in the appendices. The modified sources and binaries of TextToOnto are available on the accompanying CD-ROM.

3 Evaluation Measures

In this section different measures will be described, which evaluate the quality of an ontology. Based on these measures, the different learning procedures may be compared or optimized, so that better results are achieved with the same procedure. As it is said in (Maedche, 2002), there generally exist two different approaches of measuring the quality of an ontology: An application centered and an application neutral approach.

The goal of the application centered approach is to improve an application with the help of ontologies. Therefore results of the application are compared with each other, which were achieved with the help of different ontologies or even no ontology. An example for such an application is the task of categorizing a collection of texts, which may then be evaluated with the precision and recall measure from section 3.1. If the results are improved by using a certain ontology, it may be put down to the quality of the ontology. The application centered approach is dependent on the existence of measures for the evaluation of the achieved results.

This application centered approach has several advantages and disadvantages. An advantage is that one can be sure that the ontology really helps to improve the results of the tested application. So two birds are killed with one stone: The quality of the ontology is measured and it is shown that the tested application can be really improved by the usage of an ontology. But at the same time, this may also be a disadvantage if even the best ontology will only lead to minor improvements or if other application parameters have a stronger influence on the results. Therefore this kind of measuring ontologies has to be carefully set up and one has to remind the possible trap doors when comparing different measurements.

The other way of measuring the quality of an ontology is more application neutral. Therefore ontologies are compared to a reference ontology or gold standard which is normally hand-modeled. The main assumption is that a successfully learned ontology has to approximate the reference ontology. So it has to be ensured that the reference ontology really contains all relevant concepts, relations etc. which should be discovered by successful ontology learning procedures.

A poorly modeled reference ontology may have two effects: On the one hand it will reward ontologies which are also poorly learned and on the other hand it will penalize those ontologies which contain the elements missing in the reference ontology. So the reference ontology has to be carefully chosen, in order to get meaningful results. Then it is possible to compare the reference and the learned ontologies with the help of adapted versions of precision and recall or the taxonomic overlap (see section 3.2 and 3.3). Section 3.4 then contains a recommendation of measures which are expected to be especially useful for the evaluation of concept hierarchies.

3.1 Precision, Recall and F-Measure

This section presents precision, recall and F-measure, as they are known from information retrieval. An information retrieval system tries to identify those documents from a larger set, which would be the answer to a user query. This query may be vague and imprecise because the user doesn't exactly know which documents he searches. Examples for information retrieval systems may be search engines like Google. Information retrieval systems are normally evaluated by submitting queries, whose result document set is compared to a previously defined reference set, which represents the best possible answer. In figure 2 one can see the relationship between the automatic retrieval delivered for the query and the expected reference retrieval. The automatic retrieval is then evaluated with the precision and recall measures. These measures will be adapted to the application neutral evaluation of ontologies in section 3.2 and 3.3.

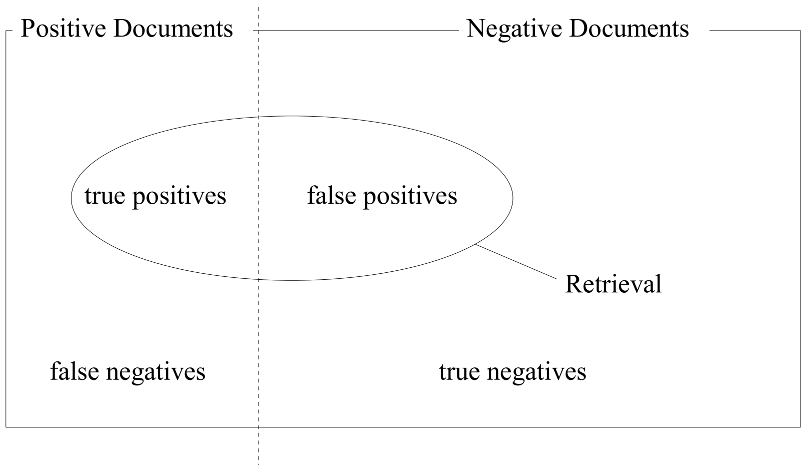


Figure 2: Relationship between reference retrieval and automatic retrieval. Based on (Maedche, 2002)

The precision of a retrieval is the proportion between the true positive documents and the overall retrieval. So if the retrieval contains documents but the set of true positive documents is empty, the precision will be 0. Given the retrieval or computed set of documents (*Comp*) and the reference set of all positive documents (*Ref*), precision is computed with the following formula:

$$P(Ref, Comp) = \frac{|Comp \cap Ref|}{|Comp|}$$

The recall of a retrieval is the proportion between the true positive documents and all positive documents. So if the set of positive documents is not empty and if the retrieval doesn't contain true positive documents, then the recall will be 0. The recall is computed with the following formula:

$$R(Ref, Comp) = \frac{|Comp \cap Ref|}{|Ref|}$$

One of the most interesting properties of precision and recall is that they are the inverse of each other. For example, if one takes a retrieval, which contains no documents in the set of true positive documents, then the precision of the retrieval will be 1 and the recall will be 0. At the same time, if one compares a trivial retrieval, containing all possible documents, with the set of positive documents, then the recall value will be 1 and the precision will be quite low.[4] That precision and recall are the inverse of each other is also shown by the following equation, which is derived from their respective definitions. It is a prerequisite that neither the retrieval nor nor the reference set is empty, because otherwise precision or recall wouldn't be defined.

$$P(Ref, Comp) = \frac{|Comp \cap Ref|}{|Comp|} = R(Comp, Ref)$$

The examples from above with them empty retrieval and the retrieval of all possible documents already demonstrated the need of balancing the precision and recall values against each other. Normally the random addition of further documents to the retrieval will result in an increase of the recall and a decrease of the precision value. The other way round, the random removal of documents from the retrieval will result in an decrease of the recall value while the precision value will normally remain quite the same, unless the retrieval is reduced to the empty set. This goal of getting a good balance between the precision and recall values is reflected by the F-measure. The F-measure is the harmonic mean of precision and recall. The retrieval with the highest F-measure is considered as the best retrieval.

$$F(Ref, Comp) = \frac{2 \cdot P(Ref, Comp) \cdot R(Ref, Comp)}{P(Ref, Comp) + R(Ref, Comp)}$$

3.2 Comparing Lexicons

In this subsection two different measures for comparing the overlap of two lexicons will be presented. Comparing the lexicons instead of the set of concepts is necessary because, as it was said in section 2.1, the concepts of an ontology have a unique identifier which may be completely artificial. So the same concept in different ontologies will normally have different identifiers and it is only possible to identify them as equivalent concepts by comparing their associated lexical entries. So it is important that ontologies preferably have a high overlap on the lexical level, even if the relation between lexical entries and concepts is a bijection. Otherwise comparing other parts of an ontology would be very difficult and influenced by the similarity of the lexicons. In the following, two different approaches will be presented. The first approach is based on the edit distance between two strings and the second on precision and recall.

In (Maedche, 2002), the *String Matching* measure is proposed for computing the similarity of two lexicons. This measure is based on Levenshtein's edit distance, described in (Levenshtein, 1966). The edit distance measures the difference between two strings. Therefore it counts the minimum number of token insertions, deletions and substitutions required to transform one string into another. For example, the edit distances between ”National Park”, ”National_Park” and ”NationalPark” are in each case 1. For the String Matching measure, the edit distance between two strings is set into relation to the lesser length of both strings. For very dissimilar strings the value of the String Matching measure will tend to 0,

4 Normally the positive documents will contain very few documents compared to the set of all possible documents, so that the precision value will tend to 0.

while a perfect match is indicated by the value 1. The following formula describes how to calculate the String Matching measure SM for two lexical entries L_i and L_j . In this formula $ed(L_i, L_j)$ stands for the edit distance between the two strings, while $|L_i|$ is the length of the string.

$$SM(L_i, L_j) := max\left(0, \frac{min(|L_i|, |L_j|) - ed(L_i, L_j)}{min(|L_i|, |L_j|)}\right) \in [0,1]$$

The definition from above may only be used for comparing two strings with each other. If two complete lexicons $\mathcal{L}_1$ and $\mathcal{L}_2$ should be compared, the Averaged String Match $\overline{SM}$ has to be used:

$$\overline{SM}(\mathcal{L}_1, \mathcal{L}_2) := \frac{1}{|\mathcal{L}_1|} \sum_{L_i \in \mathcal{L}_1} \max_{L_j \in \mathcal{L}_2} SM(L_i, L_j)$$

$\overline{SM}$ is an asymmetric measure. For example, $\overline{SM}(\mathcal{L}_1, \mathcal{L}_2)$ will be 1 if $\mathcal{L}_2$ is a superset of $\mathcal{L}_1$. But if $\mathcal{L}_2$ contains plenty more additional entries compared to $\mathcal{L}_1$, then $\overline{SM}(\mathcal{L}_2, \mathcal{L}_1)$ may tend to zero. In (Maedche, 2002) also a generalized version of the String Matching measure is described, which may be used for comparing n-tuples of lexical entries. This generalized version is necessary, if the relation between concepts and lexical entries isn't a bijection and the concepts potentially have more than one associated lexical entry. For more details on this generalized version of the String Matching measure see (Maedche, 2002).

Computing the lexical precision and recall is another approach for measuring the lexical overlap of two ontologies. Therefore the definition of precision and recall from section 3.1 will be adapted to the comparison of two lexicons, which is quite straightforward and it was already suggested in (Maedche, 2002). The lexical precision LP and lexical recall LR are then defined as follows:

$$LP(\mathcal{L}_C, \mathcal{L}_R) := \frac{|\mathcal{L}_R \cap \mathcal{L}_C|}{|\mathcal{L}_C|}$$

$$LR(\mathcal{L}_C, \mathcal{L}_R) := \frac{|\mathcal{L}_R \cap \mathcal{L}_C|}{|\mathcal{L}_R|}$$

While it is certainly desirable to have a possibly high lexical recall value for a learned lexicon, it isn't so clear how to assess the lexical precision value. This uncertainty in regard to the lexical precision is caused by the set up of comparing an automatically learned ontology with a manually modeled reference ontology. Often manually modeled ontologies are restricted in size because their creation is so labor intensive. Additionally also for humans it is difficult to find all terms relevant for a certain domain. So if one optimizes automatic procedures to have a possibly high lexical precision, one also avoids those lexical entries which may be desirable. And even if the lexicon contains many unnecessary and wrong entries, they may be easily removed in a following pruning step. So having a high lexical precision value is less desirable than having a high lexical recall.

3.3 Comparing Concept Hierarchies

Besides the lexicons also the concept hierarchy or the taxonomic relations may be evaluated. Like in the previous subsection, different measures will be presented. Most of them base on the definition of precision and recall from section 3.1. Most of these measures also take the transitivity of the taxonomic relations into account. For distinguishing them from the corresponding lexicon measures, they will be named taxonomic precision TP and taxonomic recall TR. But before presenting concrete measures, some questions have to be clarified first.

First of all, the taxonomy relates concepts with each other. As it was said earlier, those concepts only have a unique identifier which may be completely artificial. Only their associated lexical entries connect them with objects in the real world. In order to keep the following formulas simple, it will be assumed that the relation between concepts and lexical entries is a bijection. This will also not become problematic in the detailed evaluation of learning procedures in section 5.2, because there the relation between concepts and lexical entries is also a bijection. Nevertheless it is no problem to adapt the following measures to the more general case, where a concept is possibly associated with multiple lexical entries and vice versa. This adaptation can be reached by replacing each concept c with its associated lexical entries $\mathcal{F}^{-1}(c)$. An example of such an adapted measure is available in (Maedche, 2002). It contains the generalized version of TO_{sc}, which is described in subsection 3.3.2.

This also directly leads to the next problem: What happens, if a concept or lexical entry isn't available in both ontologies? Then it isn't surprising that also all relations, which involve the missing concept, aren't available in both ontologies. So a poorly learned lexicon may also influence the evaluation results of the learned concept hierarchy. But often it is desirable to separately evaluate these learning tasks. Therefore solutions have to be developed. which allow to uncouple the evaluation results of concept hierarchy and lexicon.

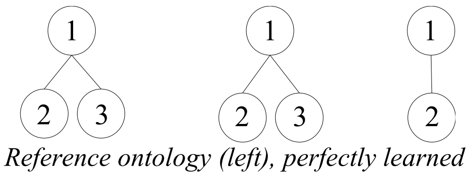


Figure 3: Reference ontology (left), perfectly learned (middle) and partially learned ontology (right)

The example in figure 3 demonstrates the effect of uncoupling the evaluation measures of the lexicon and the taxonomy. For the perfectly learned ontology one surely expects a lexical as well as a taxonomic precision and recall of 100%. But it is the question, what should be expected for the partially learned ontology. There a differentiated statement would be preferable, which says that a lexical entry is missing but that the taxonomy of the remaining ontology is perfectly learned. Even adding a new lexical entry to the learned ontology, which isn't known in the reference ontology, wouldn't change the statement about the taxonomy but only the statement about the lexicons. For this differentiated statement the influence of a low lexical overlap on the taxonomic precision and recall has to be minimized.

3.3.1 Building Blocks

This subsection describes building blocks, which may be used for creating measures evaluating the quality of concept hierarchies. These building blocks are inspired by previous work on taxonomic overlap measures, which is available in (Maedche, 2002) and (Cimiano, Hotho and Staab, 2004). Nevertheless it is started with explaining the building blocks on the example of precision and recall based measures. Subsequently they will also be used in the overlap based measures, although this chronologically is the wrong order. This subsection is finished by a discussion of the commonalities between precision and recall based measures and measures based on an overlap.

Taxonomic Precision

The precision measure, as it was presented in section 3.1, compares a reference set and a retrieved or learned set of objects. It reflects the proportion of objects in the learned set, which are also available in the reference set. In the following it will be distinguished between a local taxonomic precision value $tp(c_{1,} c_{2,} O_C, O_R)$ and the global taxonomic precision value $TP(O_C, O_R)$. The local precision compares the position of the learned concept c_1 in its concept hierarchy with the position of the reference concept c_2. while the global precision compares the complete learned hierarchy of O_C with the reference hierarchy of O_R.

For the local taxonomic precision one has to find characteristic sets of objects, representing the position of a concept in the hierarchy. Then those sets can be compared with a precision based measure. For concepts at similar positions, these sets should contain many common objects while their proportion should decrease with increasing dissimilarity. A possible and obvious solution would be to use an extract from the concept hierarchy, which characterizes the position of a concept. For example, the set of direct subconcepts sub might be used. Given such an extract from the concept hierarchy, the local taxonomic precision tp is defined as follows.

$$tp(c_{1,} c_{2,} O_C, O_R) := \frac{|sub(c_{1,} O_C) \cap sub(c_{2,} O_R)|}{|sub(c_{1,} O_C)|}$$

In this definition of the local taxonomic precision, the extract from the concept hierarchy is an important building block, which significantly influences the explanatory power of tp. Four alternatives for this building block will be explained in the following. The first alternative is the set of direct subconcepts of a concept c from the ontology O:

$$sub(c, O) := \{c_i | c_i \in C \wedge c_i < c \wedge \nexists c_j \in C : c_i < c_j < c\}$$

As one can see in table 1 and 2, this set of direct subconcepts is always empty for leaf concepts. So, according to the definition from above, the local taxonomic precision isn't defined for leaf concepts in the hierarchy of O_C. As we will see below, this leads to problems in the definition of global taxonomic precision. Another problem of using the direct subconcepts at this building block is that the lexical precision and recall of O_C have a strong influence on the local taxonomic precision values. The more concepts aren't contained in both ontologies, the higher the probability that the sets of direct subconcepts contain such concepts. So the average of all local taxonomic precision values will inevitably be lower with decreasing lexical precision and recall values.

It is tried to minimize this influence on the local taxonomic precision value, by only considering the common concepts of both ontologies. The set of direct common subconcepts is defined as follows.

$$csub(c, O_1, O_2) := \left\{ c_i \middle| c_i \in C_1 \cap C_2 \wedge c_i <_1 c \wedge \nexists c_j \in C_1 \cap C_2 : c_i <_1 c_j <_1 c \right\}$$

Starting from the concept c all subconcepts in the hierarchy are traversed until a subconcept is found, which is contained in O_1 as well as in O_2. This subconcept is then added to the set of direct common subconcepts. But although this minimizes the influence of lexical precision and recall, it increases the number of concepts for which this returns an empty set. This can be seen in table 2, where also the set of direct common subconcepts of “5” is empty.

As we've seen for the direct subconcepts as well as for the direct common subconcepts, they have the undesirable property of empty sets at the leaf concepts. The following two alternatives, based on the semantic cotopy, produce empty sets for fewer concepts of a hierarchy. In (Maedche, 2002) and (Cimiano, Hotho and Staab, 2004) the semantic cotopy was used for defining measures, based on an overlap of taxonomies (see below). The semantic cotopy of a concept contains all super- and subconcepts and the concept itself:

$$sc(c, O) := \left\{ c_i \middle| c_i \in C \wedge (c_i \leq c \vee c \leq c_i) \right\}$$

Because the concept itself is included in its semantic cotopy, it will never be an empty set. So if the semantic cotopy is used, the local taxonomic precision is defined for all concepts in O_C.

As it was the case for the set of subconcepts, there also exists a variant of the semantic cotopy, which tries to minimize the influence of lexical precision and recall on the local taxonomic precision. Therefore the semantic cotopy is restricted to the common concepts of two ontologies:

$$csc(c, O_1, O_2) := \{ c_i | c_i \in C_1 \cap C_2 \wedge (c_i <_1 c \vee c <_1 c_i) \}$$

An important thing to note is that this definition of the common semantic cotopy doesn't contain the concept itself, even if it is also contained in both ontologies. This exclusion of the concept from its own common semantic cotopy has to be seen in the context of concrete taxonomic precision measures, which are specialized on ontologies created with clustering techniques like formal concept analysis. Such a measure is explained in (Cimiano, Hotho and Staab, 2004). Unlike the semantic cotopy, the common semantic cotopy can become an empty set. This is the case if none of the super- or subconcepts is contained in both ontologies. The probability of an empty common semantic cotopy increases with decreasing lexical precision and recall values. But often it will be much lower than the probability of empty sets *sub* and *csub* .

Besides these considerations regarding the empty sets, it is also the question, in how far the local taxonomic precision values are influenced by the choice of the characteristic extract. It especially is of interest, what happens, if a concept is replaced in the hierarchy or if a new concept is introduced. This is demonstrated by the example in figure 4. There the concept “2” from the reference ontology is replaced with a new concept. An additional concept is introduced as a subconcept of “5”, which previously was a leaf concept. In table 1 and 2 one can see significant differences in the influence of these changes on *sub* , *csub* , *sc* and *csc* . For example, replacing the concept “2” has a higher impact on the single extracts of *sub* then on *sc* . Simply because the semantic cotopy always contains more concepts than the set of direct subconcept, so that replacing a single concept has a smaller effect. But this is compensated by the fact that this replacement influences more extracts than at the direct subconcepts, because it effects all sub- and superconcepts. In opposite to that, the sets of direct common subconcepts and the common semantic cotopy are not at all influenced by introducing or replacing concepts.

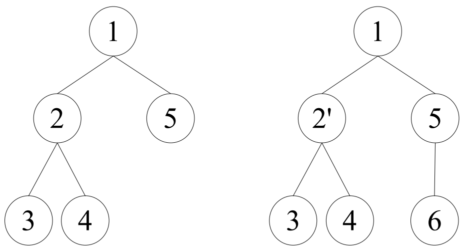


Figure 4: Reference concept hierarchy (O_R , left) and learned concept hierarchy (O_C , right)

$c \in C_R$	$sub(c, O_R)$	$csub(c, O_R, O_C)$	$sc(c, O_R)$	$csc(c, O_R, O_C)$
1	{2, 5}	{3, 4, 5}	{1, 2, 3, 4, 5}	{3, 4, 5}
2	{3, 4}	{3, 4}	{1, 2, 3, 4}	{1, 3, 4}
3	{ }	{ }	{1, 2, 3}	{1}
4	{ }	{ }	{1, 2, 4}	{1}
5	{ }	{ }	{1, 5}	{1}

Table 1: Characteristic extracts of the reference concept hierarchy in figure 4

$c\in C_C$	$sub(c,O_C)$	$csub(c,O_C,O_R)$	$sc(c,O_C)$	$csc(c,O_C,O_R)$
1	{2', 5}	{3, 4, 5}	{1, 2', 3, 4, 5, 6}	{2', 3, 4, 5}
2'	{3, 4}	{3, 4}	{1, 2', 3, 4}	{1, 3, 4}
3	{ }	{ }	{1, 2', 3}	{1}
4	{ }	{ }	{1, 2', 4}	{1}
5	{6}	{ }	{1, 5, 6}	{1}
6	{ }	{ }	{1, 5, 6}	{1, 5}

Table 2: Characteristic extracts of the learned concept hierarchy in figure 4

Besides the previously described possible extracts of concept hierarchies, there also exist further types of extracts. For example, the upwards cotopy, which contains all superconcepts, might be used for characterizing a given concept (cf. (Maedche, 2002)). This has the advantage that only the root concept of a hierarchy would be characterized by an empty extract. The last type of extract would be to use all subconcepts instead of only the direct subconcepts. But here all leaf concepts would still be described by empty sets.

In the following the concept hierarchy extract used at the local taxonomic precision will be annotated as a subscript to tp. For example, if the local taxonomic precision is computed with the help of the common semantic cotopy, this variant will be written as tp_{csc}. Those local taxonomic precision variants may then be used as building blocks in the definition of the global taxonomic precision. The global taxonomic precision compares two whole concept hierarchies with each other and it is the arithmetic mean of all local taxonomic precision values:

$$TP(O_C,O_R):=\frac{1}{|C_C|}\sum_{\underbrace{\boxed{c\in C_C}}_{\text{concept set}}}\begin{cases}\overbrace{tp(c,c,O_C,O_R)}^{\text{local taxonomic precision}} & \mathit{if}\ c\in C_R\\ \underbrace{\max_{c'\in C_R} tp(c,c',O_C,O_R)}_{\text{estimation}} & \mathit{if}\ c\notin C_R\end{cases}$$

Figure 5: Building blocks of the global taxonomic precision

The first building block of the global taxonomic precision is the set of concepts, whose local taxonomic precision values are summed up. The second building block is the local taxonomic precision, as it was previously described. It is used if the current concept is available in both ontologies. The third building block is the estimation of a local taxonomic precision value, if the current concept isn't available in the reference ontology.

Three alternative sets of concepts may be used as the first building block. The first alternative is to use the set of concepts C_C from the learned ontology. If one chooses this alternative, the global taxonomic precision is influenced by the lexical precision. For example, if the lexical precision of a learned ontology is approximately 5%, which is not an unrealistic value for a larger text corpus (see section 5.2), about 95% of the local taxonomic precision values will be an optimistic estimation because there doesn't exist a corresponding concept in the reference ontology. By computing the mean of the common concepts $C_C \cap C_R$ one can avoid this influence of the lexical precision. It makes especially sense if one uses it in combination with a local taxonomic precision measure, where the extract from the concept hierarchy only contains common concepts. As the last alternative for the set of concepts, all common concepts are excluded from the learned concepts. $C_C \setminus C_R$ is mainly used for measures, which are used for evaluating concept hierarchies learned with clustering techniques (see (Cimiano, Hotho and Staab, 2004)).

The next building block is the previously described local taxonomic precision. Here different variants may be used, which are characterized by the used extract from the concept hierarchy. As it was already said, the local taxonomic precision isn't defined for all concepts. For example, if $sub(c, O)$ is used, the local taxonomic precision isn't defined for leaf concepts of the learned hierarchy. In the following precision of 100% will be assumed, if the local taxonomic definition actually is undefined. Another possible solution would be to assume a value of 0%, but then a perfectly learned ontology would never get a perfect global taxonomic precision value of 100%. But in general it should be preferred to use a combination of building blocks, which avoids this problem. This might either be achieved by choosing another extract from the concept hierarchy or by excluding the leaf concepts from computing the average of the local taxonomic precision values. But in the latter case, this would often mean that often over 50% of the concepts of an ontology aren't included.

The last building block of the global taxonomic precision tries to make an estimation for the local taxonomic precision, if the current concept doesn't exist in the reference ontology. Here the first alternative is to search for the concept in the reference ontology, where the comparison with the current concept results in a maximal local taxonomic precision value. The other alternative is to simply assume 0%, if the current concept doesn't exist in the reference ontology. The influence of this building block is dependent on the set of concepts, for which the mean is computed. If C_C is used, the influence depends on the lexical precision of the learned ontology. If $C_C \cap C_R$ is used, the estimation component has no influence. The other extreme appears, if $C_C \setminus C_R$ is used as the concept set. Then the global taxonomic precision is the mean of estimations only. Estimating 0% should be preferred, if a low lexical precision should also decrease the global taxonomic precision. In all other cases, the optimistic estimation should be preferred.

Because the global taxonomic precision is also strongly influenced by the used extract from the concept hierarchy, it will be used as a subscript to TP, as it was the case for the local taxonomic precision.

Taxonomic Recall

Like at the taxonomic precision, there exist a local taxonomic recall and a global taxonomic recall. In parallel to the definition of recall in section 3.1, the local taxonomic recall tr is defined as follows.

$$tr(c_{1,} c_{2,} O_C, O_R) := \frac{|sc(c_{1,} O_C) \cap sc(c_{2,} O_R)|}{|sc(c_{2,} O_R)|}$$

It consists of the same building blocks, representing extracts from the concept hierarchy. The only difference is that the local taxonomic recall isn't defined if the extract from the reference concept hierarchy is empty, instead of from the learned concept hierarchy. The global taxonomic recall is the mean of the local values, but here either C_R, $C_R \cap C_C$ or $C_R \setminus C_C$ is used. Therefore the global taxonomic recall is the inverse of the global taxonomic precision.

$$TR(O_C, O_R) = TP(O_R, O_C)$$

But this equation only holds, if taxonomic precision and recall are computed with the same building blocks.

Taxonomic F- and F'-Measure

Like it is the case for precision and recall in information retrieval, also the taxonomic precision and recall have to be balanced. Therefore the taxonomic F-measure is introduced, which is the harmonic mean of the global taxonomic precision and recall. It is defined as follows.

$$TF(O_C, O_R) := \frac{2 \cdot TP(O_C, O_R) \cdot TR(O_C, O_R)}{TP(O_C, O_R) + TR(O_C, O_R)}$$

A higher taxonomic F-measure corresponds to a better quality of the concept hierarchy. The meaningfulness with regard to the overall quality of the ontology (lexical level + hierarchy) depends on the chosen building blocks. As we've seen, the taxonomic precision and recall is sometimes more and sometimes less influenced by the lexical precision and recall. The higher the influence of the lexical level on the evaluation of the taxonomy, the better is the overall quality reflected by the taxonomic. But if they are not influenced by the lexical level, then the taxonomic F'-measure has to be computed. It is a combination of the taxonomic F-measure and the lexical recall:

$$TF'(O_C, O_R) := \frac{2 \cdot LR(O_C, O_R) \cdot TF(O_C, O_R)}{LR(O_C, O_R) + TF(O_C, O_R)}$$

This measure reaches its maximum for the best learned ontology. The need for such a combined measure is best demonstrated with the example in figure 6. In this example exists a reference hierarchy of concepts O_{ref} and two learned concept hierarchies O_{comp1} and O_{comp2}. Additionally there exists a concept hierarchy, which is the result of merging O_{comp1} and O_{comp2}.

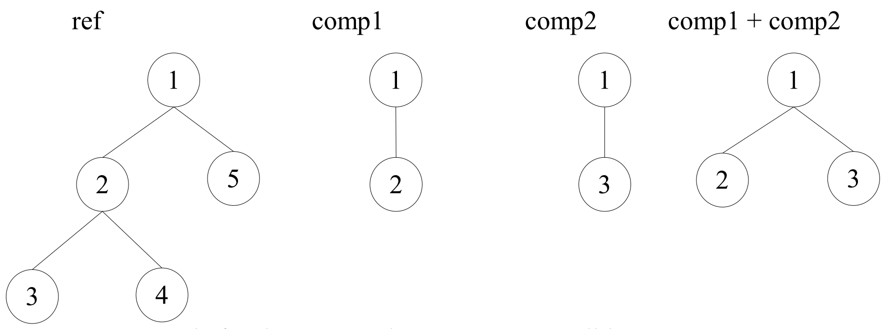


Figure 6: Example for decreasing the taxonomic recall by merging two ontologies compared to the reference ontology (left)

If one evaluates comp1, comp2 and comp1+comp2 with a measure where the extract from the concept hierarchy only contains common concepts (e.g. $csub$ or csc), then the merged ontology will have a lower taxonomic recall then the single ontologies, which also results in a lower taxonomic F-measure. This is counterintuitive because normally one expects to get at least an equal or greater recall value by adding further objects to the learned set. In table 3 one can see, how this result for $O_{comp1} + O_{comp2}$ arises. The results were achieved with TR_{csc}, which will be presented in subsection 3.3.2. It uses the common semantic cotopy and it averages the local values of the common concepts.

	1	2	3	TR_{csc}
O_{comp1}	100,00%	100,00%	–	100,00%
O_{comp2}	100,00%	–	100,00%	100,00%
$O_{comp1}+O_{comp2}$	100,00%	50,00%	50,00%	66,67%

Table 3: Local and global taxonomic recall values for the learned ontologies in figure 6

As one can see in table 4, this counterintuitive behavior of the taxonomic recall is corrected, if one looks at the taxonomic F'-measure. There the combined ontology $O_{comp1}+O_{comp2}$ has a higher TF' value than the single ontologies.

	LR	TF_{csc}	TF'_{csc}
O_{comp1}	40,00%	100,00%	57,14%
O_{comp2}	40,00%	100,00%	57,14%
$O_{comp1}+O_{comp2}$	60,00%	80,00%	68,57%

Table 4: Lexical recall and taxonomic F- and F'-measure for the learned ontologies in figure 6

Taxonomic Overlap

In (Maedche, 2002) and (Cimiano, Hotho and Staab, 2004) another measure for evaluating the quality of concept hierarchies is presented. Between the previously described precision and recall based measures and those measures based on an overlap of taxonomies exist several commonalities. For example, also the taxonomic overlap is computed for either two concepts or two complete ontologies. The local taxonomic overlap of two concepts is defined as follows.

$$to(c_{1,}c_{2,}O_{1,}O_2):=\frac{|sub(c_{1,}O_1)\cap sub(c_{2,}O_2)|}{|sub(c_{1,}O_1)\cup sub(c_{2,}O_2)|}$$

Instead of the direct subconcepts $sub(c,O)$ also other extracts from the concept hierarchy may be used (see above). But in opposite to above, the probability of an undefined local taxonomic overlap value is lower because the divisor is the union of two sets. So for an undefined value both sets must be empty, which means that the concepts are characterized by the same extract from the concept hierarchy. Therefore it would be even more reasonable than above to assume 100% as the local taxonomic overlap value, if both extracts are empty.

In opposite to the local taxonomic precision or recall, the local overlap is a symmetric measure, i.e. the following holds: $to(c_{1,}c_{2,}O_{1,}O_2)=to(c_{2,}c_{1,}O_{2,}O_1)$. This is also the cause, why in the definition of the local taxonomic overlap 1 and 2 are used as subscripts instead of C and R. In the following it will be tried to define the local taxonomic overlap in dependency on either the local taxonomic precision or local taxonomic recall. In order to keep the following formulas readable, a simplified notation will be used. In this simplified notation the local taxonomic overlap $to(c,c',O_C,O_R)$, the local taxonomic precision $tp(c,c',O_C,O_R)$ and the local taxonomic recall $tr(c,c',O_C,O_R)$ will be written as follows:

$$to=\frac{|C\cap R|}{|C\cup R|},\quad tp=\frac{|C\cap R|}{|C|},\quad tr=\frac{|C\cap R|}{|R|}$$

Additionally the local taxonomic F-measure will be introduced as

$$tf=\frac{2\cdot tp\cdot tr}{tp+tr}$$

In these formulas, C stands for the characteristic extract from the learned concept hierarchy and R for the characteristic extract from the reference ontology. By comparing the definitions of to , tp and tr one can see certain commonalities. First it will be tried to define the taxonomic overlap in dependency on the taxonomic recall. Therefore the following equations are used:

$$|C\cup R| \;=\; |R|+|C|-|C\cap R| \;=\; |R|\cdot(1+\frac{|C|}{|R|}-\frac{|C\cap R|}{|R|}) \;=\; |R|\cdot(1+\frac{|C|}{|R|}-tr)$$

$$|C\cap R| \;=\; |R|\cdot tr$$

Given these equations, it is now possible to define the local taxonomic overlap in dependency on the local taxonomic recall:

$$
\begin{aligned}
to &= \frac{|C \cap R|}{|C \cup R|} \\
&= \frac{|R| \cdot tr}{|R| \cdot (1 + \frac{|C|}{|R|} - tr)} = \frac{tr}{1 + \frac{|C|}{|R|} - tr} \\
&= \frac{tr}{tr \cdot (\frac{1}{tr} + \frac{|C|}{|R| \cdot tr} - 1)} = \frac{1}{\frac{1}{tr} + \frac{|C| \cdot |R|}{|R| \cdot |C \cap R|} - 1} = \frac{1}{\frac{1}{tr} + \frac{|C|}{|C \cap R|} - 1} \\
&= \frac{1}{\frac{1}{tr} + \frac{1}{tp} - 1}
\end{aligned}
$$

In the first step, the original divisor and dividend of the overlap are replaced with the previously shown equations. In the next step $|R|$ is canceled. After that the equation in the denominator is factorized, so that tr can be canceled, too. In the following, one of the remaining tr 's is replaced with its definition, which again leads to canceling $|R|$ from the fraction. Finally the inverse of the local taxonomic precision can be inserted instead of an equivalent fraction.

So what started as defining the local taxonomic overlap in dependency on the local taxonomic recall, finally ended with the proof that it is rather dependent on the local taxonomic recall as well as on the local taxonomic precision. Like the F-measure, the local taxonomic overlap is a combined measure of precision and recall. Figure 7 and 8 show the developing of values of the local taxonomic overlap and F-measure for a constant recall value and in dependency on the local taxonomic precision.

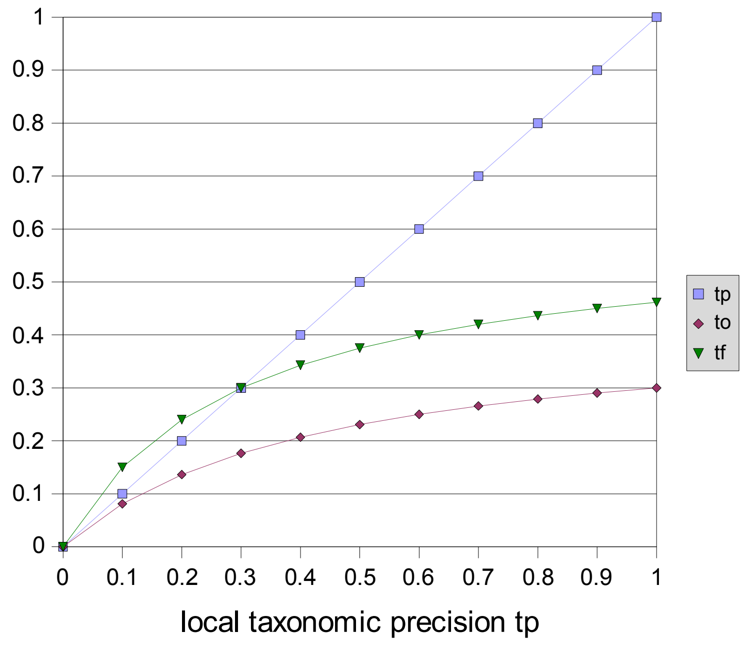


Figure 7: F-measure and overlap for a given recall of 30%, in dependency on precision

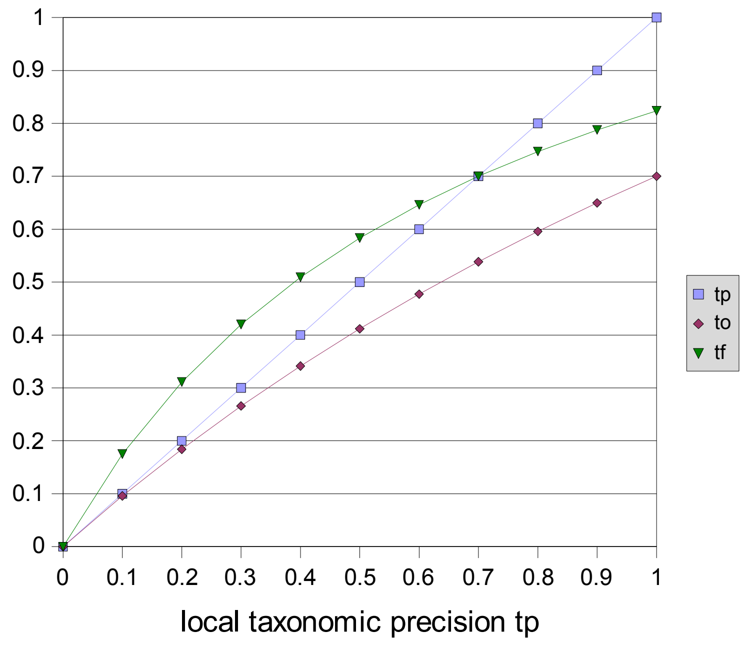


Figure 8: F-measure and overlap for a given recall of 70%, in dependency on precision

It is noticeable that the taxonomic overlap value is always lower than the F-measure. It seems that there exists a functional relationship between the local taxonomic overlap and the local taxonomic F-measure. This functional relationship is shown by the following equation, which defines the overlap value in dependency on the F-measure:

$$
\begin{aligned}
to &= \frac{1}{\frac{1}{tr}+\frac{1}{tp}-1} = \frac{tf}{tf\cdot(\frac{1}{tr}+\frac{1}{tp}-1)} = \frac{tf\cdot(tp+tr)}{2\cdot tp\cdot tr\cdot(\frac{1}{tr}+\frac{1}{tp}-1)} \\
&= \frac{tf\cdot(tp+tr)}{2\cdot tp+2\cdot tr-2\cdot tp\cdot tr} = \frac{tf\cdot(tp+tr)}{(tp+tr)\cdot(2-\frac{2\cdot tp\cdot tr}{tp+tr})} = \frac{tf}{2-\frac{2\cdot tp\cdot tr}{tp+tr}} \\
&= \frac{tf}{2-tf}
\end{aligned}
$$

In the first step, the dividend and the denominator are extended with tf. Then the F-measure in the dividend is replaced with its definition. After that the dividend is multiplied out. The result is then factorized, so that $(tp+tr)$ can be canceled from the fraction. This factorization also leads to another fraction in the dividend, which corresponds to the definition of the F-measure. Therefore the local taxonomic overlap can be defined in dependency on the local taxonomic F-measure. This dependency is shown in figure 9.

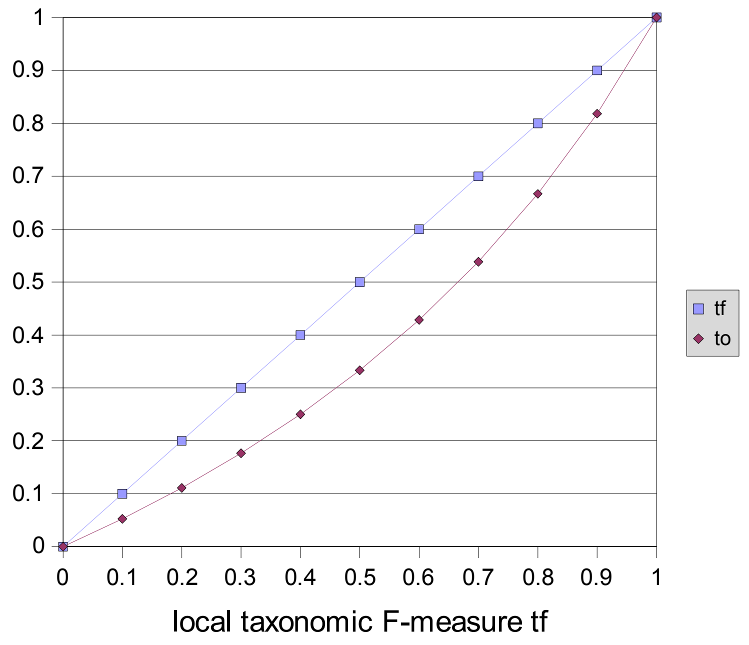


Figure 9: The local taxonomic overlap in dependency on the F-measure

All in all one can say that the local taxonomic overlap fulfills the same purpose as the local taxonomic F-measure. Nevertheless in some cases it would make a difference, whether the overlap or the F-measure is used. The difference is caused by the non-linear gradient of the overlap, if it is defined in dependency on the F-measure. This gradient is greater than the gradient of the F-measure for tf > 50% or to > 33%. It is less than the gradient of the F-measure for tf < 50% or to < 33%. This may be used for a better differentiation, if two concepts are compared with each other. For example, if the F-measures of two learned ontologies are below 50%, then the relative distance between those values will be greater than the relative distance between the corresponding taxonomic overlap values.

The global taxonomic overlap is then defined as the arithmetic mean of local taxonomic overlap values. It contains the same building blocks like the global taxonomic precision and recall.

$$TO(O_{1,}O_2) := \frac{1}{|C_1|} \sum_{c \in C_1} \begin{cases} to(c,c,O_{1,}O_2) & \textit{if } c \in C_2 \\ \max\limits_{c' \in C_2} to(c,c',O_{1,}O_2) & \textit{if } c' \notin C_2 \end{cases}$$

In general also here the same alternatives exist for the different building blocks of the global taxonomic overlap. Unlike the local taxonomic overlap, the global variant isn't necessarily a symmetric measure. This is caused by the set of concepts for which the arithmetic mean is computed. For example, if C_1 is used as the concept set, then $TO(O_C, O_R)$ will compute the average over the learned concepts, while $TO(O_R, O_C)$ computes the average over the reference concepts. The global taxonomic overlap is also asymmetric, if $C_1 \backslash C_2$ is used, but it is symmetric for $C_1 \cap C_2$. The more symmetric the global overlap measure is, the more similarity it will have with the global taxonomic F-measure.

But which conclusions about a concept hierarchy can be drawn from the potential asymmetry of the global taxonomic overlap? First of all one has to distinguish between the different concept sets used for averaging the local overlap values. For example, the global overlap will be symmetric if $C_1 \cap C_2$ is used as the concept set. In this case it is obvious that the global overlap will be dependent on the global taxonomic F-measure.

The other extreme is, if $C_1 \backslash C_2$ is used as the concept set. Then the nature of the global taxonomic overlap is dependent on the distribution of those concepts over the concept hierarchy. In (Cimiano, Hotho and Staab, 2004) it is suggested to use such a global overlap measure for computing a kind of taxonomic precision and recall, with $TP^{ov}(O_C, O_R) = TO(O_C, O_R)$ and $TR^{ov}(O_C, O_R) = TO(O_R, O_C)$.[5] The following considerations will be restricted to the characteristics of $TP^{ov}(O_C, O_R)$ but these considerations are also transferable to $TR^{ov}(O_C, O_R)$.

If $TP^{ov}(O_C, O_R)$ should have the same characteristics as $TP(O_C, O_R)$ then one has to show a strictly monotonic dependency between them.[6] Otherwise it has to be assumed that $TP^{ov}(O_C, O_R)$ still is dependent on other factors. If there exists a strictly monotonic dependency between $TP^{ov}(O_C, O_R)$ and $TP(O_C, O_R)$, the following formula must hold:

$$
\begin{aligned}
&\forall O_1, O_2, O'_1, O'_2 : \\
&((TP(O_1, O_2) = TP(O'_1, O'_2)) \Rightarrow (TP^{ov}(O_1, O_2) = TP^{ov}(O'_1, O'_2)) \wedge \\
&\ (TP(O_1, O_2) > TP(O'_1, O'_2)) \Rightarrow (TP^{ov}(O_1, O_2) > TP^{ov}(O'_1, O'_2)) \wedge \\
&\ (TP(O_1, O_2) < TP(O'_1, O'_2)) \Rightarrow (TP^{ov}(O_1, O_2) < TP^{ov}(O'_1, O'_2)))
\end{aligned}
$$

If one can find a counterexample where this formula doesn't hold then one has proven that $TP^{ov}(O_C, O_R)$ and $TP(O_C, O_R)$ don't exactly have the same characteristics. But for each new taxonomic overlap based measure one has to find a new counterexample where the formula from above doesn't hold. It would be therefore desirable to have a general procedure of how to construct such a counterexample.

5 This suggestion was made for a taxonomic overlap based measure which is designed for evaluating ontologies learned with clustering techniques. There the concepts in $C_C \backslash C_R$ will always be inner concepts of the learned hierarchy, while the concepts in $C_C \cap C_R$ are leaf concepts. Therefore the concepts in $C_C \backslash C_R$ will be quite evenly distributed over the whole hierarchy. But this isn't necessarily the case for $C_R \backslash C_C$.

6 $TP(O_C, O_R)$ uses the same building blocks as $TP^{ov}(O_C, O_R)$. But it is based on the local taxonomic precision instead of the local taxonomic overlap.

With this technique one can also show for measures which use C_C or C_R as the concept sets that there is no strictly monotonic dependency between $TP^{ov}(O_C, O_R)$ and $TP(O_C, O_R)$. This technique is applied below in subsection 3.3.2 at the measure TO_{sc}.

3.3.2 Concrete Measures

In the following concrete measures will be presented which are composed of the previously described building blocks. It starts with two quite simple measures, which use the direct subconcepts and the direct common subconcepts as the characteristic extract from the hierarchy. They are subsequently compared with two measures based on the semantic cotopy and the direct semantic cotopy. After that the taxonomic overlap based measure from (Maedche, 2002) is analyzed. This subsection is finished with a new overlap based measure, which tries to avoid the problems found in the analysis of the previous overlap based measure.

The goal of the first measures TP_{sub} and TR_{sub} is the straightforward adaptation of precision and recall to the evaluation of taxonomic relations. Therefore the concepts of two ontologies are compared on the basis of the direct subconcepts, which is very similar to comparing the tuples in $\mathcal{H}^C$ of both ontologies. The global taxonomic precision is the mean of the local values of all concepts in the learned ontology. If a concept doesn't exist in the reference ontology, then 0% will be assumed as the local taxonomic precision.

$$TP_{sub}(O_C, O_R) := \frac{1}{|C_C|} \sum_{c \in C_C} \begin{cases} tp_{sub}(c, c, O_C, O_R) & if\, c \in C_R \\ 0 & if\, c \notin C_R \end{cases}$$

$$TR_{sub}(O_C, O_R) := TP_{sub}(O_R, O_C)$$

This measure already is influenced by the lexical precision and recall of the learned ontology and therefore it isn't necessary to compute the TF'_{sub} measure. Otherwise it would be strongly influenced by the lexical recall of an ontology, once in the computation of TR_{sub} and the second time in the computation of TF'_{sub} itself.

The next measures TP_{csub} and TR_{csub} try to remove this influence by using alternative but similar building blocks. Here the direct common subconcepts are used as the concept hierarchy extract and the local values of all common concepts are averaged to the global value.

$$TP_{csub}(O_C, O_R) := \frac{1}{|C_C \cap C_R|} \sum_{c \in C_C \cap C_R} tp_{csub}(c, c, O_C, O_R)$$

$$TR_{csub}(O_C, O_R) := TP_{csub}(O_R, O_C)$$

Because the results of TP_{csub} and TR_{csub} are uncoupled from the lexical precision and recall, it is necessary to use the taxonomic F'-measure for evaluating the overall quality of an ontology. In this case TF'_{csub} is only once influenced by the lexical recall. The data in table 5 and 6 demonstrate the different behavior of the previous measures for the ontologies shown in figure 10.

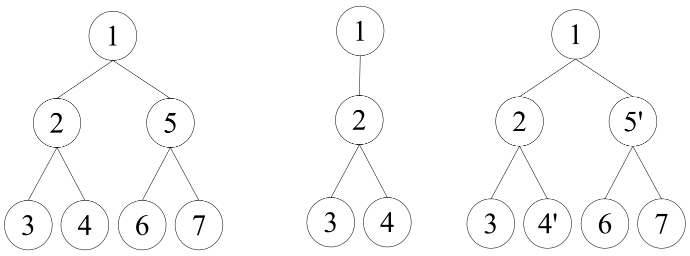


Figure 10: Reference ontology (left), O_{comp1} (middle) and O_{comp2} (right)

	LP	LR	TP_{sub}	TR_{sub}	TF_{sub}	TF'_{sub}
O_{comp1}	100,00%	57,14%	100,00%	50,00%	66,67%	61,54%
O_{comp2}	66,67%	85,71%	55,56%	64,29%	59,60%	70,31%

Table 5: Evaluation of the ontologies in figure 10 with TP_{sub} and TR_{sub}

	LP	LR	TP_{csub}	TR_{csub}	TF_{csub}	TF'_{csub}
O_{comp1}	100,00%	57,14%	100,00%	100,00%	100,00%	72,73%
O_{comp2}	66,67%	85,71%	91,67%	83,33%	87,30%	86,50%

Table 6: Evaluation of the ontologies in figure 10 with TP_{csub} and TR_{csub}

As it could be expected, the absolute values of TP_{sub} and TR_{sub} are lower than the values of TP_{csub} and TR_{csub}. This is caused by the influence of the lexical precision and recall. Only if the lexicon is perfectly learned, both tables would contain the same values, otherwise TP_{sub} and TR_{sub} will always be lower.

It is more interesting that the rank order of the ontologies doesn't change according to the taxonomic F- and F'-measure, although the taxonomic precision and recall values reveal significant differences. For example, the relative decrease of the taxonomic precision values of both ontologies is significantly higher for TP_{sub} than for TP_{csub}. Even more importantly, the rank order of both ontologies changes according to TR_{sub} and TR_{csub}.

Therefore it is very probable that the quality of the ontologies would be judged differently, depending on the chosen measure. Especially if one not only looks at the F and F'-measure but also at the taxonomic precision and recall. Altogether it will be easier to interpret the evaluation results achieved with TP_{csub} and TR_{csub}, because there each of the precision and recall values evaluates only one aspect of the concept hierarchy. Unlike the values of TP_{sub} and TR_{sub}, which are significantly influenced by the lexical precision and recall. In general the former approach of separated measures is preferable, because it allows a more detailed evaluation. If required, it is possible to combine the separate values to a new measure, which transparently weights the different components.

As it was said in subsection 3.3.1, using the direct subconcepts and direct common subconcepts as the characteristic extract from the concept hierarchy has the disadvantage that this extract will always be empty for leaf concepts. Therefore the local taxonomic precision and recall aren't defined for leaf concepts. One solution would be, to exclude the leaf concepts from computing the average. Thus often over 50% of the concepts in an ontology would be excluded from the evaluation. Another approach is, to assume a local taxonomic precision or recall value of 100% for those concepts (see subsection 3.3.1 for a motivation). The former approach leads to very low baselines and the latter to very high baselines of a measure. Measures based on the semantic cotopy avoid this problem. Such measures will be presented in the following.

The first two measures TP_{sc} and TR_{sc} use the semantic cotopy as the characteristic extract. The extract is strongly influenced by the lexical precision and recall, because it doesn't only contain common concepts. In this point the measures are similar to TP_{sub} and TR_{sub}. Also for the other building blocks, the same design decisions were made. TP_{sc} and TR_{sc} are defined as follows.

$$TP_{sc}(O_C, O_R) := \frac{1}{|C_C|} \sum_{c \in C_C} \begin{cases} tp_{sc}(c, c, O_C, O_R) & \text{if } c \in C_R \\ 0 & \text{if } c \notin C_R \end{cases}$$

$$TR_{sc}(O_C, O_R) = TP_{sc}(O_R, O_C)$$

Because of the many commonalities with TP_{sub} and TR_{sub}, also their properties will be comparable. A different behavior of those measures is expectable with an increasing proportion between leaf concepts and inner concepts. In this case the higher baseline of TP_{sub} and TR_{sub} will be perceivable. This is shown by the evaluation results in table 7. Compared with table 5 one can see that the results for O_{comp1} are closer together than for O_{comp2}, because O_{comp2} has a higher proportion of leaf concepts.

	LP	LR	TP_{sc}	TR_{sc}	TF_{sc}	TF'_{sc}
O_{comp1}	100,00%	57,14%	100,00%	51,02%	67,57%	61,92%
O_{comp2}	66,67%	85,71%	41,72%	59,35%	49,00%	62,35%

Table 7: Evaluation of the ontologies in figure 10 with TP_{sc} and TR_{sc}

But there are further causes for the different values. For example, at a semantic cotopy based measure, the influence of an error in the concept hierarchy is increased the closer it is to the root concept, because every concept below this error will contain a penalty through the missing or additional superconcepts in the cotopy. So the average depth of the hierarchy and the average number of direct subconcepts of each concept will indirectly influence the penalty. In opposite to that every error gets the same weight at the direct subconcept based measures.

Figure 11 and 12 contain two larger examples, which will be used for demonstrating this effect.

In O_{comp3}, two concepts on the third hierarchy level and in O_{comp4} two concepts on the second level are interchanged. As one can see in table 8, both ontologies get the same evaluation results, if TP_{sub} and TR_{sub} are used. Therefore both errors have the same influence on the overall result. But TP_{sc} and TR_{sc} evaluate those ontologies differently. As it was predicted, the ontology with an error closer to the root gets a higher penalty. Normally this is a desirable effect, because an error near the root is inherited to the subconcepts via the transitivity of the taxonomic relations.

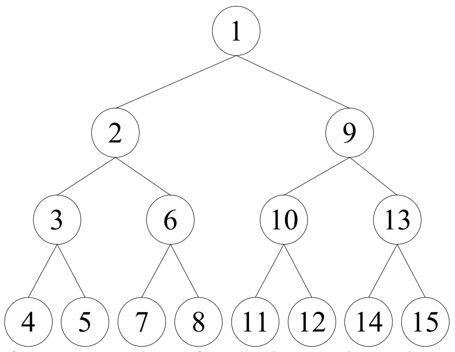


Figure 11: Reference ontology for the learned ontologies in figure 12

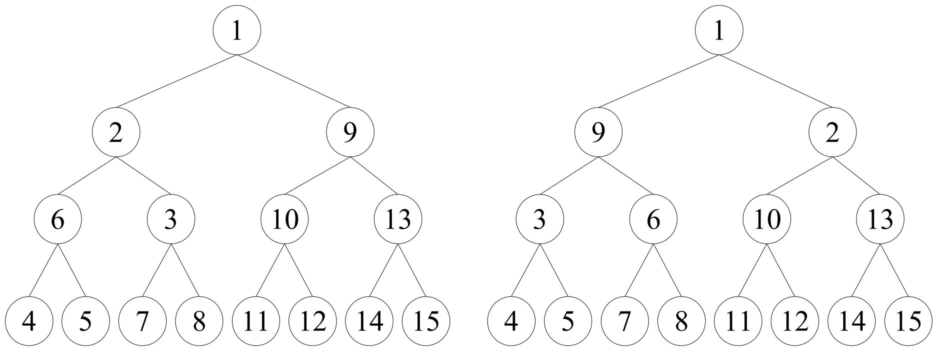


Figure 12: Learned ontology with wrong taxonomic relations near the leafs (O_{comp3}, left) and near the root (O_{comp4}, right)

	LP	LR	TP_{sub}	TR_{sub}	TF_{sub}	TF'_{sub}
O_{comp3}	100,00%	100,00%	86,67%	86,67%	86,67%	92,86%
O_{comp4}	100,00%	100,00%	86,67%	86,67%	86,67%	92,86%

Table 8: Evaluation of the ontologies in figure 12 with TP_{sub} and TR_{sub}

	LP	LR	TP_{sc}	TR_{sc}	TF_{sc}	TF'_{sc}
O_{comp3}	100,00%	100,00%	88,00%	88,00%	88,00%	93,62%
O_{comp4}	100,00%	100,00%	71,33%	71,33%	71,33%	83,27%

Table 9: Evaluation of the ontologies in figure 12 with TP_{sc} and TR_{sc}

The considerations from above are also applicable for the following measures, where the semantic cotopy is restricted to the common concepts. A further difference is that only the local values of the common concepts will be averaged to the global value. Therefore the influence of lexical precision and recall on the evaluation of the taxonomic relations is minimized. TP_{csc} and TR_{csc} are defined as follows.

$$TP_{csc}(O_C, O_R) := \frac{1}{\left|C_C \cap C_R\right|} \sum_{c \in C_C \cap C_R} tp_{csc}(c, c, O_C, O_R)$$

$$TR_{csc}(O_C, O_R) = TP_{csc}(O_R, O_C)$$

The evaluation results in table 10 and 11 confirm the considerations from above. One the one hand, the influence of lexical precision and recall is reduced, which is shown by the improved evaluation results of O_{comp2} compared to the results achieved with TP_{sc} and TR_{sc}. On the other hand, again an error closer to the root has a higher influence on the results, which can be seen by comparing the evaluation results of O_{comp3} and O_{comp4} in table 11 and 12.

	LP	LR	TP_{csc}	TR_{csc}	TF_{csc}	TF'_{csc}
O_{comp1}	100,00%	57,14%	100,00%	100,00%	100,00%	72,73%
O_{comp2}	66,67%	85,71%	100,00%	65,56%	79,19%	82,33%

Table 10: Evaluation of the ontologies in figure 10 with TP_{csc} and TR_{csc}

	LP	LR	TP_{csc}	TR_{csc}	TF_{csc}	TF'_{csc}
O_{comp3}	100,00%	100,00%	84,44%	84,44%	84,44%	91,57%
O_{comp4}	100,00%	100,00%	64,13%	64,13%	64,13%	78,14%

Table 11: Evaluation of the ontologies in figure 12 with TP_{csc} and TR_{csc}

	LP	LR	TP_{csub}	TR_{csub}	TF_{csub}	TF_{csub}'
O_{comp3}	100,00%	100,00%	86,67%	86,67%	86,67%	92,86%
O_{comp4}	100,00%	100,00%	86,67%	86,67%	86,67%	92,86%

Table 12: Evaluation of the ontologies in figure 12 with TP_{csub} and TR_{csub}

The next measure which will be presented in this subsection is taken from (Maedche, 2002). It uses the semantic cotopy as the characteristic extract from the hierarchy in the computation of a taxonomic overlap. The global value is the arithmetic mean of the local overlap values for all concepts in the first ontology. If one of those concepts doesn't exist in the second ontology, then an optimistic estimation about its value is made. Therefore the current concept is compared with all other concepts in the reference ontology and the best overlap value is taken. The global taxonomic overlap TO_{sc} is then defined as follows.

$$TO_{sc}(O_{1,}O_2) := \frac{1}{|C_1|} \sum_{c \in C_1} \begin{cases} to_{sc}(c, c, O_{1,}O_2) & \mathit{if}\ c \in C_1 \\ \max\limits_{c' \in C_2} \ to_{sc}(c, c', O_{1,}O_2) & \mathit{if}\ c \notin C_1 \end{cases}$$

The combination of building blocks used for TO_{sc} leads to a high influence of lexical precision and recall on the measure. Compared to TP_{sc} and TR_{sc}, this influence is decreased by using the optimistic estimation. But this also leads to problems if one wants to interpret the evaluation results, in order to optimize a learning algorithm or to make conclusions about the errors in the hierarchy.

TO_{sc} potentially is an asymmetric measure, which means that it is important whether the learned ontology is used as O_1 or O_2. The asymmetry is dependent on the lexical precision and recall of the learned ontology. If the ontology is perfectly learned with regard to the lexicon, then this measure will be symmetric. As it was said in subsection 3.3.1, this potential asymmetry led to the suggestion in (Cimiano, Hotho and Staab, 2004) that it might be used for computing the global value into both directions. In the following $TO_{sc}(O_C, O_R)$ will be referred as $TP^{ov}_{sc}(O_C, O_R)$ and $TO_{sc}(O_R, O_C)$ will be referred as $TR^{ov}_{sc}(O_C, O_R)$.

Those measures will be compared with TP_{sc_2} and TR_{sc_2}. They use the same building blocks but the former are based on taxonomic overlap and the latter are based on taxonomic precision and recall. This comparison will show the differences between overlap based and precision and recall based measures. TP_{sc_2} and TR_{sc_2} are defined as follows.

$$TP_{sc_2}(O_C, O_R) := \frac{1}{|C_C|} \sum_{c \in C_C} \begin{cases} tp_{sc}(c, c, O_C, O_R) & \mathit{if}\ c \in C_C \\ \max\limits_{c' \in C_R} \ tp_{sc}(c, c', O_C, O_R) & \mathit{if}\ c \notin C_R \end{cases}$$

$$TR_{sc_2}(O_C, O_R) := TP_{sc_2}(O_R, O_C)$$

As it was said in subsection 3.3.1, there has to be a strictly monotonic dependency between TR^{ov}_{sc} and TR_{sc_2} if they should have the same characteristics. This would justify the computation of the taxonomic overlap into both directions. In the following a counterexample will be given where the formula from page 28 doesn't hold for TR^{ov}_{sc} and TR_{sc_2}. The ontologies in figure 13 will be used as this counterexample, whereas table 13 and 14 contain the achieved results.

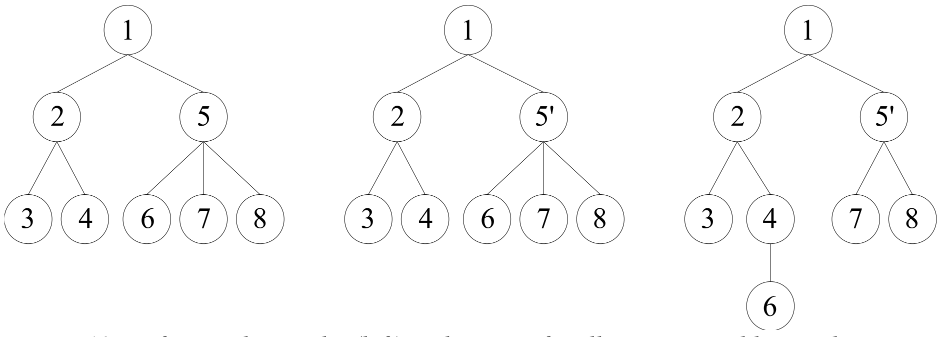


Figure 13: Reference hierarchy (left) and two artificially constructed hierarchies O_{art1} (middle) and O_{art2} (right)

	LP	LR	TP_{sc}^{ov}	TR_{sc}^{ov}	TF_{sc}^{ov}	$TF_{sc}^{ov}{}'$
O_{art1}	87,50%	87,50%	74,31%	74,31%	74,31%	80,36%
O_{art2}	87,50%	87,50%	65,35%	65,35%	65,35%	74,82%

Table 13: Evaluation of the ontologies in figure 13 with TP_{sc}^{ov} and TR_{sc}^{ov}

	LP	LR	TP_{sc}	TR_{sc}	TF_{sc}	$TF_{sc}{}'$
O_{art1}	87,50%	87,50%	83,44%	83,44%	83,44%	85,42%
O_{art2}	87,50%	87,50%	75,10%	83,44%	79,05%	83,06%

Table 14: Evaluation of the ontologies in figure 13 with TP_{sc_2} and TR_{sc_2}

According to the tables above, the following holds:

$$TR_{sc}(O_{art1}, O_{ref}) = TR_{sc}(O_{art2}, O_{ref})$$
$$TR_{sc}^{ov}(O_{art1}, O_{ref}) > TR_{sc}^{ov}(O_{art2}, O_{ref})$$

Therefore we have found a counterexample where the formula from page 28 doesn't hold. Thus TR_{sc}^{ov} and TR_{sc_2} don't have the same characteristics. Because TP_{sc}^{ov} and TR_{sc}^{ov} as well as TP_{sc_2} and TR_{sc_2} are the inverse of each other, the same ontologies might also be used as an counterexample for the strictly monotonic dependency between TP_{sc}^{ov} and TP_{sc_2}.

Nevertheless in the following the evaluation results of the ontologies in figure 10 will be presented. It will be the question, how close the values of TP_{sc}^{ov} and TR_{sc}^{ov} are to a value predicted with TF_{sc_2}. Therefore the functional dependency between the local variants of taxonomic overlap and F-measure will be transferred to the global variants. This functional dependency was discovered in subsection 3.3.1. If the values of TP_{sc}^{ov} and TR_{sc}^{ov} are close to the predicted value, then this would be an indication that also the global, asymmetric variant of the taxonomic overlap stands in a functional dependency to TF_{sc_2}.

Table 15 and 16 contain the different evaluation results of the ontologies in figure 10. For O_{comp1}, the values of TP_{sc}^{ov} and TR_{sc}^{ov} aren't close to the prediction of 64,1%.[7] But the values for O_{comp1} are neither close to the prediction nor close to one another. This is different from O_{comp2}, where the difference between TP_{sc}^{ov} and TR_{sc}^{ov} is very small. At the same time those values are close to the predicted value of 47,35%.[8]

	LP	LR	TP_{sc}^{ov}	TR_{sc}^{ov}	TF_{sc}^{ov}	$TF_{sc}^{ov}{}'$
O_{comp1}	100,00%	57,14%	89,29%	59,12%	71,13%	63,38%
O_{comp2}	66,67%	85,71%	48,33%	47,86%	48,09%	61,62%

Table 15: Evaluation of the ontologies in figure 10 with TP_{sc}^{ov} and TR_{sc}^{ov}

	LP	LR	TP_{sc_2}	TR_{sc_2}	TF_{sc_2}	TF'_{sc_2}
O_{comp1}	100,00%	57,14%	100,00%	64,12%	78,13%	66,01%
O_{comp2}	66,67%	85,71%	60,24%	68,88%	64,27%	73,46%

Table 16: Evaluation of the ontologies in figure 10 with TP_{sc_2} and TR_{sc_2}

It would be interesting to further analyze under which circumstances the difference between the values of TP_{sc}^{ov} and TR_{sc}^{ov} increases and in how far this is related to a deviation from the value which was predicted with the help of TF_{sc_2}.

All these difficulties discovered at TO_{sc} should be avoided by the following overlap based measure. It tries to reduce the influence of lexical precision and recall by using the common semantic cotopy and also avoiding the problems of a uneven distribution of the non-common concepts. The measure TO_{csc} is then defined as follows.

$$TO_{csc}(O_1, O_2) := \frac{1}{|C_1 \cap C_2|} \sum_{c \in C_1 \cap C_2} to_{csc}(c, c, O_1, O_2)$$

This combination of building blocks leads to the symmetry of TO_{csc}. It doesn't matter whether the learned ontology is used as O_1 or O_2.The information provided by this measure should be comparable to the F-measure. Therefore it also makes sense to combine TO_{csc} with the lexical recall value to get a kind of F'-measure, reflecting the overall quality of the learned ontology:

$$TF'^{ov}_{csc}(O_C, O_R) := \frac{2 \cdot LR(O_C, O_R) \cdot TO_{csc}(O_C, O_R)}{LR(O_C, O_R) + TO_{csc}(O_C, O_R)}$$

The evaluation results in table 17 show that the rank order of both ontologies is equal to all other measures except TO_{sc}. In table 18 one can see that also this measure reacts differently on errors near the root and near the leaf concepts of the concept hierarchy. This can also be deduced from using the semantic cotopy.

7 $TF_{sc_2}(O_{comp1}, O_{ref}) = 0{,}7813$; $\quad \frac{0{,}7813}{(2-0{,}7813)} = 0{,}641$

8 $TF_{sc_2}(O_{comp2}, O_{ref}) = 0{,}6427$; $\quad \frac{0{,}6427}{(2-0{,}6427)} = 0{,}4735$

	LP	LR	TO_{csc}	TF'^{ov}_{csc}
O_{comp1}	100,00%	57,14%	100,00%	72,73%
O_{comp2}	66,67%	85,71%	65,56%	74,29%

Table 17: Evaluation of the ontology in figure 10 with TO_{csc}

	LP	LR	TO_{csc}	TF'^{ov}_{csc}
O_{comp3}	100,00%	100,00%	77,78%	87,50%
O_{comp4}	100,00%	100,00%	50,36%	66,98%

Table 18: Evaluation of the ontology in figure 12 with TO_{csc}

3.4 Conclusion

In this section, different measures for evaluating learned concept hierarchies were presented. In a first step, different building blocks of such measures were defined. Normally a measure for comparing two ontologies will consist of four such building blocks. The first building block is a local measure, which compares two concepts with each other. This local measure contains another building block, which characterizes a concept with the help of extracts from the concept hierarchy. The global measure for comparing complete concept hierarchies is then defined as the arithmetic mean of the local values of a certain set of concepts. This set of concepts is the third building block. The last building block defines the behavior of the measure, if one of those concepts isn't available in both ontologies. Then either the value 0 may be assumed or an optimistic estimation is made.

For the combination of those building blocks it had to be considered, in how far the lexical precision and recall of an ontology should also affect the values of the taxonomic precision and recall. Minimizing this influence has the advantage that the different aspects of a concept hierarchy can be separately evaluated. Compared to other measures with a maximized influence of lexical precision and recall allows this a more differentiated view on the learned concept hierarchy. But sometimes it is sufficient to reduce the information about the concept hierarchy to only one or two measures. In this case measures with a maximized influence of the lexical precision and recall may be used. A further reduction of the information is achieved with the usage of a overlap based measure. All in all six different kinds of measures for the evaluation of normal hierarchies were analyzed in subsection 3.3.2. In dependency on the requirements of the evaluation, three of them can be recommended for the evaluation of concept hierarchies:

- The measures $TP_{csc}(O_C, O_R)$ and $TR_{csc}(O_C, O_R)$ should be used if one wants to evaluate the different aspects of the concept hierarchy separately. Because of the clear separation of the aspects, these measures provide the most information about the learned concept hierarchy. Additionally the lexical precision and recall as well as the TF'-measure should be computed, in order to get a complete impression of the learned ontology.

- The measures $TP_{sc}(O_C, O_R)$ and $TR_{sc}(O_C, O_R)$ should be used if one wants to distinguish between the precision and recall of the hierarchy. These measures are strongly influenced by the lexical precision and recall, so that it isn't necessary to separately compute the lexical precision and recall or the TF'-measure.
- Of all recommended measures, the measure $TO_{csc}(O_{1,} O_2)$ provides the least information. It is a symmetric measure, which means that it doesn't matter, whether O_1 or O_2 corresponds to the learned ontology. As it was shown in subsection 3.3.1, this measure has similarities with the an taxonomic F-measure, because it also balances taxonomic precision and recall. At the same time the influence of lexical precision and recall on this measure is minimized. Therefore they should be separately computed in order to get a complete impression of the learned ontology. Finally $TO_{csc}(O_{1,} O_2)$ may be combined with the lexical recall to a kind of TF'-measure.

The not recommended measures also partially provide valuable informations for evaluating ontologies. But because of several shortcomings, which were discovered in subsection 3.3.1 and 3.3.2, it should be preferred to use an equivalent from the recommended measures above. These shortcomings are the following:

- The main shortcoming of the measures $TP_{sub}(O_C, O_R)$ and $TR_{sub}(O_C, O_R)$ as well as of $TP_{csub}(O_C, O_R)$ and $TR_{csub}(O_C, O_R)$ is the problem of the undefined local measure for leaf concepts. In order to get a taxonomic precision and recall value of 100% for a perfectly learned ontology, instead of the undefined local precision and recall values also 100% has to be assumed. This leads to high baselines, if the ontology isn't perfectly learned. Additionally a wrong taxonomic relation near the root is equally penalized as an error near the leaf concepts, which may also seen as a disadvantage.
- The main shortcoming of $TO_{sc}(O_{1,} O_2)$ is its asymmetry because it isn't possible to say, which conclusions can be drawn from this fact. A counterexample was given, which showed that this shouldn't be interpreted as a kind of taxonomic precision and recall. Nevertheless this measure may be useful, if a concept hierarchy with a quite perfectly learned lexicon is evaluated because then the influence of the estimation component is minimized. But as it will be seen in the detailed evaluation in section 5.2, this is seldom the case for automatically learned hierarchies.

4 Document Types

Before explaining the learning procedures for taxonomic relations, it is important to have a look at the different kinds of possible input data first. This is necessary because some of the learning procedures make use of the characteristics of the input data to further improve the results. This section concentrates on input data consisting of natural language text although also other types of input data might be used (e.g. data base schemes; see (Kashyap, 1999) for an example). Section 4.1 starts with the characteristics of unstructured documents (e.g. websites or PDF documents), while section 4.2 concentrates on semi-structured documents, especially dictionaries and encyclopedias. Then in section 4.3, the three corpora are described, which will be used for the evaluation of the learning procedures described in chapter 5.

4.1 Unstructured Documents

Unstructured documents often contain natural language texts. Such unstructured documents, like Word and PDF documents or web sites, have no special characteristics which may be used for easing the extraction of information. Often statistical or linguistic approaches are used for identifying important information from those texts. The Hearst patterns from subsection 5.1.1 or the FCA algorithm described in (Cimiano, Hotho and Staab, 2004) are examples for learning procedures, which may be used for extracting informations from unstructured documents. Unstructured documents are by far the most prevalent sources of information for ontology extraction. For example, if a company wants to create an ontology for performing semantic searches on their archived documents, those documents or the website would be a source for unstructured documents, which cover the domain of the ontology.

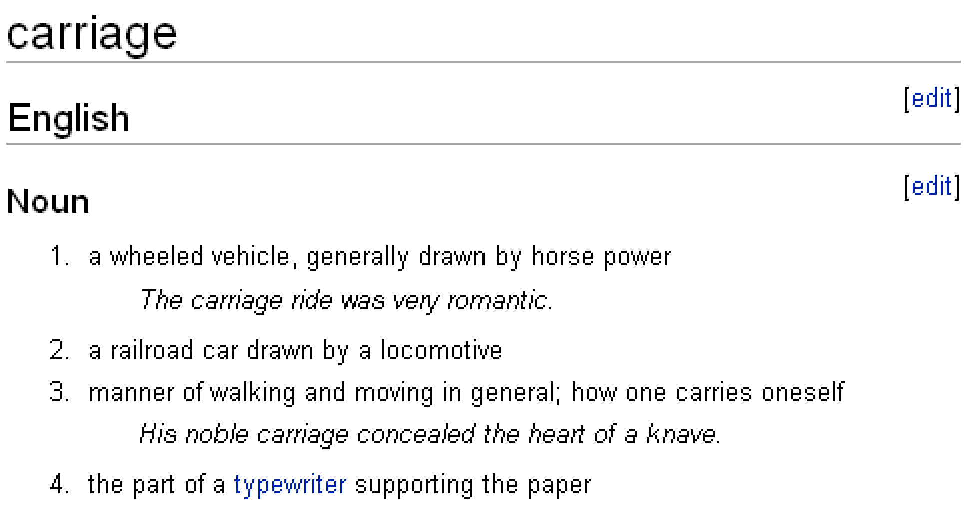
4.2 Semi-Structured Documents

Semi-structured documents contain additional structural information, which may be used for easily extracting first knowledge about the content of the document. Examples for such semi-structured documents are dictionary entries or encyclopedia articles, which are explained in more detail in the following subsections. Their structural part consists of the title of the entry or article, which narrows the possible content of the unstructured part. Another example would be categorization information available in an encyclopedia, e.g. an encyclopedia article about hotels might be categorized as a text about tourism.

Semi-structured documents can also be processed by the learning procedures for unstructured documents. But the structural information of semi-structured documents allows the design of further learning procedures. The learning procedures Head Finding and ArcRank, described in subsection 5.1.2 and 5.1.3, are examples for such specialized procedures, which try to use the structural information for enhancing the results.

4.2.1 Dictionaries

Dictionary entries can be classified as semi-structured documents. Normally a dictionary consists of several entries. The title of an entry is the word, which is defined in the entry. If the defined word has homonyms, the entry consists of several subentries for those homonyms. So the structural information given by the division into subentries may already be used for extracting important informations about a word. Often the homonymy of a word is one of the main challenges in extracting informations from unstructured documents, which is avoided by using the structural information available in dictionaries. In figure 14 one can see an example entry from a dictionary.[9] It contains several subentries for the homonyms of the word "Carriage". Each of those subentries contains a short gloss, which is an unstructured text describing the subentry.


carriage

English [edit]

Noun [edit]

1. a wheeled vehicle, generally drawn by horse power

   The carriage ride was very romantic.
2. a railroad car drawn by a locomotive
3. manner of walking and moving in general; how one carries oneself

   His noble carriage concealed the heart of a knave.
4. the part of a typewriter supporting the paper

Figure 14: Part of the "Carriage"-entry (taken from http://en.wiktionary.org)

Besides extracting information about homonyms, the structure of a dictionary also allows to identify relevant concepts. For example, if one uses a domain specific dictionary, each subentry and each entry may be used as a concept in the generated domain ontology. Additionally the glosses associated with the entries may be used for extracting the concept hierarchy.[10] Exactly this scenario of extracting concepts and their hierarchy from a dictionary was used in (Kietz, Maedche and Volz, 2000) for acquiring domain specific concepts. These domain specific concepts and their taxonomy were then further enriched with data extracted from other input sources.

The quite easy extraction of concept candidates from dictionaries is an enormous advance over their extraction from unstructured texts. There the relevant lexical entries of a domain have to be extracted by certain statistical measures like counting the frequencies of terms in a domain specific text corpus and comparing

9 The entry is taken from the Wiktionary.The full entry is available at http://en.wiktionary.org/wiki/carriage (accessed on July 18th 2005). It stands under the GNU Free Documentation License

10 The extraction of the concept hierarchy may be done by ArcRank or Head Finding. Both procedures are described in section 5.1.

them with frequencies acquired on a general text corpus. This doesn't include the difficult task of detecting and distinguishing homonyms.

But also the extraction of the concept hierarchy may be eased by certain characteristics of the glosses. For example, Merriam Webster New Pocket Dictionary and Merriam Webster Seventh Collegiate Dictionary were analyzed in (Amsler, 1981). One observation was that each textual definition was syntactically a verb or noun phrase with at least one kernel term. Those kernel terms, which correspond to the superconcepts, may then be used for building a hierarchy of the entries or concepts. The kernel term is also called genus term.

Additionally the definition text contains the differentia of a concept. The differentia describe the differences of the current concept to other subconcepts of the genus term (see (Chodrow and Byrd, 1985)). For example, in the first subentry of figure 14 the term ”wheeled vehicle” would be the genus term of ”carriage” and ”drawn by horse power” the differentia. So if one can find the genus term in a definition, it would be possible to build a concept hierarchy. In (Amsler, 1981) this extraction process was done manually, while (Chodrow and Byrd, 1985) proposes a automatic procedure for extracting the concept hierarchy. This automatic procedure bases on the previous observations and is described in more detail in subsection 5.1.2 as the Head Finding procedure.

But other characteristics of dictionaries might complicate the extraction of relations from the glosses. (Chang, Ker and Chen, 1998) observed that all words in the definition texts are drawn from a controlled vocabulary. They analyzed the Longman Dictionary of Contemporary English and counted around 2000 different words which were used in the definition texts. So, if one links a concept with its genus term as its superconcept then the hierarchy will be quite flat and the leaf concepts will normally be linked to another concept which is near to the root.

Using such a small controlled vocabulary is only possible because the authors of a dictionary often use common combinations of words instead of a more specific word. Those common combinations are called covert categories. For example, (Chang, Ker and Chen, 1998) found the pattern ”a person who studies X” (e.g. with ”history” instead of X) instead of the more precise definition ”a scholar on the subject of X”. Here ”scholar” would be the covered category and “person who studies” the combination of words which is used instead. All definitions of concepts which use this covered category would be linked to the more general concept ”person” instead of the more precise concept ”scholar”. So detecting overly general genus terms and replacing them with the covered category would lead to a deeper hierarchy of concepts.

The covert categories may especially be problematic if a specialized dictionary is used for extracting relations. Those specialized dictionaries normally wouldn't contain such general terms as “person”, so the genus term possibly doesn't exist in the dictionary. This is a general problem of specialized dictionaries that they don't contain general terms. A possible solution of this problem is suggested in (Sanfilippo and Poznanski, 1992) where it is tried to combine the data gathered from different dictionaries. So a specialized dictionary might be used for extracting domain specific terms while a general dictionary is used for supplementing the missing general terms.

Many of the papers which analyze general machine readable dictionaries use dictionaries like the Merriam Webster Seventh Collegiate Dictionary[11] or The Longman Dictionary of Contemporary English[12]. Those dictionaries are typical general dictionaries which have all of the characteristics described above. Other representatives are WordNet and the Wiktionary, which differ from typical dictionaries.

WordNet

In WordNet[13] the concepts aren't organized into entries and subentries but into synonym sets. An entry in WordNet lists all synonyms of a concept from the real world, while typical dictionaries list all homonyms for a lexeme. Figure 15 shows all synonym sets which contain the lexeme “carriage”. The organization in synsets already has one big advantage when extracting an ontology, because each synonym set might be used as a single concept which has multiple lexical entries associated. But also the information about homonyms, which is contained in typical dictionaries, is contained in WordNet. The homonyms of a certain lexeme are those synonym sets which contain the lexeme.

Besides the explicit information about the synonyms and homonyms, WordNet also contains the hypernyms, hyponyms, holonyms and meronyms of a lexeme. So it wouldn't be necessary to extract the concept hierarchy from the glosses because this information is explicitly available in WordNet. Altogether WordNet explicitly contains most parts of an ontology and actually many researchers use WordNet not as a dictionary but as an ontology (cf. (Gonzalo et al., 1998), (Moldovan and Mihalcea, 1999), (De Buenaga Rodríguez et al., 2000) and (Nakaya, Kurematsu and Yamaguchi, 2002)).

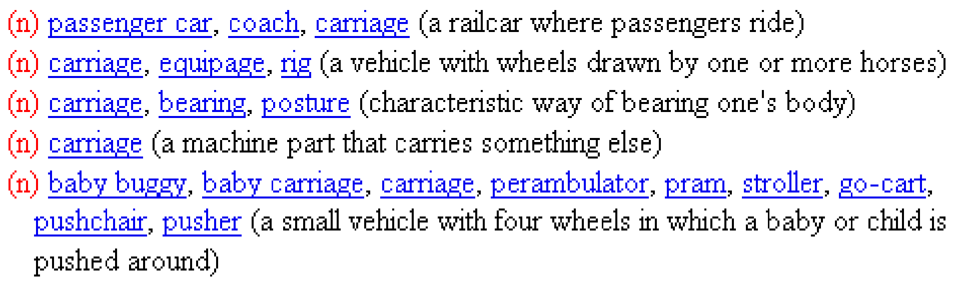

(n) passenger car, coach, carriage (a railcar where passengers ride)
(n) carriage, equipage, rig (a vehicle with wheels drawn by one or more horses)
(n) carriage, bearing, posture (characteristic way of bearing one's body)
(n) carriage (a machine part that carries something else)
(n) baby buggy, baby carriage, carriage, perambulator, pram, stroller, go-cart, pushchair, pusher (a small vehicle with four wheels in which a baby or child is pushed around)

Figure 15: Synonym sets from WordNet 2.1 which contain the lexeme "carriage"

If an ontology is only extracted from those WordNet specific characteristics, and if the unstructured informations in the glosses are ignored, then WordNet is used like a structured document and not like a semi-structured document. Algorithms

11 The Merriam Webster Online Dictionary is available at http://www.m-w.com/. The Webster's Unabridged Dictionary from 1913 is available at the Project Gutenberg http://www.gutenberg.org.
12 The Longman Dictionary of Contemporary English is available at http://www.ldoceonline.com/
13 WordNet is available at http://wordnet.princeton.edu/.

which are dependent on the additional structural information provided by WordNet aren't portable to other dictionaries, which don't have this additional information.

Wiktionary

The Wiktionary is a sister project of the more known enncylopedia Wikipedia. It is a collaborative project to produce a free multi-lingual dictionary. Everybody can participate by editing existing entries and adding new entries. For this purpose the Wiki technology is used. The Wiktionary is available under the GNU Free Documentation License and may be either downloaded as an SQL dump or as XML files, but normally it will be accessed over its web interface available at http://www.wiktionary.org.

Because of the constant further development of the Wiktionary, the quality of the entries is very mixed. Many entries use a common template, so that reliable structural information is available for those entries. Other entries only consist of natural language text, whose size may differ from single sentences to longer passages of text. Also the amount of the provided information is very mixed. For some entries only the gloss is available, while other also contain informations about synonyms, etymology or pronunciation. The minimal structural information available for all entries are the defined lexeme as the title and, if no common template was used, an unstructured natural language text as the definition of the lexeme.

Besides the mixed quality of the entries, which complicates the extraction of information, also certain characteristics of professional dictionaries do not apply. For example, the words in the glosses aren't from a controlled vocabulary, which may also result in less covert categories in the glosses. Therefore an extracted hierarchy will presumably be deeper and/or wider, but at the same time the importance of recognizing synonyms and merging them to a single concept will be increased.

This short overview over the Wiktionary project listed several difficulties, which should be expected when it is actually used for extracting parts of an ontology. But nevertheless it might be of use because of several reasons:

- With over 100,000 entries in November 2005 it contains an increasing number of entries. Therefore its extent will be comparable to those of professional dictionaries in the near future. For example, WordNet 2.1 contains 117,597 synonym sets with 155,327 unique strings.[14]
- It is available for many languages whereas the different versions are often linked with each other. So it enables testing extraction procedures also for other languages then English and the results achieved for different languages can be easily compared.
- It is freely available and everyone has access to it.

14 See http://wordnet.princeton.edu/man/wnstats.7WN.

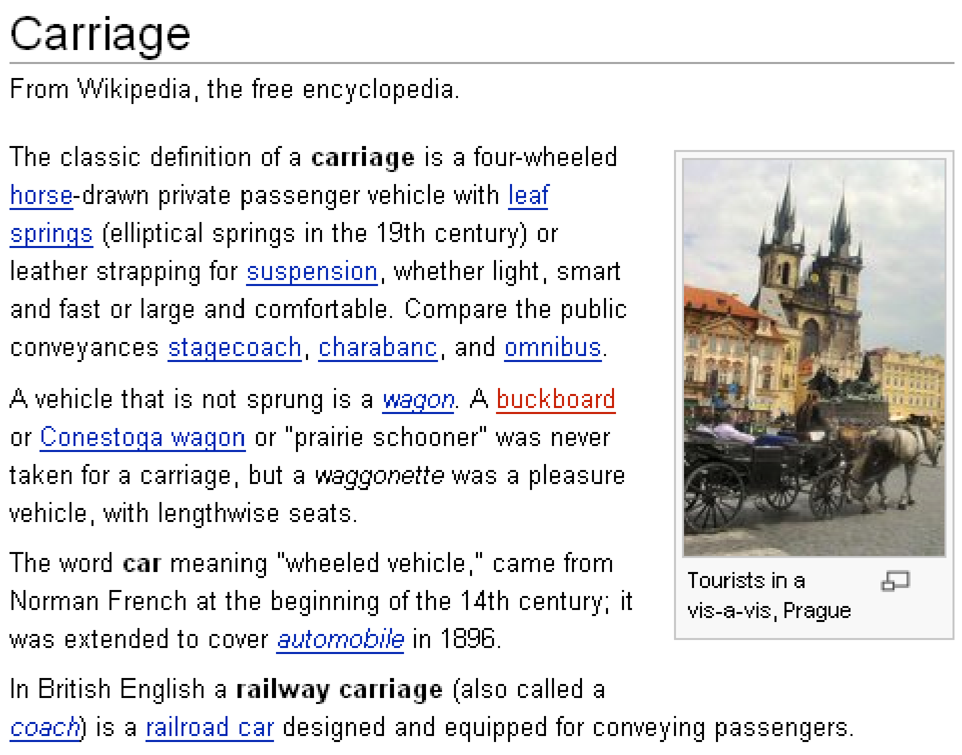
So there are several advantages and disadvantages in using Wiktionary for testing the extraction of parts of an ontology from a dictionary. But the advantages of using Wiktionary can justify further investigation of how to cope with the disadvantages and their real impact on the achievable results. One has to further distinguish between the different language editions of the Wikitonary. For example, the English edition has over 100,000 entries but at the same time those entries often consist of a single gloss, while the German edition only contains 12,000 entries, but most of those entries use a common template with extended informations like synonyms, hypernyms, example sentences or extended conjugation and declination tables.

4.2.2 Encyclopedias

Encyclopedias have very much in common with dictionaries, because they are also divided into entries or articles whereas each article contains a title and an unstructured natural language text. Sometimes the articles are further divided into sub articles which define different meanings of the title. But compared to the glosses in dictionaries, the articles in encyclopedias contain much more information. Additionally an encyclopedia also contains articles about persons, historical events etc., whose definitions aren't available in a dictionary. So while a dictionary mainly contains linguistic informations, an encyclopedia tries to provide informations about the denoted things and facts from the real world.

But if one looks closer at an encyclopedia article and compares it with a corresponding gloss from a dictionary, one can see that often the first sentence of such an article has the same function like the gloss, namely giving a short overview by mentioning the genus term and the differentia of the defined word. For example, figure 16 shows an encyclopedia article explaining the term “carriage”. The first sentence basically says that a carriage is a four-wheeled, horse-drawn passenger vehicle, which is also the content of the first gloss in figure 14. The only difference is that the gloss is a more minimalistic sentence.

Compared to a dictionary gloss, the encyclopedia article contains much more information. For example, the article additionally explains the difference between a carriage and a wagon, it contains a picture of a carriage and in the further article, which isn't shown in figure 16, it explains the historical development of carriages and lists different types, like the Fiacre and the Cabriolet.



Carriage

From Wikipedia, the free encyclopedia.

The classic definition of a **carriage** is a four-wheeled horse-drawn private passenger vehicle with leaf springs (elliptical springs in the 19th century) or leather strapping for suspension, whether light, smart and fast or large and comfortable. Compare the public conveyances stagecoach, charabanc, and omnibus.

A vehicle that is not sprung is a *wagon*. A buckboard or Conestoga wagon or "prairie schooner" was never taken for a carriage, but a *waggonette* was a pleasure vehicle, with lengthwise seats.

The word **car** meaning "wheeled vehicle," came from Norman French at the beginning of the 14th century; it was extended to cover *automobile* in 1896.

In British English a **railway carriage** (also called a *coach*) is a railroad car designed and equipped for conveying passengers.

Tourists in a vis-a-vis, Prague

Figure 16: Part of the "Carriage"-article (taken from http://en.wikipedia.org)

Not only the information provided in a dictionary gloss is minimalistic, but also the sentence structure. While the encyclopedia article contains sentences with many adjectives and subordinate clauses, the gloss normally doesn't contain those things. It is instead reduced to the minimal necessary parts of a sentence and a restricted vocabulary (see subsection 4.2.1). So an encyclopedia article has more commonalities with normal natural language texts, which may complicate their processing, compared to dictionary glosses. But the additional information and the longer definition texts in encyclopedias also have their advantages. For example, typical dictionaries seldom contain the necessary information for extracting meronyms and holonyms and for extracting concept instances. But this information might be available in an encyclopedia article.

Often the learning procedures for extracting informations from any unstructured documents are used at encyclopedias. For example, the Hearst patterns in (Hearst, 1992), were originally applied on an edition of Grolier's Academic American Encyclopedia. But unlike at the dictionaries, there doesn't exist standard encyclopedias, which are often used in the context of ontology extraction.

A further example of a general encyclopedia is the Wikipedia, which contains articles from all domains. The Wikipedia probably is the most known project hosted by the Wikimedia Foundation. The Wiktionary, which was already presented subsection 4.2.1, is another project hosted by the Wikimedia Foundation. That's also the cause why there are many commonalities between both projects.

Wikipedia is an online encyclopedia which is available under the GNU Free Documentation License at http://www.wikipedia.org and exists in different languages. In November 2005 the English edition contained 835,000 articles, followed by the German edition with over 320,000 articles. But also the editions for other languages like French, Dutch, Polish or Swedish contain over 100.000 articles. The Wikipedia is fast growing. For example, 938 articles were daily added to the English edition in March 2005. Also the other editions are fast growing. This growth is possible because everyone can add new articles or edit existing articles. Through the constant review by other users, the quality of the articles usually reaches a high level. Because often a Wikipedia article also links to the corresponding Wiktionary article and vice versa, it might be possible to use them as a combined source.

4.3 Corpora

Three corpora were used for evaluating the relation extraction procedures in section 5.1. The corpora are from the tourism and the finance domain. For the tourism domain, a corpus with unstructured documents and a corpus with semi-structured documents were used. For the finance domain only a corpus with unstructured documents was used. The corpora with unstructured documents are similar to the corpora used in the evaluation parts of (Cimiano, Hotho and Staab, 2004), (Cimiano et al., 2005) and (Hotho, Staab and Stumme, 2003).

Corpora from two different domains were used in order to show that the achieved results of the learning procedures are independent from a domain. Additionally the corpus with unstructured documents from the tourism domain and the corpus with semi-structured documents from the tourism domain were used for analyzing the influence of structured information on the results.

4.3.1 Website Corpus

For the tourism domain, a corpus of unstructured documents was created. It contains texts acquired from the web sites http://www.all-in-all.de and htttp://www.lonelyplanet.com. The first web site contains informations about accommodations and activities in Mecklenburg-Western Pomerania, a federal state in northeast Germany. The second web site contains travel informations for locations around the world. Altogether the corpus contains 12,715 documents with 10.5 million tokens.

A very similar corpus was used in the evaluation parts of (Cimiano, Hotho and Staab, 2004) and (Hotho, Staab and Stumme, 2003). But there the corpus also contained texts from the British National Corpus, a general corpus with over 100 million tokens.

4.3.2 Wikipedia Corpus

Besides the corpus with unstructured documents from the tourism domain, also a corpus with semi-structured documents from the tourism domain was created. It contains articles from Wikipedia, which was described in subsection 4.2.2. Because of the size of Wikipedia and because the corpus should contain texts from the tourism domain, not all articles were used in this evaluation. Therefore the corpus was reduced to 4,596 articles with 6.54 million tokens. As the basis of this reduction to the tourism domain, the Wikipedia snapshot of July 23th 2005 was used.[15]

As a first step for creating the corpus, all concepts from the reference ontology for the tourism domain were extracted. This reference ontology contains 294 concepts, and it was manually created by an experienced ontology engineer within the GETESS project (see (Staab et al., 1999)). In a second step, for each of those concepts a corresponding article was manually searched in Wikipedia. A corresponding article was found for 61% or 178 concepts. Additionally 28 articles were added which are a redirection to one of the other 178 concepts.[16] So, after this second step the corpus contained 206 articles.

After this manual extraction of articles, the set of articles was further extended. Therefore all articles from Wikipedia were added which either reference these 206 core articles or which are referenced by the core articles. This extension step increased the corpus to 4596 articles. It was necessary because of several reasons:

- It helped to avoid the data sparseness problem, e.g. it would be problematic to apply the Hearst patterns on only 206 articles.
- The extended set of articles contains many articles which aren't specific to the domain. For example, after the extension step the tourism corpus also contains articles like “Amplitude Modulation”, “Carbon Dioxide”, “Mathematics” or “Ronald Reagan”. So the Wikipedia corpus also contains noise like it is the case for the unstructured corpora.
- Especially for the ArcRank algorithm, which takes each article as a node and the references as edges in a graph, there is a higher probability of connections between the core articles.

For the finance domain no corpus with Wikipedia articles was created because the reference finance ontology contained too many technical terms with no counterpart in Wikipedia. This would significantly influence the evaluation results so it had no sense to also evaluate a corpus with finance articles from Wikipedia. But it doesn't mean that this procedure of using corpora with semi-structured documents is restricted to certain domains. For example, in (Kietz and Volz, 2000) a dictionary of important corporate terms was used. Equivalently, also for other domains specialized dictionaries with technical terms may be used.

4.3.3 Reuters Corpus

For the finance domain the Reuters-21578[17] news corpus was used. It contains the Reuters news from 1987. This corpus is often used in text categorization tasks,

15 A current snapshot Wikipedia is available at http://download.wikimedia.org. SQL dumps as well as XML files are available.

16 In Wikipedia the redirect mechanism is used for handling synonyms and different spellings, as well as articles whose content is included in other articles. For example, the “Steamer” article redirects to the synonym “Steamboat” and the “Musical” article contains the “Musical Theater” article.

17 http://www.daviddlewis.com/resources/testcollections/reuters21578/

and it contains 3.09 million tokens. It was also used in the evaluation parts of (Cimiano, Hotho and Staab, 2004), (Cimiano et al., 2005) and (Hotho, Staab and Stumme, 2003).

5 Concept Hierarchy Extraction

This chapter deals with the extraction of taxonomic relations from text collections. It starts with an explanation of four different learning procedures in section 5.1. The Hearst Patterns work with any any natural language text, while ArcRank and Head Finding can be applied on semi-structured documents only.

This chapter is concluded by section 5.2, where the results of the learning procedures are evaluated with the measures described in chapter 3. Therefore the learning procedures were applied on the three corpora from section 4.3. Based on this detailed evaluation, it is tried to establish an order of the learning procedures in dependency on the type of corpus (unstructured or semi-structured). Simultaneously this evaluation is used for a further examination of the evaluation measures, whether they rank the learned ontologies differently, and in how far this can be explained with the predicted behavior in chapter 3.

5.1 Learning Procedures

This section presents the learning procedures, which will be implemented and evaluated in the context of this diploma thesis. They are all used for extracting taxonomic relations from unstructured and semi-structured corpora of natural language texts. Some of the subsections contain a short description of how to adapt the procedure to learning non-taxonomic relations, too.

It isn't the goal of this section to give a complete overview of the most important learning procedures or the state-of-the-art. Those who are interested in such an overview are referred to further readings like (Shamsfard and Barforoush, 2003) or (Cimiano et al., 2005). It was instead tried to use a spectrum of procedures which covers the different classes of learning procedures (see subsection 2.2.4). Additionally this list contains procedures applicable on unstructured text corpora as well as procedures applicable on semi-structured text corpora. For example, the Hearst patterns and the Head Finding procedure are representatives of linguistic and/or pattern based approaches, while ArcRank is an example for a statistical learning procedure. From these procedures, the Hearst patterns may be used for any unstructured natural language text, while ArcRank and Head Finding only work with semi-structured documents.

5.1.1 Hearst-Pattern

The Hearst patterns are a pattern based approach for extracting information from natural language texts. Therefore it scans texts for occurrences of certain patterns and then extracts the relevant information. The original procedure is described in (Hearst, 1992). It uses 6 different patterns for the extraction of taxonomic relations. Later they were amended with additional patterns. The following list contains the original patterns suggested by Hearst (pattern 1-6) and 3 additional patterns taken from (Cimiano et al., 2005) (pattern 7-9):

```
(1) HYPERNYM(,)? such as (NP3|NP2|NP1)
(2) NP4 and other HYPERNYM
```

```
(3) NP4 or other HYPERNYM
(4) HYPERNYM, especially (NP3|NP2|NP1)
(5) HYPERNYM, including (NP3|NP2|NP1)
(6) such HYPERNYM as (NP3|NP2|NP1)
(7) HYPERNYM like (NP3|NP2|NP1)
(8) (NP3|NP2|NP1) is HYPERNYM
(9) (NP3|NP2|NP1), another HYPERNYM
```

The uppercase strings in these patterns stand for further sub patterns, which match certain parts of speech in a sentence. Amongst those sub patterns, the *HYPERNYM* pattern has a special role. It matches the noun phrase of a sentence, which contains the hypernym of the taxonomic relation, while *NP1 – NP4* match the related hyponyms. Those sub patterns are abbreviations for more complex patterns matching noun phrases or conjunctions of noun phrases:[18]

```
NOUN = (NN|NNS|NP|NPS)
NP1 = (DET )?(JJ |JJR |JJS )*(NOUN )*NOUN
NP2 = NP1 (and|or) NP1
NP3 = NP1(, NP1)+ (and|or) NP1
NP4 = NP1(, NP1)*
```

Normally the pattern approach delivers quite good results. For example, in (Hearst, 1992) the extraction of relations was tested with two corpora: The *Grolier's American Academic Encyclopedia* containing 8,6M words and a corpus of *New York Times* articles consisting of 20M words. (Hearst, 1992) only contains the results for pattern (1) applied on the Grolier's corpus. These results were evaluated by comparing them with the WordNet hierarchy. The pattern extracted 152 relations, but only at 106 relations the hypernym as well as the hyponym existed in WordNet. From these 106 relations, actually 61 relations were available in the WordNet hierarchy. Although the original paper doesn't mention explicit precision and recall values, this would correspond to a quite high taxonomic precision of approximately 57%. But concluding the taxonomic recall from the data presented in (Hearst, 1992) isn't possible.[19]

The pattern approach may not only be used for extracting taxonomic relations but also for the partOf-relation (see (Ahmad et al., 2003) and (Berland and Charniak, 1999)) or the causeTo-relation (see (Girju and Moldovan, 2002)). This adaptation to other types of relations is done by using other patterns, but often the achieved results do not have the same quality like the results for taxonomic rela-

18 NN = common noun singular, NNS = common noun plural,
NP = proper noun singular, NPS = proper noun plural,
DET = determiner, JJ = adjective, JJR = adjective comparative,
JJS = adjective superlative

19 The low absolute numbers of extracted relations allows the conclusion that the recall value will be quite low. But these low absolute numbers are caused by a restriction in the original paper, where the hypernym as well as the hyponym have to be unmodified by adjectives etc., which reduces the possible matches.

tions. In case of the partOf-relations, this can be explained with properties of the English language which complicate their extraction with a pattern based approach:

In the English language the partOf-relation may be expressed with more words than the isA-relation. According to (Iris, Litowitz and Evens, 1988), the isA-relation is almost always expressed by “is a” or “is a kind of”. Contrary at the partOf-relation: In English it is expressed by more signal words which may also be used in other contexts. Amongst those signal words “part” is the most frequently used word. Identifying the correct signal word becomes even more complicated if the different senses of the partOf-relation should be distinguished (see section 2.1).

5.1.2 Head Finding

The Head Finding procedure, as it is described in (Chodrow and Byrd, 1985), is a specialized procedure which doesn't work with unstructured natural language text but with dictionary or encyclopedia articles only. For example, the original version of this procedure was used for extracting a concept hierarchy from a dictionary. In the following the original procedure will be adapted, so that it also works with encyclopedia articles, like they are found in the Wikipedia corpus from subsection 4.3.2.

Although dictionary entries as well as encyclopedia articles can be categorized as semi-structured documents, there are significant differences. As it was said in section 4.2, dictionary entries normally contain a restricted vocabulary and are very short, while the content of encyclopedia articles is like any other unstructured natural language text. The only difference between an encyclopedia article and unstructured text is that the content of such an encyclopedia article is centered about a certain topic denoted by the title of the article. The adaptation of the Head Finding procedure to the requirements of encyclopedia articles will show commonalities to the Hearst patterns.

As it was said in subsection 4.2.1 about the characteristics of dictionaries, the gloss for each definition has two main parts, the genus term and the differentia. The genus term corresponds to the hypernym of the defined word and the differentia contains the differences to similar words. These characteristics of dictionary articles are used in the original Head Finding procedure described in (Chodrow and Byrd, 1985). It is based on the observation that the genus term of a dictionary article is typically the head word of the defining phrase. So finding the genus term can be reduced to finding the head. In the original procedure, this is done by a heuristic which avoids the full parsing of the definitions but which nevertheless has a very high accuracy. According to (Chodrow and Byrd, 1985), for noun definitions and verb definitions the head word is found with an accuracy of 98 percent and virtually 100 percent, respectively.[20]

The head word is identified in two steps. First a heuristic is used for finding the substring of the definition, which likely contains the head word. This substring

20 See (Chodrow and Byrd, 1985) for more details about the applied heuristics for extracting the head words.

is then further processed. For example, if the substring contains a conjunction, then each conjunct is used as a head word of the definition, resulting in multiple hypernyms. Otherwise, if the substring starts with a so-called empty head (like “one”, “kind”, “class”, etc.) the subsequent words are instead used as the head of the definition.

As it was mentioned above, the original Head Finding procedure was designed for dictionaries entries. They ease the extraction of the head word because of a commonly used pattern in writing such entries. For encyclopedia articles it is more difficult to use the same approach because they are much more like any other unstructured natural language text. For example, complete sentences are used in encyclopedia articles and the definition text is normally longer and more exhaustive.

But by looking at the first sentence of such an article one can see that it fulfills a similar purpose like the gloss of a dictionary entry. Often it tries to summarize the most important characteristics of the defined word. Therefore the first sentence of an encyclopedia article also contains the genus term and the differentia.[21] But often the structure of this first sentence in an encyclopedia article is more complicated than the counterpart from a dictionary.

The result is that the original heuristics used for extracting the genus terms from dictionary entries are not applicable to encyclopedia articles. Therefore a new method for identifying the genus term has to be used. It takes advantage of a common pattern found by looking at a number of Wikipedia articles. In most cases the first sentence starts with repeating the word which is defined in the article. It is followed by an inflection of “be”, potentially one of the empty heads and finally the actual genus terms of the defined word. It may be necessary to slightly adapt this pattern, if other encyclopedias than Wikipedia should be used.

The resulting pattern for extracting the genus term is shown in figure 17.[22] It is very similar to Hearst pattern number (8), described in subsection 5.1.1. The main difference is that the Head Finding pattern is only applied on the first sentence of an article, while the Hearst pattern is always applied on the whole article. Furthermore it is possible that a defined word may have associated multiple genus terms. Because of these commonalities with Hearst patterns, their implementation could be partially reused for implementing the adapted version of the Head Finding procedure (see appendix B).

```
HYPERNYM(,)? (are|is|was|were) (((a |an )?(one|any|kind|class|
manner|family|race|piece|group|complex|type|style)) of )?(NP3|NP2|
NP1)
```

Figure 17: Hearst pattern for finding the genus terms

21 For an example see subsection 4.2.2.

22 Because of the commonalities between the Hearst patterns and the Head Finding pattern, the implementation of the Hearst patterns was partially reused. But this implementation doesn't support multiple noun phrases as the hypernyms. But this is required for the Head Finding pattern. Therefore in the Hearst pattern in figure 17, the genus terms, which actually are the hypernyms, are matched by “(NP3|NP2|NP1)”, while “HYPERNYM” is used for matching the defined word or rather the hyponym.

5.1.3 ArcRank

ArcRank is a general algorithm for analyzing the relationships between nodes in a directed labeled graph. In (Jannink and Wiederhold, 1999) a directed labeled graph was extracted from the 1913 Webster's dictionary and subsequently processed with the help of ArcRank. ArcRank tries to identify those relations between nodes in a graph, which have a high importance. Therefore it computes a rank value for each of the relationships between the nodes. In the case of analyzing a dictionary, these rank values may then be used for extracting the subsuming, specializing and the kinship relation between the terms in the dictionary.

ArcRank is based on the PageRank algorithm, which computes rank values for nodes in a graph (see (Page and Brin, 1998)). PageRank distributes rank values between nodes in a graph in an iterative process. First the nodes in the graph are initialized with the same value. Then in each iteration step the complete rank value of a node is distributed to its neighbors in the graph. Each of them gets the same portion of the distributed rank value. The PageRank algorithm stops, if between two iteration steps the changes of the rank values are below a certain threshold. The final rank of a node is influenced by the number of incoming edges from other nodes and the height of these other nodes' rank values.

The idea of ArcRank is to identify for each node those edges with the highest influence on its final rank value. Therefore each edge in the graph gets the rank value assigned which it would distribute after the PageRank algorithm has reached its final state. If there are several edges between a source node and a target node their distributed rank values are summed up. Then for each node the incoming edges are ordered by the rank values they distribute. But also the outgoing edges are ordered: The more the distributed rank value influences the rank of the target node, the more important is the edge. So an edge from a high ranked node to a low ranked node is more important than an edge from the same node to another high ranked node. Finally the rank value of each edge or arc is computed based on its importance as an outgoing edge at the source and on its importance as an incoming edge at the target node.

Originally, PageRank was used for computing rank values of documents in the Internet. Thereby each document corresponds to a node, while the hyperlinks between these documents correspond to the edges in the graph. Now ArcRank is used for analyzing the structure of a dictionary. The nodes in the graph correspond to entries in the dictionary, while the edges may be extracted in two ways. Either each term in a gloss or, in the case of analyzing an encyclopedia, in an article will be converted into an edge to the describing node. Alternatively only explicit hyperlinks between dictionary entries or encyclopedia articles will be used as edges in the extracted graph.

The idea of converting a dictionary into a directed, labeled graph wasn't new. For example, (Litowski, 1978) already contains a detailed overview of how to extract such a graph. He suggests to use each word in the gloss of a definition as a directed edge to other nodes which contain the definition of the word. Additionally the defined word is used as the label of the node. But this approach of converting each word into an edge is by far more complicated than the other approach where only explicitly set hyperlinks are used as edges.

Both approaches have their advantages and disadvantages. Creating an edge for each term should be preferred for dictionaries where the glosses are short and focused. Otherwise the nodes in the graph would only be linked sparsely. But linking each term requires thorough preprocessing of the glosses. For example, each term has to be stemmed and disambiguated, so that it is linked to the correct node in the graph. Additionally, edges to stop words should be removed from the resulting graph. (Jannink and Wiederhold, 1999) contains a list of several problems which were discovered by using this approach on the 1913 Webster's dictionary.

In contrast the usage of explicit hyperlinks as edges between nodes should be preferred for encyclopedia articles, because they are normally much longer then a dictionary entry. In this case, linking each term would lead to a very high number of edges in the graph which will often point to terms irrelevant for the defined word. Only explicit hyperlinks between articles are converted to edges for the Wikipedia corpus from subsection 4.3.2.

After the successful extraction of the graph from an dictionary or encyclopedia, the actual ArcRank values are computed with the algorithm described above. Then the rank values of the edges are further analyzed. In a first step, the kinship relation between terms is identified. The kinship relation consists between similar words, but it is broader than the synonymy. In (Jannink and Wiederhold, 1999) it is assumed that nodes with similar edge structures are related by kinship. Then, in a second step, it is possible to identify the subconcepts and superconcepts of the kinship related nodes. It is assumed that, depending on the direction of the edge, their common edges point to the sub- and superconcepts.

The implementation of the ArcRank algorithm, which will be evaluated in subsection 5.2.3, uses a simplified approach for identifying the sub- and superconcepts. In this simplified approach it is assumed that the highest ranked outgoing edge connects a node with its superconcept.

5.2 Results

In the following a detailed evaluation of the learning procedures from section 5.1 can be found. The three recommended measures from subsection 3.4 will be used. Especially TP_{csc} and TR_{csc} will be used, because they provide more detailed informations about the learned ontologies then the other two measures. Those evaluation results will then be compared with the other two measures, whether they suggest the same interpretation of the learned ontologies. Finally a rank order of Hearst pattern, Head Finding and ArcRank will be given, together with a discussion of the differences between the evaluation measures and how they should be used.

5.2.1 Hearst Pattern

The detailed evaluation is started with the Hearst patterns described in subsection 5.1.1. Compared to the other learning procedures, by far the most evaluation data was collected for the Hearst patterns, because they could be applied on the Wikipedia corpus, the Website corpus as well as on the Reuters corpus. Additionally the results of the Hearst patterns are influenced by a threshold value, which may be used for restricting the extracted taxonomic relations (see below). Also it is possible to analyze the influence of the different patterns on the final results.

If the Hearst patterns are applied on a collection of texts, it is very likely that the same relation is extracted more than once. This information can be used for defining a confidence value in the extracted relation. The confidence is increased, with the number of occurrences. The most often extracted relation gets a confidence value of 1.0. With descending occurrences, this value drops to 0.0.

pattern	*LP*	*LR*	TP_{csc}	TR_{csc}	TF_{csc}	TF'_{csc}
1 (t=0.0)	1,74%	32,65%	75,83%	49,87%	60,17%	42,33%
2 (t=0.0)	3,53%	19,39%	86,55%	54,51%	66,89%	30,06%
3 (t=0.0)	7,14%	11,90%	88,57%	67,82%	76,82%	20,61%
4 (t=0.0)	6,61%	7,82%	100,00%	48,34%	65,17%	13,97%
5 (t=0.0)	3,40%	23,47%	85,42%	51,21%	64,04%	34,35%
6 (t=0.0)	6,13%	8,50%	89,33%	53,87%	67,21%	15,10%
7 (t=0.0)	3,06%	18,03%	90,57%	46,13%	61,12%	27,84%
8 (t=0.0)	1,59%	41,50%	30,81%	76,41%	43,91%	42,67%
9 (t=0.0)	8,57%	3,06%	100,00%	100,00%	100,00%	5,94%
All (t=0.0)	1,00%	49,66%	22,26%	83,81%	35,18%	41,18%

Table 19: Evaluation results of the different Hearst patterns for the Wikipedia corpus

In table 19 one can see the evaluation results of the different Hearst patterns applied on the Wikipedia corpus. The first column contains the number of the pattern (see subsection 5.1.1) together with the threshold value, for which the highest TF' value was achieved. The last row then contains the results achieved by merging the results of all patterns. The following points are noticeable:

- Judging from the TF' measure, which was said to indicate a kind of overall quality, all patterns achieved their best results with a threshold of 0.0. So no relations were removed from the results.
- Pattern (1), (2), (5), (7) and (8) seem to be the most productive patterns with regard to the lexical recall.
- If one looks at the taxonomic precision, one can see a qualitative difference between pattern (8) and all other patterns. For pattern (8), the taxonomic precision is approximately 30%, while for all other patterns it is significantly higher. For most of them it is above 85%.
- This significant difference at the taxonomic precision is only partially made up by a higher taxonomic recall for pattern (8), which lays at 76%. Most of the other patterns lie between 45% and 55%. So if one balances the taxonomic precision and recall in form of the taxonomic F-measure,

pattern (8) gets the lowest value with 44%, while most of the other patterns lie between 60% and 76%.

All in all, the lexical as well as the taxonomic precision and recall of pattern (8) suggest that it more or less “accidentally” finds correct lexical entries and taxonomic relations. In table 20 one can see the evaluation results of pattern (8) in dependency on the application of a threshold. These results partially relativize the impression that pattern (8) only accidentally finds correct lexical entries and taxonomic relations. Because if one removes all relations from the results, which occurred below average (i.e. with a confidence value < 0.25), the quality of the taxonomic relations is comparable to the other patterns. But this increase of quality on the taxonomic level leads to a major decrease on the lexical recall level.

threshold	*LP*	*LR*	TP_{csc}	TR_{csc}	TF_{csc}	TF'_{csc}
0.0	1,59%	41,50%	30,81%	76,41%	43,91%	42,67%
0.3	10,77%	15,31%	80,00%	60,80%	69,09%	25,06%
0.6	10,53%	4,08%	72,22%	72,73%	72,47%	7,73%
0.9	13,46%	2,38%	71,43%	83,33%	76,92%	4,62%

Table 20: Evaluation results of pattern (8) for the Wikipedia corpus, in dependency on the threshold

In table 21 and 22 one can see the influence of pattern (8) on the overall results. Therefore the evaluation results of merging the relations of all patterns are compared with the merged results of all patterns but pattern (8). All in all in both cases it seems to have a very positive effect on the quality of the taxonomic relations, if relations with a below average occurrence are removed. The inclusion of pattern (8) leads to a good compromise between the quality of the taxonomic relations and the lexical precision and recall.

threshold	*LP*	*LR*	TP_{csc}	TR_{csc}	TF_{csc}	TF'_{csc}
0.0	1,00%	49,66%	22,26%	83,81%	35,18%	41,18%
0.3	7,27%	22,79%	81,01%	59,60%	68,67%	34,22%
0.6	12,09%	11,22%	83,08%	62,11%	71,08%	19,39%
0.9	17,04%	7,82%	84,06%	73,85%	78,62%	14,23%

Table 21: Evaluation results for the merged relations of all patterns, in dependency on the threshold.

threshold	*LP*	*LR*	TP_{csc}	TR_{csc}	TF_{csc}	TF'_{csc}
0.0	1,27%	39,80%	34,88%	76,37%	47,89%	43,47%
0.3	7,98%	13,95%	96,14%	65,94%	78,23%	23,67%
0.6	13,70%	6,80%	91,67%	75,57%	82,84%	12,57%
0.9	22,73%	5,10%	94,44%	86,19%	90,13%	9,66%

Table 22: Evaluation results for merging the relations of all patterns but pattern (8), in dependency on the threshold

If one looks at further statistical values, one gets additional support for the “unhealthiness” of the unrestricted ontology. For example, the first row of table 23 contains the statistical values of the tourism domain reference ontology, against which the learned ontologies were compared. Then the following rows contain the statistical values of the learned ontologies. One can see that the unfiltered concept hierarchy contains 4973 loops (i.e. a concept is also one of its own superconcepts), while a leaf concept has 119 superconcepts in average. Also it is interesting to look at the branching factor of the hierarchy. There one can see that a concept has 3.57 direct subconcepts in average, with a very high deviation of 53.2 from the average. Also the average number of direct superconcepts is quite high with 1.52 and a deviation of 2.2. The statistical values of the ontologies filtered by the threshold are more like the values from the reference ontology

threshold	concepts	loops	avg. depth	avg. sub	sub dev.	avg. super	super dev.
ref	294	1	5.14	5.22	4.42	1.03	0.17
0.0	14569	4973	119.29	3.57	53.2	1.52	2.2
0.3	893	97	3.8	2.81	14.89	1.22	0.87
0.6	246	24	3.29	2.68	8.39	1.16	0.78
0.9	116	2	3.17	2.76	6.06	1.08	0.35

Table 23: Additional statistical values of the reference ontology for the tourism domain and of the ontologies learned with all Hearst patterns from the Wikipedia corpus.

This recommendation of applying a threshold is not only restricted to the Wikipedia corpus. In fact the same characteristics of the Hearst pattern can also be seen for the Website corpus as well as for the Reuters corpus (see table 24 and 25). There the taxonomic precision and recall is also significantly improved by applying the threshold, while the lexical recall drops at the same time. The detailed evaluation results for the Hearst patterns applied on the Website corpus and the Reuters corpus are available on the accompanying CD-ROM.

patterns	*LP*	*LR*	TP_{csc}	TR_{csc}	TF_{csc}	TF'_{csc}
All (t=0.0)	1,59%	42,18%	30,28%	76,85%	43,45%	42,80%
All (t=0.3)	10,90%	11,56%	79,41%	73,30%	76,23%	20,08%
1-7,9 (t=0.0)	2,64%	32,99%	77,42%	53,07%	62,97%	43,30%
1-7,9 (t=0.1)	3,07%	21,43%	88,89%	57,73%	70,00%	32,81%

Table 24: Evaluation results of the Hearst patterns for the Website corpus, in dependency on the threshold and the chosen patterns

patterns	*LP*	*LR*	TP_{csc}	TR_{csc}	TF_{csc}	TF'_{csc}
All (t=0.0)	11,56%	26,74%	36,26%	38,43%	37,31%	31,15%
All (t=0.3)	25,17%	9,08%	75,24%	61,35%	67,59%	16,00%
1-7,9 (t=0.0)	12,75%	20,85%	50,19%	44,82%	47,35%	28,95%
1-7,9 (t=0.3)	24,54%	6,54%	75,98%	61,96%	68,26%	11,94%

Table 25: Evaluation results of the Hearst patterns for the Reuters corpus, in dependency on the threshold and the chosen patterns

After this detailed evaluation of the Hearst patterns with the help of TP_{csc} and TR_{csc}, the next interesting question is, whether the other two recommended measures from section 3.4 would suggest the same optimizations. In table 26 one can see the evaluation results achieved with TP_{sc} and TR_{sc}, while table 27 contains the results for the TO_{csc} measure. It exists a significant difference to the preciously achieved results because TF'_{csc} reached its best value for the merged relations at a threshold of 0.0. But TF'_{sc} as well as TF'^{ov}_{csc} reach their maximum at a threshold of 0.3. This value is clearly higher than the corresponding value at a threshold of 0.0. So what previously required a thorough analysis of TP_{csc} and TR_{csc} is here reflected by the F'-measure. This can be put down to another weighting of taxonomic and lexical precision and recall in the final F'-measures.

threshold	LP	LR	TP_{sc}	TR_{sc}	TF_{sc}	TF'_{sc}
0.0	1,00%	49,66%	0,10%	27,84%	0,21%	0,41%
0.3	7,27%	22,79%	3,23%	8,67%	4,71%	7,80%
0.6	12,09%	11,22%	6,44%	3,61%	4,63%	6,55%
0.9	17,04%	7,82%	10,40%	2,53%	4,07%	5,35%

Table 26: Evaluation results of all patterns for the Wikipedia corpus

threshold	LP	LR	TO_{csc}	TF'^{ov}_{csc}
0.0	1,00%	49,66%	10,76%	17,69%
0.3	7,27%	22,79%	50,81%	31,47%
0.6	12,09%	11,22%	52,21%	18,48%
0.9	17,04%	7,82%	58,63%	13,80%

Table 27: Evaluation results of all patterns for the Wikipedia corpus

Although in this case the weighting would have led to the correct decision for optimizing the Hearst patterns, it might not work as well in other cases. Compared to TP_{csc} and TR_{csc}, those other two measures are more like a black box where one has to trust that the final taxonomic F'-measure correctly reflects the own notion of a good ontology. This is also necessary because the values of TP_{sc}, TR_{sc} and TO_{csc} are influenced by more than one characteristic of the concept hierarchy, so that it would be very difficult to draw any conclusions from them.

5.2.2 Head Finding

All ontologies in this section were learned with the implementation of the Head Finding procedure as it is described in appendix B. Because it is specialized on semi-structured documents, it was only applied on the Wikipedia corpus. As it was said in subsection 5.1.2, the Head Finding procedure applies a pattern on dictionary entries or encyclopedia articles. This applied pattern is very similar to pattern (8) of the Hearst patterns (see subsection 5.1.1). Therefore it would be interesting to compare the results of the Hearst pattern with the Head Finding pattern.

The main difference between both approaches is that the Head Finding procedure uses structural information contained in an encyclopedia for improving the results. The first structural information it makes use of is the title of the current article. Unlike the Hearst patterns, which extract every information from a text, the Head Finding pattern is restricted to those taxonomic relations, where the current title is the hyponym. Furthermore the Head Finding pattern is only applied on the first sentence of an article (cf. subsection 5.1.2).

In the first row of table 28 one can see the evaluation results of the Head Finding pattern if it is applied on the complete article without using the additional structural information. The achieved results are very similar to those of pattern (8) in table 22. This shows that the following improvements can be put down to the additional structural information and not on the minor difference between the Head Finding pattern and pattern (8). Already the restriction of the extracted relations to those where the current title is also the hyponym leads to a major improvement of the taxonomic precision and recall values. But at the same time the lexical recall remains quite high compared to pattern (8), if its results are filtered by a threshold. Therefore the additional structural informations available in encyclopedia articles are a better filter for the learned relations than the threshold from the Hearst patterns. The effect of the filter is further improved regarding the concept hierarchy, if the Head Finding pattern is only applied on the first sentence. But this improvement on the taxonomy level is accompanied by a decrease of the lexical recall. All in all, the best results according to TF'_{csc} are achieved with only using relations where the current title is the hyponym.

	LP	*LR*	TP_{csc}	TR_{csc}	TF_{csc}	TF'_{csc}
Article	1,13%	47,96%	22,81%	84,40%	35,91%	41,07%
Titles	2,55%	39,12%	86,80%	45,83%	59,99%	47,35%
First	3,23%	34,35%	94,22%	46,84%	62,58%	44,36%

Table 28: Evaluation results for the Head Finding pattern applied on the complete article (Article), restricted to relations with the current title as hyponym (Titles) and to the first sentence (First)

If one looks at the additional statistical values of the learned ontologies, one can see the same trend. If the Head Finding pattern is applied on the complete article without using structural information, the values look as “unhealthy” as in table 23. Restricting the taxonomic relations to those with the current title as the hyponym leads to a major improvement of those values, although there still exist a number of loops in the concept hierarchy.

The best statistical values are achieved with restricting the Head Finding pattern to the first sentence. Only the deviation at the number of direct subcon-cepts is an indicator for remaining problems in the hierarchy. Given the average depth of the hierarchy and the average number of direct subconcepts compared to the total number of concepts, it is very likely that this deviation indicates a deformed hierarchy. In this deformed hierarchy a few concepts have a lot of direct subconcepts, while all other only have one or two direct subconcepts. But this doesn't differentiate the Head Finding hierarchy from the hierarchy extracted by the Hearst pattern, because there the additional statistical values allow the same interpretation.

	concepts	loops	avg. depth	avg. sub	sub dev.	avg. super	super dev.
ref	294	1	5,14	5,22	4,42	1,03	0,17
Article	12494	5311	99,54	3,01	58,63	1,6	3,03
Titles	4516	1092	3,13	3,13	45,47	1,47	1,16
First	3127	15	3,09	3,17	31,61	1,05	0,3

Table 29: Additional statistical values of the reference ontology from the tourism domain (ref) and of the ontologies learned with the Head Finding pattern from the Wikipedia corpus

In table 30 one can see the evaluation results achieved with TP_{sc} and TR_{sc}, while table 31 contains the results for the TO_{csc} measure. According to these measures the variants of the Head Finding pattern filtered by structural information get a different rank order. TF'_{sc} favours the filtering the results by title and first sentence while TF'^{ov}_{csc} favours the filtering by titles only. This can presumably put down to a stronger influence of the taxonomic and the lexical precision on the final TF' value.

	LP	LR	TP_{sc}	TR_{sc}	TF_{sc}	TF'_{sc}
Article	1,13%	47,96%	0,12%	26,53%	0,23%	0,46%
Titles	2,55%	39,12%	1,00%	14,22%	1,87%	3,57%
First	3,23%	34,35%	1,51%	12,18%	2,68%	4,97%

Table 30: Evaluation results of the Head Finding pattern for the Wikipedia corpus

	LP	LR	TO_{csc}	TF'^{ov}_{csc}
Article	1,13%	47,96%	11,87%	19,03%
Titles	2,55%	39,12%	41,27%	40,16%
First	3,23%	34,35%	45,01%	38,97%

Table 31: Evaluation results of the Head Finding pattern for the Wikipedia corpus

5.2.3 ArcRank

ArcRank is the last learning procedure, which will be evaluated. Because ArcRank is also specialized on semi-structured documents, it was only applied on the Wikipedia corpus. For this evaluation the implementation described in appendix C was used. This implementation of ArcRank doesn't provide any parameters for improving the results. Compared to the original ArcRank in (Jannink and Wiederhold, 1999), a simplified procedure for determining the concept hierarchy was used (see subsection 5.1.3).

In table 32 you can see the evaluation results of ArcRank for the Wikipedia corpus. The lexical recall is comparable to the Head Finding procedure in table 28, while the lexical precision is slightly below the value of Head Finding. Also the taxonomic precision and recall are below the values of Head Finding. The taxonomic precision value nevertheless is quite good, while the taxonomic recall is the worst of all previous procedures.

LP	LR	TP_{csc}	TR_{csc}	TF_{csc}	TF'_{csc}
2,40%	34,35%	79,65%	26,18%	39,41%	36,71%

Table 32: Evaluation results of ArcRank for the Wikipedia corpus

If one compares ArcRank with the Hearst patterns in table 21, one gets a mixed result. Compared to the unfiltered results of the Hearst patterns one would probably judge ArcRank as better because of its better taxonomic precision values. On the downside, ArcRank has a lower taxonomic recall and lexical recall. Contrary, the filtered Hearst patterns are better in almost every category but the lexical recall. In this case ArcRank has a significantly higher lexical recall. So in both cases one has to weight for oneself, which aspects of the learned concept hierarchy are more important.

But if one looks at the additional statistical values of the learned hierarchy in table 33, one gets the best results compared to all of the previous learned ontologies. Given the high number of concepts in the ontology, the average depth of the hierarchy is still reasonable. This is also reflected by the very good values for the average number of direct subconcepts and the deviation from this average value. These values suggest that the learned hierarchy isn't deformed like the previously learned hierarchies, although the average number of direct subconcepts is quite low.

	concepts	loops	avg. depth	avg. sub	sub dev.	avg. super	super dev.
ref	294	1	5,14	5,22	4,42	1,03	0,17
ArcRank	1426	0	31	2,15	4,03	1	0,04

Table 33: Additional statistical values of the reference ontology from the tourism domain (ref) and of the ontology learned with ArcRank from the Wikipedia corpus

As one can see in table 34 and 35, the other two measures would lead to similar results if ArcRank is compared with the Hearst pattern and the Head Finding procedure.

LP	LR	TP_{sc}	TR_{sc}	TF_{sc}	TF'_{sc}
2,40%	34,35%	0,48%	8,96%	0,91%	1,78%

Table 34: Evaluation results of ArcRank for the Wikipedia corpus

LP	LR	TO_{csc}	TF'^{ov}_{csc}
2,40%	34,35%	21,40%	26,37%

Table 35: Evaluation results of ArcRank for the Wikipedia corpus

5.2.4 Discussion

Especially the measures TP_{csc} and TR_{csc} have proved their usefulness in the detailed evaluation of learning procedures in the previous subsections. Compared to TP_{sc} and TR_{sc} the interpretation of their values is easier because they aren't also influenced by the lexical precision and recall. Compared to TO_{csc} they simply provide more information.

But this additional information and the uncoupling of the taxonomic measures from the lexical measures has really to be used for analyzing concept hierarchies. It isn't sufficient to only compare the TF'_{csc} of ontologies. Instead one has to weigh the importance of the different measures for oneself. This was seen in the evaluation of the Hearst patterns in subsection 5.2.1, where only looking at TF'_{csc} wouldn't have led to filtering the extracted relations with a threshold.

Additionally it has been shown, that the computation of less sophisticated measures is also useful for distinguishing the quality of different learning procedures. Examples are the average depth of the concept hierarchy or the average number of sub- and superconcepts and the standard deviation from these simple measures. They helped to discover significant differences in the concept hierarchies, which weren't noticed by the more sophisticated measures.

In table 36 one can see the rank order of the learned concept hierarchies in dependency on the chosen measures and the used learning procedure. For the Hearst pattern, the concept hierarchy learned from the Wikipedia corpus will be ranked. This concept hierarchy merges the results of all patterns and it was filtered by a threshold of 0.3. The evaluation results of this concept hierarchy are available in table 21, 26 and 27. For the Head Finding procedure the concept hierarchy learned from the Wikipedia corpus will be ranked. It is filtered by the title information and the first sentence. The evaluation results are available in table 28, 30 and 31. The evaluation results for the concept hierarchy learned with ArcRank are available in table 32, 34 and 35.

Only the name of the global taxonomic precision or overlap will be mentioned for the rows in the table. But also the taxonomic recall or the taxonomic F- and F'-measure as well as the additional statistical information will be taken into account for determining the rank order.

	Hearst pattern	Head Finding	ArcRank
TP_{csc}	2 – 3	1	2 – 3
TP_{sc}	1	2	3
TO_{csc}	2	1	3

Table 36: Rank order of concept hierarchies learned with the different learning procedures from section 3.3.2, in dependency on the

Especially ranking the learning procedures with the help of TP_{csc} and TR_{csc} isn't so easy, because the different factors have to be carefully weighed. The concept hierarchy learned with the Head Finding procedure was ranked as the best learned hierarchy because almost every measure has a superior value compared to the other learned concept hierarchies. It is followed by the concept hierarchy learned with the Hearst patterns. Compared to the concept hierarchy learned by ArcRank one gets a mixed results. If one compares the TF_{csc} values in table 21 and 32, the hierarchy extracted with the Hearst patterns will be judged as significantly better. But judged from the lexical level and the additional statistical values, the concept hierarchy learned with ArcRank is the better hierarchy. Therefore the rank order of both concept hierarchies will be dependent on weighing the different measures.

It is noticeable that the concept hierarchies are ranked differently, depending on the used measure. For the other two measures the decision about the order was easier because it wasn't necessary to weigh the different components of the measure. Especially TP_{sc} and TR_{sc} lead to a very different rank order of the hierarchies, while the usage of TO_{csc} leads to a rank order comparable with TP_{csc} although the order of the Hearst patterns hierarchy and the ArcRank hierarchy is clear.

The below average evaluation results of the concept hierarchy learned with ArcRank aren't necessarily meaningful and representative for the best achievable results. For example it wasn't analyzed in how far the number of documents in the corpus influence the results of ArcRank. Additionally a simplified procedure for extracting the concept hierarchies from the raw data of ArcRank was used (see subsection 5.1.3).

6 Conclusions & Future Work

This work presented a framework of building blocks which may be used for creating new similarity measures for the comparison of concept hierarchies. But also already existing measures like the taxonomic overlap presented in (Maedche, 2002) may be integrated into this framework. An important part of this framework is the differentiation between the global variant of a measure, which might be used for comparing whole concept hierarchies, and a local variant of the same measure, which compares two concepts with each other.

This common framework was used for making similarity measures comparable with each other. It directly proved its usefulness in the comparison of taxonomic overlap based measures with taxonomic precision and recall based measures. It was possible to show that the local variant of the taxonomic overlap could be defined in dependency on the local taxonomic F-measure. But at the same time for a concrete global measure a counterexample was given, that the potential asymmetry of this measure shouldn't be interpreted as a kind of precision and recall.

The findings regarding the global taxonomic overlap are only based on examples. It would be desirable to generalize those findings in order to draw further conclusions about the explanatory power of deviations between global taxonomic overlap values computed into both directions.

In this diploma thesis mainly considerations were made about the general explanatory power of evaluation measures, i.e. whether they are always meaningful and can be easily interpreted. It is one important result of these considerations that it is desirable to uncouple the development of the taxonomic precision and recall values from the development of their counterparts for the evaluation of lexicons. But little work was done on the topic which kind of errors are successfully detected in concept hierarchies by those measures. Also here further research of this topic would be interesting in order to improve the detection of different error classes in concept hierarchies.

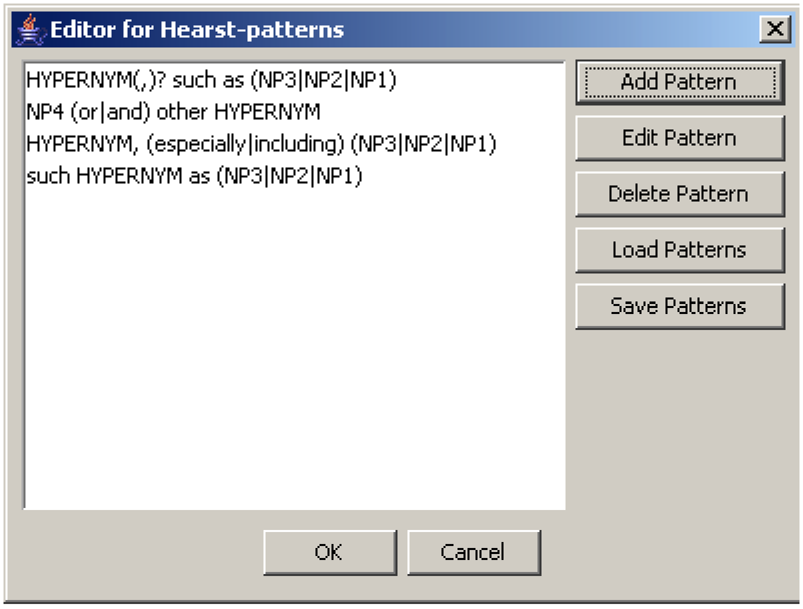
All in all three different measures were recommended for evaluating concept hierarchies. These measures were then tested on their usability in a comparison of three learning procedures, namely the Hearst patterns, the Head Finding procedure and the ArcRank algorithm. Especially the measures TP_{csc} and TR_{csc} proved their usefulness because there the effects of changing different aspects of an ontology didn't superpose each other in the computation of the evaluation results. Therefore the values were easy to interpret. During this evaluation of the concept hierarchies it also became clear, that it is very likely that one may draw different conclusions about the quality of a concept hierarchy depending on the chosen evaluation measure.

Another side effect of this prototypical evaluation was the conclusion that it is more effective to filter the results of a learning procedure with the help of structural information available in encyclopedias instead of a confidence value based on the number of occurrences of a taxonomic relation.

A Hearst-Pattern

A.1 Implementation

Although the original implementation of TextToOnto, which is available at http://kaon.semanticweb.org, already contains an implementation of the Hearst patterns, it was completely rewritten for this diploma thesis. This was necessary for improving the flexibility. For example, in the rewritten version of the Hearst patterns it is possible to change the set of used patterns and to test the new patterns before using them on a larger text corpus. Figure 18 and figure 19 show the user interface for defining and testing new patterns. They can be accessed from the TaxoBuilder module of TextToOnto.

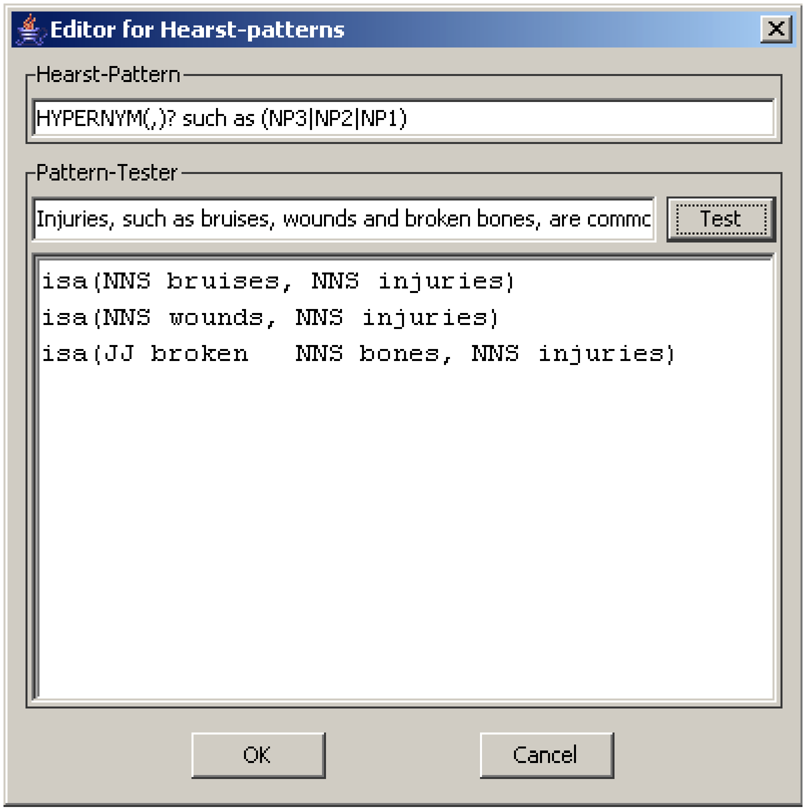

Editor for Hearst-patterns

```
HYPERNYM(,)? such as (NP3|NP2|NP1)
NP4 (or|and) other HYPERNYM
HYPERNYM, (especially|including) (NP3|NP2|NP1)
such HYPERNYM as (NP3|NP2|NP1)
```

Figure 18: List of currently used patterns

The implementation of the procedure can be found in the *taxobuilder* sub project of TextToOnto. It is available on the accompanying CD-ROM. It consists of the files *HearstPattern.java* and *HearstMatch.java* (see figure 20). Both files can be found in the package `edu.unika.aifb.texttoonto.taxobuilder.pattern`. An instance of the HearstPattern-class corresponds to a single pattern. A pattern is written by the user in the editor (see figure 18) and then translated by the HearstPattern instance into a regular expression which may be used for recognizing it in a POS tagged text.

After creating a new HearstPattern instance it may be used for extracting relations from POS tagged texts. The corresponding method *extractRelations* returns a list of HearstMatch instances which were found in the text. Each HearstMatch instance represents one match and contains the extracted hypernym and a list of corresponding hyponyms.



HYPERNYM(,)? such as (NP3|NP2|NP1)

Injuries, such as bruises, wounds and broken bones, are commc

```
isa(NNS bruises, NNS injuries)
isa(NNS wounds, NNS injuries)
isa(JJ broken   NNS bones, NNS injuries)
```

Figure 19: Editor for creating and testing new patterns

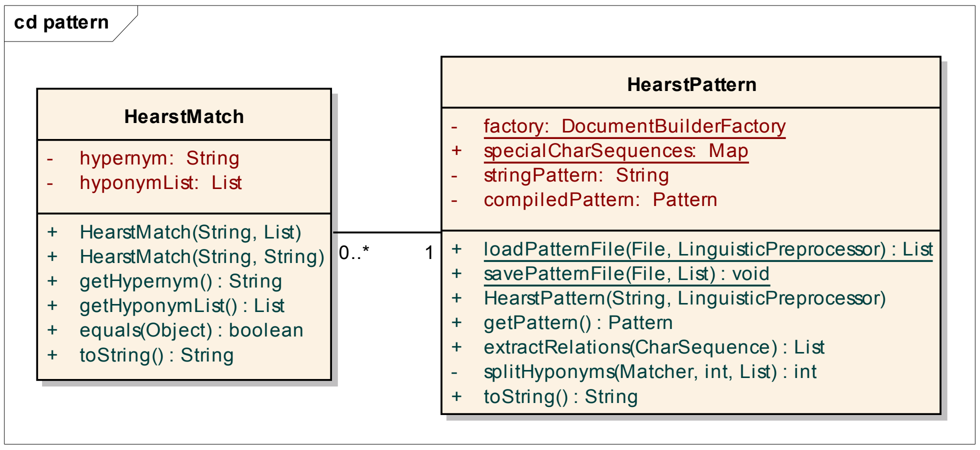

cd pattern
HearstMatch
- hypernym: String
- hyponymList: List
+ HearstMatch(String, List)
+ HearstMatch(String, String)
+ getHypernym() : String
+ getHyponymList() : List
+ equals(Object) : boolean
+ toString() : String
0..*
1
HearstPattern
- factory: DocumentBuilderFactory
+ specialCharSequences: Map
- stringPattern: String
- compiledPattern: Pattern
+ loadPatternFile(File, LinguisticPreprocessor) : List
+ savePatternFile(File, List) : void
+ HearstPattern(String, LinguisticPreprocessor)
+ getPattern() : Pattern
+ extractRelations(CharSequence) : List
- splitHyponyms(Matcher, int, List) : int
+ toString() : String

Figure 20: UML diagram of the classes implementing a Hearst pattern

A.2 Test Cases

The following test cases were used for testing the implementation of the Hearst patterns. They are available in the test folder of the *taxobuilder* sub project. For each of the patterns used in this diploma thesis at least one pattern exists which shows that it can be used for successfully extracting a relation with this pattern. All these tests are contained in *PatternTest.java* in the package `edu.unika.aifb.texttoonto.taxobuilder`.

Name	testExtractRelations1
Input	Pattern: ”HYPERNYM(,)? such as (NP3|NP2|NP1)” Text: ”Injuries, such as bruises, wounds and broken bones, are common in sports.”
Result	Hypernym: "NNS injuries" Hyponyms: "NNS bruises", "NNS wounds", "JJ broken NNS bones"

Name	testExtractRelations2
Input	Pattern: ”HYPERNYM(,)? such as (NP3|NP2|NP1)” Text: ”Smaller injuries such as bruises and a wound are common in sports.”
Result	Hypernym: "JJR smaller NNS injuries" Hyponyms: "NNS bruises", "DT a NN wound"

Name	testExtractRelations3
Input	Pattern: ”HYPERNYM(,)? such as (NP3|NP2|NP1)” Text: ”Injuries such as bruise wounds and broken bones are common in sports.”
Result	Hypernym: "NNS injuries" Hyponyms: "NN bruise NNS wounds", "JJ broken NNS bones"

Name	testExtractRelations4
Input	Pattern: ”NP4 (or|and) other HYPERNYM” Text: ”Bruises, wounds and other injuries are common in sports.”
Result	Hypernym: "NNS injuries" Hyponyms: "NNS bruises", "NNS wounds"

Name	testExtractRelations5
Input	Pattern: ”NP4 (or|and) other HYPERNYM” Text: ”Bruises, wounds, broken bones or other injuries are common in sports.”
Result	Hypernym: "NNS injuries" Hyponyms: "NNS bruises", "NNS wounds", "JJ broken NNS bones"

Name	`testExtractRelations6`

Input	`Pattern: ”HYPERNYM, (especially|including) (NP3|NP2|NP1)”`
	`Text: ”Injuries, especially bruises and broken bones, are common in sports.”`

Result	`Hypernym: "NNS injuries"`
	`Hyponyms: "NNS bruises", "NNS wounds", "JJ broken NNS bones"`

Name	`testExtractRelations7`

Input	`Pattern: ”HYPERNYM, (especially|including) (NP3|NP2|NP1)”`
	`Text: ”Injuries, including bruises and wounds, are common in sports.”`

Result	`Hypernym: "NNS injuries"`
	`Hyponyms: "NNS bruises", "NNS wounds", "JJ broken NNS bones"`

Name	`testExtractRelations8`

Input	`Pattern: ”such HYPERNYM as (NP3|NP2|NP1)”`
	`Text: ”Such injuries as bruises and wounds are common in sports.”`

Result	`Hypernym: "NNS injuries"`
	`Hyponyms: "NNS bruises", "NNS wounds"`

Name	`testExtractRelations9`

Input	`Pattern: ”such HYPERNYM as (NP3|NP2|NP1)”`
	`Text: ”Such injuries as bruises are common in sports.”`

Result	`Hypernym: "NNS injuries"`
	`Hyponyms: "NNS bruises"`

Name	`testExtractRelations10`

Input	`Pattern: ”such HYPERNYM as (NP3|NP2|NP1)”`
	`Text: ”Such injuries as bruises are common in sports.\n Such injuries as bruises and wounds are common in sports.\n”`

Result 1	`Hypernym: "NNS injuries"`
	`Hyponyms: "NNS bruises"`

Result 2	`Hypernym: "NNS injuries"`
	`Hyponyms: "NNS bruises", "NNS wounds"`

Name	`testExtractRelations11`

Input	`Pattern: ”HYPERNYM like (NP3|NP2|NP1)”`
	`Text: ”Injuries like bruises and wounds are common in sports”`

Result	`Hypernym: "NNS injuries"`
	`Hyponyms: "NNS bruises", "NNS wounds"`

Name	`testExtractRelations12`

Input	`Pattern: ”(NP3|NP2|NP1) is HYPERNYM”`
	`Text: ”A bruise is an injury common in sports.”`

Result	`Hypernym: "DT an NN injury"`
	`Hyponyms: "DT a NN bruise"`

Name	`testExtractRelations13`

Input	`Pattern: ”(NP3|NP2|NP1), another HYPERNYM”`
	`Text: ”A bruise, another injury common in sports, doesn't hurt very much.”`

Result	`Hypernym: "NN injury"`
	`Hyponyms: "DT a NN bruise"`

B Head Finding

B.1 Implementation

In the original version of TextToOnto no implementation of the Head Finding procedure was available. It was completely new written based on the descriptions in (Chodrow and Byrd, 1985) and in subsection 5.1.2. One of the main challenges was to introduce the ability to add semi-structured documents to the corpus. A description of the semi-structured document implementation can be found in appendix C about ArcRank.

The implementation of the Head Finding procedure can be found in the package `edu.unika.aifb.texttoonto.taxobuilder.headfinding` of the `taxobuilder` subproject of TextToOnto. It is available on the accompanying CD-ROM. It consists of the single file *HeadFinder.java*. As it was explained in subsection 5.1.2, the implementation of the Hearst patterns could be partially reused for the Head Finding procedure. For each document which should be processed, `HeadFinder.extractRelations(String, CharSequence)` has to be called. It returns a list of `HearstMatch`-instances.

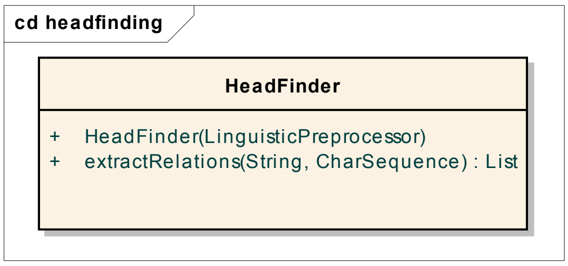

cd headfinding
HeadFinder
+ HeadFinder(LinguisticPreprocessor)
+ extractRelations(String, CharSequence) : List

Figure 21: UML diagram of the HeadFinder class

C ArcRank

C.1 Implementation

In the original version of TextToOnto no implementation of the ArcRank algorithm was available. It was completely new written based on the descriptions in (Jannink and Wiederhold, 1999) and (Haveliwala, 1999). One of the main challenges in the implementation was to introduce semi-structured documents to the corpus implementation of TextToOnto. They need to support a title for the document and they have to extract the explicit links to other documents and make them available through an API.

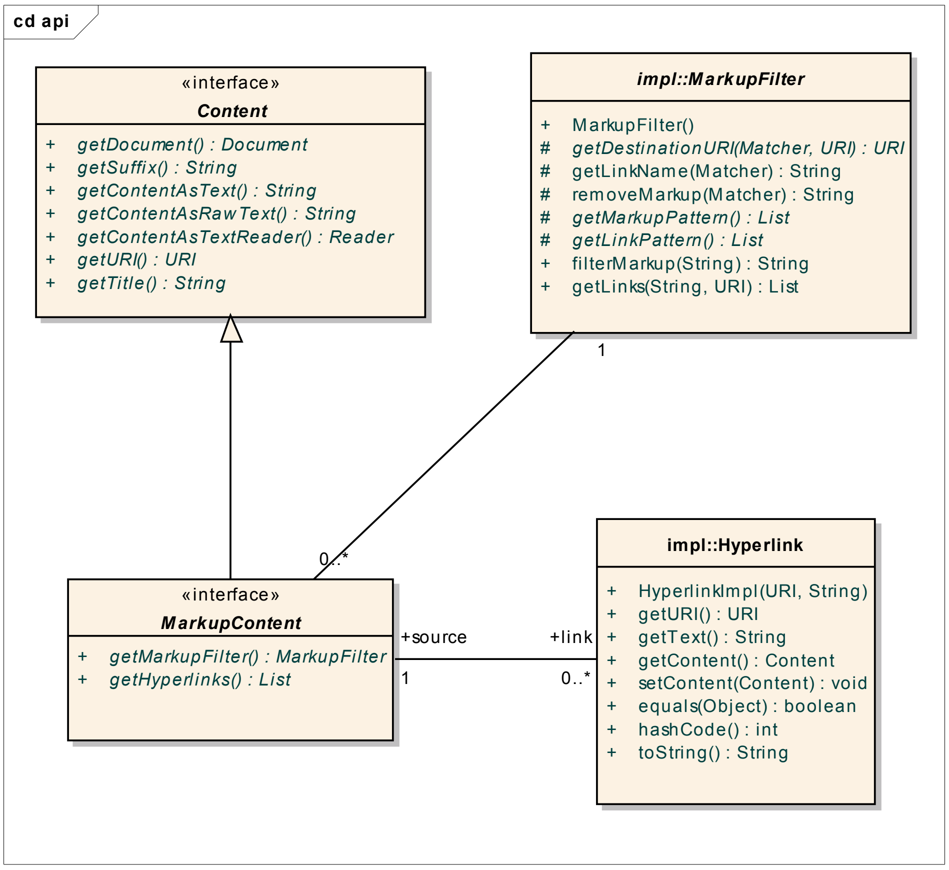

cd api
«interface»
Content
+ getDocument() : Document
+ getSuffix() : String
+ getContentAsText() : String
+ getContentAsRawText() : String
+ getContentAsTextReader() : Reader
+ getURI() : URI
+ getTitle() : String
impl::MarkupFilter
+ MarkupFilter()
# getDestinationURI(Matcher, URI) : URI
# getLinkName(Matcher) : String
# removeMarkup(Matcher) : String
# getMarkupPattern() : List
# getLinkPattern() : List
+ filterMarkup(String) : String
+ getLinks(String, URI) : List
1
0..*
«interface»
MarkupContent
+ getMarkupFilter() : MarkupFilter
+ getHyperlinks() : List
+source
1
+link
0..*
impl::Hyperlink
+ HyperlinkImpl(URI, String)
+ getURI() : URI
+ getText() : String
+ getContent() : Content
+ setContent(Content) : void
+ equals(Object) : boolean
+ hashCode() : int
+ toString() : String

Figure 22: UML diagram of the classes for handling texts with markup

In figure 22 one can see some of the newly introduced or modified classes in the `corpus` subproject of TextToOnto. The interface `Content` was extended by `getTitle()`, so every content may have a title. The interface `MarkupContent` is derived from `Content` and adds the new method `getHyperlinks()` which returns a list of `Hyperlink` instances. Two classes implement the `MarkupContent` interface. One class handles content with HTML markup and the other one handles content with markup in the Wiki style. The latter implementation is necessary for processing articles from Wikipedia or one of its sister projects, like the Wiktionary.

Those classes for HTML content and Wiki content only differ in the used implementation of the `MarkupFilter` class. The `MarkupFilter` is used for removing the markup from the content so that it may be subsequently tagged with POS labels. Another purpose of the `MarkupFilter` is to scan the content for markup denoting a hyperlink. This markup is used for creating corresponding `Hyperlink` instances.

The class `Hyperlink` represents a link between two content instances. The source of the link is returned by `getContent()`, while the link target is returned by `getURI()`. The target is denoted by a URI because the referenced content isn't necessarily in the corpus. Representing the target as a URI allows to lookup, whether the denoted content is available in the corpus and, if required, to add the referenced content to the corpus.

These newly added and modified classes in the `corpus` subproject of TextToOnto form the basis for the implementation of the ArcRank algorithm. The implementation of ArcRank can be found in the package `edu.unika.aifb.texttoonto.taxobuilder.arcrank` of the `taxobuilder` subproject of TextToOnto. It is available on the accompanying CD-ROM. It consists of the single file *ArcRank.java*, which contains the class `ArcRank` and several private member classes. The member classes are internally used for representing the nodes and edges of the directed, labeled graph extracted from a corpus which contains documents with `MarkupContent`.[23] The output of the ArcRank algorithm is a hierarchy of concepts whereas the title of each document was converted into a concept in the hierarchy.

C.2 Test Cases

The extraction of links from markup content is a central task for a proper functioning of the ArcRank algorithm. For this diploma thesis, two filters for extracting links were implemented, one implementation for HTML content and one for wiki content[24]. These implementations are tested with the following test cases. They can be found in the test folder of the *corpus* subproject of TextToOnto and there in the file *LinkFilterTest.java* in the package `edu.unika.aifb.texttoonto.corpus`.

23 See section 5.1.3 for details about the ArcRank algorithm.

24 An overview of the markup for links in Wikipedia is available at http://en.wikipedia.org/wiki/Wikipedia:How_to_edit_a_page. Because of the wide variety of link types only the types important for ArcRank will be recognized by the `WikiFilter`. This includes links to other Wikipedia articles in arbitrary languages and namespaces . Links to external http-addresses are currently not recognized but it could be added if necessary .

Name	`testWikiFilterLinks1`

Input	`Source URI: ”jdbc:mysql://localhost:3306/wikidb/ArticleName”`
	`Content: ”[[Sandbox]]\n[[Wikipedia:Contributing to Wikipedia]]\n[[Wikipedia:Tutorial|any text]]\n[[:Category:Astrophysics]]”`

Result 1	`Target URI: ”jdbc:mysql://localhost:3306/wikidb/Sandbox”`
	`Link name: ”Sandbox”`

Result 2	`Target URI: ”jdbc:mysql://localhost:3306/wikidb/Wikipedia:Contributing_to_Wikipedia”`
	`Link name: ”Contributing to Wikipedia”`

Result 3	`Target URI: ”jdbc:mysql://localhost:3306/wikidb/Wikipedia:Tutorial”`
	`Link name: ”any text”`

Result 4	`Target URI: ”jdbc:mysql://localhost:3306/wikidb/:Category:Astrophysics”`
	`Link name: ”Astrophysics”`

Name	`testWikiFilterLinks2`

Input	`Source URI: ”http://en.wikipedia.org/wiki/ArticleName”`
	`Content: ”[[wiktionary:house]]\n[[wiktionary:house|]]\n[[:de:Michael Jackson]]\n[[w:de:Michael Jackson]]\n[[:de:Österreich|Österreich]]\n[http://www.google.de Google]”`

Result 1	`Target URI: ”http://en.wikipedia.org/wiki/wiktionary:house”`
	`Link name: ”house”`

Result 2	`Target URI: ”http://en.wikipedia.org/wiki/wiktionary:house”`
	`Link name: ”house”`

Result 3	`Target URI: ”http://en.wikipedia.org/wiki/:de:Michael_Jackson”`
	`Link name: ”Michael Jackson”`

Result 4	`Target URI: ”http://en.wikipedia.org/wiki/w:de:Michael_Jackson”`
	`Link name: ”Michael Jackson”`

Result 5	`Target URI: ”http://en.wikipedia.org/wiki/:de:Österreich”`
	`Link name: ”Österreich”`

Name	`testHTMLFilterLinks1`

Input	`Source URI: ”http://en.wikipedia.org/wiki/ArticleName”`
	`Content: ”<a target="_blank"\nhref="Sandbox"/>\n<a\nhref="Contributing_to_Wikipedia">Contributing to Wikipedia</a>\n<a href="/wiki/Wikipedia:Tutorial">any text</a>”`

Result 1	`Target URI: ”http://en.wikipedia.org/wiki/Sandbox”`
	`Link name: ”Sandbox”`

Result 2	`Target URI: ”http://en.wikipedia.org/wiki/Contributing_to_Wikipedia”`
	`Link name: ”Contributing to Wikipedia”`

Result 3	`Target URI: ”http://en.wikipedia.org/wiki/Wikipedia:Tutorial”`
	`Link name: ”any text”`

Name	`testHTMLFilterLinks2`

Input	`Source URI: ”http://en.wikipedia.org/wiki/ArticleName”`
	`Content: ”<a href=\"http://www.google.de\" target=\"_blank\"/>”`

Result	`Target URI: ”http://www.google.de”`
	`Link name: ”http://www.google.de”`

D Taxonomic Relation Evaluation

D.1 User Interface

The application, which was used in this diploma thesis for producing the evaluation results, is available in the *Diplomathesis* subproject. The main class is `startscripts.TaxonomyComparison`. In figure 23 one can see a screenshot of this application. The first two text fields contain the information, which should be used for loading the reference and the learned ontology. It is possible to select multiple files containing learned ontologies. In this case one ontology after the other is compared with the reference ontology. At the moment two formats for saving an ontology are supported:

- The KAON file format, which is also used by TextToOnto. See http://kaon.semanticweb.org/ for more details on this format.
- A plain text format, which contains taxonomic relations in the form "isa(hyponym, hypernym) = 0.3". Each of these relations must be in a separate line. After the actual taxonomic relation, one can optionally supply a confidence value. If it is missing, a confidence value of 1 will be assumed (see below for an explanation of the confidence value).

It will be no problem to add further file formats as long as a mapping to the plain text format can be found. For example, the KAON file format, which basically is XML, is converted by XSLT to the plain text file format.

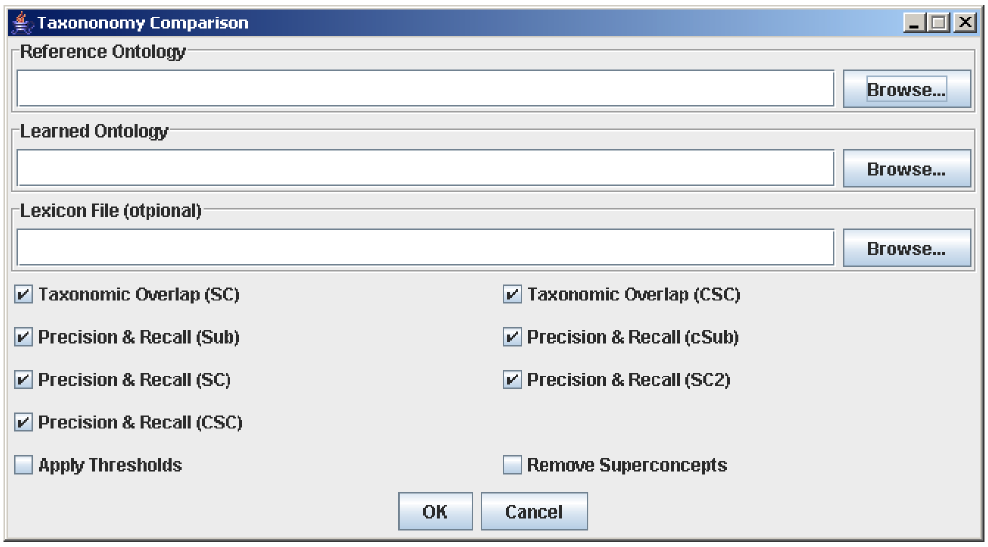


Figure 23: Screenshot of the TaxonomyComparison application

After the selection of the reference ontology and the learned ontologies one can optionally supply a lexicon. This lexicon is used for mapping lexical entries in the learned ontologies to lexical entries in the reference ontology. The DTD, which describes the format of the lexicon, is available in figure 24. This file format was originally used for providing a mapping between the titles of articles in Wikipedia and the corresponding lexical entry in the reference ontology (see subsection 4.3.2). If it is used as a lexicon, then the `title`-attribute of the `wikiarticle`-element contains a lexical entry in the learned ontology. The `value`-attribute of the surrounding concept-element then contains the corresponding lexical entry from the reference ontology.

```
<!ELEMENT mapping (concept*)>

<!ELEMENT concept (wikiarticle*)>
<!ATTLIST concept value CDATA #REQUIRED>

<!ELEMENT wikiarticle EMPTY>
<!ATTLIST wikiarticle title CDATA #REQUIRED
                      id CDATA #IMPLIED>
```

Figure 24: DTD which describes the format of the lexicon file

After providing the necessary information about the reference and the learned ontologies and the optional lexicon, one can choose between the different evaluation measures, which were presented in section 3.3. If multiple evaluation measures are selected, they will be consecutively applied on each of the learned ontologies.

After that, one can choose whether the ontologies should be evaluated with different thresholds. If this option is selected then an ontology will be evaluated multiple times with different thresholds. If a threshold is applied, all relations in the ontology whose value is below this threshold will be ignored. It is started with a threshold of 0.0 and it is increased in steps of 0.1, until 1.0 is reached.

Finally one can select the option “Remove superconcepts”. It performs normalization steps on the learned ontologies, which often contain concepts with more than one superconcept. This might adulterate the evaluation results. Therefore such superconcepts are removed in three steps:

- In the first step, all loops are removed from a learned ontology. A loop is detected by traversing the subconcepts, beginning at the root of the hierarchy. If one passes the same concept a second time before reaching a leaf concept the corresponding taxonomic relation will be removed from the hierarchy. This is not a deterministic procedure, because it is dependent on the traverse order of the subconcepts.
- In the second step, all shortcuts are removed from the hierarchy. A shortcut is a taxonomic relation, connecting two concepts directly, which are also connected via several other taxonomic relations. Those shortcuts aren't necessary in an ontology because of the transitivity of the taxonomic relation.
- In the last step, for each concept with several superconcepts all taxonomic relations to the superconcepts will be removed, except those with the highest threshold.

After pressing the OK button, the evaluation results will be computed. They are printed to the standard output. Directly thereafter one can change the parameters and perform the next evaluation.

D.2 Implementation

Figure 28 contains an overview of the classes, which are used for the evaluation of ontologies. The most important class is the `TaxonomyComparator`-class. It implements all building blocks described in subsection 3.3.1. It can be configured to use this building blocks in any combination. Therefore the `ComparatorConfiguration`-class is used. It is possible to provide different configurations for the computation of the precision and the recall value. Further parameters of the constructor of `TaxonomyComparator` are the learned and the reference ontology which should be evaluated.

The classes `TaxonomicOverlap` and `PrecisionRecallComparator` contain the configurations, which were used for the evaluation in this diploma thesis. The definition of a new measure is practically reduced to writing a new configuration. Only if a previously unknown extract from the concept hierarchy should be used, some additional work has to be done. Therefore the method `precomputHierarchyExtracts(CoreOntology, ComparatorConfiguration)` of the class `TaxonomyComparator` has to be overridden. It returns the characteristic extracts for all concepts in the given ontology. Further implementation details of the classes in figure 28 are available in the JavaDoc of the classes.

D.3 Test Cases

The implementation of the `TaxonomyComparator` is tested with the following test cases. They can be found in the test folder of the *Diplomathesis* subproject and there in the file *TaxonomyComparisonTest.java* in the package `startscripts.utils`. They should test the different building blocks of an evaluation measure. This is done by comparing the two learned ontologies in figure 26 and 27 with the reference ontology in 25. These examples are taken from (Cimiano, Hotho and Staab, 2004). The following tables contain the expected evaluation results for these ontologies and/or the characteristic extracts from the concept hierarchy.

Name	testGetMeasures1			
Input	O_{gold} and O_{gen1}			
Results		*TP*	*TR*	*TF*
	TP_{cl}^{ov}	91,67%	100,00%	95,65%
	TP_{sc}^{ov}	21,19%	21,60%	21,40%
	TP_{sub}	50,00%	54,54%	52,17%
	TP_{csub}	100,00%	100,00%	100,00%

Name	testGetMeasures2			
Input	O_{gold} and O_{gen2}			
Results		TP	TR	TF
	TP^{ov}_{cl}	100,00%	46,67%	63,63%
	TP^{ov}_{sc}	27,86%	25,39%	26,57%
	TP_{sub}	85,71%	54,54%	66,67%
	TP_{csub}	100,00%	100,00%	100,00%

Name	testGetSemanticCotopies1	
Input	O_{gold} and O_{gen1}	
Results	$c \in C_{gen1}$	$csc(c, O_{gen1}, O_{gold})$
	excursion	{}
	trip	{}
	hotel	{}
	bike	{}
	car	{}
	apartment	{}
	planable	{excursion}
	joinable	{excursion, trip}
	rideable	{excursion, trip, bike}
	driveable	{excursion, trip, bike, car}
	rentable	{excursion, trip, bike, car, apartment}
	root	{excursion, trip, bike, car, apartment, hotel}

Name	testGetSemanticCotopies2	
Input	O_{gold} and O_{gen1}	
Results	$c \in C_{gold}$	$csc(c, O_{gold}, O_{gen1})$
	excursion	{}
	trip	{}
	hotel	{}
	bike	{}
	car	{}
	apartment	{}
	activity	{excursion, trip}
	two-wheeled vehicle	{excursion, trip, bike}
	vehicle	{excursion, trip, bike, car}
	object to rent	{excursion, trip, bike, car, apartment}
	root	{excursion, trip, bike, car, apartment, hotel}

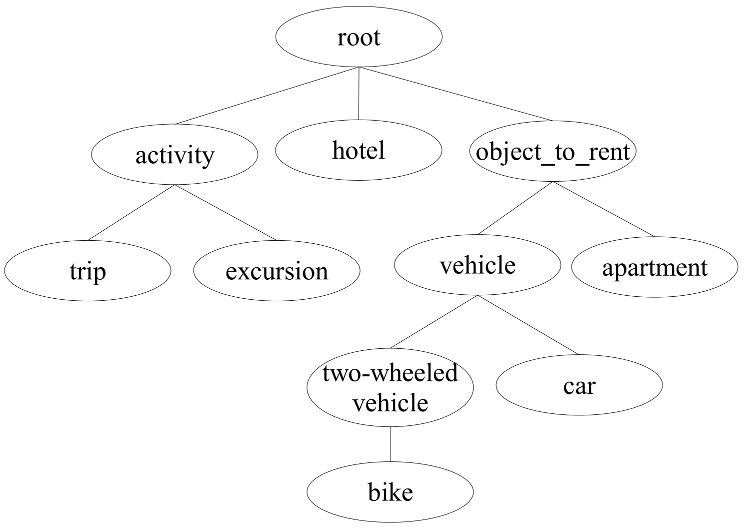


Figure 25: Reference ontology O_{gold}

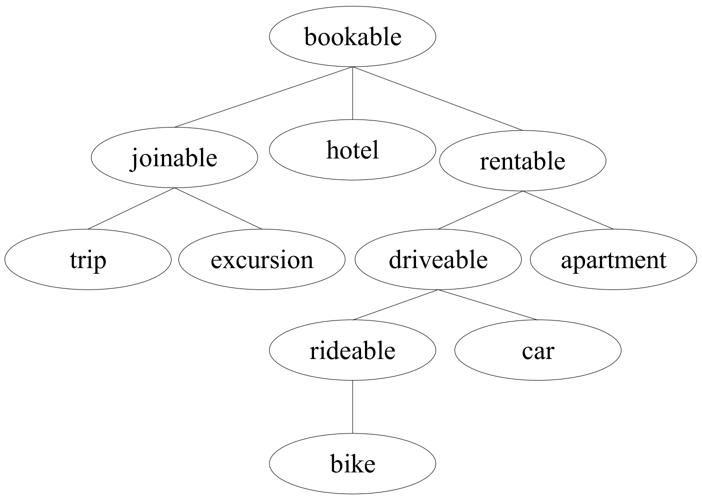


Figure 26: Learned ontology O_{gen1}

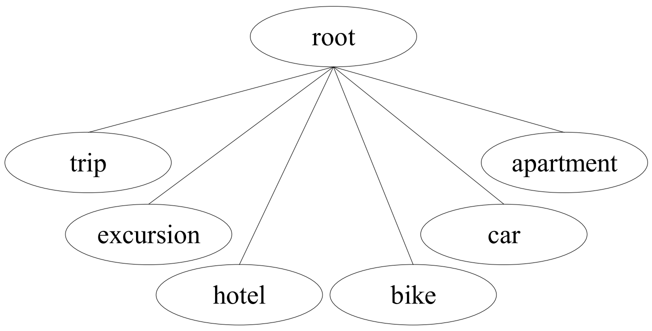


Figure 27: Learned ontology O_{gen2}

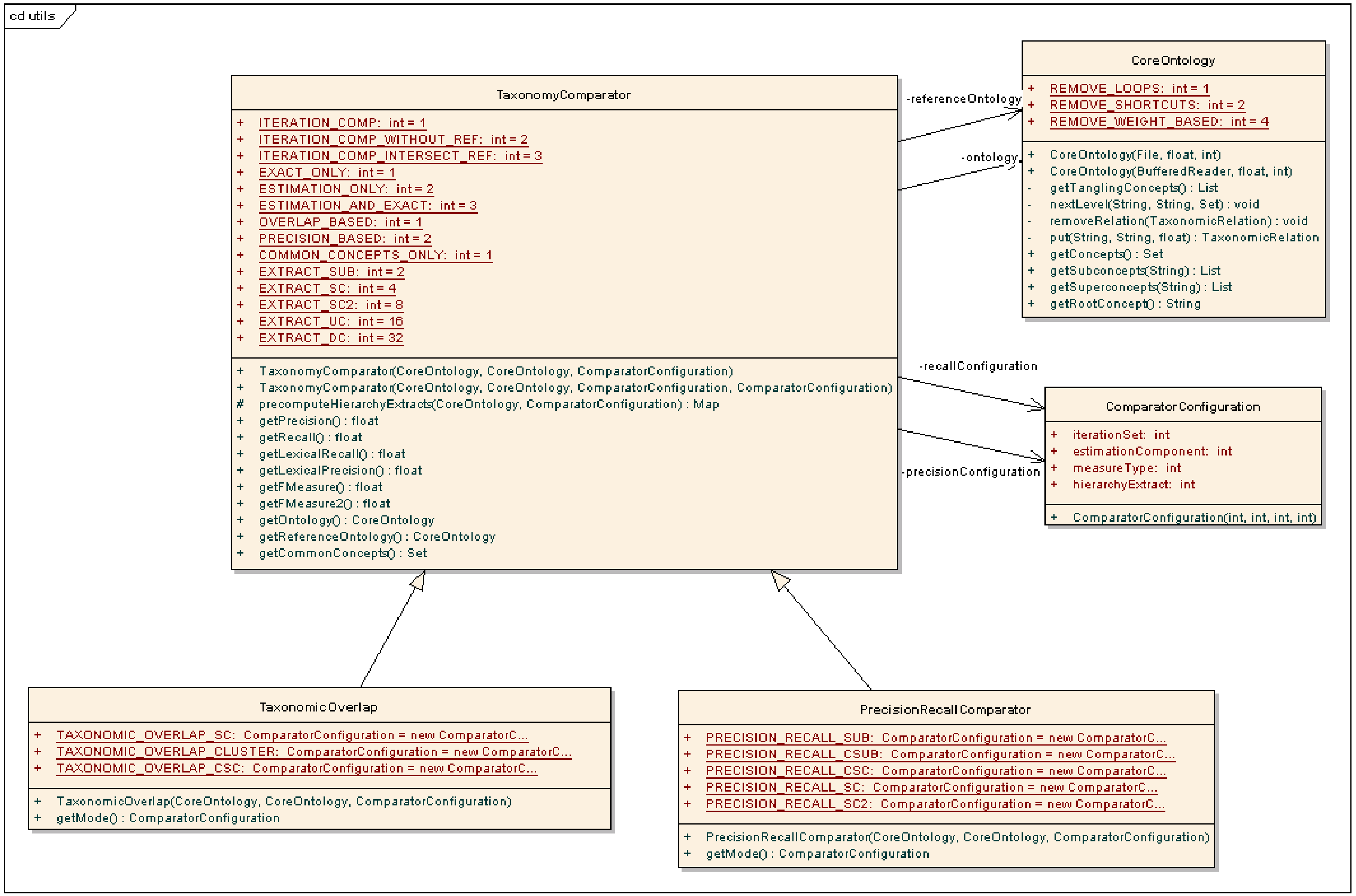

cd utils
TaxonomyComparator
+ ITERATION_COMP: int = 1
+ ITERATION_COMP_WITHOUT_REF: int = 2
+ ITERATION_COMP_INTERSECT_REF: int = 3
+ EXACT_ONLY: int = 1
+ ESTIMATION_ONLY: int = 2
+ ESTIMATION_AND_EXACT: int = 3
+ OVERLAP_BASED: int = 1
+ PRECISION_BASED: int = 2
+ COMMON_CONCEPTS_ONLY: int = 1
+ EXTRACT_SUB: int = 2
+ EXTRACT_SC: int = 4
+ EXTRACT_SC2: int = 8
+ EXTRACT_UC: int = 16
+ EXTRACT_DC: int = 32
+ TaxonomyComparator(CoreOntology, CoreOntology, ComparatorConfiguration)
+ TaxonomyComparator(CoreOntology, CoreOntology, ComparatorConfiguration, ComparatorConfiguration)
# precomputeHierarchyExtracts(CoreOntology, ComparatorConfiguration) : Map
+ getPrecision() : float
+ getRecall() : float
+ getLexicalRecall() : float
+ getLexicalPrecision() : float
+ getFMeasure() : float
+ getFMeasure2() : float
+ getOntology() : CoreOntology
+ getReferenceOntology() : CoreOntology
+ getCommonConcepts() : Set
-referenceOntology
-ontology
CoreOntology
+ REMOVE_LOOPS: int = 1
+ REMOVE_SHORTCUTS: int = 2
+ REMOVE_WEIGHT_BASED: int = 4
+ CoreOntology(File, float, int)
+ CoreOntology(BufferedReader, float, int)
- getTanglingConcepts() : List
- nextLevel(String, String, Set) : void
- removeRelation(TaxonomicRelation) : void
- put(String, String, float) : TaxonomicRelation
+ getConcepts() : Set
+ getSubconcepts(String) : List
+ getSuperconcepts(String) : List
+ getRootConcept() : String
-recallConfiguration
-precisionConfiguration
ComparatorConfiguration
+ iterationSet: int
+ estimationComponent: int
+ measureType: int
+ hierarchyExtract: int
+ ComparatorConfiguration(int, int, int, int)
TaxonomicOverlap
+ TAXONOMIC_OVERLAP_SC: ComparatorConfiguration = new ComparatorC...
+ TAXONOMIC_OVERLAP_CLUSTER: ComparatorConfiguration = new ComparatorC...
+ TAXONOMIC_OVERLAP_CSC: ComparatorConfiguration = new ComparatorC...
+ TaxonomicOverlap(CoreOntology, CoreOntology, ComparatorConfiguration)
+ getMode() : ComparatorConfiguration
PrecisionRecallComparator
+ PRECISION_RECALL_SUB: ComparatorConfiguration = new ComparatorC...
+ PRECISION_RECALL_CSUB: ComparatorConfiguration = new ComparatorC...
+ PRECISION_RECALL_CSC: ComparatorConfiguration = new ComparatorC...
+ PRECISION_RECALL_SC: ComparatorConfiguration = new ComparatorC...
+ PRECISION_RECALL_SC2: ComparatorConfiguration = new ComparatorC...
+ PrecisionRecallComparator(CoreOntology, CoreOntology, ComparatorConfiguration)
+ getMode() : ComparatorConfiguration

Figure 28:UML diagram of the classes implementing the evaluation of concept hierarchies

References

Ahlswede, T. and Evens, M. (1988). Parsing vs. Text Processing in the Analysis of Dictionary Definitions, In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*

Ahmad, K., Tariq, M., Vrusias, B. and Handy, C. (2003). Corpus Based Thesaurus Construction for Image Retrieval in Specialist Domains, In *Proceedings of the 25th European Conference on Advances in Information Retrieval (ECIR)*

Alfonso Ureña López, L., De Buenaga, M., Garcia M. and Gómez, J. (1998). Integrating and Evaluating WSD in the Adaptation of a Lexical Database in Text Categorization Task, In *Proceedings of the 1st Workshop on Text, Speech, Dialogue*

Amsler, R. (1981). A Taxonomy for English Nouns and Verbs, In *Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics*

Berland, M. and Charniak, E. (1999). Finding Parts in Very Large Corpora, In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*

Chang, J., Ker, S. and Chen, M. (1998). Taxonomy and Lexical Semantics - From the Perspective of Machine Readable Dictionaries. *Lecture Notes in Computer Science Volume 1529/1998*, Springer Verlag:Heidelberg , pages 199-212

Chodrow, M. and Byrd, J. (1985). Extracting Semantic Hierarchies from a Large On-Line Dictionary, In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*

Cimiano, P., Pivk, A., Schmidt-Thieme, L. and Staab, S. (2005). Learning Taxonomic Relations from Heterogenous Sources of Evidence. In P. Buitelaar, B. Magnini and P. Cimiano (eds.). *Ontology Learning from Text: Methods, Applications, Evaluation*. IOS Press:Amsterdam

Cimiano, P., Hotho, A. and Staab, S. (2004). Learning Concept Hierarchies from Text Corpora using Formal Concept Analysis. *Technical Report*, Institute AIFB, University of Karlsruhe

De Buenaga Rodríguez, M., Gómez-Hidalgo, J. and Díaz-Agudo, B. (2000). Using Wordnet to Complement Training Information in Text Categorization. In Nicolov, N. and Mitkov, R. (eds.). *Recent Advantages in Natural Language Processing*. John Benjamins:Amsterdam

Ganter, B. and Wille, R. (1999). *Formal Concept Analysis - Mathematical Foundations*. Springer Verlag:Heidelberg

Girju, R. and Moldovan, D. (2002). Text Mining for Causal Relations, In *Proceedings of the FLAIRS Conference*

Gonzalo, J., Verdejo, F., Chugu, I. and Cigarrán, J. (1998). Indexing with WordNet synsets can improve text retrieval, In *Proceedings of the COLING/ACL '98 Workshop on Usage of WordNet for NLP*

Haveliwala, T. (1999). Efficient Computation of PageRank. *Technical Report*, Stanford University, Stanford, CA

Hearst, M. (1992). Automatic Acquisition of Hyponyms from Large Text Corpora, In *Proceedings of the 14th International Conference on Computational Linguistics*

Hotho, A., Staab, S. and Stumme, G. (2003). Text Clustering Based on Background Knowledge. *Technical Report 425*, Institute AIFB, University of Karlsruhe

Iris, M., Litowitz, B. and Evens, M. (1988). Problems of the part-whole relation. In Evens, M. (eds.). *Relational Models of the Lexicon - Representing Knowledge in Semantic Networks*. Cambridge University Press:Cambridge

Jannink, J. and Wiederhold, G. (1999). Thesaurus Entry Extraction from an On-line Dictionary, In *Proceedings of Fusion '99*

Kashyap, V. (1999). Design and Creation of Ontologies for Environmental Information Retrieval, In *Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management*

Kietz, J.-U. and Volz, R. (2000). Extracting a Domain-Specific Ontology from a Corporate Intranet, In *Proceedings of Learning Language in Logic Workshop (LLL-2000)*

Kietz, J.-U, Maedche, A. and Volz, R. (2000). A Method for Semi-Automatic Ontology Acquisition from a Corporate Intranet, In *Proceedings of Learning Language in Logic Workshop*

Levenshtein, I. (1966). Binary Codes capable of correcting deletions, insertions and reversals.. *Cybernetics and Control Theory*, 10(8):707-710

Litowski, K. (1978). Models of the semantic structure of dictionaries. *Journal of Computational Linguistics*, 81:25-74

Lyons, J. (1977). *Semantics*. Cambridge University Press:Cambridge

Maedche, A. and Staab, S.(2000), The TEXT-TO-ONTO Ontology Learning Environment*, Software Demonstration at ICCS-2000 - Eight International Conference on Conceptual Structures*

Maedche, A. (2002). *Ontology learning for the Semantic Web*. Kluwer Academic Publishing:Boston

Moldovan, D. and Mihalcea, R. (1999). Improving the Search on the Internet by Using WordNet and Lexical Operators. *Technical Report No. NLP-TR-99-09*, NLP Lab, Southern Methodist University

Nakaya, N., Kurematsu, M. and Yamaguchi, T. (2002). A Domain Ontology Development Environment Using a MRD and Text Corpus, In *Proceedings of Joint Conference on Knowledge Based Software Engineering*

Oberle, D., Volz, R., Staab, S. and Motik, B. (2004). An Extensible Ontology Software Environment. In Staab, S. and Studer, R. (eds.). *Handbook on Ontologies*. Springer Verlag:Heidelberg

Page, L. and Brin, S. (1998). The Anatomy of a Large-Scale Hypertextual Web Search Engine, In *Proceedings of the 7th Annual World Wide Web Conference*

Sanfilippo, A. and Poznanski, V. (1992). The Acquisition of Lexical Knowledge from Combined Machine-Readable Dictionary Sources, In *Proceedings of the 3rd Conference on Applied Natural Language Processing*

Shamsfard, M. and Barforoush, A. (2003). The State of the Art in Ontology Learning: A Framework for Comparison. *Knowledge Engineering Review*, 18(4):293-316

Staab, S., Braun, C., Bruder, I., Düsterhöft, A., Heuerr, A., Klettke, M., Neumann, G., Prager, B., Pretzel, J., Schnurr, H.-P., Studer, R., Uszkoreit, H. and Wrenger, B. (1999). Getess - Searching the Web Exploiting German Texts, In *Proceeding of the 3rd Workshop on Cooperative Information Agents*

Staab et al. (2002). KAON - Towards a Large Scale Semantic Web, In *Proceeding of EC-Web 2002*

White, J. (1988). Determination of Lexical-Semantic Relations For Multi-Lingual Terminology Structure. In Evens, M. (eds.). *Relational Models of The Lexicon - Representing Knowledge in Semantic Networks*. Cambridge University Press:Cambridge

White, J. (1991). Lexical and World Knowledge: Theoretical and Applied Viewpoints, In *ACL/SIGLEX Workshop on Lexical Semantics and Knowledge Representation*