Text-Mining for Semi-Automatic Thesaurus Enhancement

Diploma Thesis

presented by
Robert Meusel
Matriculation Number 1020135

Adviser: Kai Eckert

submitted to the
Lehrstuhl für Künstliche Intelligenz
Prof. Dr. Heiner Stuckenschmidt
University Mannheim

August 2009

Acknowledgment

I want to thank Kai Eckert for his competent supervision of my thesis and his constructive ideas and reviews which head my research onto a procreative trail. I also want to thank Prof. Dr. Heiner Stuckenschmidt for his conceptual inputs during the time of my work. Moreover I want to thank all co-workers at the Knowledge Representation and Knowledge Management Research Group of the University of Mannheim for their support and the pleasant atmosphere.

I thank all my friends and fellow students who have provided me with further suggestions. Especially I am indebted to my girl friend Claudia Janoske who was a great support in the last weeks and my friends Thomas Markotschi, Patrick Deege and Tobias Gummer for proofreading and decoding my thoughts.

Last but most important, I want to thank my parents Andrea and Axel Meusel for supporting me on all I did with great patience, words and deeds.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The ongoing dispersion of the internet and digital stored media press the research in the area of information retrieval ahead. Media indexing and retrieval, as it is provided by the popular web search engine companies *Google* and *Yahoo* has become also in companies more and more popular for internal use. These organisations try to support the knowledge management of their employees. This includes the creation and storing of meta data and in addition the providing of a retrieval system. Caused by the enormous costs of such processes, more and more automatism has been included, already in the past.

A structure which supports the automatic indexing and retrieval is a thesaurus. A thesaurus is a structured dictionary, which is focused on the representation of a limited set of semantic relations between different concepts. In the process of automatic document indexing the thesaurus can be used to deliver headwords. Moreover while retrieving documents this kind of dictionary could add additional information to improve the retrieval and by that quality and quantity of the results.

Although a thesaurus can be used to reduce the manual effort in the different information retrieval processes, it still need to be constructed and maintained manually, which causes a lot of costs.

First investigations in automating or even supporting the construction of thesauri by computers were made by Grefenstette in 1994 (cf. [12]), who tried to explore thesaurus structures in raw texts. Since this time the research have been advanced and in 2007 Bellegala et al. discovered a method to extract semantic structures with the help of web search engines (cf. [3]). Additionally these approaches help to maintain and improve already existing structures of thesauri if these structures are over- or under-determined.

Detecting weaknesses in existing thesauri is one of the main goals of the *Semtinel* which was implemented by Eckert in 2007. Eckert discovered a method, called *IC Difference Analysis*, which is able to identify over- and under-determined substructures in the thesaurus according to a document corpus. This statistical analysis calculates based on the frequency of different index terms the allocation in the various levels of a thesaurus. Generally, the allocation of frequency should decrease from common concepts to more specific concepts (cf. [18] and [9]).

Corresponding to the problem detection of Eckert, the problem addressed in this thesis is, given a (possibly empty) thesaurus and a set of documents, which have a similar topic, the enhancement of this thesaurus by extracting relevant concepts out of the documents and using similarity and clustering methods to find locations in the existing structures where these candidates fit most. This enhancing process is supported by *Thencer*, which is a extension to *Semtinel* and was implemented during this research. *Thencer* is designed to solve under-determination weaknesses by adding relevant concepts and improving the richness of detail in the thesaurus.

Giving a short introduction into the topic the main idea of information retrieval and the used techniques are described in the chapter 2. Additionally former and ongoing research according to the different topics are described briefly. At the end of this chapter the main questions which are answered by this thesis are listed and described. Chapter 3 is concerned with the description of the complete method which is evaluated in this thesis. Additionally specific aspects according to the implementation of the *Thencer* tool are discussed. The experiments and their setup, which are done to evaluate the approach, are described in chapter 4. Additionally the evaluation strategy is explained. Chapter 5 and 6, include the results of the experiments mentioned in chapter 4 and the resulting answers to the questions of chapter 2. Issues, which were identified in chapter 5 will be discussed in the last chapter of this thesis. Moreover different solutions are explained briefly.

Chapter 2

Theoretical Foundation

This chapter provides an overview of the topic of information retrieval in general. Moreover the theoretical background which is necessary to understand the different approaches presented in this chapter is explained briefly. Section 2.4 and 2.5 give an introduction into research done in terms of relation extraction and thesaurus construction which are related to this thesis. They are summarised in section 2.6 where additionally the methods used in the approach of this thesis are listed. Corresponding to this approach a detailed problem description and the central questions which are relevant are included in the last section.

2.1 Information Retrieval

2.1.1 Overview and Definition

Information Retrieval (IR) is generally defined as the gathering of items which satisfy the need of information of the user. In general this means that IR does not claim to answer specific questions but to retrieve documents which include the answer for this question.

The idea of IR, as we understand it today, was first presented in the article *As We May Think* by Vannevar Bush and the first systems on computers searching for relevant information came up in the 1950s. By the dispersion of the internet and the thereby enormous increase of published and available documents, efficient and automatic methods for IR become necessary. To handle this amount of items more and more automation was included in the process of indexing and retrieval. In contrast to the former library systems, where documents were indexed by the librarians themselves, automatic indexing systems and search engines were created which do not require a logical based query language. Today most of the popular web search engines use techniques like term frequency weighting approaches (tf-idf) to extract important terms out of documents and vector space approaches to rank the results according to their relevance for the query. Detailed information about this topic can also be found in C. Manning et al.'s book *Introduction to Information Retrieval* [22].

2.1.2 Retrieval Evaluation

Coming up with always new approaches to improve indexing and retrieval for digital stored information methods are needed to define the quality of the approaches. Although there are different points of view, which could be included in the evaluation of an IR approach this thesis mainly focus on the correctness, which is in this case the accuracy or precision and recall of a method and the runtime. An overview about possible other evaluation aspects is shown in table 2.1 which is an extract out of *Introduction to modern information retrieval* from Salton et al. (cf. [28]).

Precision and Recall

Precision P and recall R are two measurements, that give a hint about the quality of the retrieved answers or items a document search presents. In this raw form they are used to evaluate unranked retrievals. The precision is defined as the number of items, that are returned by the search engine and include relevant information for the search query divided by the total number of items returned (cf. equation 2.1 where I is a set of items. I_{rel} is the set of the items that is returned by the search engine and I_{ret} the items that are relevant for a query.).

$$P = \frac{|I_{ret,rel}|}{|I_{ret}|} \tag{2.1}$$

In comparison, the recall is defined as the number of items that are returned by the search engine and include relevant information for the search query divided by the total number of relevant items in the collection.

$$R = \frac{|I_{ret,rel}|}{|I_{rel}|} \tag{2.2}$$

The aim of a search engine is to maximize both values and also for the evaluation of a search engines it is important to have a look on both

User criteria	Selected related parameters
Recall and precision	Indexing exhaustivity (the more exhaustive, the better the recall)
	Specificity of indexing language (the more specific, the better the precision)
	Provisions indexing language of improving recall (synonym recognition, recognition of term relations, etc.
	Provisions indexing language for improving precision (use term weights, use of term phrases)
Response time	Type of storage device and storage organization
	Query type
	Location of information centre
User effort	Characteristics of device permitting access
	Location of accessing and storage devices
Form of presentation	Type of accessing and display device
	Size of stored information file
	Type of output
Collection coverage	Type of input device and type and size of storage device
	Ease of content analysis
	Demand for service (the demand increases with greater coverage)

Table 2.1: User Performance Criteria and Related Parameters

measures. To give a short example: If there is a query and only one item is returned in total that is by chance also relevant for the query, than P would be 1. But if there are like 10000 relevant items in the total collection R would just be $\frac{1}{10000}$ which is absolutely no sign for a high quality search engine. The other way around, when the search engines returns always all items out of the collection R is be 1. But when only $\frac{1}{100}$ of all items are relevant P would also be only $\frac{1}{100}$. Moreover it is important to mention that for the recall it is necessary to know the total number of all relevant items. This could be an issue in a library as well as in the internet. Mostly when new approaches are tested and evaluated a standard corpus with preevaluated queries is used.

2.1.3 Query Expansion

One way to raise the correctness of a search engine a technique called *query expansion* is used. The general idea behind this is to attach additional concepts to the search query of the user[1]. These attached concepts could be user specific information in case of a personalized search. For example it is known that the user only uses the search to find Java libraries and with this technique it is possible to attach to all queries of the user the concept *Java library* to improve the quality of the highest ranked results for this specific user. Another worthwhile utilization of this technique is the expansion of the query with synonyms[2], hypernyms[3] or hyponyms[4]. Such words are provided by a thesaurus.

2.1.4 Thesaurus

Overview and Definition

A *thesaurus* (old Greek *thesaurós*, meaning treasure or treasury) is like a dictionary, but not in the conventional sense. In a *thesaurus*, concept descriptions, plurals etc. can be found rarely, but synonyms for concepts, broader and narrower concepts. A thesaurus has a kind of a tree structure with well-defined relations (cf. table 2.2) between the entries [5].

In addition to these relations sometimes a *scope note (SN)* is added to a concept. This scope note includes a detailed description about how this concept should be used or in which way it is defined. Also a scope note could be added if there have been potential problems in the past to avoid complications in the future. A possibility to structure a thesaurus more clearly is usage of *node labels*. These nodes are no own concepts in the thesaurus but they are included to help organising and building up the structure of the thesaurus. A brief example of a thesaurus concept is shown in table 2.3. This table shows three different concepts *nose*, *nose rings* and *smeller* and their semantic relations.

In addition to the usage of query expansion the concepts of thesauri are used to provide a standard and uniform collection of words which could be

[1]This technique is effective if the search engine does not use the logical expression *and* between the single tokens of the search phrase and ranks the returned items with the help of a vector space distance approach. Most of the web search engines use these techniques.

[2]A synonym is another concept for an existing one which has the same meaning.

[3]A hypernym is a superordinated concept for a concept with extended content.

[4]A hyponym is a sub concept of a concept and is in general more precise.

relation	description
USE	A reference from a non-preferred term to a preferred term
UF	Use For: a reference from a preferred term to a non-preferred
BT	Broader Term: a reference to terms which are more general in scope
NT	Narrower term: a reference to term which are more specific in scope
RT	Related term: a reference to a term which is related in some way other than BT or NT

Table 2.2: Thesaurus relationships

nose	
BT	face
RT	nose ringing
RT	nose rings
UF	smeller
UF	nostrils
SN	smelling organ of animate being
nose rings	
BT	handling equipment
RT	nose
RT	nose rings
smeller	
USE	nose

Table 2.3: Thesaurus extract

used to index items. This limits the variety of stored meta data[5] and makes the manual retrieval easier. Additionally thesauri can be used for simple browsing activities to learn more about a specific topic.

[5]One of the most popular meta data standards, developed in the last 10 years is *Dublin Core (DC)*

Thesaurus Classification

During the development of information retrieval, including the rise of web search engines and the storage of meta data different kinds of dictionaries were used to improve information retrieval approaches. One of this is the so called *controlled vocabulary*. A *controlled vocabulary* contains the normalized forms of all the words that are important for a specific domain. These collections of technical and well-defined concepts are normally created by specialists.

A dictionary which includes more semantic relations between the concepts is called an ontology. An *ontology* contains (abstract) concepts and their dependencies that try to formally represent the knowledge of a specific domain. It is primarily used for reasoning or inference and thus often contains fundamental information. A small graphical shortcut out of an existing ontology about wine and its background can be found in figure 2.1[6].

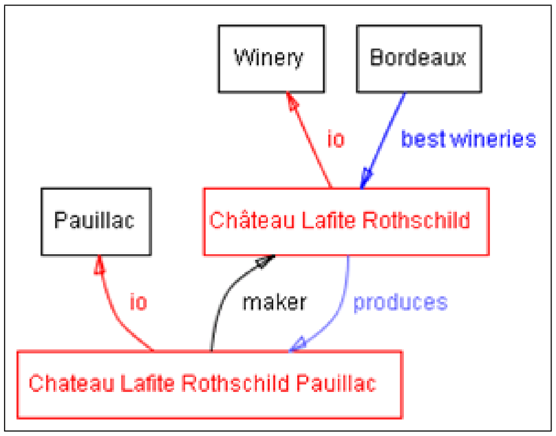


Figure 2.1: Graphical extract out of a wine ontology

[6]The textual description of this thesaurus can be found under http://oaei.ontologymatching.org/tests/102/onto.html

2.2 Relevance

In the beginning of the process of enhancing a thesaurus it is primarily necessary to find concepts which are missing and need to be included into the thesaurus or in other words which are relevant for the thesaurus. The definition of *relevance* given by [15] explains the context more clearly.

Something (A) is relevant to a task (T) if it increases the likelihood of accomplishing the goal (G), which is implied by T.[7]

In this specific case T is the thesaurus itself and G in general are all goals the thesaurus is required for in the context of information retrieval. A is every concept which increases the capability of the thesaurus to improve all processes of information retrieval where it is used.

To extract concepts for a thesaurus it is primarily necessary to find documents which are relevant for the thesaurus. The likelihood of these documents to include thesaurus relevant concepts is obviously higher than taking non relevant documents. The issues that occur while searching and identifying a possible document corpus for a thesaurus are described in the journal article *Problem der Textauswahl für einen elektronischen Thesaurus* of Bergmann which was published in December 2003 [2] in respective of the creation of a German thesaurus out of German dictionaries from different eras.

The thesaurus could be improved by adding words from the document corpus. But obviously not every word that appears in the corpus has the same relevance for the thesaurus. A ranking method is needed to identify the most relevant concepts out of the documents.

2.2.1 Term Frequency, Document Frequency and Stopwords

By making the assumption, that the used document corpus and the used thesaurus have the same goal or in other words they are concerned with the same topic, it is possible to reduce the problem of the relevance of the document corpus.

The most often used factors to decide about the relevance of concepts according to the whole document are the document and the term frequency. Caused by the simplicity of these factors they are so popular when talking about concept weighting.

[7]cf. [15]

Document frequency or short df is defined as the number of documents the same concept or term occur in. Here it does not matter if in one document the term appears once or twenty times. The document frequency is a term-specific factor and mostly indicated as df_t where t is a token. Terms with a high document frequency mostly indicate that they are general. Terms whose document frequency is (almost) equal to the total number of documents are mostly stopwords.

Term frequency or short tf means the number of occurrences of the same concept or term in a document. So the term frequency $tf_{d,t}$ is given for a specific document i and a specific term t. Having a high term frequency in a document indicates that this term is very important and that is the reason why the author always repeats it. But this term could also be just a filling word like *and*. This factor alone does not help enough.

Stopwords or stopword lists are used to diminish the difficulty coming out of the problems discussed in the two former paragraphs. A stopword is a term which does not have any topic-specific relevance but appears really often in a document and a corpus. Stopwords are language dependent and may differ from one topic to the other. Some examples for the English language are *about, a, of, the*[8].

2.2.2 TF-IDF Weighting

tf-idf (*term frequency - inverse document frequency*) weighting is a method to calculate the relevance of a term according to a document. In this case the goal of the document is to inform the reader. The most relevant term is the one which informs the reader most. This measure combines the term and document frequency to calculate a unique relevance value or weight for a concept in a document. This method of relevance calculation is used in different variations by most of the search engine companies and libraries. The measure assumes that the importance of a term in a document corresponds to the number of appearances in this document as well in all documents of the whole corpus. A term is more important than another if the term has more occurrences in the document than another. But the importance decreases if a term occurs in more documents of the corpus. Following this

[8]A complete list of English stopwords including 429 words can be found under http://www.lextek.com/manuals/onix/stopwords1.html. German stopwords can be found under http://feya.solariz.de/wp-content/uploads/stopwords.txt.

assumption the term frequency is combined with the inverse of the document frequency by multiplication. The great achievement of this measure in text indexing is that *stopwordlists*, which are common to eliminate non-context relevant words, are not necessary any more. As stated above there are different alternatives to calculate the tf-idf relevance value of a term in a document. Mostly these alternatives differ in the way the tf- and *df*-component are modified to calculate the total value. In 1988 Salton et al. [27] already made different experiences with the variation of modifications for document and query weighting. They calculated that there are at this time more than 1800 different possibilities to calculate the relevance value of a term but they only picked 287, which they thought were distinct. Some of the possible modifications are shown in table 2.4 being an extract out of [27].

The outcome of their experience (cf. equation 2.3 where i is a term in the current document) is that the best way to calculate the relevance weight for a document is the fully weighted system tfc:

$$\frac{tf \cdot log\frac{N}{n}}{\sqrt{\sum(tf_i \cdot log\frac{N}{n_j})^2}} \tag{2.3}$$

The most sophisticated way to calculate the relevance weight for a query turns out to be the nfx:

$$(0.5 + \frac{0.5tf}{max(tf)}) \cdot log\frac{N}{n} \tag{2.4}$$

2.3 Word Similarity

After identification of possible candidates which have a high relevance for the thesaurus a measurement is needed to determine concepts that are similar to these candidates. This does not mean to identify *synonyms* because they should have a similar meaning but to identify concepts which have something to do with each other.

In 1998 Dekan Lin [21] has presented a universal definition for the similarity of terms. Similarity in this case does not mean that two terms have to look similar like 'beer' (German 'Bier') and 'bear' (German 'Baer'). The similarity mentioned by Lin is concerned with the similar meaning of two concepts. His definition out of [21] includes the following three intuitions:

Term Frequency Component		
b	1.0	binary weight equal to 1 for terms present in a vector (term frequency is ignored)
t	tf	raw term frequency (number of times a term occurs in a document or query text)
n	$0.5 + 0.5\frac{tf}{max(tf)}$	augmented normalized term frequency (tf factor normalized by maximum tf in the vector, and further normalized to lie between 0.5 and 1.0)
Collection Frequency Component		
x	1.0	no change in weigh; use original term frequency component (b, t, or n)
f	$log\frac{N}{n}$	multiply original tf factor by an inverse collection frequency factor (N is total number of documents in collection, and n is number of documents to which a term is assigned (same as df))
p	$log\frac{N-n}{n}$	multiply tf factor by a probabilistic inverse collection frequency factor
Normalization Component		
x	1.0	no change; use factors derived from term frequency and collection frequency only
c	$\frac{1}{\sqrt{\sum w_i^2}}$	use cosine normalization where each term weight w is divided by a factor representing Euclidean vector length

Table 2.4: Term-weighting components

- Intuition 1: The similarity between A and B is related to their commonality. The more commonality they share, the more similar they are.

- Intuition 2: The similarity between A and B is related to the differences between them. The more differences they have, the less similar they are.

- Intuition 3: The maximum similarity between A and B is reached when A and B are identical, no matter how much commonality they share.

As the similarity theorem he uses the equation 2.5 where the similarity between two concepts is the division of the ratio of between the amount of information, which are needed to describe both concepts in common ($P(common(A, B))$) and the amount of information which are needed to describe the specific characteristics of each concept ($P(description(A, B))$) (cf. [21]).

$$sim(A, B) = \frac{logP(common(A, B))}{logP(description(A, B))} \tag{2.5}$$

This definition was also used by Lin in [20] to measure automatically constructed thesaurus and compare them to handmade one, like *WordNet* or *Roget Thesaurus*.

The most popular *similarity measures* are based on this equation and are briefly presented and discussed in the next section.

2.3.1 Similarity Measures

In the following definitions the variables A and B represent sets of items as normally used in math. The intersection out of both sets includes only the items, which are in set A and also in set B. The union of both sets include all items of A and all of B.

The variables P and Q are search engine phrases. These phrases could consist out of a single concept or a combination out of more than one concept. Assuming that P consists out of the concept *fracture* and Q consists out of the concept *heart* and the concept *disease* the resulting phrase out of intersection $P \cap Q$ consists out of the concept *fracture*, *heart* and *disease*[9]. $H(P)$ represents the number of pages returned from the search engine for the phrase P.

The Jaccard index is also known as *Jaccard similarity coefficient* and is an index of the similarity of two sets of samples. The Jaccard index J is defined as shown in the following equation where A and B are sets of

[9]Most of the most popular web search engines, like *Google* and *Yahoo* use a *or* conjunction between all words in the search phrase

samples and the $J(A, B)$ is the intersection divided by the union of both sets.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.6}$$

The modification which was done by [3] to adjust this measure to the situation of a web search engine and the delivered page count is the *WebJaccard* measure for the two concepts P and Q.

$$WebJaccard(P, Q) = \begin{cases} 0, & \text{if } H(P \cap Q) < c \\ \frac{H(P \cap Q)}{H(P)+H(Q)-H(P \cap Q)}, & \text{otherwise} \end{cases} \tag{2.7}$$

In this equation $H(P \cap Q)$ is the page count of the combined phrase out of P and Q. In addition Bollegala et al. (cf. [3]) added the condition $0, \text{if } H(P \cap Q) < c$ to reduce the effect of random co-occurrence when the page count of the combination of both words / phrases is too small.

The Dice coefficient like the *Jaccard index* is used to measure the similarity between two sets of samples.

$$dice(A, B) = \frac{2|A \cap B|}{|A| + |B|} \tag{2.8}$$

In comparison to *Jaccard* the amount of the intersection is doubled and divided by the sum of the amounts of both single sets. Adapted for the web search engine Bollegala et al. defined the *WebDice* as the doubled page count of the combination of the concepts $P \cap Q$ divided by the sum of the page counts of each single concepts.

$$WebDice(P, Q) = \begin{cases} 0, & \text{if } H(P \cap Q) < c \\ \frac{2H(P \cap Q)}{H(P)+H(Q)}, & \text{otherwise} \end{cases} \tag{2.9}$$

Again there is a case differentiation to reduce the site effects of random co-occurrence.

Overlap (Simpson) coefficient is preferential not a real way to measure the similarity of two sets of samples.

$$overlap(A, B) = \frac{|A \cap B|}{min(|A|, |B|)} \tag{2.10}$$

It is sooner a indicator of how many samples of set B are also covered by set A. This is also the weakness of the *Overlap* because it is maximal if A is a subset of B, no matter if $|B| >> |A|$ or $|B| = |A|$. The adaptation done for the web search engines is similar to the one done for the other coefficients. Again there is a case differentiation and the amounts are replaced by the page counts. This whole adaptation makes this measure suitable for calculating similarity using page counts because there are no real subsets.

$$WebOverlap(P, Q) = \begin{cases} 0, & \text{if } H(P \cap Q) < c \\ \frac{H(P \cap Q)}{min(H(P), H(Q))}, & \text{otherwise} \end{cases} \tag{2.11}$$

The Point-wise mutual information (PMI) is like the three one above also a measure of similarity or association of two outcomes.

$$PMI(A, B) = \log \frac{p(A, B)}{p(A)p(B)} \tag{2.12}$$

In this equation $p(A)$ is the probability of the coincidence of A alone and $p(A, B)$ is the probability of the coincidence of the joint of the intersection of A and B. Bollegala et al. defines the probability p for the usage of page counts.

$$p(A) = \frac{H(A)}{N} \tag{2.13}$$

Here N is the total number of pages indexed[10] by the used search engine. The resulting equation for the *WebPMI* including the case differentiation looks as followed:

$$WebPMI(P, Q) = \begin{cases} 0, & \text{if } H(P \cap Q) < c \\ \log(\frac{\frac{H(P \cap Q)}{N}}{\frac{(H(P)}{N}\frac{H(Q)}{N}}), & \text{otherwise} \end{cases} \tag{2.14}$$

[10]In their work Bollegala et al. assumed that *Google.com* has indexed $N = 10^{10}$ [3]

2.3.2 Machine Learning

Another possible method to calculate the similarity of concepts is provided by clustering methods. Using such methods the clusters could be divided into similarity and non - similarity clusters. Moreover the concepts of similarity could be split into smaller sub clusters like *synonyms* or *hyponyms*. Although these methods could be used to learn several different relations by its own, in the case of this thesis where a limited amount of relations has to be extracted a special kind of clustering methods is interessting. Machine learning does not create primarily clusters on its own but learns how different attributes and their parameter values fit into preset clusters.

Generally machine learning handles objects which are represented by several parameters and their significant values and categorized depending on the learned model of the approach. To create a model for an machine learning approach, a set of training data, which includes beside the specific value parameters the category or cluster the object belongs to. Two different approaches, the supported vector machine and the decision tree are presented in the following two paragraphs.

Support Vector Machine (SVM) commonly uses the input training data, which are represented as vectors to create a separating hyperplane between the possible input categories. The kind of the hyperplane could be defined by the user and is mostly either linear or non-linear. In the second case mostly the radial basic function, shortly RBF is used. During the phase of training every training vector is seen as a constraint. The SVM tries to calculate a hyperplane between the categorized vectors without violating one of these constrains. If this is possible and more than one hyperplane could be calculated, the hyperplane with the highest sum of the smallest distances of every vector to the hyperplane is chosen for the final separation. If the training data could not be strictly separated, a hyperplane is chosen, where least constraints are violated.

A small graphical example of a linear and a non-linear hyperplane which were created using the same training vectors is shown in figure 2.2 where the violated constraints are marked red.

More information about how to adjust a SVM especially when trying to recognize patterns can be found in the book *Kernel Methods for Pattern Analysis* by Shawe-Taylor and Christianini [29].

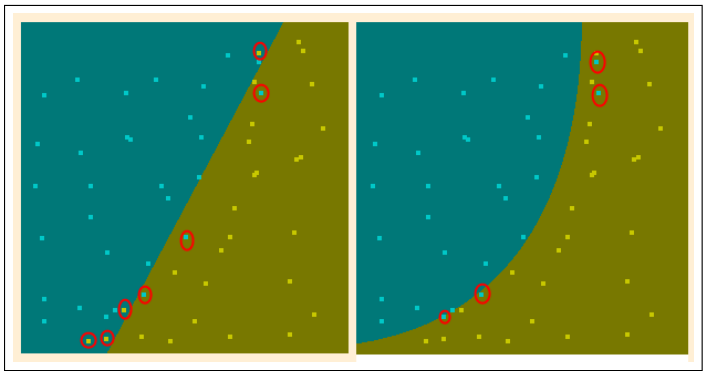


Figure 2.2: SVM: linear and non-linear hyperplane

Decision Trees are the second approach used in this thesis in terms of categorization. These trees do not only appear in machine learning approaches but also in economics when deciding whether an investigation should be done or not. The nodes in the tree are events and the leafs represent the final decision according to the outcomes of the overlying events.

Decision tree approaches in general calculate based on a set of training data the combinations of parameters and parameter values which lead to the different possible decision. Based on this calculation a decision tree model is created. An example of a decision tree model is given in figure 2.3. More information including results from different studies and ways of modification can be found in the article of J. R. Quinlan in the journal *Machine Learning* of 1986 [25].

2.4 Relation Extraction

Beside the calculation of similarity between concepts which only gives hints if two concepts have a similar meaning the relation between those concepts have to be identified. Regarding first of all, a candidate and a concept in a thesaurus that are similar, it is the question if these two concepts have a synonym or a genus/species relation. Regarding ontologies the number of possible relations is higher.

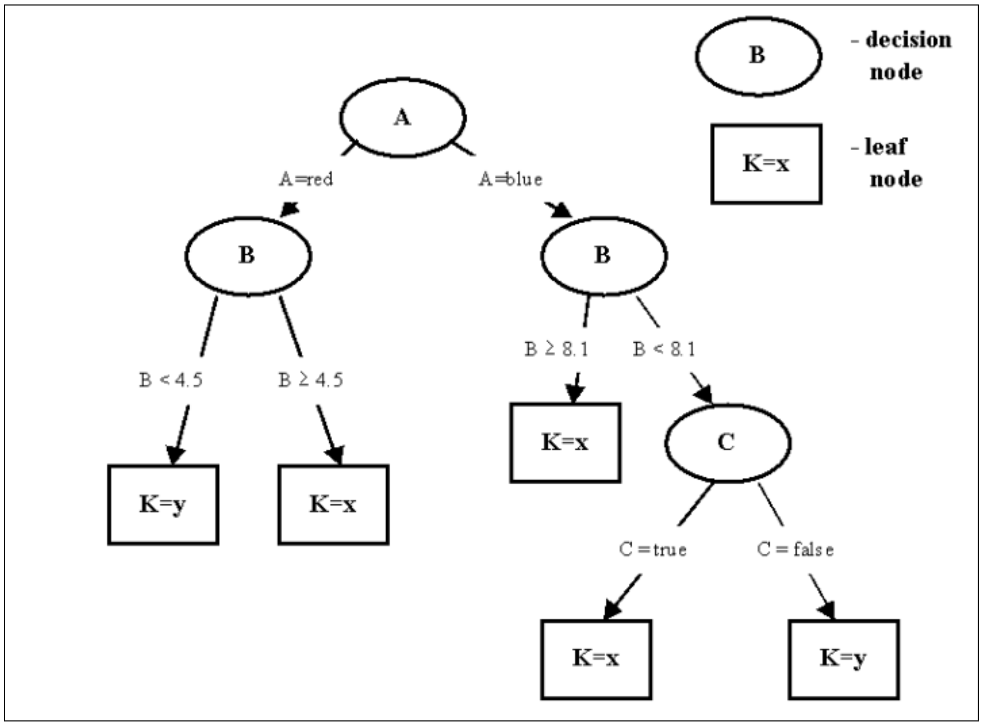


Figure 2.3: Decision Tree Model

2.4.1 The Approach of Roussinov

An approach, which gives a good example how the different similarity measures and methods could be used to identify similar words and what kind of attributes can be taken into account to represent the items, is described in [26]. The approach of Roussinov is concerned with the *textual classification* and relation extraction using the neural network technique self-organizing map (SOM), which was first discovered by Kohonen [19] to categorize concepts and define their relations between each other.

The SOM approach maps vectors from an input layer to a mapping (output) layer. The values of the dimensions of these vectors are occurrences of attributes which are characteristic for the objects the vectors represent. The output layer is initialized with random numbers. Then, a winning node, the node with the smallest Euclidean distance between mapping and input vector is calculated. Winning input and mapping vector are adjusted to reduce the Euclidean distance and also the vectors in the neighbourhood are adjusted proportionally. So a multi-dimensional input network could be represented by a two-dimensional map [19].

In the approach of Roussinov the vectors represent terms and the attributes are the documents where the terms occur in. The amount of dimensions is equal to the number of the included documents in the corpus. A vector has the parameter value of one in a dimension when the term, the vector represents occurs in a document. The value is zero if the term does not occur in the document. Interesting to mention is the fact, that Roussinov improves the algorithm to reduce the complexity and to make the clustering more scalable.

2.4.2 The Approach of Pantel and Lin

Another approach which uses also several clustering methods to explore the sense of a word is the one of Pantel and Lin [24] published in 2002. In contrast to Roussinov they do not take the documents as attributes for the input vectors but the content in which a word occurs. The main idea behind this assumption is based on the *Distributional Hypothesis* of Harris [14] which in general assumes that terms which *occur in the same contexts tend to have similar meanings* [14]. This idea is also used by Grefenstette to construct a thesaurus from the scratch (cf. section 2.5.1).

Pantel and Lin created in 2002 a clustering method called *CBC* which uses different clustering techniques to assign a term to multiple clusters. Each cluster represents in this case a sense. Rawly they calculate for every word similarity attributes. In a second step they form out of the highest ranked attributes or similar-elements, how they call them, clusters (committees) and assign the terms corresponding to their attributes to these committees.

2.4.3 The Approach of Nguyen

Another possibility how different relation between concepts could be extracted out of documents was discovered by the group around Nguyen [23]. They use the semi-structured Wikipedia[11] to extract several relations out of the articles. In a first step they had to face the problem that although Wikipedia has a kind of a structure for articles it does not have an application programming interface (API) to access it directly from computer programs. This fact makes the extraction for Nguyen et al. bit more complicated. Another problem they had to face was the uncontrolled structure of this encyclopaedia and the resulting partly incorrect usage of different fields in the article (e.g. Definition, Introduction, etc.). After resolving these problems Nguyen and his team detect main entities in the English Wikipedia and extracted relations out of the summary part of the encyclopaedia. With these extracted relations they build up data to learn patterns which they weight and use to categorize a relation.

The interesting fact of this research regarding this thesis is not the extraction from Wikipedia data but the extraction of relations and patterns only from the summaries, which are condensed shortcuts of the whole topic. Although only these rather small textual parts are used the results are sufficient.

2.4.4 The Approach of Bollegala

In respect of relation extraction a relative new idea by Bollegala et al. [3] which was first presented at the 16th World Wide Web Conference in 2007 is to mention. Bollegala et al. use web search engines, e.g. Google.com to gather information about the combination of two concepts, whose relation they want to analyse. In a first step, the amount of returned pages - the page count - of the search engine for the single concepts and the combination of both is collected and used as input for four different similarity measures, such as Jaccard, Overlap (Simpson), Dice and PMI (point-wise mutual information) (cf. chapter 2.3.1). In a second step, the small pieces of text - the snippets - the search engines returns for every hit are analysed. Bollegala et al. replace the searched concepts in the snippets with variables (concept one - X, concept two - Y) and extracts on that way patterns (e.g. ‘X is Y’). Before they run the final method the most significant patterns for synonyms

[11]Wikipedia is a free encyclopaedia which is kept alive by the users of the internet. The concept of this encyclopaedia is that the users are creating the entries themselves and also correcting and maintaining them. Wikipedia is available in different languages. The main site is `www.wikipedia.org`

are detected by using 5000 synonyms from WordNet. The 200 patterns for synonyms, which have the most information content, are used to create 200 more inputs for the final similarity measuring. Out of these 200 values and the 4 measures out of step one a vector is created which is the input for a trained two-class supported vector machine (SVM). The SVM is trained with extracted synonyms out of WordNet and self-created non-synonyms. A trained SVM could be used to categorize input vectors into the different clusters. The improvement of this method is quite obvious. For calculating the similarity of two concepts only a web search engine is needed. This approach uses less own resources and makes use of the whole information provided by the web. Caused by the huge amount of information examined, the ill-defined concepts which are extracted do not play a significant role.

2.5 Thesaurus/Ontology Construction

Having all the tools and methods to identify relevant concepts, calculate similarities between them and extract relations just a small step is missing to start constructing (semi-)automatically thesauri and ontologies with the collected information. One of the first approaches which leads into this direction was done be Grefenstette.

2.5.1 The Approach of Grefenstette

G. Grefenstette presented in 1994 [12] a system, called *Sextant* to gather words from rare texts and calculate similarities between the words to construct a complete thesaurus from the scratch. Grefenstette abdicate all hand-build and non-renewable meta information such as domain-dependent knowledge-structures or human-oriented dictionaries. In a first step *Sextant* identifies concepts out of a raw text corpus by looking up the words in a dictionary and adding the deposited grammatical category. If a concept has more than one grammatical category Grefenstette uses the frequency of the different grammatical categories of this concept in the *Brown corpus* to select one single for his corpus. For each single noun concept the tool evaluates the relations to other grammatical categories which occur with this word and calculates attributes for every concept. Finally the *Sextant* calculates for every pair of concepts the similarity with the help of a weighted Jaccard measure which is described in equation 2.15 where *ua* stands for unique attributes.

$$\frac{\sum_{ua} \min\left(weight(object_m, attribute), weight(object_n, attribute)\right)}{\sum_{ua} \max\left(weight(object_m, attribute), weight(object_n, attribute)\right)} \tag{2.15}$$

Evaluating his results, Grefenstette found out, that *knowledge-poor techniques such as the selective natural language processing of Sextant could extract semantic similar words from raw texts* (cit. [11]). In all his evaluations, the semantic pairs of concepts, extracted by *Sextant* can also be found in thesauri which were hand-made out of the same corpus [11].

In a separate experiment, Grefenstette enriched the hand-made thesaurus WordNet with the help of the information gathered by *Sextant*. The problem, at what level of hyponym should the newly discovered concept be placed, is solved by calculating the most similar word in the existing thesaurus, and comparing the concept with all sub levels of the most similar concept, adding it to the most fitting one. Unfortunately there is no case discrimination mentioned, if the new concept is probably only a synonym of the most similar one in the thesaurus and also no case if the new concept is a completely new leaf in the tree. Grefenstette assumes that the basic structure, meaning all sub trees is already discovered. (cf. [12] page 114)

2.5.2 The Approach of Kageura

Another approach when talking about the automatic construction of thesauri was made by Kageura et al. in 2000 [17]. This approach takes parallel or comparable input corpora in two different languages as origin for their construction and tries to construct a bilingual thesaurus for technical translations.

In a first step the different corpora are analysed separately and relevant concepts in both languages are extracted with the help of morphological and lexical methods. In a second step, Kageura et al. use a translation module, which also uses statistical weighting methods to delimit the varieties of translation pairs. In a last step a bilingual concept graph is created and by using graph theoretic approaches proper clusters are generated.

The performance and the outcome of this method is described in [17] as *fairly good* and lead to the assumption that a translation of concepts could cause better results.

2.5.3 Ontology Construction

Another field where research of automatic generation is continued is the construction of ontologies. Because ontologies include more different types of relations and information in comparison to thesauri, ontologies have a larger range of operational area but are also even more complicate to create and maintain.

All three following papers [8] [13] [10] are concerned with the (semi-)automatic creation of ontologies from raw text corpora, which are domain specific. The commonality of all three approaches is the usage of similarity measures or clustering approaches to learn relations and categorize them into learned clusters. Gillam et al. use a combination of statistical and linguistic approaches to identify possible relations between concepts or terms out of the documents. In this case it is to mention that the relations were not predefined by the researchers.

With another alignment Brank et al. [4] use machine learning techniques and similarity measures to predict extensions of hierarchies in ontologies. Especially they analyse how it is possible to predict changes in structures, to add new subcategories to concepts, to extend the ontology and make it more suitable for the future. The difference in respect to thesauri is that ontologies are divided into a structural description and instances of nodes of this description. But out of these circumstances also a stronger need arose to adapt an ontology because it is more flexible as a thesaurus. To adapt this changes quickly, they observed most of the changes which happened in the past and tried to learn with machine learning out of the change of instances how the change influence the structural framework. This approach is interesting for this thesis because it could give a hint how the detection of under-determined concepts and sub trees in the thesaurus could be predicted and through this the thesaurus could be adjusted earlier.

2.6 Summary and Approaches Used

In the sections above, the necessary and most used methods concerning the creation of a thesaurus were described. Summarizing these methods it is remarkable that on the one hand co-occurrence methods based on the *Distributional Hypothesis* by Harris [14] are used to calculate similarity between concepts. Moreover, by regarding the extraction of specific relations out of texts it is almost impossible to achieve a sufficient goal without using a clustering or machine learning approach. Beside this, none of the known clustering methods turned out to be the most sufficient. Although different groups and research focus in the direction of automatic construction of thesauri or ontologies it emerged that none of the actual approaches could do it without the human interaction or an existing structure to orientate on.

These facts lead this thesis, which has the demand to reduce the human interaction during the process of thesaurus enhancement to a minimum, to the decision to use the following approaches. To identify the most relevant concepts out of the corpus a approach based on the tf-idf weighting will be used, because this approach combines the most significant and measurable factors of occurrence of concepts and terms. In a second step a co-occurrence approach, similar to the one of Grefenstette and his *Sextant* will be implemented into the whole process to calculate similar concepts of the index and the thesaurus. This method is used because it has a almost static runtime and once the attribute of a term is calculated the effort of calculating the similarity to others is minimal. Moreover, caused by the results from [23] which lead to the assumption that if a summary of a topic is enough to extract relations from it, it would also provide enough information about the relevant concepts which are included to calculate the similarity. In a final step the relations between two concepts which were earlier just identified as similarity are categorized with the help of a clustering approach. In addition the attributes of the concepts, which are extracted out of the abstract, are not used as input for the clustering method. Different patterns which are identified with the help of a web search engines, similar to the approach of [3] are collected.

2.7 Problem Description and Central Questions

The problem addressed in this thesis is, given a (possibly empty) thesaurus and a set of documents the classification of candidates, which are extracted out of these documents into the thesaurus at a relevant position to enhance the thesaurus. The relevant position in this thesis means that the candidate is assigned to a concept where it is a synonym, a hyponym or a hypernym. A *candidate* in this context is defined as a domain relevant concept out of the documents which is not already included in the thesaurus. Concepts of the thesaurus which have a high similarity to the candidate are called *possible adding locations*. To calculate the similarity according to the co-occurrence approach which is used it is necessary to assign the co-occurrence attributes of a candidate, which are terms appearing more often in the surrounding of the candidate. Caused by this circumstance in the following, these attributes are called the *surrounding* of a concept.

The central questions which arise out of the problem description, the primary goal of this thesis and the chosen approaches out of the section above are listed below:

1. Could the tf-idf weighting method or a modification be used to identify and rank possible concepts out of a document corpus with the goal to enhance an existing thesaurus?
2. Are the information provided by an abstract of a document enough to calculate the similarity between two concepts using a co-occurrence approach?
3. Is the method of pattern recognition out of web search engines snippets also suitable for specific thesauri and could the analysis which is done by a supported vector machine decide between subclass and synonym?
4. Is the whole process adaptable for real world organisations to reduce the costs of maintaining their thesauri?

Chapter 3

Design and Implementation

Having defined the problem and the central questions of this thesis in chapter 2, where also the theoretical foundations are described which are relevant, in section 3.1 of the current chapter the approach which is used to achieve the goal of this thesis is explained. Ongoing, aspect of the implementation of *Thencer* are stated in section 3.2.

3.1 Thesaurus Enhancement Method

The method which is used in this thesis and is evaluated is a combination out of the different approaches which were mentioned in the section 2.6 and own considerations.

In a first step the document corpus is analysed with the help of the tf-idf weighting method to extract relevant candidates for the thesaurus. These candidates are ranked according to their global relevance for the current issue which is done with separated methods. These methods are discussed in chapter 5 according to their practicability. For each candidate, possible adding locations are identified with a co-occurrence approach, similar to the approach in [12] but with less effort in analysing and creating the surrounding of the different concepts. The final categorization of the different pairs and their relation toward each other is done by two different machine learning approaches in parallel. The input vectors for these clustering approaches are created by extracting several patterns out of a web search engine, similar to the approach in [3].

3.1.1 Identification of Relevant Concepts with TF-IDF

According to the studies of Salton et al. (cf. section 2.2.2) this thesis use a modification of the tf-idf weighting approach to extract relevant concepts out of the document corpus. In comparison to the described approach of Salton the weight of a single term t in a document d is calculated as shown in the equation 3.1.

$$w_{t,d} = \frac{(1 + log(tf_{t,d})) * log\frac{N}{df}}{\sqrt{\sum w_i^2}} \tag{3.1}$$

The equation does not use the raw term frequency but this part is altered to $1 + log(tf_{t,d})$. This modification softens the occurrence of terms in a document and takes away the proportional increase. The effect is demonstrated in table 3.1[1], where only the unnormalised weights are taken into account. Moreover it leads to the advantage, that terms which occur twice as often as others in a document are not twice as important and also that terms which occur in all documents are ignored (document frequency part is zero if $N = n$ because the $log(1) = 0$). Although the corpus size stays the same over the time of evaluation the weights are normalized because the length of abstracts may differ enormously and the chance that a concept is used more often in a longer abstract than in a short one is even higher.

tf	1	2	10	100
$w_{t,d}$ Salton	1	2	10	100
$w_{t,d}$ Thesis	1	1.3	2	3

Table 3.1: Term Frequency Modification

According to the master thesis of Ackermann [1] who discovered that the precision of an index could decrease if terms are stemmed, this method is not used for the approach of this thesis. Beside this, a tagged and a untagged corpus are indexed. In general tagging adds the grammatical category to every word and will be done with a Part-of-Speech tagger (cf. [7]). Using such a technique the differentiation of terms could increase because some terms, especially in the English language are written identical although they have another grammatical category. Indexing a untagged corpus will ignore

[1]The document frequency part in this example stays always the same and is one.

this problem. Another assumption which goes together with grammatical categories is that only noun terms are interesting for the enhancement of a thesaurus, which would even more indicate the usage of a tagged corpus. For the tagging process this thesis will use the *Stanford POS-Tagger*[2], which is a free available tool of the *Stanford Natural Language Processing Group* of the *Stanford University*.

Having calculated the weight of all terms in respect to the document they appear in, a possibility is needed to rank the different concepts according to their global relevance for the thesaurus. Unfortunately this is not really what tf-idf weighting is made for. Taking the average tf-idf weight for every term and ranking them again with the new weight will not satisfy the demand of identifying the most relevant terms for the whole corpus or the thesaurus. Imagining a collection where the top 50 terms have a tf-idf weight around 0.6 and a document frequency of 1, which means they appear just once in the whole corpus. Unlike the number 51 is a term with a weight around 0.4 and a document frequency of 30 (which is maybe just 1.0 percent of the whole document corpus). Remembering the intention of relevance, concepts which appear in more than one document but have a lower weight are more relevant for the thesaurus than concepts which appear just once. The identification of the most relevant terms which are not already included in the thesaurus is reviewed using different techniques. For all used techniques the assumption that the global weight of a term is the average weight of all appearance of this term in the corpora (cf. equation 3.2) is valid. In one of the approaches a fixed weight level is set and all terms beneath this level are eliminated. The rest of the terms is ranked depending on the document frequency. In another approach the average tf-idf weight is used in combination with the document frequency of the candidates to calculate a new weight of a concept. This combination is shown in equation 3.3 where l defines the level of the document frequency. Terms beneath this level are ignored. The equation is based on the assumption, that the most relevant terms according to their average tf-idf weight and their document frequency are located in the middle. On the one hand the issue of this way of weighting is the automatic elimination of terms with a not sufficient document frequency (l defines the level of document frequency above which all document frequencies are taken into account.) and on the other hand in the imbalance toward the tf-idf weight which is considered stronger when it is

[2]The tagger is licensed under the GNU GPL. Information, FAQ and the download of the tagger are available under http://nlp.stanford.edu/software/tagger.shtml.

high. In comparison the influence of the document frequency is softened if it is high.

$$w_t = \frac{\sum w_{t,d}}{df} \tag{3.2}$$

$$w(c) = (\log df - l + 1) \times tf - idf_{avg}^2 \tag{3.3}$$

3.1.2 Matching of Index and Thesaurus Concepts

To identify candidates from the collection of extracted concepts it is necessary to eliminate all concepts which are already included in the thesaurus. According to the way of indexing, which is done in the first step, concepts which have more than one term have to be fragmented and matched term by term with the created index. The assumption is made, that when a term out of the index is included in any concept of the thesaurus it is no longer a candidate for the thesaurus enhancement. Depending on the structure of the thesaurus this assumption could lead to the deprivation of different hyponyms and hypernyms. When the term *illness* can be found for example in the index and the thesaurus has a concept which is labelled *illnesses and diseases* the term *illness* is no longer be seen as a candidate for the thesaurus although it could be a possible hyponym of the thesaurus concept. The admissibility of this assumption is discussed in the chapter 5 and is also an issue which can be found in the chapter 7. But currently, with this kind of index it is not really possible to remedy this problem.

3.1.3 Calculating the Weight of a Thesaurus Concept

In order to find a specific level of weight, which could give a hint about which candidates should be included in a thesaurus; it is useful to calculate the weight of concepts included in the thesaurus.

Paying attention to the fact that concepts could consist of more than one term, a way to combine the weights of the included terms has to be found. The three most obvious techniques to solve the problem are:

- Minimum - weight: $min(w_i, w_j, ...)$
- Maximum - weight: $max(w_i, w_j, ...)$
- Average - weight: $\frac{sum(w_i,w_j,...)}{count(w_i,w_j,...)}$.

Reckoning this issue from a semantic point of view it follows that the first technique, the minimum weight, is the best approximation for this problem, although it is expected to be higher than the weights would be in reality. Although the other possibilities of combining the weight do not have a strong semantic background in this case, they have been tested.

3.1.4 Preselecting Adding Locations for the Candidates

In a second step, possible adding locations in the thesaurus for all candidates that are chosen from the index need to be identified. Although there are two different approaches which are theoretically able to handle this problem the co-occurrence method, similar to the one of Grefenstette and his *Sextant* is used. The reason for this choice is the infeasible runtime of the pattern recognition approach. This approach needs three different web search engines requests for every relation which should be analysed. Even if the creation of one attribute vector, consisting out of the appearance of patterns in the snippets, consumes only one second, an analysis of one candidate and the whole MeSH thesaurus[3] would need over 7 hours. In comparison the estimated runtime to calculate the surroundings of all existing tokens in the index, which is not even necessary would need 4.5 hours[4]. Moreover the calculated surroundings could be reused for every similarity calculation and do not need to be calculated again. This means the co-occurrence approach needs constantly less than 5 hours. Trying to calculate an adding location for two different candidates with the pattern recognition approach for the whole thesaurus would consume more than 14 hours.

The modifications done in this thesis, in comparison to the original *Sextant* approach, just affect the effort of assigning the surrounding of a term. There is not a four step sentence analysis trying to identify non noun terms which influence noun terms, in detail. The simplification done by this thesis assumes that a term belongs to the surrounding of a concept if it appears in the same abstract and is not equal to the concept. According to this the surrounding of a global concept S_t is assigned by using the union of all surroundings of this specific term $s_{a,t}$ from all documents it appears in (cf. equation 3.4).

[3]The MeSH Thesaurus of 2008 consists out of 25,501 concepts.

[4]This runtime was the result out of different pre-tests which were done for this thesis. For an index of 18,000 noun concepts the surroundings of each was calculated. A rate of 100 surroundings per minute was the average creation rate. Using the amount of 25,500 concepts the calculated time which would be needed for calculation is around 255 minutes.

$$S_t = s_{a1,t} \cup s_{a2,t} \cup ... \cup s_{ax,t} \tag{3.4}$$

Having assigned the surrounding of all terms, these surroundings are ranked according to the tf-idf weights of the single terms they include. Afterwards the *stopwords* are eliminated because they do not include any content specific information[5]. In ongoing steps, different sizes of the surrounding, always including the top ranked concepts, are evaluated according of their results. Because of the critical runtime of the pattern recognition approach the goal is to keep the amount of collected relevant adding locations around 50[6].

The final similarity calculation, based on co-occurrence of concepts, is done with a weighted Jaccard measure (cf. equation 2.15), similar to the *Sextant* approach. Here the tf-idf weight of the concepts is used as input values for the measure.

The mode of operation of the used co-occurrence approach can be seen in the small example in figure 3.1. In this example the similarity of the concepts *cat* and *dog* is calculated based on two different surrounding sizes.

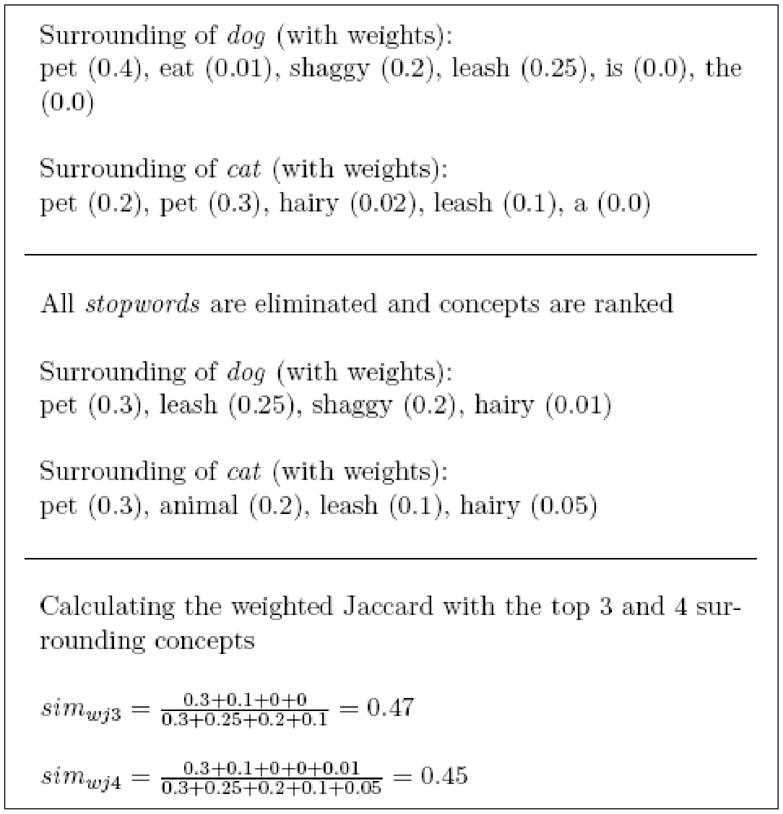
3.1.5 Categorizing Relations with Pattern Recognition

Finally, after having extracted candidates and possible adding locations for each candidate the relation between these pairs are categorized with the help of a machine learning approach. The attributes of the concept pairs for this approach are patterns, which occur in the snippets delivered by web search engines. The whole approach is similar to the one of Bollegala et al. in [3].

The candidates and each of their possible adding locations in the thesaurus are combined to tuples. From the content of each tuple a query for a web search engine is formed. Out of the result and the included snippets, the page counts and the occurring patterns for this specific query are extracted. The combination of these attributes creates an attribute vector for the tuple, which is categorized by two different machine learning approaches - a supported vector machine as in [3] and a decision tree.

[5]This elimination is done with the help of the tf-idf weights as mentioned in section 2.2.1.

[6]Collecting the attributes vectors for 50 different concepts pairs in a arguable time is possible. Assuming again that every calculation would need around one second leads to a runtime of a minute for one concept. Of course this is just a average value and in special cases the runtime could be even higher.



Surrounding of *dog* (with weights):
pet (0.4), eat (0.01), shaggy (0.2), leash (0.25), is (0.0), the (0.0)

Surrounding of *cat* (with weights):
pet (0.2), pet (0.3), hairy (0.02), leash (0.1), a (0.0)

All *stopwords* are eliminated and concepts are ranked

Surrounding of *dog* (with weights):
pet (0.3), leash (0.25), shaggy (0.2), hairy (0.01)

Surrounding of *cat* (with weights):
pet (0.3), animal (0.2), leash (0.1), hairy (0.05)

Calculating the weighted Jaccard with the top 3 and 4 surrounding concepts

$$sim_{wj3} = \frac{0.3+0.1+0+0}{0.3+0.25+0.2+0.1} = 0.47$$

$$sim_{wj4} = \frac{0.3+0.1+0+0+0.01}{0.3+0.25+0.2+0.1+0.05} = 0.45$$

Figure 3.1: Simularity Calculation Example

In detail, this approach uses three similarity measures (WebJaccard, WebDice and WebOverlap) (cf. section 2.3) to create the attribute vector. In comparison to Bollegala et al. the WebPMI is not taken into account because on the one hand it was ranked in the second half of the important measures with a importance weight of 0.0001 for the used SVM and on the other hand an assumption has to be made over the whole number of index documents of the used search engine. Additionally a set of different preextracted patterns are used to calculate the remaining attributes for the vector. This calculation is done by extracting the snippets out of the result of the web search engine and searching the different patterns in these snippets. Afterwards the dimensions are filled with the normalized amount of appearance of the respective pattern.

These patterns are extracted in a preprocessing step out of the web search engine by entering known synonym and genus/species pairs which are taken out of the MeSH or the WordNet thesaurus. Above this, concept pairs which do not have a connection to each other are used to extracted patterns. For each pattern a contingency table is created as in table 3.2. P and N within this table represent the sum of all frequencies of the captured patterns. p_v is the frequency of the pattern v in the complete collection. n_v represents the same as p_v but for the other class. Bollegala et al. just used this table for synonyms and non-synonyms but in this thesis this contingency table is also used for genus/species in contrast to non-genus/species and in contrast to synonyms.

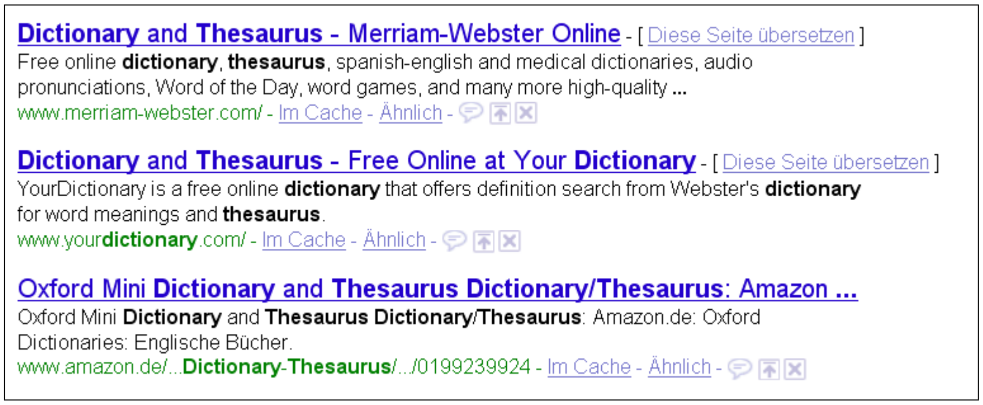
	v	other than v	All
Freq. in snippets for *class 1* word pairs	p_v	$P - p_v$	P
Freq. in snippets for *class 2* word pairs	n_v	$N - n_v$	N

Table 3.2: Contingency table

The extracted patterns for synonyms and genus/species relation pairs are ranked according to the information they contain in respect to represent the relation, similar to the approach in [3]. The $x^2 - weight$ (cf. equation 3.5) calculates the weight for the patterns of one of these groups according to the patterns of pairs that do not share one of these relations. Having calculated these weights the patterns are ranked and the first 60(80) patterns are used to form the dimensions of the attribute vectors.

$$x_v^2 = \frac{(P+N)(p_v(N-n_v) - n_v(P-p_v))^2}{PN(p_v+n_v)(P+N-p_v-n_v)} \tag{3.5}$$

In another preprocessing step the machine learning approaches need to be trained. The training data is created by using known synonyms, genus/species and disjunct pairs to create attribute vectors for these pairs. The machine learning approaches is not only trained, in contrast to the approach of Bollegala et al., with synonym and disjunct pairs, but also separated with genus/species and disjunct pairs and in a third training session with synonym and genus/species pairs. At the end three different models are used to categorize the relations.


Dictionary and Thesaurus - Merriam-Webster Online - [Diese Seite übersetzen]
Free online dictionary, thesaurus, spanish-english and medical dictionaries, audio pronunciations, Word of the Day, word games, and many more high-quality ...
www.merriam-webster.com/ - Im Cache - Ähnlich -

Dictionary and Thesaurus - Free Online at Your Dictionary - [Diese Seite übersetzen]
YourDictionary is a free online dictionary that offers definition search from Webster's dictionary for word meanings and thesaurus.
www.yourdictionary.com/ - Im Cache - Ähnlich -

Oxford Mini Dictionary and Thesaurus Dictionary/Thesaurus: Amazon ...
Oxford Mini Dictionary and Thesaurus Dictionary/Thesaurus: Amazon.de: Oxford Dictionaries: Englische Bücher.
www.amazon.de/...Dictionary-Thesaurus/.../0199239924 - Im Cache - Ähnlich -

Figure 3.2: Google Result for Thesaurus and Dictionary

To clarify the creation of attribute vectors, figure 3.2 shows a small example. Here the original query is *Thesaurus Dictionary* and the shown results are an extract from the *Google* search result. In the snippets of the three results four different patterns can be found:

- dictionary, thesaurus (X, Y)
- dictionary for word meanings and thesaurus (X for word meanings and Y)
- dictionary and thesaurus (X and Y)
- dictionary/thesaurus (X/Y).

Assuming that the dimensions of the attribute vectors include only the four different patterns:

1. X, Y
2. X and Y
3. X (Y
4. X/Y

the corresponding non normalized attribute vector for the tuple *Thesaurus Dictionary* is (1—1—0—1).

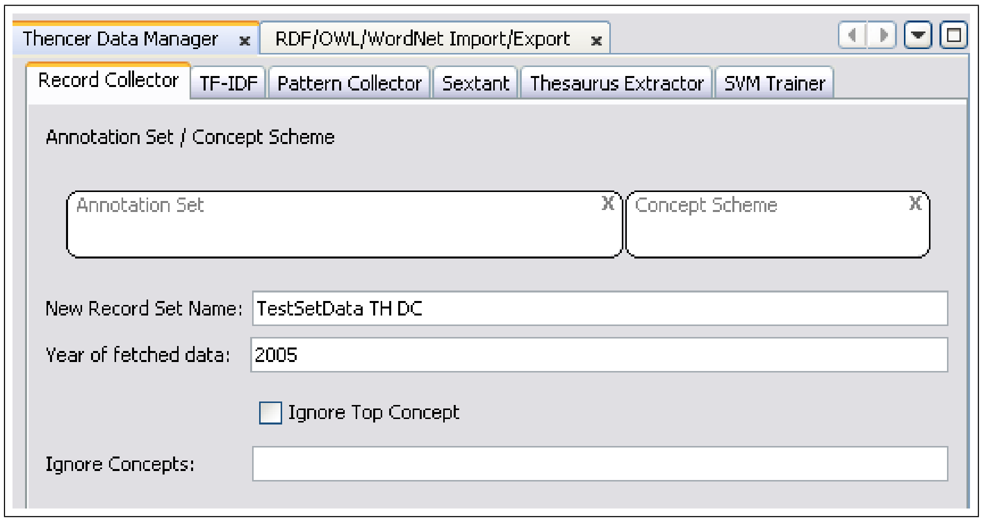
The search engine which is used in this thesis to fetch data for this final approach is Yahoo.com. Yahoo.com still provides an API which allows requesting more than 100 results for one query. Unfortunately it could happen that after 1000 requests the requests of the application identifier are blocked and any further results will be empty till the next day [7]. Another possible search engine is the well known Google.com. Unfortunately *Google* shut down their former API which was based on SOA (service oriented architecture). The new provided API is mainly designed to be used by JavaScript and is only usable by Java with the help of a JavaScript Notation Objects (JSON). Moreover the results of this API only contain 64 items. Therefore this was not a real alternative, because to create the pattern vectors, it is necessary to collect more than 100 results to get a sufficient filled vector. Another search engine, which was just launched during the research of this thesis is named *Bing* and published by *Microsoft*. This search engine also provides an API but first tests turned out that there are a lot of bugs, which are not completely fixed. This is the reason why *Bing* was not used in this thesis. Beside these three search engines, there are a lot more which are available on the internet but *Google* and *Yahoo* are the current market leaders and so it seems to be a good choice to take one of them. For more information about (web) search engines it may be a good idea to have a look at `www.searchenginehistory.com`.

[7]During the research it just happened twice in more than 30 days that *Yahoo* blocked the requests of the *thencer* application identifier and a waiting time was necessary to go on with the research.

3.2 Thencer - The Thesaurus Enhancement Tool

The whole implementation of the *Thencer* is done in the programming language Java. For creation and testing the *NetBeans Platform 6.5*[8] framework was used. The *Thencer* combines all functions and modules necessary for the execution of the experiments and is designed to be an expansion of the *Semtinel*[9] which was created and is still maintained by Kai Eckert.

The main module, which including all functionalities that are directly needed to run the process of thesaurus enhancement are located in the *Thencer Data Manager* module. A screenshot of the graphical user interface of this module is shown in figure 3.3. Additionally two more modules that extend the basic functionalities of the *semtinel* were implemented during the work for this thesis.



New Record Set Name: TestSetData TH DC
Year of fetched data: 2005
☐ Ignore Top Concept
Ignore Concepts:

Figure 3.3: Screenshot of the Thencer Data Manager GUI

3.2.1 Data Base Creation

To create a sufficient and well-founded data corpus with a large variety of records the *Record Collector* of the *Thencer Data Manager* module provides the functionality to enrich an existing record and annotation set with new records and annotations or to fetch a new record and annotation set for a

[8]More information and the download can be found under www.netbeans.org.

[9]The *semtinel* can be downloaded under www.semtinel.org. Information about new versions and the ongoing work can be found in the blog of Kai Eckert blog.kaiec.org.

thesaurus. This function uses the *PubmedSearchSFS* in the *Pubmed Search* module of the *Semtinal framework*. The *PubmedSearchSFS* access Pubmed via the API to collect data for the label of a concept. The function is implemented to store records which do not have an empty abstract and whose MeSH annotations are not empty. Additionally it is possible to adjust this class in a way that certain concepts and the children of these concepts in the thesaurus will not be taken into account when fetching records. With this setting possibility a split harvesting of records on different computers for the same thesaurus is feasible. Moreover, to avoid data loss by complications with the class (missing i-net connection, database access problems or out-of-memory errors) the class buffers the data and stores it in packages[10] into the database of the *Semtinel*.

3.2.2 Indexing and Concept Matching

Another functionality is provided under the tab *TF-IDF* in the *Thencer Data Manager* module. This class is designed to index a provided record set and in an ongoing step to match the terms of an index with the concepts of a thesaurus.

The indexing functionality takes every record which is deposited in the record set and fetches the stored abstracts. Depending on the setting the *Indexer* will use a tagger[11] before the real indexing process is started. In case of a tagged corpus the *Indexer* uses the combination out of term string and grammatical category to uniquely identify a token. If the corpus is untagged, the term string alone is the unique identifier of a token. In detail, the `left3words-wsj-0-18.tagger` model is used, which should have a accuracy of 89.03[12] percent on new words but it is the fastest model. A model with a higher accuracy but also a longer runtime is the `bidirectional-wsj-0-18.tagger` model with an accuracy of 89.30 percent on unknown words. Because the increase of accuracy would be so slight and the runtime is higher the first model is chosen for this thesis.

Having created the index, the second available functionality in this tab could be used to match a concept with a concept out of the index. The underlying class uses a thesaurus and the index identifier of a stored index

[10]For this thesis a fetching buffer of 50 searches and a storage buffer of 5000 records was used.

[11]As already mentioned in the section 3.1 this thesis uses the Stanford POS-Tagger to add grammatical categories to the terms in a sentence.

[12]This value is taken from the documentation, which is included in the POS-Tagger download.

as input and tries to match concepts out of the index with the tokenized preferred and alternative (synonym) labels which are included in the thesaurus. Finally, a list is stored including for each concept in the thesaurus the tokens from the index which are included in any of its labels[13]. Additionally the weight of a concept is stored by taking the minimum of the global tf-idf weights of all tokens included in the label.

The storing is done via a direct database connection to the *Thencer* specific H2 database *thencer.db* (cf. section 3.2.6 for a detailed description of this database).

3.2.3 Similarity Calculation with Co-occurrence

Under the tab *Sextant* - according to the software of Grefenstette - the co-occurrence based similarity calculation between candidates and concepts in the thesaurus could be done. The user has the possibility to variegate the length of the vector which means in practice if there are more attributes available than the size of the surrounding allows only the top attributes, ranked according to their tf-idf weight are used. When executing the process for the first time, it is advisable to set up the largest size of surrounding which should be evaluated, because they are precalculated if they are not already stored in the database[14]. If larger surroundings are required for any evaluation and precalculated vectors already exist, it is necessary to select the *delete precalculated vectors* option to get a valid result. The result is presented in a table and sorted descending according to the similarity weight. When the similarity measure, which is actually a weighted Jaccard measure should be switched `calcSimJac` function in the `sextant.java` file has to be adjusted.

3.2.4 Pattern Recognition in Snippets

The following section describes the classes and functions which cover all tasks according to the pattern recognition approach. This includes the pattern extraction to create the dimensions for the attribute vectors, the cre-

[13]For tagged tokens only nouns are matched. If no tagging was done all kind of tokens could be matched to the concept stopwords with a really low weight.

[14]This could take a while, but after storing all vectors the calculation is really fast, because only the vector of the new token has to be calculated. Storing is done into the table *vector* (see section 3.2.6 for a description of the database).

ation of training data sets for the machine learning approaches and also an implementation of a supported vector machine[15].

Collecting patterns from snippets could be done under the tab *Pattern Collector* in the *Thencer Data Manager* module. Patterns are extracted out of the snippets which are included in the results of web search engine. In general the functionality uses two concept labels, builds one query phrase out of them and sends this query to the search engine. It is important that the two labels that are taken to build the phrase are put into quotes, so that the search engines do not factorise the single terms included in one concept (cf. listing 3.1). A final query would look like *"information retrieval" "web search engine"* where *information retrieval* and *web search engine* are the labels of the two concepts whose relation shall be examined.

Listing 3.1: Phrase creation for pattern collection for synonyms

```
        String concept1 = c1.getPrefLabel().toString();
        String concept2 = "";
        if (c1.getAltLabels().size() > 0)
                concept2 = c1.getAltLabels().get(0).toString();
        if (concept2.length() > 1)
                String searchphrase = "\"" + concept1.trim() +
                    "\"" + "␣"\"" + concept2.trim() + "\"";
```

The function always works in the same way, with some small variations depending on the relation between the two concepts where patterns should be searched for. The given thesaurus is walked through using a depth-first search. For the capturing of genus/species and synonym patterns only one stack is necessary. For a concept the synonyms (`getAltLabels()`) or the species (`getAltLabels()`) are used to create a new query. To collect patterns for concepts, which do not share any relation two stacks are filled with concepts from the thesaurus. The first stack includes the concepts in the same order as used for the two other relations. The second stack is filled with the vertical mirrored thesaurus. In addition a randomization is included in the process of collecting concepts out of the stack, so that it is possible to extract patterns using always different concept pairs with the same relation out of the same thesaurus[16]. The patterns are stored with the relation which linked the used concepts into the table *pattern* (cf. section

[15]The decision tree approach is not already implemented in this module. Tests and evaluations going together with this approach are solely done in the *WeKa 3*

[16]This randomization works sufficient and provides a large variety of used sub trees out of the thesaurus. The standard swell is 0.7. If the whole thesaurus should be run through the randomization has to be eliminated by setting the swell to 1.0.

3.5). The used shortcuts are syn for synonym, sub for subclass (species) and dis for disjunct.

The second important function which is also accessible in the *Pattern Collector* tab handles the ranking of the extracted patterns according the opponent relation. Actually the equation 3.5 is used to calculate this ranking. The calculated weights are stored into the table *patternrank*.

The creation of training data for the later training and testing of the machine learning approaches is implemented in a similar way as the pattern creation is done. Again the function uses an imported thesaurus to extract labels according to the chosen relation. To set the relation, that shall be used the option 1 for synonym, 2 for subclass or 3 for disjunct relations has to be chosen. In addition a plain text file is needed including the different patterns (representing the dimensions of the final attributes vector), which should be searched in the snippets of the training data. The function uses a `BufferedReader` to read this file line by line, so every line should include one pattern (cf. listing 3.2)[17]. The results from the web search engines of all tuples are fetched and reviewed if and how often the different patterns appear in the result. The three similarity measures as stated above are calculated. Retrieving the three necessary page counts, three independent web search engine requests are necessary. Out of the first two requests, only the page count is used. Beside the page count, for the combination of the two concepts the last request provides a list of all snippets. The appearance numbers of the tokens are normalized and the final vector which is built just contains values between 0 and 1. The normalization of the appearance number is done by dividing the total number by the number of all found tokens.

Listing 3.2: Pattern file example

```
        X and Z
        X (Z
        X, Z
         .
         .
         .
        X; Z
         [emptyline]
```

[17]The last line in the pattern file has to be empty.

The final vector is stored in a semi-sparse vector representation style to a plain text file. Each line in this text file represents one vector. The first char sequence represents the final class of the vector (1 = synonym, 2 = subclass, 3 = disjunct). The rest of the line is a representation of the dimensions where the value is not zero[18]. The first number in front of the double point is the dimension, the number behind is the value (cf. listing 3.3). In the example, the vector in the first line represents a synonym relation (first char sequence is *1*). Beyond the first three dimensions, which are the similarity measures, it has an appearance of pattern 11 (14 - 3) and 12 (15 - 3). The second vector represents a disjunct relation. This format could be read by the SVM implementation `libsvm.lib` which is used in this thesis. Moreover on the same tab a converter function is implemented which could convert the just created vector files from the `libsvm` format to a format *WeKa 3*[19] could import [16].

Listing 3.3: Vector file example

```
        1 1:1.0 2:0.4552 3:0.0045 14:0.022344112 15:0.99
        3 1:0.0 2:0.1 3:0.0 11:0.1
        ...
```

Supported vector machine learning and testing could be done after having used all the functions stated before with the collected data. This thesis uses the SVM implementation of the `libsvm.lib`[20] java library according to [6]. This implementation imports the vectors out of the format stated above, but is also able to read non-sparse vectors. To create a model, which is afterwards used by the SVM to classify vectors where the class is unknown, the function *SVM Model Trainer* under the *SVM Trainer* tab in the *Thencer Data Manager* module could be used. Unfortunately there is no possibility

[18]Except the first three dimensions, which represent the similarity measures out of the page count. These values are always included in the vector file - although they are sometimes zero. Furthermore the user has the ability to ignore vectors which only include the first three dimension by selecting the corresponding option.

[19]WeKa 3 is the actual version of a data mining software which is written in java and has a variety of machine learning methods and functions included. WeKa is a project of the University of Waikato. To learn more about WeKa please visit `http://www.cs.waikato.ac.nz/ml/weka/`.

[20]More information about all possible configurations, other implementations in languages other than java and current information can be found under `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`. Moreover a beginner guide is available at this page and describes the standard work flow and functionalities of SVM and especially libsvm. In addition a GUI servlet is included.

to configure the SVM via the GUI. If a special configuration should be used the user has to have a look into the code of the `SVMTrainer.java` file in the SVM package of the *Thencer Data Manager* module. The result of this function is a model file. This model represents the hyperplane between the classes with which the SVM was trained. Running first evaluations of the created model is possible by the *SVM Predictor* functionality. This method needs a SVM model and again some training or test vectors which have to have the same format as the train vectors (if possible not the same vectors with which the SVM was trained) as input. The accuracy will be calculated by loading the model and classifying the vectors without respect to the class they belong to. Afterwards the *SVM Predictor* counts the number of right and wrong classified vectors and calculates the accuracy based on this counting.

At the moment there is no implementation of the decision tree approach included in the *thencer* extension. If the evaluation shall be done with a decision tree *WeKa 3* has to be used to run the classification.

3.2.5 Additional Modules and Functionalities

In addition to the main module two more modules were developed during this thesis to extend the functionalities of the basic functions of the *Semtinel*.

The OWL/RDF/WordNet - Importer was designed to extend the variety of possible import formats of thesauri. Beneath thesauri which are available in an owl/rdf xml format it is possible to import the WordNet thesaurus from its data files. The *RDF / OWL Import* function could be used to import thesauri which are available in an owl/rdf xml format. At the moment the tags, which are used to identify the different entities of the thesaurus (synonyms, subclass, etc.) are not customizable via the GUI and are hard coded in the function. In the listing 3.4 there are two new concepts. The first would have the URI 'C33779' and the second which is a subclass of the first the uri 'C33128'. 'C33779' has the preferred label 'dog' and the synonym 'Canis familiaris'. Additionally the 'dog' concept would have a scope note which can be found under the tag 'Definition' in the example. Technically seen the file is read by an `rdfparser`[21] and all statements are buffered into a graph. Having build up this graph completely all concepts are stored followed by the relations which link them.

[21] In this thesis the rdfparser from org.openrdf.rio.RDFParser was chosen whose library is downloadable via `openrdf.org`.

Listing 3.4: OWL / RDF Importer: Associated Relations and Entities

```
<owl:Class rdf:ID="C33128">
  <oboInOwl:hasRelatedSynonym>
    <oboInOwl:Synonym>
      <rdfs:label>
        Canis familiaris
      </rdfs:label>
    </oboInOwl:Synonym>
  </oboInOwl:hasRelatedSynonym>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="C33779" />
  </rdfs:subClassOf>
  <rdfs:label>
        dog
  </rdfs:label>
  <oboInOwl:hasDefinition>
    <oboInOwl:Definition>
      <rdfs:label>
        A member of the genus Canis (probably descended from
            the common wolf) that has been domesticated by man
            since prehistoric times; occurs in many breeds.
      </rdfs:label>
    </oboInOwl:Definition>
  </oboInOwl:hasDefinition>
</owl:Class>
<owl:Class rdf:ID="C33779">
  rdfs:label>
        Canis
  </rdfs:label>
  ...
</owl:Class>
```

The *RDF / OWL Export* function, which could be used to export any imported thesaurus into a owl xml file, similar to the format which can be found in the listing 3.4 writes all concepts from a thesaurus into the file in sorted form and groups all information according to one concept. This function does not use an `rdfwriter` to create an output.

The *WordNet Import* function uses the locally stored data file of WordNet[22]. Additionally the user has the possibility to enter the starting concept for the import. With this setting it is also possible to just import a WordNet sub thesaurus.

[22]Please note that WordNet has first to be installed (`wordnet.princeton.edu`), or at least the data files must be available locally on the computer.

In detail, the functionality takes the synsets and stores each synset as a new concept of the imported thesaurus. The first word, which appears in the synset, will always be the preferred label.

Web Search Engines is the second module which does not belong to the main module, but was used during the thesis. It includes three[23] different search interfaces - two for the web search engines *Google.com* and *Yahoo.com* and one for a locally stored *WordNet* data files. Both web search engines are accessed via the APIs provided by the publishers themselves. Although *Google.com* shut off their former API which used SOA technologies, a way to grab the information with the help of *JSON* objects is implemented. Unfortunately only a limited number of results are returned by this API (around 64). The API of *Yahoo.com* is included in the same way, but returns more results if requested. Both search engine functionalities do return a JSON object and additional a result string.

3.2.6 Database Model

The following section gives a short overview and an introduction on the used database structure. The *Semtinel* framework database (semtinal.db) is not extended. The *Thencer* specific database (*thencer.db*) includes all data concerning indexing, weighting of indexed concepts, matching thesaurus and index concepts (cf. figure 3.4) and also everything which belongs to the pattern extraction for the machine learning (cf. figure 3.5).

The index is stored in the table *index*. Here a generated *id* as identifier and the *name*, which should describe the index, are the attributes. All tables in figure 3.4 depend on the *index* table. There is always a *on delete cascade* setting included in the references, so if an index is deleted all reference entries in the different tables are deleted too. The table *record*, storing information about the other records which are also stored in the *semtinal.db*, depends on the *index* table in the place. Here the *name* attribute is a reference to the internal *id* of the records in the *semtinel.db*. Actually the stored tf-idf weight is the sum of all tf-idf weights of the tokens included in this record. This weight is used to normalize the single weights. In the tables *token* every unique token is stored, what means that the combination out of *name* and *type* has to be unique. If the type of a token is unknown because

[23] Actually there are four APIs included but the ask.com API seems to have been shut down some months ago.

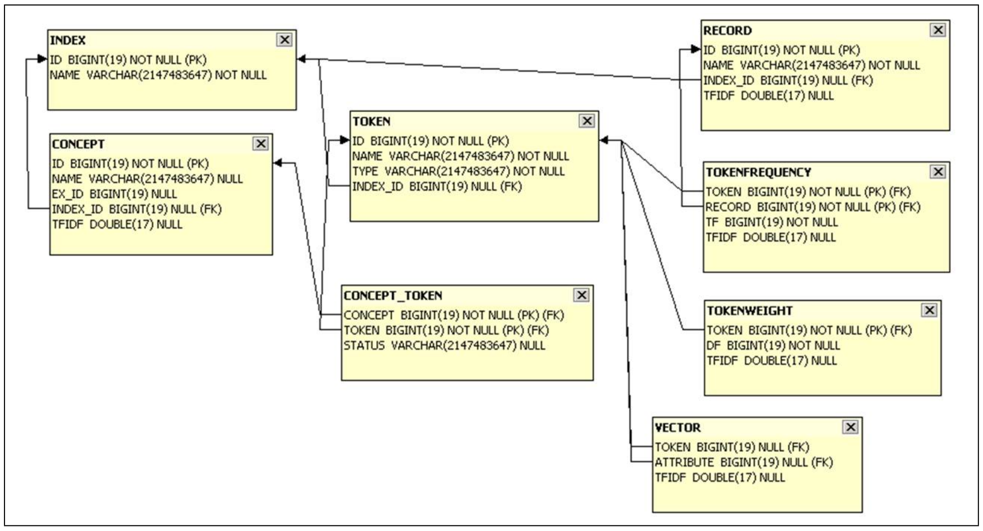

INDEX
ID BIGINT(19) NOT NULL (PK)
NAME VARCHAR(2147483647) NOT NULL
RECORD
ID BIGINT(19) NOT NULL (PK)
NAME VARCHAR(2147483647) NOT NULL
INDEX_ID BIGINT(19) NULL (FK)
TFIDF DOUBLE(17) NULL
TOKEN
ID BIGINT(19) NOT NULL (PK)
NAME VARCHAR(2147483647) NOT NULL
TYPE VARCHAR(2147483647) NOT NULL
INDEX_ID BIGINT(19) NULL (FK)
CONCEPT
ID BIGINT(19) NOT NULL (PK)
NAME VARCHAR(2147483647) NULL
EX_ID BIGINT(19) NULL
INDEX_ID BIGINT(19) NULL (FK)
TFIDF DOUBLE(17) NULL
TOKENFREQUENCY
TOKEN BIGINT(19) NOT NULL (PK) (FK)
RECORD BIGINT(19) NOT NULL (PK) (FK)
TF BIGINT(19) NOT NULL
TFIDF DOUBLE(17) NULL
CONCEPT_TOKEN
CONCEPT BIGINT(19) NOT NULL (PK) (FK)
TOKEN BIGINT(19) NOT NULL (PK) (FK)
STATUS VARCHAR(2147483647) NULL
TOKENWEIGHT
TOKEN BIGINT(19) NOT NULL (PK) (FK)
DF BIGINT(19) NOT NULL
TFIDF DOUBLE(17) NULL
VECTOR
TOKEN BIGINT(19) NULL (FK)
ATTRIBUTE BIGINT(19) NULL (FK)
TFIDF DOUBLE(17) NULL

Figure 3.4: Database design for tf-idf index storing

there has not been done tagging while indexing, the type is *undefined* and the uniqueness just refers to the name. When tagging has taken place it is possible that there are two identical names (e.g. 'fancy') but different types. The different weights of the tokens in connection to the documents are stored in the table *tokenfrequency*. Additionally the term frequency can be found in this table. The table *tokenweight* contains the document-frequency *df* and the calculated global tf-idf weight. The surrounding of a token, which is needed in the co-occurrence approach to calculate similarities is stored in the table *vector*. This table includes no additional information and is completely deducible from the other tables but makes the calculation essentially faster.

During the process of matching thesaurus and index, the tables *concept* and *concept token* are used to store the generated information. The first one includes the concepts and the *external id* attribute creates the link between *thencer.db* and *semtinal.db*. The concepts are also dependent from the index, because on this way it is possible to create more than one mapping between a thesaurus and an index. When trying to find out the differences between a tagged and an untagged index. The *status* attribute is an identifier if the token was found in the preferred label of the concept or in an alternative label (synonym).

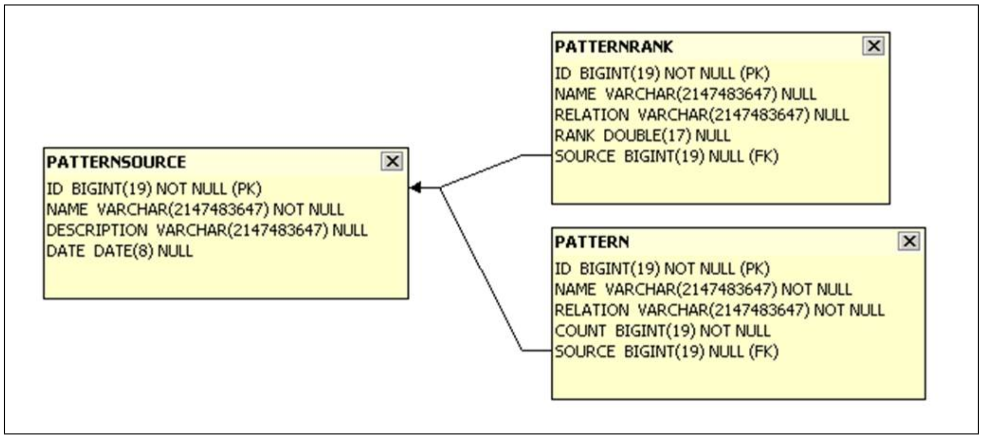

PATTERNRANK
ID BIGINT(19) NOT NULL (PK)
NAME VARCHAR(2147483647) NULL
RELATION VARCHAR(2147483647) NULL
RANK DOUBLE(17) NULL
SOURCE BIGINT(19) NULL (FK)
PATTERNSOURCE
ID BIGINT(19) NOT NULL (PK)
NAME VARCHAR(2147483647) NOT NULL
DESCRIPTION VARCHAR(2147483647) NULL
DATE DATE(8) NULL
PATTERN
ID BIGINT(19) NOT NULL (PK)
NAME VARCHAR(2147483647) NOT NULL
RELATION VARCHAR(2147483647) NOT NULL
COUNT BIGINT(19) NOT NULL
SOURCE BIGINT(19) NULL (FK)

Figure 3.5: Database design for pattern storing

The patterns which are collected in the process of the thesaurus enhancement and especially for the final machine learning approaches are stored in the table *pattern*. Every time a new collection of pattern is fetched from the search engine, a new entry in the table *patternsource* is stored to group these patterns. Although it is not possible to collect patterns for different relations in the same run at the moment, the *relation* attribute is included in the *pattern* table to obtain this possibility. The third table *pattern* includes the patterns ranked in contrast to other relations. This function is available in the same tab than the pattern collection. Important to notice is, that the *id* is not the original *id* attribute from the table *pattern* but a newly created one, because here two patterns with a similar string but different *identifier*s from different *relations* are compared. As *source* always the id of the primary pattern source is stored. When the user compares the patterns of pattern source 1 and 5, 1 is stored in the *patternrank* table under the *source* attribute. The relation attribute indicates what kind of relations where compared.

Chapter 4

Experimental Setup and Evaluation

After the approach used in this thesis has been described in chapter 3 the experiments and the setup of them can be found in this chapter. Additionally the different aspects, which are taken into acount for the evaluation are listed in section 4.2.

4.1 Test Data Corpus

The evaluation and testing of the methods are done based on the MeSH (Medical Subject Headings) thesaurus published by the National Library of Medicine (NLM). With the help of concepts provided by this thesaurus a data corpus is collected via Pubmed Search[1]. A specific feature of the results of this search engine, which are mostly publications out of the medical environment, is that most of the retrieved documents are tagged with the headwords out of the MeSH. To create a sufficient data corpus for each concept of the MeSH 2008 thesaurus publications out of the year 2005[2] are collected with the help of Pubmed. Always the ten first documents are stored, which do not have a blank abstract. Although most of the documents are from the year 2005 there is no subject focusing, because Pubmed does extend the restrictions if sufficient documents could not be found. The effect of subject focusing could appear in years where a topic was really present, for example the *H1N1 virus (swine flu)* in 2009. The

[1]The internet browser version of this search engine, which is also a service of the NLM can be found under `www.pubmed.gov`.

[2]This year is the last one, where almost every publication is tagged.

data corpus includes 218,761 records[3] split into 16 separated collections - one for each top concept of the MeSH. The largest collection (Drugs and Chemicals) includes over 46,000 documents and the smallest (Publication Characteristics) around 1,000 documents (For a complete overview of the data corpus see appendix B). Additionally to the records almost 3 million annotations (tags of a document, which refer to a concept in the MeSH) are included in this corpus. Additionally the WordNet thesaurus[4] which was imported into the data structure of the *Semtinel* is used. WordNet 3.0 includes around 75,000 concepts[5] and under the head note *entity* it is divided into three large subcategories: *thing*[6], *physical entity* and *abstract entity*. In contrast to MeSH WordNet is really global and more general, what is recognizable not only in the topic of the concepts but also in the length of the concept labels. Most of MeSH's concept labels include even more than two terms. The concept labels of WordNet, including maximal three terms, are really easy understandable for everyone. This is also the reason why WordNet is often used to create training sets and test methods (cf. [3]). Unfortunately the WordNet thesaurus does net reflect the richness of detail that has to be expected in a thesaurus a company or organisation uses to represent their own structure. A lot of concepts in specific thesauri include of specifications to curtail their topic (e.g. the MeSH 2008 concept *Central Nervous System Parasitic Infections*).

[3]This value is the sum calculated out of the record count of all 16 top-concepts of the MeSH. So this value could include duplicate records.

[4]The WordNet thesaurus is a hand made thesaurus of the *Princeton University*. More information and the download can be found under `wordnet.princeton.edu`

[5]This number of concepts reflects size of the imported thesaurus in the *Semtinel*. The structure of WordNet differs from the one used in the *Semtinel* in some aspects. In contrast, WordNet stores every concept as a single entity. Concepts which are in a synonymous relation are stored in *synsets*. So it is to assume that WordNet includes around 75,000 *synsets*.

[6]The sense of this node is not clear, because it includes only a couple of leafs.

4.2 Evaluation Methods

The evaluation of the complete process of thesaurus enhancement is an important step to validate its adaptability and usage. Caused by the existing diversity of methods, included in the aggregated approach, different evaluation techniques are used. All methods and the complete process are evaluated under the aspects of:

- accuracy
- runtime
- adaptability to other thesauri and circumstances

4.2.1 Finding the Important Concepts

In this part a detailed evaluation is complicated. The correctness of the outcome of the indexing and identification of candidates with the tf-idf weighting approach is difficult to measure because there is no *correct* division of the candidates. The *correctness* is in the eye of the beholder. The allocation does absolutely depend on the usage and the topic of the thesaurus. In general it is to state, that there is no real *wrong* when trying to enhance a thesaurus. The only issue is, that a term is added on a absolutely *wrong* position. This would cause failures when this *wrong* thesaurus is used.

Regardless these circumstances, it makes sense to have a look at the density of nouns in the top ranked candidates after applying the approach to the index. In general, when looking at the tf-idf approach and its intention, it needs to be estimated that the top concepts should be mostly nouns with a really high relevance for the thesaurus.

Moreover the time which is needed to create a tagged and untagged index is measured and compared. In addition the added value created by the tagged index is evaluated in comparison to the untagged index.

4.2.2 Co-occurrence to Decrease Problem Complexity

In contrast to the identification of candidates, the concept of *correctness* could be applied to the similarity calculation between concept pairs based on the co-occurrence approach. In general the measurement of this approach is done by the deletion of existing concepts in the thesaurus and the recovery of the deleted structures. Caused by the commonality of the concept of *similarity* it is not supposed that the real position is retrieved, but that concepts in the surrounding are identified as similar. This means that the *right position* has to be defined as a position in the surrounding of the exact position. The accuracy A_{co} of this method is calculated by counting the right results and dividing it by the sum of right and wrong results (cf. equation 4.1). Because this method is used to reduce the possible adding locations it is essential that the accuracy of this method is sufficient high. If the *right* adding location is not included, the following approach is not able to categorize this position.

$$A_{co} = \frac{\sum Results_{right}}{Results_{right} + Results_{wrong}} \tag{4.1}$$

Moreover the time which is needed to create the surrounding of a concept is evaluated according to the time the final methode needs to categorize the relation between two concepts.

4.2.3 WeKa 3 - Machine Learning Software

The pattern recognition approach offers a lot of possible evaluation methods but also a lot of setscrews to modify the results. This thesis will use *WeKa 3* [30] which offers a couple of clustering methods and evaluation techniques. These methods will be trained in a first step and the accuracy and correctness will be evaluated with another set of data afterwards. Both used machine learning approaches are compared not only according to their accuracy but also according to their recall and precision. *WeKa 3* uses a own importing format for data but all imported data is available for each clustering method used what makes the tool perfect to run different tests on a couple of datasets. Beside this, while training and testing different configurations of clustering methods, WeKa tool provides a uniform result sheet which presents an overview about how accurate a method and configuration is according to the set of data. Moreover some clustering method results could be visualised.

In this constellation the extracted data is analysed with a linear and a RBF supported vector machine. In combination with these evaluations, the different dimensions of the vector are verified according to their influence on the decision whether a combination of two tokens has a subclass or a synonym relation. In the article of Bollegala et al. a correlation of 0.83 is achieved with a linear kernel so this value is taken as a goal. Moreover different sets of trainings data will be taken into account. Additionally, for this last categorizing step it is not only important that the right category is found by the machine learning approach but also that are not too many other candidates categorized in the same way. This would lead in the end to too much manual work for the user. According to this the final set of possible adding locations should be as small as possible but still should include the best fitting position.

4.3 Experiments

This section explains the experiments which are done during the evaluation phase of this thesis and give a brief explanation why these experiments are accomplished.

Experiments on indexes are done in the first place. To form an opinion about the necessity of tagging, indexes which are created based on tagged and untagged documents are analysed. It is of interest, if the high ranked concepts according to the tf-idf weighting in the different indexes variegate significantly or if a high percentage of concepts is similar. Moreover, to find a possible limit for the tf-idf weights among which it is no longer meaningful to add the concept to the thesaurus the distribution of the tf-idf weight in the thesaurus is analysed. Additionally the assumption for concepts consisting out of more than one word to take the minimum weight is part of another experiment. In a last experiment which is concerned with the created indexes, the three different methods, presented in section 3.1.1 are used to extract candidates out of the thesaurus.

Co-occurrence experiments are done in a second step. This part is mainly focused on the reduction of possible adding locations in the thesaurus to reduce the effort and the runtime of the last approach low. In a first experiment an automatic evaluation is done using synonyms and also species that are included in the index and in the thesaurus to evaluate the degree of correctness of this method. In a second experiment for different candidates the most similar tokens in the thesaurus and in the index are calculated. The correctness of these results is proofed by looking up both concepts manually in the internet. The execution of these experiments should avoid that the results are absolutely wrong and the abstracts provide too less information which are needed to calculate a sufficient surrounding of a concept for a co-occurrence analysis.

Experiments on the machine learning approaches are done at the end. In a first step the two different approaches are evaluated with the training data extracted based on MeSH and WordNet. The results are compared and should give hints, if the tested approaches are influenced by the specificity of a thesaurus, because, according to [3], this should not play any role. Moreover the two models which were created to decide between synonym (species) and disjunct relation are compared with the model which categorizes between all three (two) categories. The outcome is really of interest, because if the machines could be trained by this model and the accuracy is sufficient high it is no longer necessary to use the two single models.

Chapter 5

Results

This chapter presents the outcome of the experiments described in chapter 4.2 corresponding to the approach of this thesis which was explained in chapter 3. The experiments are grouped by the three different sub methods of the approach. A final evaluation of the complete approach can be found in section 5.4.

5.1 Analysis of Index and Weighting Method

5.1.1 Tagged versus Untagged Indexes

In the first place the indexing of all documents at once would not be expected to happen in real world companies. The new documents which appear and are relevant for the companies would be indexed directly or buffered within a week. According to this the working load would be just a part of what is done in this thesis at once. The runtime of the `Indexer` heavily depends on the settings chosen by the user concerning tagging. Although using the fastest tagging model which still has a sufficient accuracy the time needed for the indexing process increases by 5,400 percent. Evaluating two different record sets (humanity set with 1468 records and anatomy set with 13,392 records) the runtime for one record without tagging is in both experiments around 0.024 seconds[1]. In comparison the indexer needs for one record with tagging is around 1,3 seconds. Calculating the time the `Indexer` would need for the complete set of 218,761 records without tagging it takes 5,250 seconds (around 87 minutes). Trying to tag and index the complete record

[1]The time of the experiments is measured including the period the program needs to fetch all records into the main memory.

set 284,389 seconds are necessary (around 3 days and 8 hours). Over a longer time it is to reckon with around 3.8 to 4.0 seconds[2]. Additionally the program is currently designed to load the complete set of records into the main memory. Fetching only the records (around 218,000 records)[3] of the MeSH this process could take several minutes. Record sets with around 50,000 records do not cause any problems, except that loading times increase proportional with the amount of records. Trying to index the complete set of records has caused a *Java Heap Space Error* and the program generates around 2.3 GB of swap files before throwing an exception. An alternative is to load record by record which needs approximately one second per request (again around 3 days for the complete set). The compromise is to buffer the records in bundles and index them. Unfortunately the index already created has to be updated the whole time or has to be hold in the main memory during this phase ensure the consistency. For the following evaluation the two sub thesauri humanity and anatomy and the corresponding record sets are chosen. Using a sub thesaurus instead of the complete MeSH thesaurus reflects the real world problem better, because there is not a almost complete technical thesaurus available in a company.

Looking at the tagged or untagged index in terms of finding candidates for the thesaurus, the composition of verbosity is critical. At the first glance, in the two tagged indexes nothing unexpected can be recognized (cf. table 5.1). Remembering that there are eight lexical categories in the English language, the only issue is the high number of nouns within the index.

Kind of Word	Anatomy Index	Humanity Index
Nouns	59.25 percent	55.34 percent
Other Kinds	40.75 percent	44.66 percent

Table 5.1: Composition of Word Kinds in the Index

Sorting the complete index descended according to their weight[4] the circumstances change a bit. Still the global dispersal of the verbosity remains equal. But looking at the development of this dispersal, starting from in the first ten percentages and extending the amount step by step to the whole index, it is notable that the density of nouns is much higher in the first

[2]The indexing considering tagging of the organisms record set which includes 29,061 records took over 30 hours.

[3]For the test a Java Heap Space of 1,512 MB was used.

[4]With the weight of a token or a concept in this case and all the following always the tf-idf weight will be meant, except something different is explicitly stated.

percentages than in the complete index. This is shown in figure 5.1 and figure 5.2 where it is apparent that the density of nouns decreases by around 10 to 15 percent in the complete development.

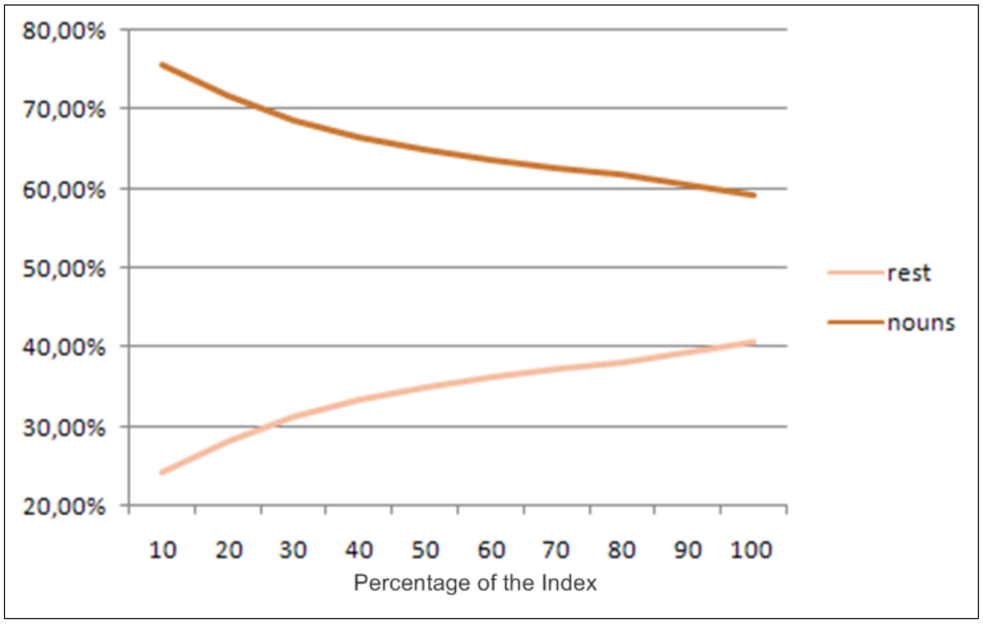


Figure 5.1: Anatomy Index: progress of noun density in a ranked index

Analysing the composition of the nouns in detail, what means looking at the different noun tags, which are added by the tagger[5] it is conspicuous that the density of common nouns (nn and nns) remains almost constant over the complete development of the ranked index (cf. figure 5.3 and figure 5.4). The reason for the global decrease of the noun density is only the density of proper nouns (nnp and nnps).

The influence of the proper nouns is especially obvious in the anatomy index, where the density of those kinds of nouns starts at 43 percent in the first ten percent and decreases to fewer than 25 percent in the whole ranked index. This trend is already visible in the first ten percent of the ranked index (cf. figure 5.5), where the density already decreases by five to seven percentage points.

[5]The Stanford Tagger has four different noun tags which it uses to divide between common noun singular (nn), common noun plural (nns), proper noun singular (nnp) and proper noun plural (nnps).

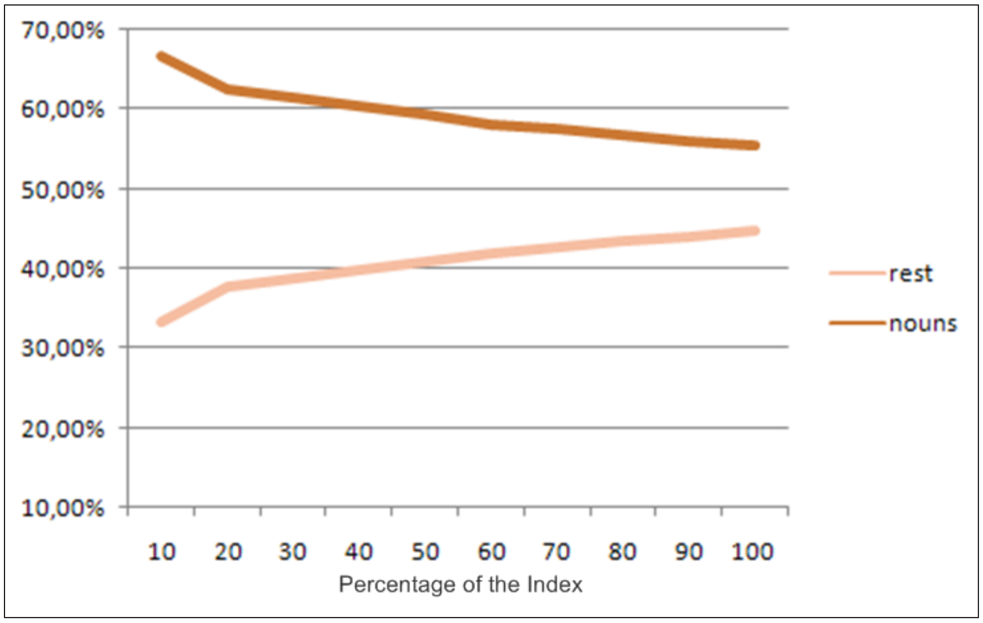


Figure 5.2: Humanities Index: progress of noun density in a ranked index

To compare tagged and untagged indexes, on the one hand a simple string comparison need to be made and on the other hand the grammatical categories need to be added belatedly to get an overview of the composition of verbosity of the untagged indexes. This assignment has been done by a standard string matching method which only takes the names of tokens and not their types into account. In case one token of the untagged index could be matched with different grammatical categories from the tagged index, the weightiest is added. This avoids, that those high ranked tokens of the tagged index, which have additional appearances in lower ranks but with a higher document frequency, are tagged with the lower frequency category.

In general the untagged index includes around 80 percent of the variety of concepts in relation to the tagged one which is listed in table 5.2[6]. This could be explained with table 5.3[7]. The untagged tokens are not divided in the grammatical categories although one token may appear in different grammatical categories. Although there are less concepts in the untagged index the distribution of verbosity in the ranked index is not affected. Words

[6]Cleaned is this content means, that all tokens, which are obviously only a senseless combinations of letters and numbers have been deleted.

[7]The amount in the table reflects always just one of the tokens, with multiple types - meaning if 10 tokens appear with 2 types than there are all in all 20 tokens in this group.

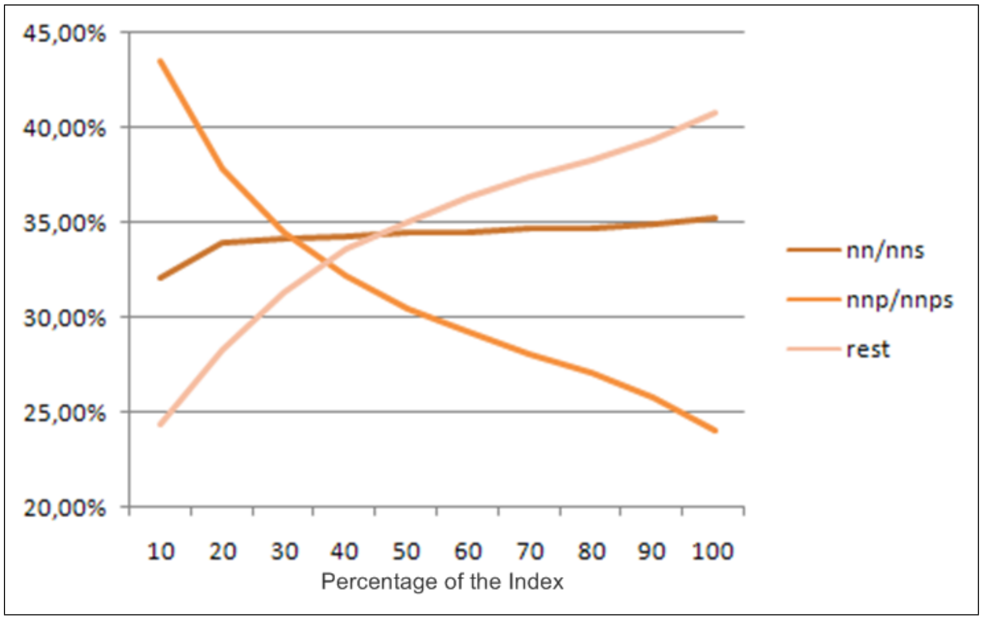


Figure 5.3: Anatomy Index: comparison of proper and common nouns in a ranked index

Index	Amount of tokens (in tagged version)	Amount of tokens (in untagged version)
Anatomy (complete)	62877	48112
Anatomy (cleaned)	60180	45763
Humanities (complete)	20905	17605
Humanities (cleaned)	20055	16761

Table 5.2: Overview of tagged and untagged indexes

are not split up depending on their grammatical categories and so the weight, which reflects the average weight of all single concepts, may be lower or higher than it actually is. A resulting distribution of nn(s), nnp(s) and the other types is shown in figure 5.6 for the anatomy index and in figure 5.7 for the humanities index.

The distribution differs at some points from the distribution of the tagged index. The proper nouns have a high density in the first percentages of

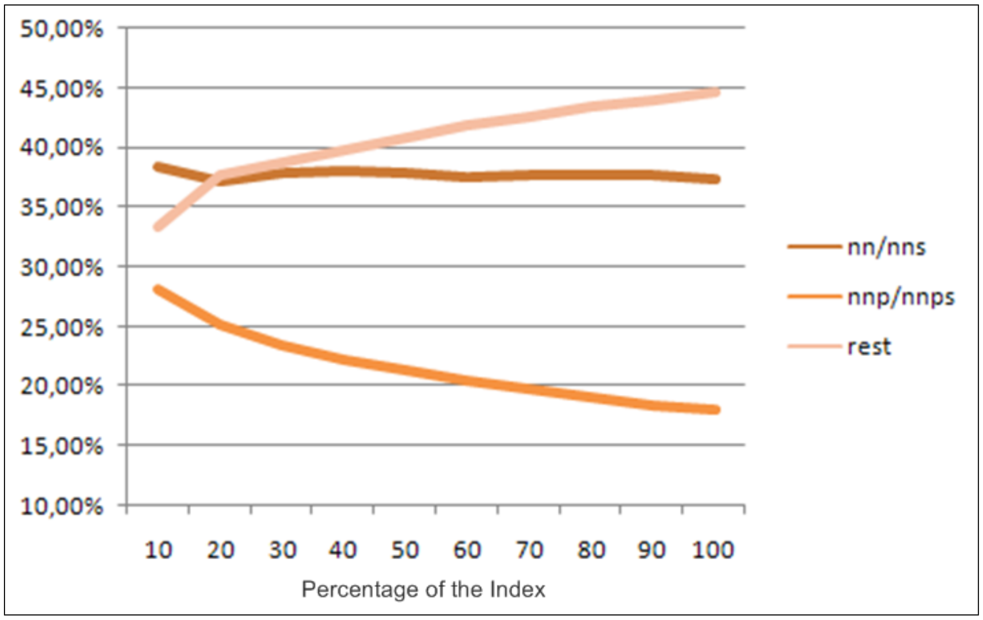


Figure 5.4: Humanities Index: comparison of proper and common nouns in a ranked index

Kinds of Types	Anatomy Index Tokens	Humanities Index Tokens
1	35077	14299
2	7956	2290
3	2731	542
4	621	70
5	177	18
more than 5	51	5

Table 5.3: Listing of Token Types Appearance

the ranked index, which decreases by around 20 percentage points through the complete index. Using the first 10,000 tokens of the untagged ranked anatomy index and searching them in the tagged ranked anatomy index over 80 percent of the tokens are found.

According to the evaluation results concerning the indexes it is not necessary to use a tagged index to identify candidates, because the allocation

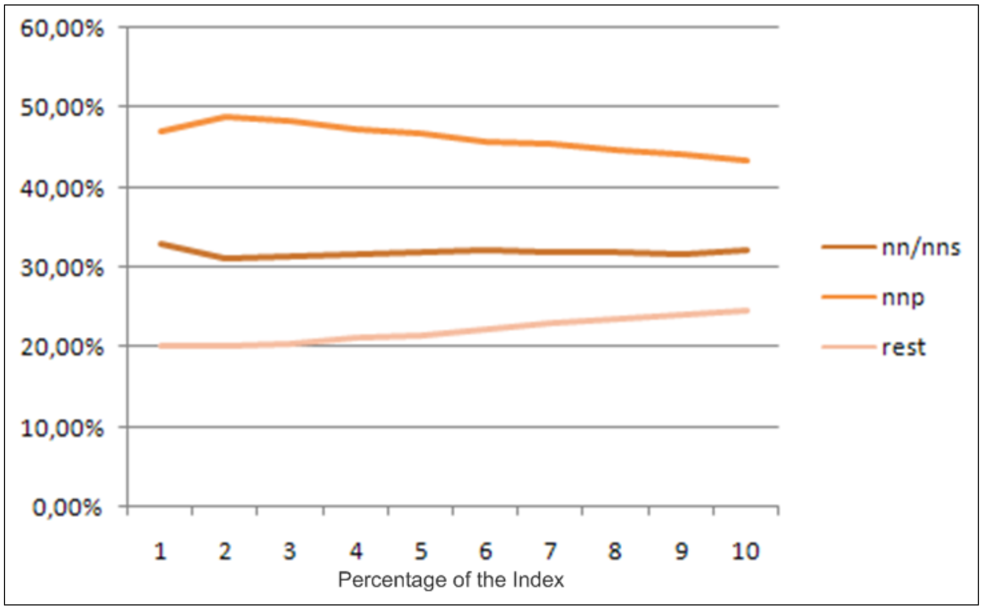


Figure 5.5: Anatomy Index: Proper noun trend in the first 10 percentage of the ranked index

of grammatical categories in the upper percentages of the index is almost identical and only some concepts are ranked differently.

5.1.2 Recovering Thesaurus Concepts in the Index

Using the two created indexes and the corresponding MeSH sub thesauri - anatomy and humanities - the matching of index and thesaurus concepts it turned out that almost all concepts in the thesaurus can be found in the index. Only 4 percent cannot be matched in no way (cf. table 5.4), what means that neither the tokens out of the preferred label nor any tokens out of the alternative labels can be matched to the index.

This outcome leads to the assumption that the set of documents is relevant for the thesaurus.

Moreover from the matched concepts a maximal amount of 2 percent of the preferred labels cannot be found in the index[8]. In these cases it is questionable whether the preferred token is really the one which should be

[8]This matching only has to include one of the tokens of the preferred label. Labels with more than one token/word included are matched if at least one word can be matched to the index.

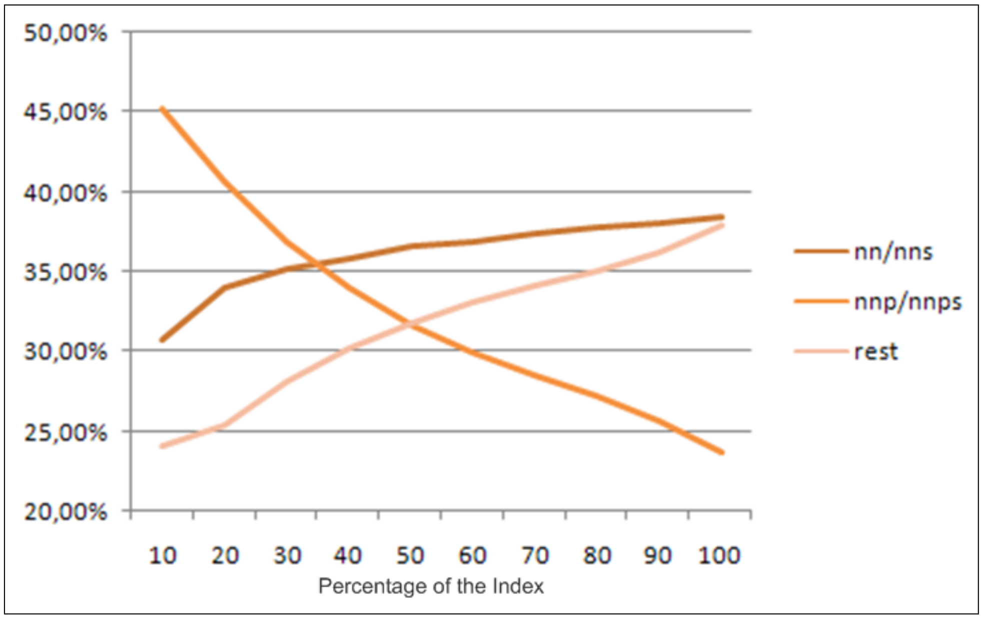


Figure 5.6: Anatomy Index Untagged: Distribution of Type Density

chosen to get the *USE* tag in the thesaurus, or if it would not be better to use an alternative label instead. The same idea could arise when comparing the weights of the preferred and alternative labels. More than 35 percent of the alternative labels in the humanities thesaurus and over 20 percent of the alternative labels in anatomy thesaurus have a higher weight than the corresponding preferred label.

Index	Number of Concepts	Unmatched Concepts	Unmatched Preferred Labels[9]
Anatomy (Untagged)	1163	11 (0.95%)	7 (0.60%)
Anatomy (Tagged)	1163	12 (1.03%)	8 (0.69%)
Humanities (Untagged)	114	5 (3.47%)	3 (2.08%)
Humanities (Tagged)	114	5 (3.47%)	2 (1.39%)

Table 5.4: Overview: Matching Concepts and Index

In a next step the progress of the weight of the concepts in a thesaurus is analysed. As stated in the theoretical background (cf. section 3.1) to

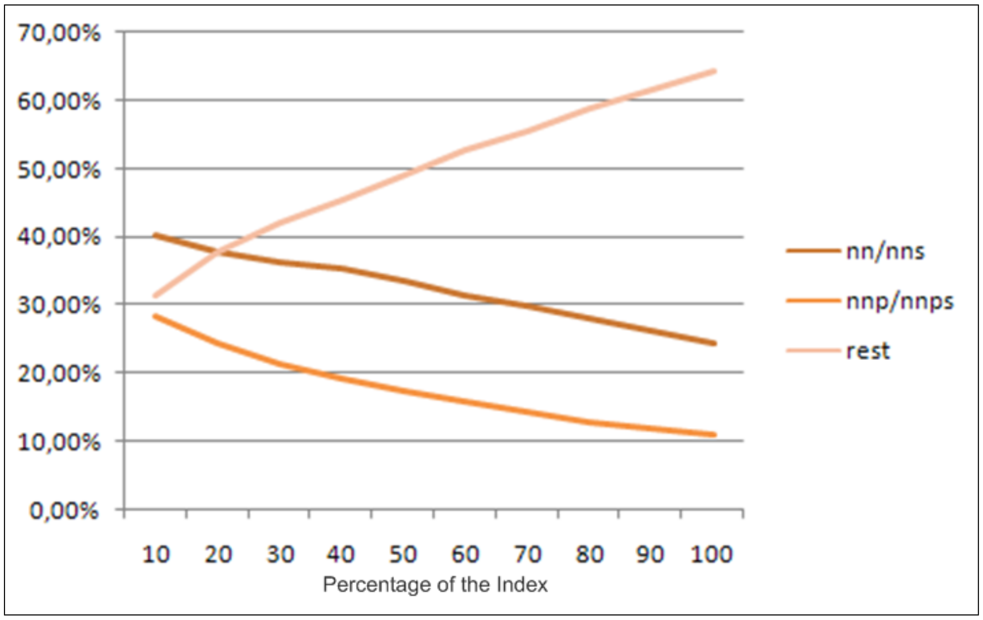


Figure 5.7: Humanities Index: Proper noun trend in the first 10 percentage of the ranked index

calculate the weight of a thesaurus concept, which does include more than one token in the preferred label all weights of all included tokens are taken and the minimum of those weights is used. To proof this semantically based assumption, also the progress of the maximum weight and the average weight is outlined in figure 5.8 for the untagged anatomy index and in figure 5.9 for the tagged anatomy index. The horizontal axis shows the rank of the concept. In this figures the concepts are ordered for every of the three progresses newly.

It is obvious that weights decrease more rapid during the first 50 ranks than while the rest of the progress. In the end the weight converge more and more against the horizontal axis. There is not a level above which the weight of the concepts remains. Minimum values for the anatomy thesaurus, ignoring the unmatched concepts are 0.03796 and for the humanities thesaurus 0.0408. Having a closer look at the different weights of a concept - meaning the weight of the different terms which appear in one concept (preferred and alternative label) it stands out that in many cases the weight of the tokens deviate a lot from each other. Regarding figure 5.10 the ground line represents the minimum weights of all tokens that are combined in the concept (brown curve). The highest cure (strong orange curve) is the maximum

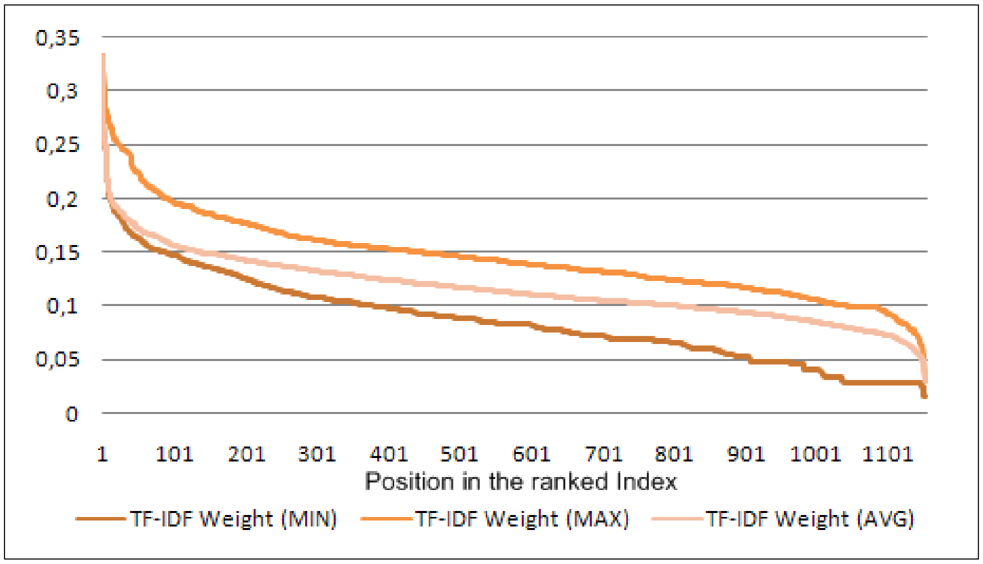


Figure 5.8: Concept weights of anatomy thesaurus matched with untagged Index

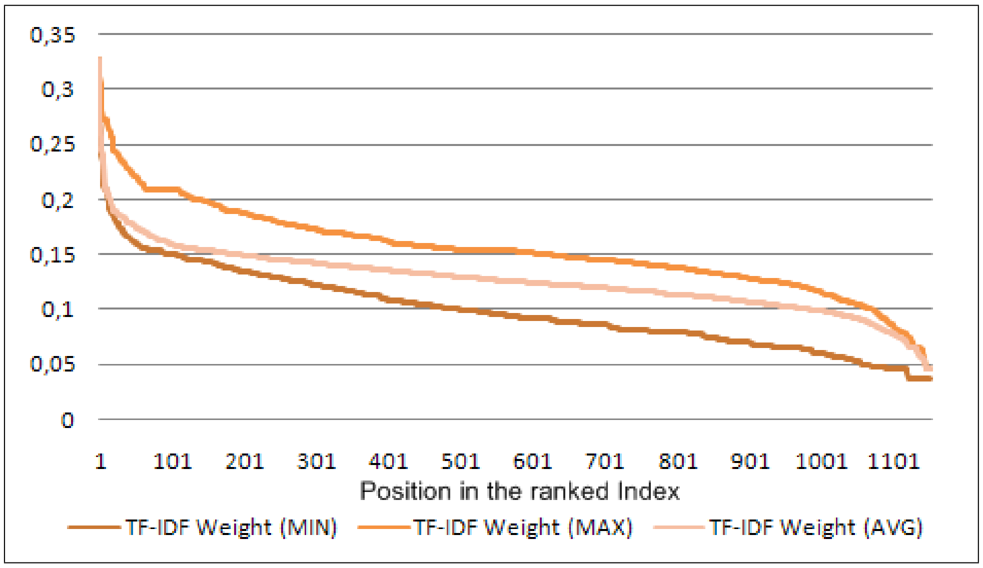


Figure 5.9: Concept weights of anatomy thesaurus matched with tagged Index

of the weights which are combined in one concept. The tokens are sorted descended by the minimum appearing weight firstly, secondly by the maximum and lastly by the average (light orange curve). This figure strongly underlines the inhomogeneity of the different fragments which are combined in one concept.

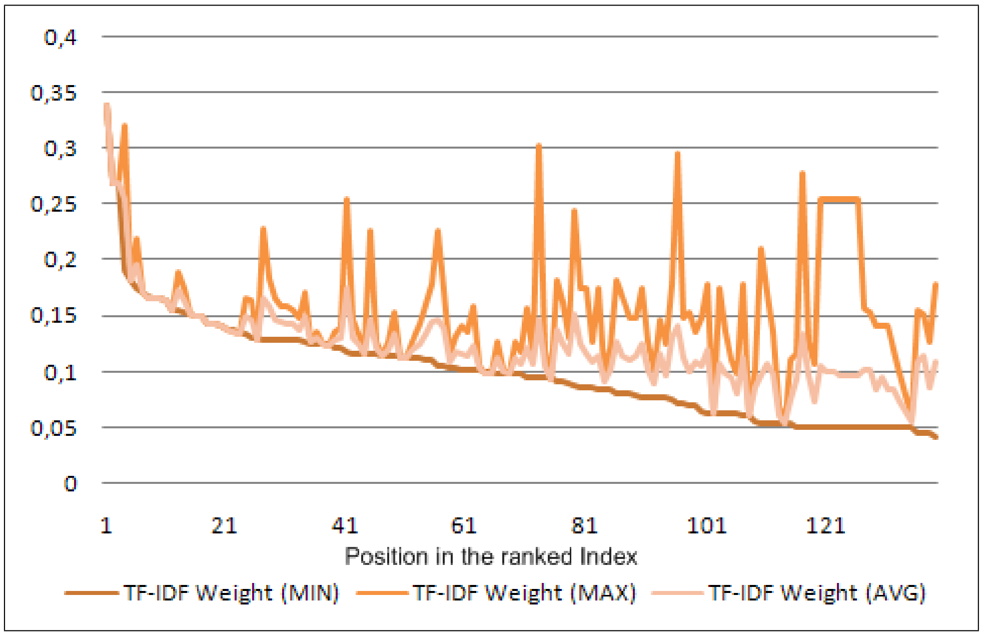


Figure 5.10: Breakup of the different weight calculation methods for thesaurus concepts

In a second step the matched tokens are split into common nouns and proper nouns because this two groups, although they belong together do not behave in the same way when they are ranked, as already stated in the section above. It is essential that the common noun curve (cf. figure 5.11) look almost the same as in figure 5.10. However the proper noun curves differ. The curves of the three calculation methods (minimal, maximal and average tf-idf weight) only have a countable number of points were they disperse, which means, most of the concepts just include one proper noun in their labels (cf. figure 5.12). But it is also to mention that only 54 percent of all concept labels in the humanity thesaurus do even include a proper noun.

Overall these results underline the assumption which was made, based on the semantic background of relevance calculation, because both other

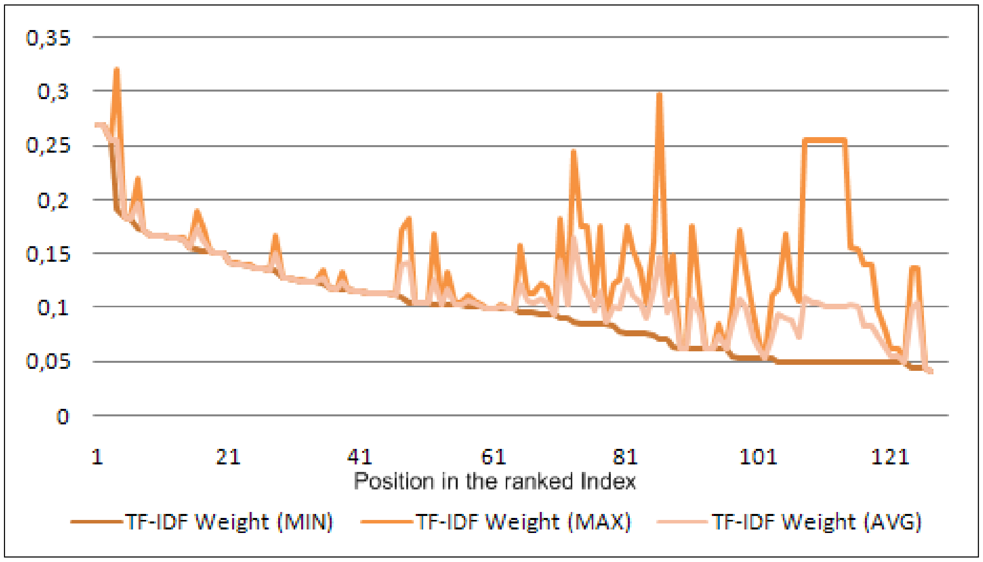


Figure 5.11: Detailed breakup of the different weight calculation for common nouns

calculation methods are strongly influenced by inhomogeneity of the weight dispersal in the words of the labels and lead to an overestimation of several concepts. But no matter which calculation method is used, a weight level which is significant for the thesaurus could not be identified.

5.1.3 Finding Candidates to Enhance a Thesaurus

By using the described method to rank the different tokens a list of all concepts is created which is ranked corresponding to their weights. As described above these indexes have certain combinations and densities of the different grammatical categories and they have a significant appearance in the ranked index. It is also apparent that it does not make a difference if a tagged or an untagged index is used.

In table 5.5 a brief overview of the amount of possible candidates, which can be extracted out of the two record sets, is given. These candidates are characterised by a weight over 0.0 and by the fact that they do not appear in any of the concepts of the used thesaurus.

Regarding only candidates that are tagged as nouns (nn, nns, nnp, nnps), at the first glance, it is difficult to separate those candidates which are more relevant and should be reordered according to their document frequency as it is planned in section 3.1. In the figures 5.13 and 5.14 the development of

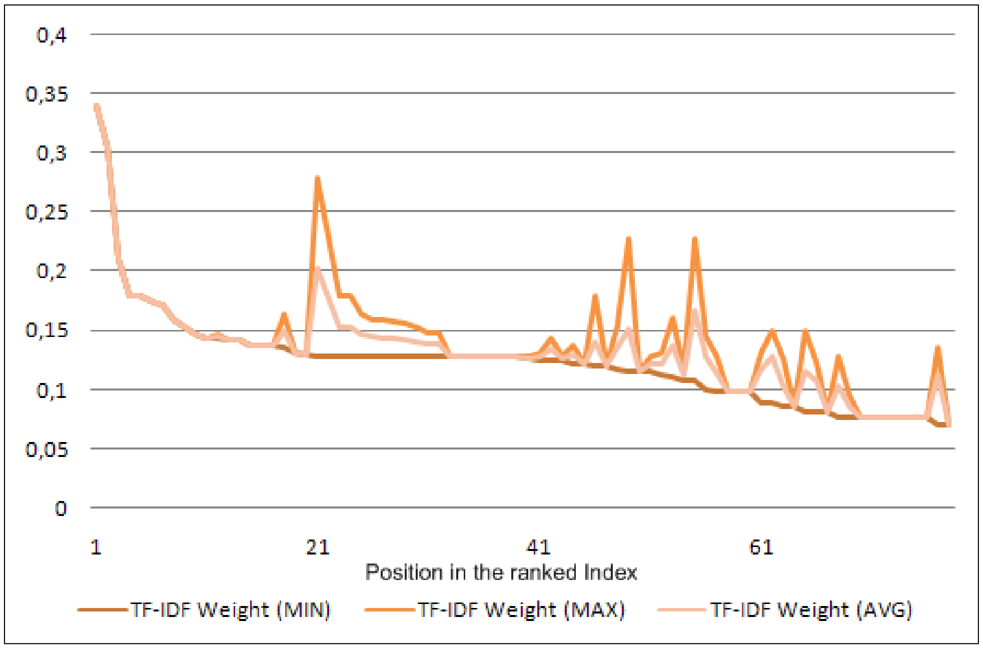


Figure 5.12: Detailed breakup of the different weight calculation for proper nouns

Anatomy		Humanities	
Untagged	Tagged (nouns)	Untagged	Tagged (nouns)
46449	61576 (35780)	17289	20726 (11297)

Table 5.5: Number of Candidates for Thesaurus Enhancement

the tf-idf weight (right side) and the document frequency (left side, ranked according to the tf-idf weight) is shown. The trait which could be deduced out of these figures is the strong decrease of the tf-idf weight over the first ranks and the strong increase of the document frequency within the last ranks.

There are different possibilities where the line for the subsequent ranking according to the document frequency can be drawn. The first idea, mentioned in the section 3.1 was a limit according to a fixed percentage of ranks. The main issue of this and the following ideas is that a limit has to be drawn, where a sufficient allocation of document frequency is available. When using for example the first 250 candidates of the anatomy index all candidates would have a document frequency of 1 and a further ranking would be unnecessary. Enlarging this limit up to the first 20 percent

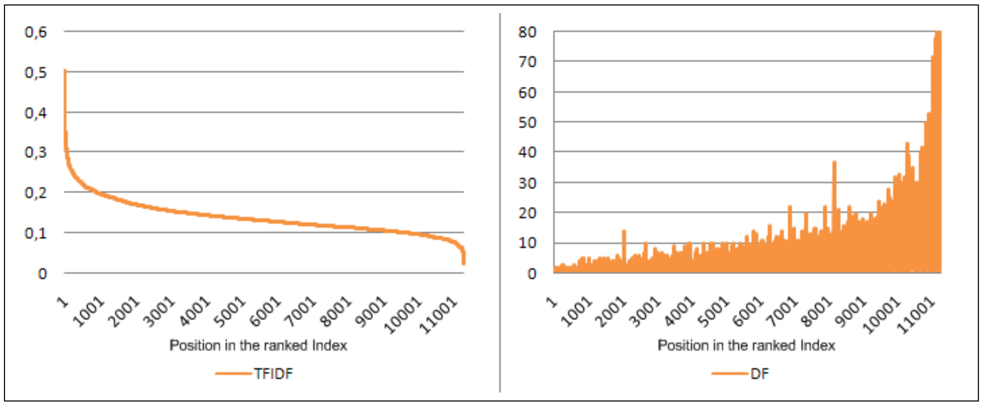


Figure 5.13: Candidates for Humanities Thesaurus Enhancement

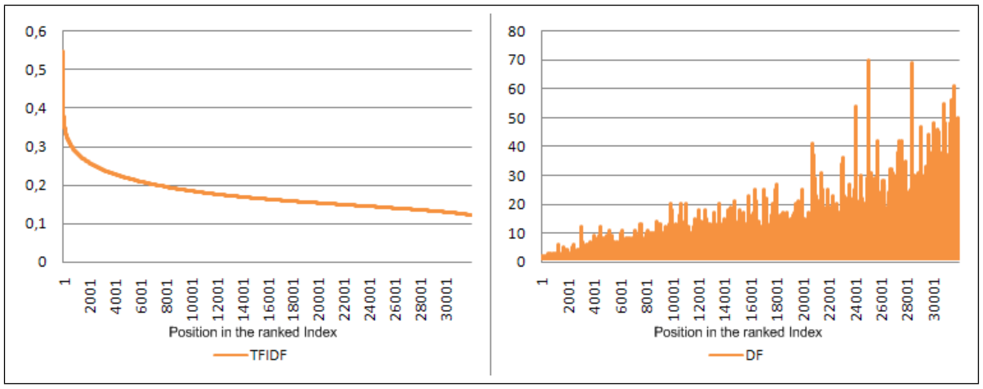


Figure 5.14: Candidates for Anatomy Thesaurus Enhancement

of the candidate corpus, which would include around 7000 candidates the maximum document frequency which can be found is 12 and the average document frequency is 1.33. This means that there are a couple of candidates which appear in more than one document. But additionally it is to state that candidate number 7,000 has just a tf-idf weight of 0.20274 (highest value in this candidate corpus is 0.54883 - which is more than the twice of this value).

Another possibility is to assign the limit of weight to the point of the tf-idf distribution curve gradient maximum (cf. left diagrams in figures 5.13 and 5.14). In figure 5.15 the inverse gradient curve of the humanities candidates is explicitly drawn. This curve always alternates between zero and points larger than zero when two tokens have the same weight. When interpolating

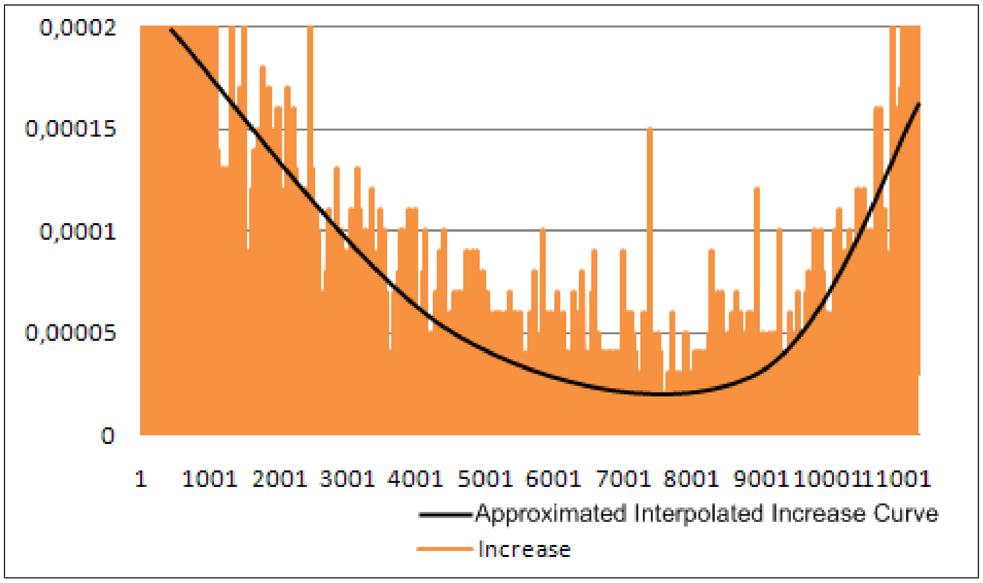


Figure 5.15: Gradient Curve of Weight Distribution of Humanities Candidates

this curve (cf. black curve) the minimum of this inverted gradient curve is located between rank 7000 and 8000 which is an percentage of around 66 percent. Here the weight of the token has decreased to 0.12.

The last alternative, which was stated in the section 3.1, is the attempt to combine document frequency and tf-idf weight of a concept to calculate a new weight. This idea is deduced from the increasing average level of the document frequency while the tf-idf weight is decreasing (cf. figures 5.13 and 5.14). Using the basic function without any further modification and the document frequency limit of one the new ranking method ranks 4,341 candidates according to their new weight. All other candidates are not taken into account because their new weight is zero. In table 5.16 the progress is drawn. Some of the high ranked candidates for the humanities thesaurus are for example *ptsd* (df = 14, tfidf = 0.1705) which is the abbreviation for *post-traumatic stress disorder* and *helicobacter* (df = 3, tfidf = 0.25308) which is kind of a bacterium.

Again *right* and *wrong* could not be determined when looking at the different possibilities but setting the focus on the fast adaption of the approaches and a minimal effort for the user the last approach, the combination of tf-idf weight and document frequency seems to be the best choice.

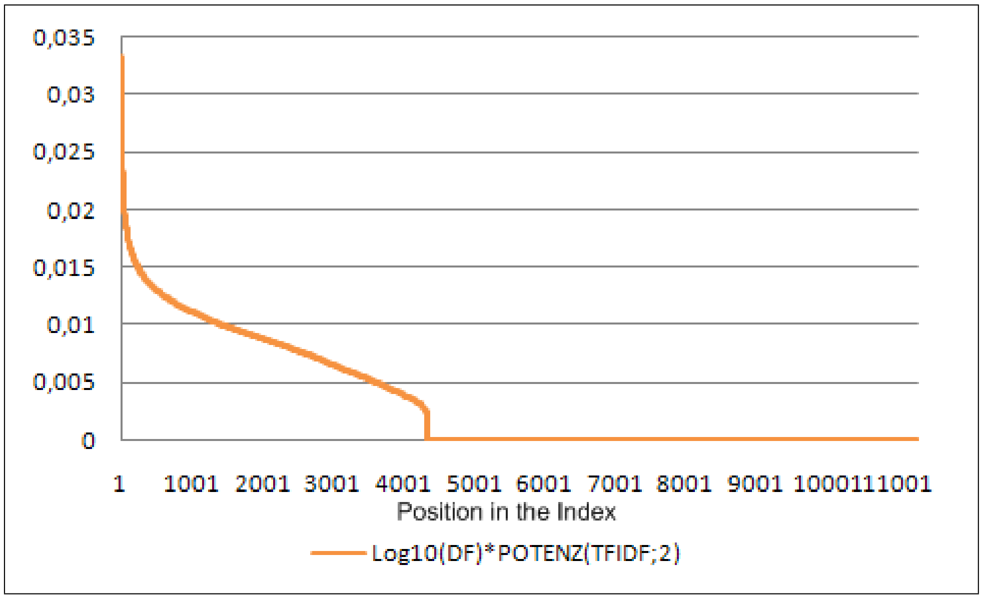


Figure 5.16: Ranked Humanities Candidates with Ranking Function

5.2 Co-occurrence in Abstracts

As mentioned above the aim of this intermediate step on the way of enhancing a thesaurus is designed to reduce the possible adding locations in the thesaurus. To calculate the surrounding of a token only the abstract of the document are used which leads to a really small amount of words occurring around each candidate. As already mentioned in section 3.1, this thesis abdicates completely on a fixed window size and a deep and extensive sentence analysis to keep the runtime of this method low.

In a first step the surrounding for each concepts is calculated which is included in the thesaurus (cf. section 5.1.2)[10]. In this case the creation of the vectors needs around 120 minutes for the complete humanities index with 12,000 nouns, what leads to a creation rate of 100 per minute. The calculated surroundings are stored in the database. This improves the similarity calculation between concepts in the thesaurus and candidates enormously. The needed surroundings are just retrieved out of the database and compared. This step takes just several seconds. Having calculated the surroundings with a large amount of attributes it is easily possible to reduce the amount without a recalculation because the items, contained in the surrounding are

[10]For this evaluation the size of the surrounding was set to 500 concepts.

ranked. The complexity of the implementation and the adaption to other systems or into a company should be possible without grave modifications. There is no special database necessary and also the used way of implementation could be translated into other programming languages as well.

The automated measuring of the correctness of this method, which is based on the MeSH thesaurus and the corresponding indexes completely failed. Caused by the chosen kind of index, the extracting of suitable synonyms and hyponyms leads to problems. Only around 40 concepts could be match one to one with the index, what is only possible if they consist out of one word could be extracted. Moreover all of these 40 matches do not lead to any mentionable accuracy of this method, because on the one hand they do not have a synonym or a hyponym and on the other hand the related labels could not be found in the index. To back this result up and avoid domain test data specific apparitions, in a second step the WordNet thesaurus was taken and synonyms were extracted containing also only one word. Different evaluations have been made with different vector length. Although using a vector length of 500 and accepting a result set size above 4,000 the accuracy did not reach the 100 percent.

This unsatisfactory result of the automatic evaluation with different thesauri has several reasons. It is possible that the pure collection of words which appear around the candidate is not sufficient for the co-occurrence analysis and that a more intensive investigation has to be done like Grefenstette did it in [12]. Another reason is maybe buried in the size and structure of abstracts. In abstracts the authors normally try to condense the information which could lead to unusual sentences. In this case the calculated surrounding of the concepts may not reflect the real conditions and create an artificial, for abstracts significant surrounding. But one of the strongest reasons is, that the thesauri and the corresponding index have a domain specific content. In this case the evaluation done with the WordNet thesaurus which has a completely different focus and a universal domain could definitely not lead to a top accuracy.

Doing some outstanding manual test with the co-occurrence analysis the problems mentioned before are pointed out. But the manual tests also show that this analysis and prefiltering of the concepts in the thesaurus works. Using the top ranked candidates from the tagged humanities index the most similar concepts are shown in table 5.6. *ptsd* the shortcut for post-traumatic stress disorder is highly ranked with *war*, *conflict* and *event* which is at the first glance surprising because there is nowhere a similarity with *disease* or *disorder*. But regarding the domain of the thesaurus *humanities* the calculated similar words can be explained. Another candidate *hallmarks* which is the umbrella term for pictures formed out of stars was fitted to *research*, *day* and *astrology* which is a sufficient base to perform the last categorization with the machine learning approach on.

Candidate	Top Ranked Similarity Words
ptsd	war, conflict, event
helicobacter	research, prize, renaissance
psychologists	psychology, awards, witnesses, law
hallmarks	research, day, astrology, sixteenth

Table 5.6: Co-occurrence Analyze of Humanities Index

Overall it is notable, that a surrounding size of 100 which exists for about 5 to 6 sentences is sufficient to ensure that each concept obtains at least over 20 similarity proposals. The outstanding aspect in these results is that the outcome really depends on the document corpora and the topic of the content. In general, the adaptability for other thesauri is given but unexpected similarity candidates could occur and moreover the expected one could not be included in the result.

5.3 Recognition of Patterns and Categorisation of Relations

This part of the evaluation is concerned with machine learning approaches. According to [3] the same circumstances are evaluated in a first step, which means that pattern, extracted based on WordNet thesaurus data are used to decide, whether two concepts are synonyms or not. Moreover the categorization between species and disjunct relations and also between synonyms and species was evaluated with the help of the WordNet patterns. These patterns can be found in the appendix B. Furthermore new patterns, extracted based on MeSH thesaurus data were used to analyse all three different categorizations for a more specific thesaurus. In addition to [3] all evaluations were also done with a decision tree approach.

Before considering the accuracy of these methods it is important to mention that for the creation of one attribute vector, which represents the relation between two concepts three independent web search engine requests are necessary. The time which is needed for one request depends a bit on the used web search engines but the whole process takes around 1 second, which leads to a creation time of 3 seconds per vector. This time does not really depend on the size of the vector because only a fractional amount of time is needed to parse all results.

5.3.1 Machine Learning for Relation Categorization

Different sets of patterns have been collected for this thesis because the machine learning should not only be performed for the categorization of synonyms but also for species. It is important that for every combination of relations (synonyms vs. disjunct, species vs. disjunct, synonyms vs. species), the patterns which are used to create the attribute vectors have to be selected newly, depending on their content of information for this specific combination. The complete list of the used patterns can be found in the appendix B. With these patterns, training data set have been collected. The three similarity measures are added to the dimensions, formed by the patterns. At this point it is notable, that two different ways were used to create the training vectors. On the one hand, for all word pairs which were taken the corresponding vector has been stored and used to train the machines. This leads to a lot of vectors which have really small values in the first three dimensions (the similarity measures) and no other occurrence of the other dimensions. On the other hand training data was captured by skipping such almost *zero vectors*. Strictly spoken the second procedural method is a corruption of the data but later evaluations with the machine learning approaches showed that this corrupted data works better.

First evaluations on the different machines with different settings were done with the extracted data. It turned out that three different combinations are interesting for this thesis: the supported vector machine with a linear kernel and a radial basis function kernel and the decision tree J48. An overview of the outcome can be found in table 5.7[11].

The used decision tree approach (J48) delivers more satisfying results in all cases. Overall the correctness level is around 11.6 percentage points better than the linear supported vectors machine and 24.1 percentage points better than the radial basis function based support vector machine. Comparing the results of the different thesauri the WordNet data is more stable and the produced results are 10 percentage points better than the results of the MeSH data. This could be a hint, that the outcome of the pattern recognition approach is influenced by the domain and the detailedness of the used thesaurus.

[11]The table shows the degree of correctness using the respective trainings data and a 10 folds cross validation. SVM configuration stays the same except the kernel type is switched between linear and radial basis function.

Trainings Data	SVM (lin)	SVM (RBF)	Decision Tree
WordNet - Synonym/Disjunct	86 %	54 %	98 %
WordNet - Subclass/Disjunct	73 %	63 %	82 %
WordNet - Synonym/Subclass	70 %	50 %	71 %
WordNet - Synonym/Subclass/Disjunct	58 %	47 %	70 %
MeSH - Synonym/Disjunct	71 %	59 %	85 %
MeSH - Subclass/Disjunct	74 %	60 %	87 %
MeSH - Synonym/Subclass	53 %	52 %	68 %
MeSH - Synonym/Subclass/Disjunct	51 %	40 %	68 %

Table 5.7: Correctness of Trained Machine Learning Approaches

A significant difference came up while analysing the failure of both machines. The supported vector machine approaches is mostly not prepared to take any risks concerning the first class. This means in the case of categorizing between synonym and disjunct relations or species and disjunct relations the most failures were made by the machine while categorizing synonyms or species. Either a few or no failures occur during the categorization of disjunct relations most of the time. This means particularly, if the SVM categorizes a relation as synonym or a species the correctness is even higher than the overall correctness (cf. table 5.8)[12]. From this it follows that the overall percentage of right assigned categories is lower using a linear supported vector machine but if the linear supported vector machine categorize a relation as synonym or species the chance that the machine is wrong is smaller than using a decision tree, where the failures are distributes equally.

In a separate evaluation the direct decision between the relations synonym and species was figured out. Caused by the missing accuracy of the machine learning approaches in connection with the training data it is not possible to decide with the SVM or the decision tree to which category a relation depends for MeSH data. Although the accuracy of WordNet data is much higher for this decision it is still just around 70 percent and a decision based on this approach is not recommended.

[12]For the comparison between species and synonyms the data was not sufficient to calculate a founded value, taking MeSH patterns.

	MeSH		WordNet	
Relations	SVM(lin)	J48	SVM(lin)	J48
synonym vs. disjunct	96.4 %	87.3 %	92.1 %	93.3 %
subclass vs. disjunct	93.9 %	85.9 %	89.4 %	84.6 %
synonym vs. subclass	-	-	89.3 %	75 %

Table 5.8: Correctness of SVM and Decision Tree according to synonyms or hyponyms versus disjunct relation categorization

5.3.2 Information Content of Patterns

In a second step the different methods of operations are analysed in connection with the used input vectors.

The supported vector machine uses all the dimensions of the input vectors to learn the separating hyperplane between the different categories. To rank these dimensions according to the influence they have on the final results a SVM specific dimension evaluation is used.

One of the most useful and highest ranked dimensions for synonym and species relation decisions is the WebOverlap which has an average rank of 2.40. Using only the overlap pattern to decide the category of a relation, the linear SVM has still a correctness of 68 percent (for species relation decisions on the WordNet data). The WebDice, which was ranked highly in the paper of Bollegala et al. has just an average rank of 26.5 and is only in the second quarter (WebJaccard has an average rank of 43.5). Regarding the tables 5.9 and 5.10, which provide an overview of the important dimensions for the SVM, it stands out that most of the high ranked patterns have either a high x^2 - value or a really low one.

Although a lot of SVM high ranked patterns have a low x^2 - value they are needed to draw the separating hyperplane between the categories. According to some additional evaluations with the species decisions on the WordNet thesaurus the correctness of the machine does not decrease at all if the dimensions 20 to 70 are removed. Although using just the first 10 and the last 5 dimensions the correctness just decrease by 2 percentage points.

The decision tree approach (J48) mostly generates results with a higher correctness. Although this method is more precise it uses less dimensions to calculate the category of a vector. Regarding the decision trees in figure

Pattern	Average Rank	Rank according to x^2 - value
X, Z	1	4
X optical Z	2	83
WebOverlap	3.8	-
X/ Z	4	82
X , Z	5	11
X (also Z	6	81
X (Z	7.2	5
X), Z	8	80
WebDice	8.4	-

Table 5.9: WordNet synonym pattern ranking according to support vector machine evaluation

Pattern	Average Rank	Rank according to x^2 - value
X, see Z	1	83
X. n. Z	3	82
WebOverlap	3	-
X, Z	3.4	4
X ‘Z	5	81
X. Z	5.6	7
X by a Z	7	80

Table 5.10: WordNet subclass pattern ranking according to support vector machine evaluation

5.17 and 5.18 of the synonym and species relation decision on the MeSH the proposition mentioned above is underlined.

Especially in the synonym decision tree it stands out that after four decisions (three of them are based on similar measures) 90 percent of all vectors could be categorized (cf. red circles). Overall this tree just uses four dimensions at all. In the species decision tree six different patterns 85 percent could be categorized after evaluating. Something remarkable in the decision trees is that all used patterns in the trees are out of the first 8 ranked patterns according to the x^2 - value ranking and that all similarity measures are used to calculate a decision. Surprisingly, when regarding the

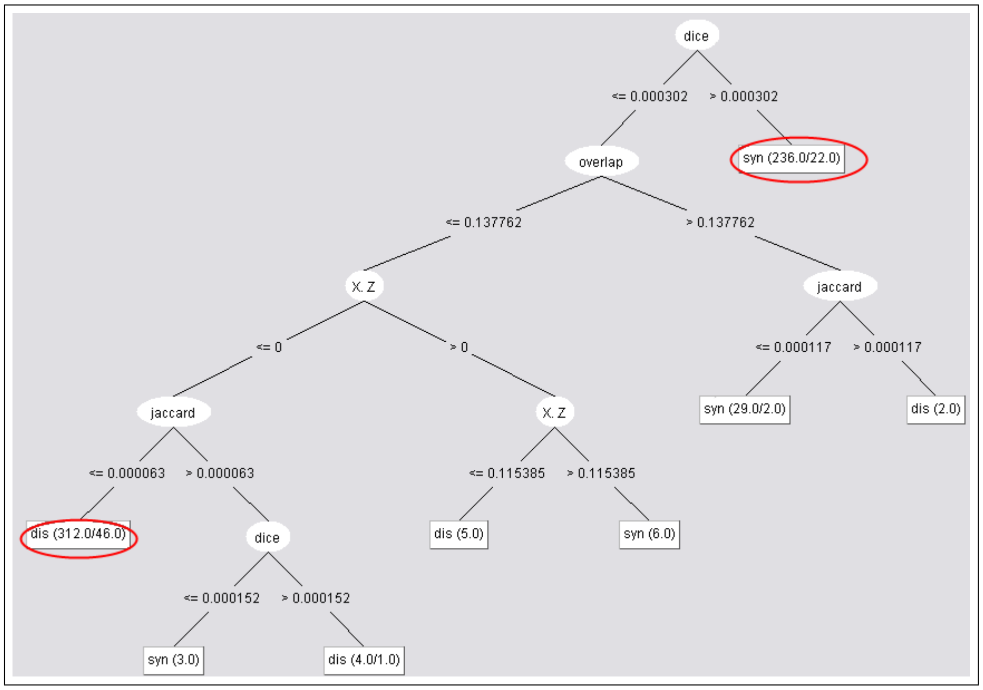


Figure 5.17: Decision Tree for synonym categorization on MeSH Data

synonym decision tree (cf. figure 5.19) based on WordNet, the *WebJaccard* similarity measure is used on a important node although it was weighted by the supported vector machine evaluation as one of the lowest.

5.4 Enhancing a Thesaurus

In a final experiment the combination of the three different approaches is evaluated as an aggregated approach. Applying the weighting techniques, which combines tf-idf weight and document frequency of candidates from the humanity index a list of possible concepts, which are not included in the humanity thesaurus yet, is created. The top 100 candidates of this list can be found in the appendix B. Regarding these candidates through the eyes of a non-domain specialist they all seem to be somehow relevant for the humanity thesaurus and there are almost no terms, striking directly to the eye, that are misplaced in this list. Except the tokens *ms* (position 74), *mrs* (position 78) and *or* (position 97) do maybe not fit in here, because actually they are stopwords or general concepts. But all of them are tagged as proper nouns which could lead to believe that these terms could also be a shortcut (e.g. *or* for operating room). Additionally it is to mention that a lot of shortcuts as *cbt (cognitive behaviour therapy)* or *mdd (medical device directive)* can be found in the top 100 candidates.

In the following the top 10 candidates are used to perform the next step of the approach.

1	ptsd	6	arts
2	helicobacter	7	asthma
3	psychologists	8	stressors
4	hallmarks	9	cbt
5	memories	10	faculty.

Using the co-occurrence approach for these 10 candidates to identify possible adding locations in average 59.4 locations were returned. All of these calculated surrounding include terms which are relevant for the candidate. The following three candidates and their surrounding extracts provide a brief example of the results.

Token	Tokens included in the surrounding
ptsd	events, war, conflict, attacks, history, ...
helicobacter	prize, research, renaissance, world, ...
psychologist	psychology, awards, witnesses, law, sex, ...

In the example some of the adding locations do not have anything in common with the candidate (cf. *helicobacter*, firstly discovered in 1982, and *renaissance*). But others are obviously connected (cf. *psychologist* and *psychology*). In addition, when regarding the topic of the humanities thesaurus most of the suggestions for *ptsd* make sense.

Finally for all the combinations out of the 10 candidates and the corresponding adding locations the pattern vectors are created and entered into a trained linear supported vector machine[13]. The result of the machine learning approach always reduced the possible number of adding locations and only in one case, which is explained also later, none of the adding locations, suggested by the co-occurrence approach, was categorized as synonym or species relation.

[13]The SVM is used because of the higher accuracy concerning first class categorization.

In the following three different candidates are selected to demonstrate the possible outcomes and the evaluation of the whole approach. An example where a new structure in the thesaurus could be discovered is the *ptsd* concept. The 44 suggestions from the co-occurrence approach were reduced to 20. 11 of these 20 are supposed to be synonyms (e.g. *war* and *attacks*). The remaining nine are supposed to be genus/species (e.g. *research*). It is obviously that *war* is not a synonym for *ptsd* but for this pair the first two SVM models supposed it is either a synonym or a genus/species relation. The third model belatedly supposed it is a synonym relation. Caused by the low accuracy of this model this categorization may be a failure. Additionally the results are a good example for complete under-determined structure because *ptsd* is a disorder and disorders are not yet represented in the humanity index anyhow. Another example, where a structure could not be discovered is *helicobacter*. In fact, this is not really a failure because a bacterium does not fit in the concept of *humanities*. The third example *psychologist*, which has obviously something to do with *humanities*, has a result set out of 25 possible adding locations after the categorization. Eight of them are supposed to be genus/species categories (e.g. *surgeon* or *nursing*). The relation to *psychology* was categorized as synonym relation. Relations which were categorized as disjunct are e.g. *witnesses* and *awards*.

In general it is notable, that the division between synonym and genus /species relations should not be trusted because it is mainly based on the last SVM model which just has an accuracy below 50 percent for the MeSH thesaurus. The reason for this is maybe founded in the undefined stripline between those categories. Moreover the lower accuracy of the MeSH thesaurus models influences the outcome of the categorization.

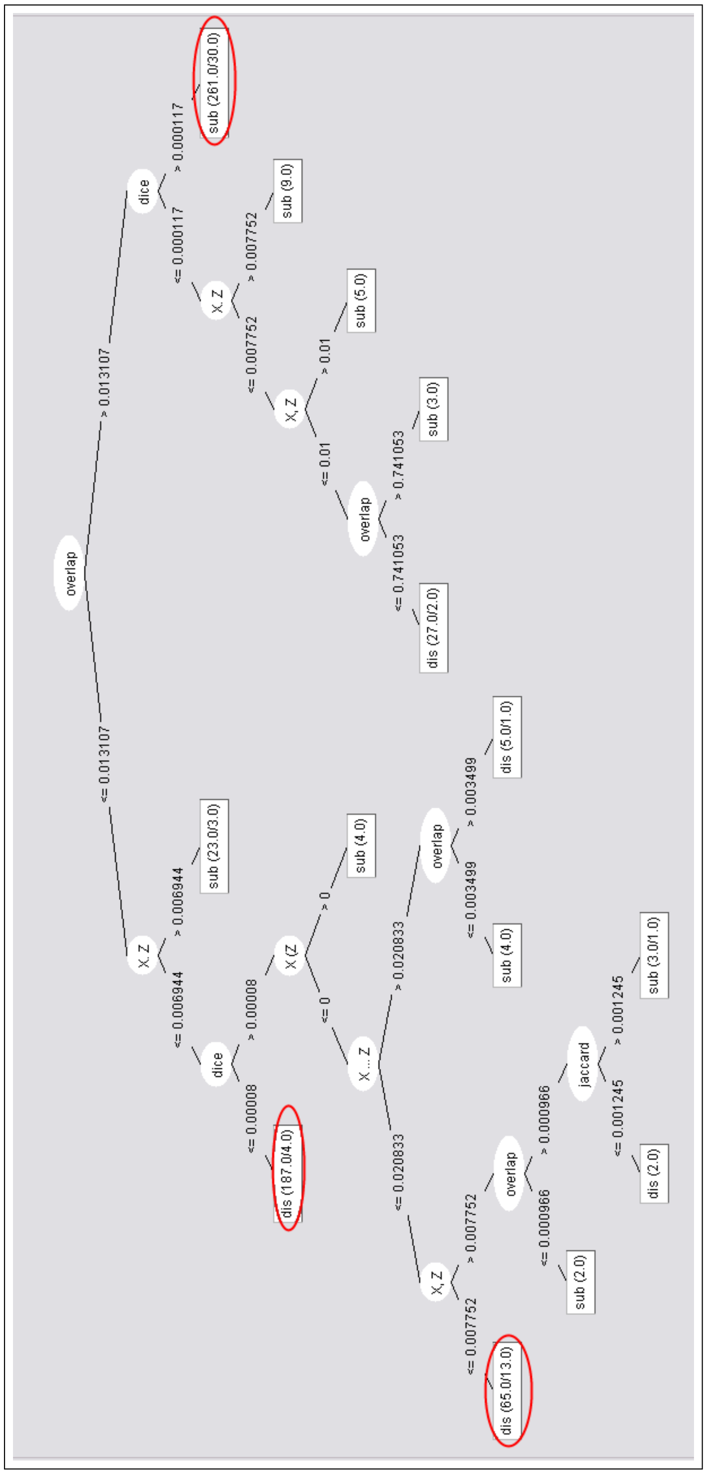


Figure 5.18: Decision Tree for subclass categorization on MeSH Data

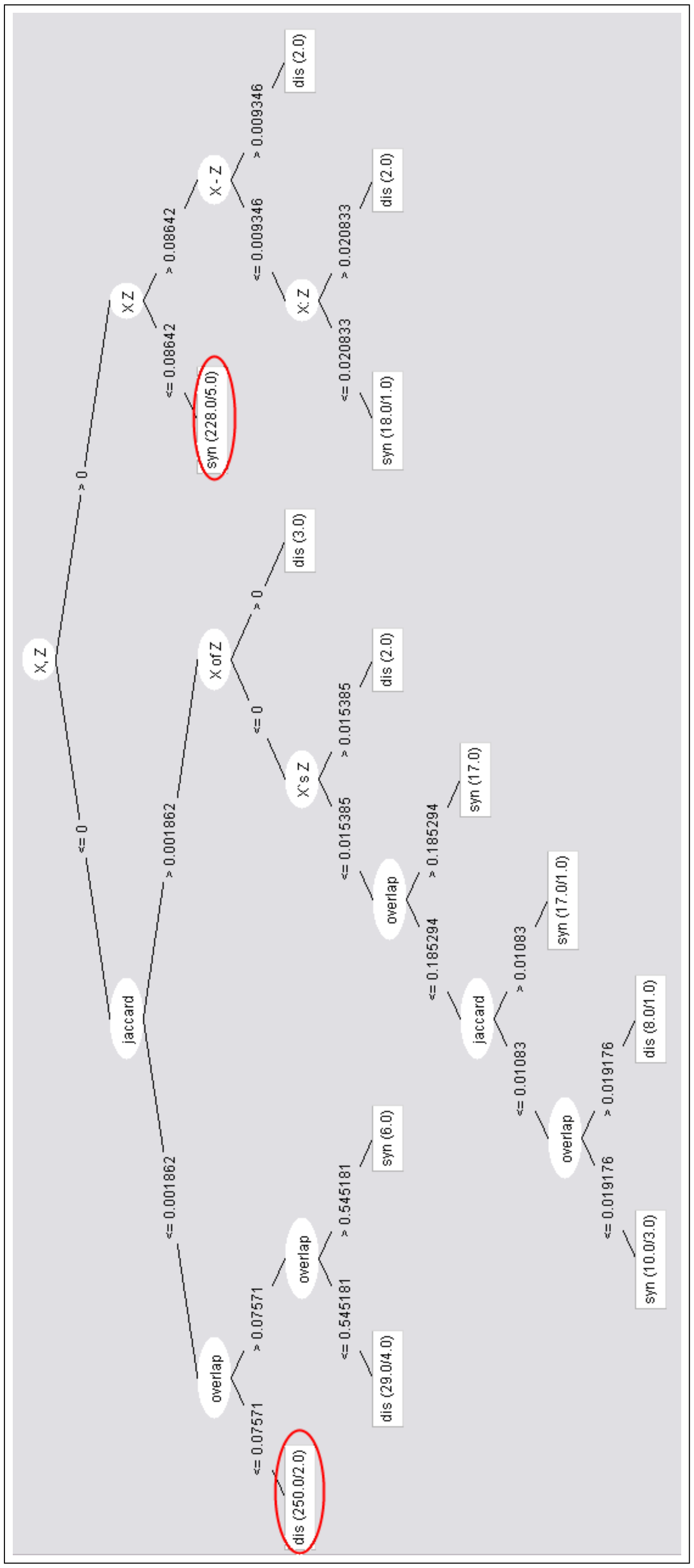


Figure 5.19: Decision Tree for synonym categorization on WN Data

Chapter 6

Conclusion

As mentioned in the introduction, this thesis describes an approach to replace the manual process of thesauri enhancement partly as explained in section 3.1. The three general methods, which form the approach, are:

- A modified tf-idf weighting method to identify concepts for enhancing the thesaurus, which is shown in section 3.1.1.
- A co-occurrence approach to reduce the amount of concepts in the thesaurus where an identified concept could be added to (cf. section 3.1.4).
- A final categorization of the relation between two concepts with a pattern recognition approach based on machine learning, which is explained in section 3.1.5.

All these three methods meet the requirements of this thesis (cf. chapter 5) and the complete approach supports the process of enhancing thesauri with new concepts and provides a well-founded selection of possible adding locations as the results in section 5.4 underline. But the final decision about the relation between two concepts has still to be done by ourselves.

In the following, the specific questions concerning the problem description from section 2.7 are answered in detail.

Could the tf-idf weighting method or a modification be used to identify and rank possible concepts out of a document corpus with the goal to enhance an existing thesaurus? As seen in the results in section 5.1, the tf-idf weighting method is primarily a method to weight tokens in a document. The assumption that the global weight of a token could be calculated by the average tf-idf weight of all appearance of this concept was useful and could satisfy the demand of extracting the most relevant concepts for the corpus, which is shown by the results in section 5.1.3. The reason for this is due the high weights of tokens which only appear in one document and the average of them is similar to the single weight in this specific document. The presented possibilities to modify the raw weight of the tokens pervade sufficiently the demand. Although the recalculation of the weight of concepts by using the global tf-idf weight in combination with the document frequency (cf. equation 3.3) works in the most sufficient way. The equation allows setting a level for the document frequency and only concepts with a document frequency above this level will be taken into account. The method that ignores concepts which does not have a sufficient tf-idf weight and reorders the sufficient concepts according to their document frequency could also be used (cf. section 3.1.1). But it is recommended to use the maximal gradient of the tf-idf weigt development to determine the weight limit. The advantage of this limit identification method is the higher feasibility of automation in comparison to the manual assignment of the level.

Regarding this whole part in terms of adaptability and runtime it is notable that this method is easy to adapt to an existing system because the indexing methods can be found in almost every programming language preimplemented. The runtime, which was really high in this thesis is in real world practice not that high and also not a bottle-neck because not all documents are index at once. They could be indexes just when they are available. Also it would be possible to execute the index event over night or in background. Once the documents are indexed and the results are stored only the global weights need to be recalculated occasionally.

One of the largest problems which influence the later discussed co-occurrence analysis and the calculation of similarity are founded in the tokenization of the index. Regarding complex thesauri like the MeSH most of the concepts consists out of more than one word. This makes the matching of thesaurus and index concepts really difficult as described in section 3.1.2. The approximation of taking the minimal weight of nouns included in the thesaurus maybe is not a completely admissible method but under the circumstances of this thesis it is feasible. Using another kind of index - for example a positional index - the weights of the concepts would even be lower and a big amount of concepts would not be matchable at all. There is no significance in the used concepts of a thesaurus according to the devolution of tf-idf weights in the two different analysed thesauri. But some of the preferred labels which are used have a significant lower weight than their alternative labels, and so it is questionable if the actual preferred label is really the best choice (cf. section 5.1.3).

Additionally the results of the evaluation underline that it is not necessary to use a tagged index if using the tf-idf weighting method because most of the high ranked concepts are nouns.

Are the information provided by an abstract of a document enough to calculate the similarity between two concepts using a co-occurrence approach? In the first place the specific structure of an abstract was outstanding. Although most of the important words of a document are included in this abstract, the structure of the sentences is not the structure which is normally used. This is one of the reasons why the co-occurrence method can lead to unexpected results of the co-occurrence analysis as it is comprehensible in section 5.2. This unpredictability makes it really difficult to evaluate this method because *right* or *wrong* mostly depends on the point of view of the spectator or on the topical orientation of the thesaurus. Although this artificial surrounding is present it is possible to extract adding locations which share a semantic relation with the candidates. Another fact which influences the quality of the results is the flat sentence and surrounding analysis (cf. section 3.1.4) which was used in this thesis. Furthermore the number of possible adding locations can be controlled by reducing or increasing the amount of items included in the surrounding.

Another point which was already mentioned above is concerned with the kind of index which was used. The surrounding was always calculated for a single token out of the index and not for a complete concept. The combination of the surroundings of all in a concept included tokens may lead to even more unexpected results, because tokens could have more than one meaning and with a tokenized, non positional index it is not measurable if the tokens appear in the same content as in the concept.

Is the method of pattern recognition out of web search engines snippets also suitable for specific thesauri and could the analysis which is done by a supported vector machine decide between subclass and synonym? As seen in the paper of Bollegala et al. [3] the pattern extraction based on WordNet data works really sufficient. In this thesis a correctness of 98 percent was achieved with the extracted training data. Regarding the MeSH based extracted patterns and achieved correctness the conclusion of Bollegala et al. that this method could be used to calculate synonyms cannot be confirmed absolutely. In the evaluations done in section 5.3, mostly the correctness lies above the one of WordNet which is absolutely founded in the complexity of concepts. Also this thesaurus is really specific and the similarity measure which are based on the page count are still significant for the later machine learning approaches the values are quite small and the page counts decrease going deeper into the thesaurus. Additionally the collection and identification of significant patterns in the snippets for the more specific MeSH thesaurus was more difficult than for the really general WordNet thesaurus. To reach a sufficient level of patterns in this thesis, which are listed in the tables of appendix B also patterns including the three significant dots have to be taken into account. This was not necessary for the patterns collected with WordNet. But it was also not possible to use the patterns which were collected with WordNet on the MeSH thesaurus. Here the correctness of the later used machine learning approach decreases. If the method of pattern recognition based on machine learning as explained in section 3.1.5 should be adapted to a thesaurus the patterns have to be extracted and identified for this specific thesaurus again to achieve better results.

Regarding the second part of the question it is notable that a direct decision upon a synonym, genus/species or disjunct relations between two concepts is almost not possible with the methods used and the data which was extracted. The most common fault which occurs during the categorisation is the confusion of the SVM between synonym and genus/species relations. Analysing the issues in separate categorisation steps is maybe possible. In a first step training the supported vector machine on synonyms versus disjunct and subclass versus disjunct relations where it has a sufficient degree of correctness shown in table 5.7. If the relation is categorized into one of the first classes (synonym or genus/species) using a SVM the correctness of this decision is really high according to table 5.8. If the token is categorized into both or none of the categories the incorrectness grows because most failures of the SVM are made while categorizing synonym/genus/specie as disjunct relation and also the categorization between genus/species and synonym do not have the high degree of correctness of the other two categorizations. Using the MeSH as base it is almost better to role the dice than relying on the result of the support vector machine.

This leads to a final point which is really grave. In all the tests of this thesis the decision tree delivers better results than the supported vector machine. Moreover it uses less attributes of the vectors. Unfortunately the allocation of failures is really balanced which is not necessarily an advantage. But overall if a high correctness is the goal the decision tree is the better choice to categorize the tokens.

Is the whole process adaptable for real world organisations to reduce the costs of maintaining their thesauri? The last question which also connects somehow the outcomes of the previous questions with the motivation of this thesis can be answered affirmatively. The whole process is adaptable to different programming languages with some effort and also the runtime should not be the bottleneck in a real world scenario. As a supporting tool for the responsible person the method works sufficient, but an exact decision has to be finally done by the users themselves.

The indexing part which requires the most time at all could be done piece by piece as described above. The calculation of the co-occurrence vectors needs a time but just have to be done once and is possible in background. Accessing the web search engines with a limited amount of possible linking points it is possible to execute and walk through the complete process needing less time than doing all manually. Also it is possible, that a person, who is not an expert for thesauri, can do the final assignment.

Chapter 7

Future Work

Two different aspects influence the results of the approach presented in this thesis and are worth to do deeper investigations in. On the one hand the kind of index, which is used for the document corpora and on the other hand the way the surroundings of the tokens are calculated. Improving these topics may lead to a better final result of the complete approach. Moreover additional investigations in the collection of patterns for the decision tree could increase quality of the results too.

Using a positional index could lead to a more sophisticated analysis on the one hand of the thesaurus itself and on the other of the index. It would be also helpful to have a kind of phrase detection to avoid the tokenization of concepts which belong semantically together (e.g. *viral marketing*). Combining these two techniques a more sophisticated analysis should be possible. The data corpus unfortunately has to be extended to these new circumstances and also it would possible be not sufficient to capture only ten documents per thesaurus concept anymore but even 100 to provide a matching of a sufficient amount of concepts in the thesaurus.

Investigating in the sentence structure could be a helpful improvement to increase the hitting precision of the co-occurrence method for abstracts. At the moment this precision strongly depends on the topical orientation of the document corpus and the thesaurus. By increasing the sentence analysis it could be possible to reduce the influence of the topic and to make the similarity calculation more general.

Improving the information content of patterns in respect to the decision tree could increase the output of this clustering approach. Caused by the approach in [3] the method of identifying and collecting patterns was mainly focus on the usage for the SVM. The issues, that maybe other patterns or another selection process is more supportive for decision trees is currently not considered in this thesis.

Appendix A

Thencer - Detailed Module Overview

The table A.1 gives a short overview about the main module and the included classes and functionalities. The table A.2 shows an overview about the other additional modules which were implemented for this thesis.

Java Class (alphabetically sorted)	Description
AnnotationEnricher	Uses an imported thesaurus (MeSH) and fetches with the help of Pubmed for every concept (only preferred label) 10 records which have an abstract and the MeSH mapping filled. As input the thesaurus could be entered directly or an existing annotation set could be used. If the annotation set is inserted, the underlining thesaurus is used. Additionally it is checked if there are already records in the annotation set included which belong to a concept. The records are stored into the semtinel database.
ConceptMatcher	Uses an imported thesaurus and matches its concepts with an index, which was created before. The matching is based string similarity of the tokens of the index and the single tokens of the concept labels. Preferred labels and Alternative Labels are taken into account.

Table A.1: Thencer Data Manager Module Overview

Java Class (alphabetically sorted)	Description
Index	Helper class which represents an index object.
IndexInserter	Handles the transactions between database and java runtime environment concerning the index topic. Includes also some prepared statements which return data from the database.
KeyVal	Helper class which implements a comparator who compares depending on the value of the object.
PatternCollector	Manages the whole pattern collecting process. Uses an imported thesaurus to collect for a specified relation (synonym, subclass or disjunct) patterns from the Yahoo web search engine. Stores the data into the database.
PatternInserter	Handles the transactions between database and Java runtime environment concerning the pattern topic. Includes also some prepared statements which return data from the database.
Sextant	Manages the calculation of the surrounding of tokens and the calculation of the similarity measure based on a weighted Jaccard measure.
SextantEval	A methods which uses the sextant and a thesaurus as input to automatically generate an evaluation for the calculated surrounding vectors. Here only concepts out of the thesaurus are taken which consists of one single token.
ThesaurusExtractor	Extracts out of an imported thesaurus a sub thesaurus starting from a specified concept.
SVMPredicter	Implementation from the `libsvm`. Uses an existing SVM model and a test data set to calculate the accuracy of the model.
SVMRun	Categorizes a input vector with the help of a precreated model.
SVMTrainer	Creates a SVM model using a train vectors.

Table A.1: Thencer Data Manager Module Overview

Java Class (alphabetically sorted)	Description
SVMUtil	Calculates the contingency table and the ranking of the patterns. Also includes methods to collect test and training vectors and stores them into plain text files (This includes also functionality which read pattern files and creates the dimensions of the vectors.).

Table A.1: Thencer Data Manager Module Overview

Module Name	Java Class (alph. sorted)	Description
TFIDF	Indexer	Handles the indexing of an imported annotation set. Runs throw all the records in the annotation set and decompose the abstracts into tokens. Function could be adjusted so that the abstracts are tagged with the help of the Stanford POS-tagger before the abstracts are decomposed. Term-frequency, document-frequency and tf-idf weight are stored into the database.
RDF/OWL /WordNET Import/Export	OwlConcept	Represents a concept, which is similar to the concept which is included in an imported thesaurus. Similar to ConceptImpl but public.
	OwlExporter	Exports an imported thesaurus into an owl xml file. rdfs:subClassOf = getBroader, oboInOwl:Synonym = getAltLabels, oboInOwl:hasDefinition = getScopeNote
	RDFImporter	Imports a thesaurus from an owl/rdf xml file into the semtinal framework. Actual the relations could not be adjusted via GUI and have to be changed in the function itself.

Table A.2: Additional Thencer Module Overview

Module Name	Java Class (alph. sorted)	Description
	WordNetImporter	Imports the WordNet, whose database files have to be anywhere locally on the computer into the Semtinal framework. WordNet synsets are represented by concepts. Always the first word in the synset is used as preferred label.
Search-Engines	googleSearch	API implementation to access Google search API via Java. The maximum of returned value is at the moment 64, but Google does not restrict the number of requests per day. The results are returned from google in a JSON object.
	wordNetSearch	Access the locally stored WordNet database file and returns search requests.
	yahooSearch	API implementation to access Yahoo search API via Java. This API allows to request as many results as possible but according to Yahoo only 1000 requests per day per application ip are allowed. The results are also returned via JSON objects.

Table A.2: Additional Thencer Module Overview

Appendix B

Collected Data

Extracted Patterns for MeSH and WordNet

The tables B.1 and B.2 show the extracted patterns, ranked according to their x^2 value for MeSH and WordNet thesaurus, which were used to create the attribute vectors for the machine learning approach. The patterns differ depending on the chosen relations.

Dim	Synonyms vs. Disjuncts	SubClass vs. Disjuncts	Synonym vs. SubClass
1	X, Z	X, Z	X, Z
2	X (Z	X and Z	X (Z
3	X. Z	X Z	X of Z
4	X or Z	X. Z	X and Z
5	X; Z	X or Z	X , Z
6	X of Z	X/Z	X Z
7	X/Z	X (Z	X. Z
8	X , Z	X: Z	X; Z
9	X Z	X; Z	X mean?) Z
10	X and Z	X of Z	X. encyclopedia: Z
11	X: Z	X - Z	X of the Z
12	X, or Z	X, or Z	X (redirected from Z
13	X on Z	X / Z	X, also called Z
14	X mean?) Z	X, and Z	X, also known as Z
15	X - Z	X. encyclopedia: Z	X , Z
16	X / Z	X of the Z	X or Z
17	X’s Z	X, the Z	X, and Z
18	X, also known as Z	X on Z	X. derived forms: Z
19	X (or Z	X and the Z	X. from wikipedia, the free encyclopedia (redirected from Z
20	X (redirected from Z	X &amp;gt; Z	X in Z
21	X-Z	X-Z	X (or Z
22	X of ... Z	X. the Z	X, or Z
23	X, also called Z	X the Z	X takes Z
24	X, more... additional searches for Z	X of something that shines with reflected Z	X [syn: Z
25	X. from wikipedia, the free encyclopedia (redirected from Z	X in Z	X. symbol: Z
26	X , Z	X; ”in the Z	X fire Z
27	X. derived forms: Z	X [syn: Z	X. hypernyms (”Z
28	X, the Z	X’s Z	X also Z
29	X the Z	X, respect, Z	X. search for Z

Table B.1: Patterns for WordNet thesaurus

Dim	Synonyms vs. Disjuncts	SubClass vs. Disjuncts	Synonym vs. SubClass
30	X. learn more about ”Z	X is the Z	X (Z
31	X is a Z	X takes Z	X. retrieved from ”http://en.wiktionary.org/wiki/Z
32	X,Z	X fire Z	X in the Z
33	X. retrieved from ”http://en.wiktionary.org/wiki/Z	X. define Z	X is available with a prescription under the brand name Z
34	X ... Z	X antonyms. information about ... Z	X. define Z
35	X in the free online english ... Z	X is Z	X, i'm a Z
36	X, a Z	X before Z	X (generic), Z
37	X. symbol: Z	X after Z	X is also known as Z
38	X. hypernyms (”Z	X — Z	X(Z
39	X] ... definition: Z	X,Z	X. brands: Z
40	X (Z	X” Z	X(s) or Z
41	X also Z	X (or Z	X. learn more about ”Z
42	X and a Z	X. respect. Z	X - oral (Z
43	X. search for Z	X. lead poisoning. Z	X, interruption, Z
44	X(Z	X: a Z	X toZ
45	X — Z	X Z	X. and Z
46	X is Z	X, selfhood, selfness, Z	X synonyms. Z
47	X is available with a prescription under the brand name Z	X called Z	X (also called Z
48	Xs. Z	X regarding Z	X. why is Z
49	X” or ”Z	X character or Z	X. symbol. Z
50	X Z	X to the Z	X (dZ
51	X is also known as Z	X = Z	X” or ”Z
52	X, i'm a Z	X [Z	X” and ”Z
53	X is the Z	X rule score Z	X and a Z
54	X (generic), Z	X ”Z	X (also spelled Z
55	X and the Z	X.Z	X, a Z
56	X” and ”Z	X as Z	X called Z
57	X. brands: Z	X is called Z	X. brand names: Z
58	X(s) or Z	X , Z	Xs, Z
59	X Z	X at Z	X is called Z
60	X = Z	X is a Z	X at Z
61	X (also called Z	Xmen of Z	X such as Z
62	X [Z	X: the evolution of Z	X. brand name: Z
63	X. n. Z	X toZ	X, symbol Z
64	X - oral (Z	X and personal Z	X meaning ”Z
65	X. the Z	X as a Z	X Z
66	X, interruption, Z	X (”Z	X (also Z
67	X synonyms. Z	X would Z	X (also known as Z
68	X. why is Z	X, insubordination, Z	Xs. Z
69	X. symbol. Z	X:Z	X). buy Z
70	X (also known as Z	X. learn more about ”Z	X is a Z
71	X (dZ	X. synonyms: Z	X, 'Z
72	X (”Z	X and. Z	X and the Z
73	X”, ”Z	X? Z	X. i'm a Z
74	X,” ”Z	X (the Z	X's foot, Z
75	X (also spelled Z	X. a Z	X. dZ
76	X. brand names: Z	Xed Z	X. view all brands. Z
77	X), Z	X by a Z	X; a Z
78	X (also Z	X 'Z	X's and Z
79	X/ Z	X. n. Z	X:Z
80	X optical Z	X, see Z	X”, ”Z

Table B.1: Patterns for WordNet thesaurus

Dim	Synonyms vs. Disjuncts	SubClass vs. Disjuncts	Synonym vs. SubClass
1	X (Z	X. Z	X. Z
2	X: Z	X + Z	X (Z
3	X. ... Z	X: Z	X) 0 (Z
4	X. Z	X) 0 (Z	X information including symptoms, causes, diseases, ... Z
5	X, Z	X. ... Z	X, Z

Table B.2: Patterns for MeSH thesaurus

Dim	Synonyms vs. Disjuncts	SubClass vs. Disjuncts	Synonym vs. SubClass
6	X; Z	X and Z	X,Z
7	X or Z	X (Z	X or Z
8	X: ... synonyms: Z	X, Z	X; Z
9	X of Z	X; Z	X. genus: Z
10	X and Z	X. scope note: ... Z	X. from wikipedia, the free encyclopedia (redirected from Z
11	X/Z	X Z	X, or Z
12	X - Z	X of Z	X/Z
13	X, or Z	X categories. ... Z	X: Z
14	X Z	X or Z	X such as Z
15	X. from wikipedia, the free encyclopedia (redirected from Z	X - Z	X genus: Z
16	X,Z	X such as Z	X(Z
17	X. genus: Z	X are Z	X. ?Z
18	X: ... Z	X* Z	X / Z
19	X, also known as Z	X, ... Z	X. aka/or. Z
20	X, ... Z	X categories. Z	X. a Z
21	X : ... Z	X with Z	X to search Z
22	X genus: Z	X, and Z	X (Z
23	X. from wikiprofessional. concept ... Z	X, such as Z	X (syn. Z
24	X(Z	X: 1. Z	X. ... Z
25	X to search Z	X. additional relevant mesh terms: Z	X. from wikiprofessional. concept ... Z
26	X / Z	X include Z	X preparation, Z
27	X. aka/or. Z	X ... Z	X ... the Z
28	X. a Z	X to Z	X: ... Z
29	X Z	X — Z	X, also known as Z
30	X , Z	X *Z	X) ... Z
31	X ... Z	X/Z	X. from wikiprofessional. concept identifier: ... Z
32	X (Z	X terms and definitions ... Z	X ... synonyms:Z
33	X (syn. Z	X, particularly Z	X , Z
34	X (redirected from Z	X, also known as Z	X. genus Z
35	X are Z	X Z	X (or Z
36	X preparation, Z	X: ... Z	X (also called the Z
37	X — Z	X are: Z	X. species: Z
38	X. from wikiprofessional. concept identifier: ... Z	X Z	X (redirected from Z
39	X ... synonyms:Z	X can be converted into Z	X Z
40	X. genus Z	X. ... one types is Z	X - Z
41	X (or Z	X-Z	X. synonyms. Z
42	X (also called the Z	X $Z	Xs, Z
43	X) jump to: navigation, search. Z	X is termed the Z	Xe. Z
44	X. species: Z	X (therapeutic use) Z	Xs; Z
45	X. the Z	X, including Z	X, the Z
46	X. synonyms. Z	X? Z	X; ... Z
47	X. synonyymit. Z	X. ... the Z	X include Z
48	Xs, Z	X" ... Z	X (the Z
49	Xe. Z	Xs. Z	X?Z
50	Xs; Z	X ... these Z	X;Z
51	X, the Z	X) ... 0 (Z	Xe, Z
52	X" ... Z	X Z	X, a Z
53	X) 0 (Z	X in the absence of Z	X at dictionary.com with free online dictionary, ... also called Z
54	X (annexin iv) (Z	X from Z	X – Z
55	X; ... Z	X ... children: Z	X. from wikiprofessional ... Z
56	X to Z	X in the columbia encyclopedia, ... Z	X (formerly Z
57	X... Z	X. categories. Z	X include: class Z
58	X ... 0 (Z	X mean? ... Z	X, disease database information ... Z
59	X (the Z	X ... keywords: central Z	X (sometimes called Z
60	X?Z	X. the drugs Z	X can be used for: Z

Table B.2: Patterns for MeSH thesaurus

Candidates from the Humanity Index

The table B.3 the top 100 candidates from the humanity index are listed according to their weight calculated on the equation 3.3.

Rank	Candidate	PoS	Rank	Candidate	PoS
1	ptsd	nnp	51	survivors	nns
2	helicobacter	nnp	52	reasoning	nn
3	psychologists	nns	53	complication	nn
4	hallmarks	nns	54	nursing	nnp
5	memories	nns	55	therapy	nnp
6	arts	nns	56	joint	nn
7	asthma	nn	57	ethical	nnp
8	stressors	nns	58	fish	nn
9	cbt	nnp	59	healers	nns
10	faculty	nn	60	deformation	nn
11	schizophrenia	nn	61	witness	nnp
12	mindfulness	nn	62	debates	nns
13	interpreter	nn	63	apparatus	nn
14	meditation	nn	64	norms	nns
15	crimean	nnp	65	sculpture	nn
16	combat	nn	66	psychiatry	nn
17	ibs	nnp	67	mdd	nnp
18	chaplains	nns	68	gulf	nnp
19	anaesthesia	nn	69	team	nn
20	jews	nns	70	portraits	nns
21	welfare	nn	71	september	nnp
22	psychoanalysis	nn	72	curriculum	nn
23	corporations	nns	73	immigrants	nns
24	adhd	nnp	74	ms	nnp
25	neurofeedback	nn	75	competency	nn
26	tobacco	nn	76	cd	nnp
27	nightingale	nnp	77	hospice	nn
28	florence	nnp	78	mrs	nnp
29	sin	nn	79	technology	nn
30	advice	nn	80	aids	nnp
31	jehovah	nnp	81	eating	nn
32	theater	nn	82	root	nn

Table B.3: Top 100 Candidates from the humanity index

Rank	Candidate	PoS	Rank	Candidate	PoS
33	epilepsy	nn	83	classifications	nns
34	lds	nnp	84	violence	nn
35	donation	nn	85	scientific	nnp
36	coins	nns	86	insurance	nn
37	stem	nn	87	attributes	nns
38	dissemination	nn	88	arena	nn
39	nurses	nns	89	warren	nnp
40	retrieval	nn	90	orthodox	nnp
41	es	nnp	91	bioethics	nns
42	veterans	nns	92	emergency	nnp
43	residents	nns	93	disclosure	nn
44	dentists	nns	94	leukemia	nn
45	ocb	nnp	95	physics	nns
46	euthanasia	nn	96	coin	nn
47	hypnosis	nn	97	or	nnp
48	conceptualizations	nns	98	lifetime	nn
49	story	nn	99	dilemmas	nns
50	caregivers	nns	100	examination	nnp

Table B.3: Top 100 Candidates from the humanity index

Document Corpus Overview

The table B.4 shows the composition of the MeSH data corpus. The sum of all concepts included in the sub thesauri is higher than the number of concepts included in the MeSH (around 25,500) because there are loops between the concepts of the sub thesauri. Also the union of all records may include duplicates caused by the separated harvesting of the different sub thesauri. The name of the record sources is similar to the name of the sub top concept with the added prefix *TestData Record Mesh2009*. The same is true for the name of the annotation sources. Here the added prefix is *TestData Annotation Mesh2009*.

Concept		Recordsource		Annotationsource	
Name	#	ID	#	ID	#
Anatomy	1611	1	13392	1	185289
Organisms	3563	6	29061	6	359055
Diseases	4333	47	15070	47	204026
Chemicals and Drugs	5600	50	46716	50	665428
Analytical, Diagnostic and Therapeutic Techniques and Equipment	2612	62	25054	62	339078
Psychiatry and Psychology	1619	71	14830	71	192964
Biological Sciences	1930	74	18774	74	253894
Natural Sciences	1558	78	14881	78	193179
Anthropology, Education, Sociology and Social Phenomena	755	69	6836	69	88252
Technology, Industry, Agriculture	799	67	7378	67	95361
Humanities	186	16	1468	16	17157
Information Science	506	17	4368	17	54517
Named Groups	198	15	1848	15	25312
Health Care	1584	59	14559	59	188589
Publication Characteristics	143	14	1028	14	11709
Geographicals	377	13	3498	13	44147
Sum	27374		218761		2917957

Table B.4: Overview of MeSH Data Corpus

Bibliography

[1] Martin Ackermann. Statistische Korpusanalyse zum Extrahieren von semantischen Wortrelationen. Master's thesis, University of Hildesheim, November 2000.

[2] Rolf Bergmann. *Probleme der Textauswahl für einen elektronischen Thesaurus*. S. Hirzel Verlag, 1996.

[3] D. Bollegala, Y. Matsuo, and M. Ishizuka. Measuring semantic similarity between words using web search engines. In *16th International World Wide Web Conference (WWW2007)*, 2007.

[4] Janez Brank, Marko Grobelnik, and Dunja Mladenic. Predicting category additions in a topic hierarchy. In *The Semantic Web*, volume 5367/2008 of *Lecture Notes in Computer Science*, pages 315–329. Springer Berlin / Heidelberg, 2008.

[5] Vanda Broughton. *Essential thesaurus construction*. Facet Publishing, 2006.

[6] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[7] Eugene Charniak. Statistical techniques for natural language parsing. *AI Magazine*, 18:33–44, 1997.

[8] Philipp Cimiano, Andreas Hotho, and Steffen Staab. Clustering ontologies from text. In *Conference on Lexical Resources and Evaluation (LREC)*, pages 1721–1724. ARTIPOL, Portugal, May 2004.

[9] Kai Eckert, Heiner Stuckenschmidt, and Magnus Pfeffer. Interactive thesaurus assessment for automatic document annotation. In *Proceedings of The Fourth International Conference on Knowledge Capture (K-CAP 2007)*, 2007.

[10] Lee Gillam, Mariam Tariq, and Khurshid Ahmad. Terminology and the construction of ontology. *Terminology*, 11:55–81, 2005.

[11] Gregory Grefenstette. Automatic thesaurus generation from raw text using knowledge-poor techniques. In *In Making Sense of Words. Ninth Annual Conference of the UW Centre for the New OED and text Research*, 1993.

[12] Gregory Grefenstette. *Explorations in Automatic Thesaurus Discovery*, volume 278 of *The Springer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, 1994.

[13] Marko Grobelnik, Janez Brank, and Dunja Mladeni. Using dmoz for constructing ontology from data stream. *Information Technology Interfaces, 2006. 28th International Conference on*, pages 439–444, 2006.

[14] Zellig Harris. Distributional structure. *The Philosophy of Linguistics*, 1985.

[15] Birger Hjorland and Frank Sejer Christensen. Work tasks and socio-cognitive relevance: A specific example. *J. Am. Soc. Inf. Sci. Technol.*, 53(11):960–965, 2002.

[16] G. Holmes, A. Donkin, and I.H. Witten. Weka: A machine learning workbench. In *Proc Second Australia and New Zealand Conference on Intelligent Information Systems*, Brisbane, Australia, 1994.

[17] Kyo Kageura and N. Aizawa. Automatic thesaurus generation through multiple filtering. In *In Proceedings of 18 th International Conference on Computational Linguistics*, pages 397–403, 2000.

[18] Magnus Pfeffer Kai Eckert and Heiner Stuckenschmidt. Assessing thesaurus-based annotations for semantic search applications. *International Journal on Metadata, Semantics and Ontologies*, 2008.

[19] T. Kohonen. *Self-Organizing Maps.* Springer, Berlin, Heidelberg, New York, 1995, 1997, 2001.

[20] Dekang Lin. Automatic retrieval and clustering of similar words. In *unknown*, pages 768–774, 1998.

[21] Dekang Lin. An information-theoretic definition of similarity. In *In Proceedings of the 15th International Conference on Machine Learning*, pages 296–304. Morgan Kaufmann, 1998.

[22] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval.* Cambridge University Press, 2008.

[23] Dat P. T Nguyen, Yutaka Matsuo, and Mitsuru Ishizuka. Exploiting syntactic and semantic information for relation extraction from wikipedia. In *In IJCAI07-TextLinkWS*, 2007.

[24] Patrick Pantel and Dekang Lin. Discovering word senses from text. In *In Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 613–619, 2002.

[25] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1), March 1986.

[26] Dmitri G. Roussinov and Hsinchun Chen. A scalable self-organizing map algorithm for textual classification: A neural network approach to thesaurus generation. *Communication Cognition and Artificial Intelligence*, 15:81–112, 1998.

[27] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. In *Information Processing and Management*, pages 513–523, 1988.

[28] Gerard Salton and Michael J. McGill. *Introduction to modern information retrieval.* McGraw-Hill Book Company, 1983.

[29] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, illustrated edition edition, June 2004.

[30] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2nd Edition, 2005.

Ehrenwörtliche Erklärung

Ich versichere, dass ich die beiliegende Diplomarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Mannheim, den 14.08.2009 Robert Meusel