RADBOUD UNIVERSITY

MASTER THESIS

Methods For Automatically Generating a Legal Thesaurus

Author:
Hugo DE VOS
s4193695

Supervisor:
Dr. Iris HENDRICKX

A thesis submitted in fulfillment of the requirements for the degree of Master

in the

Center for Language Studies

September 3, 2017

Radboud University

Abstract

Faculty of Arts
Center for Language Studies

Master

Methods For Automatically Generating a Legal Thesaurus

by Hugo de Vos
s4193695

Automatic thesaurus generation is a desired technique for the reason that a thesaurus is a useful tool in NLP, but manually making a thesaurus is expensive and time consuming. In this thesis, the process of thesaurus generation is divided up in two parts: term extraction and relation extraction. Term extraction being the process of automatically finding candidate terms for a legal thesaurus and relation extraction is the process of finding which terms are hypernyms of each other. For term extraction different termhood measures are used: Log Likelihood, Kullback Leibler Divergence and the measure as assigned by the TExSIS tool. For relation extraction, different classifiers are trained to classify whether two terms have a hypernym-relation. The conclusion of this thesis is that no system could be built that can autonomously build a thesaurus and that in the short term it is better to look for a system to assist humans in making a thesaurus.

Contents

Chapter 1

Introduction

Almost everyone who has written a text recently, probably visited Thesaurus.com a few times. A useful website if you quickly need a synonym or other information about a word.

Thesauri are useful tools in many occasions: writing a text, looking for keywords or getting the meaning of a word. However, making a thesaurus is a lot of work. For this reason it would be a good thing if this task could be done by a machine.

Thesauri are known to have a positive effect on many Natural Language Processing (NLP) systems like text mining systems and information retrieval systems. (Aronson, Rindflesch, and Browne, 1994; Jarmasz, 2012). An example of applications in which thesauri are used are search engines (IR-systems) (Bhogal, MacFarlane, and Smith, 2007), especially search engines that have less (computational) resources than giants like Google and Bing. Such search engines are usually focused on a single domain. In these domain specific search engines, a domain specific thesaurus can make a difference in the performance. Because smaller search engine companies have less computational resources, it us usually beneficiary to add lexical knowledge in the form of a thesaurus.

1.1 Problem Description

A general problem with thesauri is that they are expensive to build and maintain (Voorhees, 1994; Aitchison, Gilchrist, and Bawden, 2000). Creating a domain specific thesaurus requires specialists within the domain. Someone from outside the domain would not be able to classify the domain specific terms and link them to other terms. People with such specialist knowledge about a domain are usually costly. Making a complete thesaurus with thousands of entries therefore takes a lot of time. For this reason, the total costs of creating a thesaurus are high.

Legal Intelligence[1] (LI) is a Dutch company that offers a search engine for looking up documents in the legal domain. LI uses a hand made thesaurus of legal concepts to improve their search engine and provide their customers with lexical information about their search terms. LI is looking for an automatic way to improve and maintain their thesaurus, for the reason that it is expensive to perform this task by hand.

[1]https://www.legalintelligence.com/

1.2 The Legal Domain

The legal domain is linguistically a unique domain that poses a number of challenges (but also) opportunities for applying language technology.

With respect to information retrieval (IR), Van Opijnen and Santos (2017) distinguishes thirteen features in which legal documents are different from most other types of documents. But most of those features also apply to other fields in NLP. These features illustrate the advantages and disadvantages of applying NLP to documents in the legal domain. These features are:

1. Volume
2. Document size
3. Structure
4. Heterogeneity of document types
5. Self-contained documents
6. Legal hierarchy
7. Temporal aspects
8. Importance of citations
9. Legal terminology
10. Audience
11. Personal data
12. Multilingualism
13. Scatteredness of legal resources

Volume is the reason why Information Retrieval techniques are needed, but also a reason why the legal domain could serve as a good corpus for NLP-systems: it is easy to get a collection with a lot of text. It differs a little per country and legal tradition, but every day large amounts of legal text are produced within the domain of law. From verdicts to contracts, to laws to academic material. New legal texts are added to the total collection in a fast pace.

The average *document Size* in the legal domain is larger than the size of most non-legal texts. Information Retrieval application often use summaries of the documents, but for the reason that nuance is generally of great importance in legal texts, the document must be processed or read as a whole to get extract knowledge.

Structure: within a legal genre, documents tend to be very structured. For example, contracts are all structured roughly the same way. In the Netherlands, verdicts are also all structured according to the same manner. This could be an advantage for language technology. However, between the different legal genres the structures tend to differ to a large extent.

This brings us to the point of *heterogeneity of document types*. There is no such thing as 'the legal document'. Many types of legal documents exist and they differ in multiple ways. Contracts are different in structure, language use and length from treaties, court notes or academic texts about law. This means that in most cases there is not one method that fits all types of legal texts.

Self contained documents are documents that are '*...not just 'about' the domain ...*' and '*... actually contain the domain itself and hence [...] have specific authority, depending on the type of documents*' (Van Opijnen and Santos, 2017, p. 68). In my opinion, this is a rather vague point and not relevant to most research.

Legal Hierarchy is about that legal documents form a hierarchy according to which they apply. An example of this is the hierarchical principle *lex posterior derogat legi priori* (The newer law displaces the older) (Verheugt, 2013). This means that if a newer law is more important than an older law when those laws contradict. Other examples of such principles are *the specific law displaces the general law* and *the higher law displaces the lower law*. These principles are especially important in IR-systems that automatically try to find laws that apply to a case. (Moens, 2001)

Temporal Aspects also refers to the *lex posterior*-principle above. But also to the fact that a law can be totally retracted or be explained differently dependent on time. An example of this is the Runescape Case (Hoge Raad, 2012). In this verdict, the theft of a virtual amulet in the game of Runescape was punished by the High Court. Since this case, the meaning the word *good* also includes virtual goods in games. According to Dutch law, only *goods* can be stolen and therefore also the exact meaning of the word *theft* changed as a result of this case. This illustrates that the meaning of laws and legal terms, constantly changes over time.

Citations are an important part of many texts in general. However, in legal texts they are probably more important. For example in court, the whole argumentation is built on references to law articles and precedents. This is one of the reasons why IR-systems are a welcome asset in the legal domain, because a large amount if references need to be retrieved to build a case in court.

Legal terminology seems to be clearly defined by itself. *Murder* for example, seems to be well defined in the law. However, the exact meaning of *murder* can change with the different interpretations judges might give to it through time. Furthermore, definitions of legal terms need to be vague and precise at the same time (Coulthard and Johnson, 2009): for example, the definition of *murder* must be general enough to apply to all cases of murder, but specific enough to leave no doubt to what exactly murder is.

Another feature of legal terminology opposed to for example biomedical terminology is that many legal terms originate in common language (Dozier et al., 2010). For example the term Avonex is a rare term. If such a term appears in a biomedical text, it is almost certain it is a biomedical (named) entity. Legal terms and (named) entities are more based in common language. For example, lawyer firms are in many cases called after a person. An example of this is the company *Dirk Zwager Advocaten* (Eng: Dirk Zwager lawyers). *Dirk Zwager* is the name of a person and *lawyers* is a very common term in the legal domain. For none of the terms in *Dirk Zwager Advocaten* is very unique, this may cause problems in for example Named Entity Recognition if no attention is payed to this problem.

A third property of legal terminology is that many legal terms consist of multiple words. Examples of such terms are *algemene maatregel van bestuur* (Eng: administrative order) and *Algemene wet bestuursrecht* (Eng: General Administrative Law Act). Later in this thesis there will be more attention for this.

Another example is the word *murder*, this terms is a legal term as well as a common used term. Although the meanings of the legal *murder* and the general *murder* overlap, the legal meaning of murder has a more specific and narrower meaning than the common use of murder.

With *Audience*, Van Opijnen and Santos (2017) refers to the fact that legal texts have a wide range of possible readers. From academic staff to lawyers to judges to the laymen. Many people need the law and read legal documents for different reasons. As a consequence of this, legal texts are consulted by people with variate types of education. Therefore legal texts are in most cases not aimed at a single public. A Dutch juridical verdict, for example, must ideally be readable by all Dutch citizens, but also mus contain information detailed enough for a lawyer or scholar.

Legal texts often concert people and companies and therefore contain a lot of *Personal Data* like names and addresses. On the one hand this makes them suitable for training Named Entity Recognition systems, but on the other hand many texts are anonymized to some degree. In the verdict corpus used later in this thesis names and addresses of suspects, witnesses and other people are left out.

Multilingualism and *multi-jurisdictionality* are problems in legal language technology. Contrary to other domains in science, the juridical domain differs per country. First of all the language: in many scientific domains like linguistics or biomedicine, the main language is English and the large majority of the literature is in English. In the legal domain however, there is no standard language. In the legal domain, every country uses its own language. Dutch verdicts and laws are published in Dutch, English verdicts are published in English and German verdicts in German. A consequence of this is that NLP-systems need to be separately developed and or trained for every language. This makes it harder for the field of legal-NLP to develop than NLP in other areas as it is scattered over different languages.

Multi-jurisdictionality means that the way a legal system is organized differs per country. For example the difference between common law as practiced in the UK and America versus the codification of law in the European mainland (Verheugt, 2013). This influences the way legal texts are structured and this also means that legal NLP systems need to be adapted to different juridical cultures.

The conclusion of the points from Van Opijnen and Santos (2017) is that the collection of all legal text is not homogeneous. The the properties of legal texts change with time, discipline, country, audience and many more factors. In this thesis, a specific subset of all legal documents is considered (namely verdicts). The collection verdicts are relatively homogeneous regarding the points above. However, to really represent the whole legal domain, the results of this study should be generalized.

1.3 Aim of the thesis

The aim of this thesis is to research methods for building a legal thesaurus from a corpus of Dutch juridical verdicts. This thesis will focus on two components of a thesaurus learning system. As will be further explained in later chapters, these two components of automatic thesaurus generation are: *term extraction* and *relation classification* (Velardi, Faralli, and Navigli, 2013, e.g.). Term extraction is the task of finding candidate terms for the thesaurus. Relation classification is the task of recognizing whether two words are hypernym and hyponym of each other. This thesis contains two main studies focusing on these two tasks that play important roles in automatic thesaurus learning

This thesis does not aim to build a full thesaurus learning system, but to provide insights into the different ingredients of a system. These insights can be used in further studies in which a full system is built.

1.4 Research Questions

The main research questions of this thesis are:

1. What is the best method for extracting legal terms from a Dutch verdict corpus?
2. What is the best method for relation classification on Dutch legal terms?

Both questions can be divided into two sub questions:

- What is the best method for the task according to the literature?
- How does that method work on the legal verdict corpus?

1.5 Outline of the thesis

- In Chapter 2, a literature review is presented on literature about the different aspects of legal thesaurus learning. This chapter consists of four parts: a part on NLP in the legal domain in general, a part about thesaurus learning in general, and two parts about the two steps in most thesaurus learning: term extraction and relation finding.
- In Chapter 3, the methodology of the experiments in this thesis is described. This chapter is divided in two main parts. The first containing the methodology for relation extraction and the second containing the methodology about relation extraction.
- In chapter 4, the results of the term extraction experiments are presented and analyzed.
- In Chapter 5 the results of the experiments about relation extraction are presented and analyzed.
- In Chapter 6 the results are discussed, linked to the literature. Also future lines of research explored.

Chapter 2

Related work

This chapter consists of four parts. The first part is a general discussion about language technology in the legal domain. The second part is about previous attempts on thesaurus learning. Parts three and four concentrate on the studies that are conducted in this thesis. Part three discusses literature about term extraction and works towards the methodology chosen for this component of thesaurus learning. Part four discusses literature on relation extraction and relation classification.

2.1 Language Technology in the legal domain

Although the need for language technology in the legal domain seems obvious due to the large amount of text, there is not very much literature on this topic (Francesconi et al., 2010; Liu and Chen, 2017). Of course, some research has been done, but the topics of those studies are rather scattered. The different sub-domains of NLP in the legal domain are behind to the development of NLP in other disciplines, especially in comparison with the biomedical domain, in which the potentials of language technology are widely acknowledged and the different topics like term extraction or ontology learning are widely studied.

A task that got relatively much attention in legal language technology is automatically predicting a judges' decision in a legal case (Aletras et al., 2016; Luo et al., 2017; Liu and Chen, 2017; Katz, Bommarito II, and Blackman, 2017).

Aletras et al. (2016) try to predict whether there is a violation of an article of the European Convention on Human Rights (*the Convention*). For three articles of the Convention they trained a binary classifier to classify whether it is violated or not based on the case application. As features they use bag of words N-gram features and as classifier a SVM is used. With this method, they achieve an accuracy of on average .79.

The ultimate goal of Luo et al. (2017) is to develop a system to help anyone understand law. The aim is that that anyone (even a layman) could give a description of a case in their own words and then get relevant law articles and what kind of punishment could be expected for this case. For their system Luo et al. (2017) train a SVM and a neural network that are able to extract relevant law articles given a case description as well as to predict the charge given the case description.

Liu and Chen (2017) had an objective that was similar to Aletras et al. and Luo et al.: namely predicting the severeness of a charge. The exact objective of Liu and Chen is to classify a cases into jail-time categories (for example 1-2 years and 2-5 years). This is achieved by performing a sentiment analysis that was trained on bag of words features of case descriptions of precedents.

These studies are examples of a line of research that gets relatively high attention in legal NLP.

Next, related work on automatic thesaurus learning will be discussed which is closer to the focus of this thesis.

2.2 Thesaurus and Ontology Learning

Thesaurus and ontology learning is a topic that got much attention through the history of NLP.

One of the earliest approaches on thesaurus learning is the one by Jing and Croft (1994). They already recognize the possibilities of using a thesaurus in an IR-system. They present a system called Phrasefinder. The system works by extracting relations using rules. However, the paper is not really clear on what a relation exactly is.

Kageura, Tsuji, and Aizawa (2000) build a Thesaurus based on translations between Japanese and English. The idea behind this approach is that words that different Japanese words that translate into the same English word have a semantic relation. Apart from this main idea, they also use Part of Speech information. This information is used to do a cluster analysis and terms that fall in the same cluster are assumed to be related. As well as in Jing and Croft (1994), there is no clear definition of what a relation is. They are not explicitly looking for hypernym relation, or a synonym relation. Just words that are semantically related. For this reason this study is less applicable to this thesis.

A similar technique based on translations (this time Mandarin to English) was used by Yang and Luk (2003). They learn a legal thesaurus based on parallel translations of Hong Kong law texts. What makes this study relevant is that this study extensively describes the term extraction part. In this study, the tf-idf measure was used to extract relevant terms. Although the tf-idf measure itself is not used in this thesis, the termhood measures used in this thesis are similar to tf-idf as will be explained later in this thesis.

The second reason why the study of Yang and Luk (2003) relates to this thesis is of course that a thesaurus is built for the legal domain. The problem is that it is not possible to adapt or test their methodology for Dutch. The reason for this is that there does not exist a parallel corpus for Dutch law. Maybe Europarl and the European laws and treaties might serve this goal. However these only cover the very specific domains in law like, for example, trade-law and agricultural law. Areas like criminal law will not be covered.

Lenci, Montemagni, and Pirrelli (2006) describe a system for learning a legal thesaurus. This study divides the thesaurus learning algorithm in two stems: term extraction and relation finding. *Term extraction* is the process of extracting candidate terms that could be part of the thesaurus. *Relation finding* is the part where relations between those candidate terms are searched. The advantage of splitting the process in two parts is that the search field for relations is a lot smaller. The scope can even be reduced to a single domain (like the legal domain). The idea of this two phase thesaurus learning has also been applied in other studies (Velardi, Faralli, and Navigli, 2013).

This two step phase has also been applied in this thesis.

2.3 Term Extraction

Term extraction focuses on finding domain relevant words or phrases (terms). The domain relevance can be expressed as 'termhood', which is generally defined as:

the extent to which a term is distinctive for the document or domain of interest.

According to this definition, the notion *termhood* is relative to the domain. If, for example, you have a document about a forest the term *oak tree* might have a high termhood, while in the domain of linguistics, oak tree has a low termhood. The scope of this thesis is the legal domain, in particular the domain of Dutch verdicts. Because of this, a definition of termhood that better applies to this thesis is:

The extend to which a term is distinctive for Dutch legal Verdicts

Apart from these definitions, *termhood* also consists of a second measure: *phraseness*. *Phraseness* can be defined as follows:

The extend to which a term is a cohesive entity

This concept will be explained further in section 2.3.3

2.3.1 Measuring Termhood

There are a lot of measures to measure the termhood of a candidate term, of which most originate in information retrieval. In this section I will give a brief overview of some of these methods.

A good place to start is tf-idf (Salton and Buckley, 1988). This is the prototypical measure to determine how much a term says about a document. This measure works by multiplying the term frequency with the inverse document frequency. There are different definitions of tf-idf, but they all come down to weighing some kind of term frequency with the inverse document frequency.

The terminology *some kind of term frequency* has been chosen as there are different types of document frequency among which are *Boolean frequency* (word is present (1) or absent (0)), *absolute frequency* (the number of times a term occurs in a document), *relative frequency* (absolute frequency divided by document size) and the *log frequency* that can either be the logarithm of absolute or the relative frequency.

The inverse document frequency is equal to one divided by the number of documents the term is present in. TF-IDF could be described as the term frequency weighed by the inverse document frequency. The purpose of this weighing is to down weigh stop words that occur often in a single document (high tf) but also occur in many other documents (low idf) and up weigh words that occur a lot in a document (high tf) but not in many other documents (hence the high idf). The intuition is that words with a high tf-idf value are indicative for what a text is about. In a text about language, for example, words like *verb* and *noun* will have high tf-idf and words like *the* or *computer* will have a low value as either their term frequency is low or their inverse document frequency is low.

A disadvantage of the tf-idf measure is that it works on the document level. Per document in a collection, the tf-idf is determined for each word. The research in this thesis does not try to find out whether a terms is descriptive for a document but whether it is descriptive for a domain (law) and hence whether it is descriptive for a collection of documents representing this domain.

A review of different measures to determine the termhood of a word is presented in (Verberne et al., 2016). As reported in this paper, the termhood of a *term* can be determined based on a combination of two measures: *informativeness* and *phraseness*: *informativeness* is the extent to which a term gives information about the topic of a document or document collection and *phraseness* to which a candidate term is a full phrase.

For example, the word combination *college van burgermeester* (Eng: College of Mayor) will have a high informativeness, because it will occur frequently in the legal domain and not often in a non legal domain. However, it will have a low *phraseness*, as it is incomplete. The complete phrase is *college van Burgermeester en Wethouders* (Eng: College of Mayor and Aldermen). In this case you only want to find the second complete option as a term and not the first incomplete option. To achieve this, you must take into account both informativeness and phraseness.

2.3.2 Informativeness

Informativeness is the amount of information a term gives about a document or collection of documents. In other words, how much a certain term says about a domain or how representative that word is for a domain.

For example, in the legal domain the term *penalty clause* will have a high informativeness, because this is a combination of words that occurs often in the legal domain and not often outside the legal domain. The word *apple* will not have a high informativeness, as it is not unique for the legal domain.

Another way to put it, *informativeness* is how easy it is to classify a document based on a term. If you find the word *penalty clause* occurs in a text, you can say with reasonable certainty that the text belongs in the legal domain, however, when the word *apple* occurs in a document, it is not possible to say (on this information alone) whether the document is a legal document.

In (Verberne et al., 2016) five informativeness measures are identified: *Parsimonious language models (PLM)*, *Kullback–Leibler divergence for informativeness (KLI)*, *Frequency Profiling*, *Co-occurrence based method (CB)*.

Those models differ in the details, but they essentially compare the occurrence of a word in a corpus or document with the occurrence of that same word in a background corpus. That background corpus can be either a totally different corpus or all the documents in the same collection as is the case with tf-idf.

An method where two corpora are compared is the *pointwise* Kullback-Leibler Divergence for Informativeness (Verberne et al., 2016; Tomokiyo and Hurst, 2003). Kullback-Leibler Divergence for Informativeness (KLI) is defined in equation 2.1 in which $P(t|D)$ is the probability of a term (t) occurring in a document and $P(t|C)$ the probability of a term occurring in a document collection. It is, possible but not necessary for the document (D) to occur in the background corpus (C). As a result of this, any document or corpus can be compared with any other document or corpus with this method.

$$KLI(t) = P(t|D) log \frac{P(t|D)}{P(t|C)} \tag{2.1}$$

The KLI-measure measures the informativeness based on relative frequency of words (or n-grams) in different corpora. Other measures like Frequency Profiling and Parsimonious Language Models (Hiemstra, Robertson, and Zaragoza, 2004) are based on the observed frequency versus the expected frequency. In parsimonious language models the expected frequency of a word is estimated based on the collection this document is part of and then, for each n-gram, its observed frequency is compared with its real frequency. For this measure, the 'target document' must be part of the background collection.

This is not the case with frequency profiling (Rayson and Garside, 2000). This method is used for comparing two corpora or document collections with each other.

First the expected frequency of term t in corpus C ($E(t, C)$) given corpus C and corpus D is computed with equation 2.2 in which $count(t, D)$ is the absolute occurrence count of a word in corpus C and $count(t, D)$ the absolute count in corpus D. $|C|$ and $|D|$ are the size of corpus C and D respectively. This formula is based on the same principle of *expectedness* as in for example χ^2. Also the expected frequency for every term in corpus C is computed with this formula.

$$E(t, C) = |C| * \frac{count(t, C) + count(t, D)}{|C| + |D|} \tag{2.2}$$

With the expected frequency for each term, the log-likelihood of a term occurring in a corpus can be computed. The log-likelihood (LL) of a term t occurring in corpus C can be computed with equation 2.3. As explained in (Gelbukh et al., 2010), this formula does not discriminate between the foreground and the background corpus. It only signals to what extent a word contributes to the difference between the frequency distributions between the two corpora. As Verberne et al. (2016, p. 517) put it: *The term with the largest LL value is the word with the most significant relative frequency difference between the two corpora.*

This is visible from the formula, as the LL will be high when a term occurs more than expected in the domain corpus, but also when the term is unique for the background corpus.

In Gelbukh et al. (2010), this *symmetry* is solved by applying the following heuristic: *If the relative frequency of the term in the background corpus is higher than the relative frequency in the foreground corpus, the log-likelihood is multiplied by -1.* As a consequence of this, terms with a high positive value are representative for the foreground corpus and terms with a 'high' negative value will be representative for the background corpus.

$$LL = 2 * \left(count(t, D) * log \frac{count(t, d)}{E(t, D)} + count(t, C) * log \frac{count(t, C)}{E(t, C)} \right) \tag{2.3}$$

2.3.3 Phraseness

In Verberne et al. (2016), two measures for phraseness are described: *C/NV-value* and the Kullback-Leibler Divergence for Phraseness (KLP). Both methods rely on applying higher weights to longer terms. In the example above about *college van Burgermeester en Wethouders* (Eng: college of Mayor and Aldermen), every substring of this 5-gram occurs at least as often as the full 5-gram. It is even very likely that every subset of words of this n-gram, occurs more than the full five words together. These substrings however, are most of the times not a term on their own.

The C-value (Frantzi and Ananiadou, 1999) works by multiplying the number of occurrences of an n-gram with the number of words in an n-gram. This way, longer n-grams get a higher weight and an n-gram such as *Burgermeester en Wethouders* will get a lower C-value that the full phrase.

The C-value raises a problem, namely that terms that are a subset of another term, may be overlooked. For example, the C value will favour *College van Burgermeester en Wethouders* above the single word term *Burgermeester*. Of course, the count of *Burgermeester* will be higher than the full 5-gram, but this method is not very sensitive for this problem.

A measure that is more refined with this is the Kullback-Leibler Divergence for phraseness. The formula for this measure is presented in 2.4. In this formula, $P(t|D)$ is the probability of a term in document or corpus D. $P(u_i|D)$ is the probability of each unigram i u_i from term t in D.

For example, the term *college van Burgermeester en Wethouders* consists of 5 unigrams. $P(u_0|D)$ is the probability of *college* occurring in D.

A result of this formula is that longer terms gain a higher weight. If the unigrams a term consists of are rare, they will hardly occur outside the context of the larger term. And the lower the probabilities of the unigrams, the higher the KLP will be. On the contrary: when a unigram occurs a lot on itself, it may be a term of itself. In this case the KLP for the larger phrase it is a part of will be lower, because the individual term also occur.

$$KLP(t) = P(t|D) * log \frac{P(t|D)}{\prod_{i=1}^{n} P(u_i|D)} \tag{2.4}$$

The Kullback-Leibler divergence for phraseness (KLP) can be combined with the Kullback-Leibler difergence for informativeness (KLI) in one termhood measure called the *Kullback-Leibler divergence for Informativeness and Phraseness (KLIP)*. This formula is presented in 2.5. The KLIP is essentially a weighed sum of the KLI and the KLP. The gamma (γ) is a parameter that determines how heavily the KLI and KLP are weighed. The value of the gamma is always between zero and one. A gamma close to zero yields a high weight for the KLI. A gamma close to one means that the phraseness is more important .

$$KLIP(t) = (1 - \gamma) * KLI(t) + \gamma * KLP(t) \tag{2.5}$$

2.4 Relation Extraction and Relation Classification

The task that is done in this thesis is *relation classification*: classifying whether two words are hypernym and hyponym from each other. This is different from *relation extraction*, because in relation extraction a system actively searches for words that have a relation. However, to get a good understanding of relation classification, we need to start at relation extraction.

2.4.1 Pattern Based Approaches

One of the earliest approaches to automatic hypernym relation extraction is the one by Hearst (1992). This method works by finding patterns that signal a hypernym relation.

Finding these patterns is done by using a seed-list of known pairs with a hypernym relation. A large corpus is searched for places where those terms co-occur. At these places the algorithm searches for patterns of words and POS tags that also regularly co-occur with these pairs. Then it is inferred that such patterns signify hypernym relations. These patterns are then used to find other pairs

For example: a pattern like in 2.6 can signal a hypernym relation. From a sentence like 2.7, this pattern can extract that injury is a hypernym of *bruise*, *wound* and *broken bone*.

(2.6) NP {, NP} * , 'or other' NP

(2.7) Bruises, wounds, broken bones or other injuries

A later variant on the algorithm by Hearst is the Snowball algorithm (Agichtein and Gravano, 2000). This is a bootstrapping algorithm for extracting semantic relations. This is a method begins the same as the method above, but when new word pairs are found, these word pairs are then used as seed pairs to find more patterns that then can be used to find new pairs. This method lead to rather high results of a precision and recall both about .85 after 3 iterations through the cycle.

A difference between the snowball method and the method by Hearst (1992) is that snowball was designed for more concrete relations like a organization-location relation. For this reason, the system was also tested on this type of relations and not on hypernym relations. As a result of this, both systems are difficult to compare directly. However, as the method of snowball shows similarities with the method of Hearst, it is not unthinkable to use snowball for hypernym relation extraction.

2.4.2 Word Embeddings

A reason why the pattern based methods described in the previous section may not be useful for the verdict corpus used in this thesis[1] is that the so called Hearst patterns tend to be rare in a corpus (Fu et al., 2014). And I expect them to be even rarer in the verdict corpus. The reason for this is that the verdict corpus is not explanatory in nature and also that the texts are specific to one case.

For this reason, a method with more abstraction is needed. A way to get an abstract representation of a word's meaning is via word embeddings.

Word embeddings is a way to represent words in a vector space. Word embedding vectors represent words based on their contexts. The state of the art of embedding models at the moment are word2vec (Mikolov et al., 2013b) and GloVe (Pennington, Socher, and Manning, 2014). The general working of both of these models is to train a neural network to predict context of a word and then take the weights of the hidden layer of the neural network as vector representing a word.

The intuition behind embedding models like Word2Vec or even Latent Semantic Indexing (Deerwester et al., 1990) is that words that occur in the same context have similar meaning. *Context* can either be direct context (for example a window of 5

[1]See section 3.2.1 for the description of this corpus.

words left and right) or a larger context like a whole document. A simple example to illustrate is that both words *cow* and *horse* both can occur in the sentence *'the * eats grass in the meadow.'*. From this it can be deduced that *cow* and *horse* have similar meanings. (Namely: both mammals, farm animals, grass eaters etcetera.)

The reason why word embeddings might be suited well for extracting hypernym relations is that Mikolov et al. (2013a) showed that semantic relations between words are intuitively present in these models. For example: Figure 2.1 shows that the location of *Moscow* relative to *Russia* in the vector space is the same as the position of *Beijing* in relation to China.

Another classic example that shows the same is that if you take the vector representing *queen* and subtract from it the vector of *woman* and add the vector for *man* to that result, you end up close to the vector for *king*. What this and the previous example show, is that semantic relations in the real world are represented in the vector space models achieved by word2vec and GloVe [2].

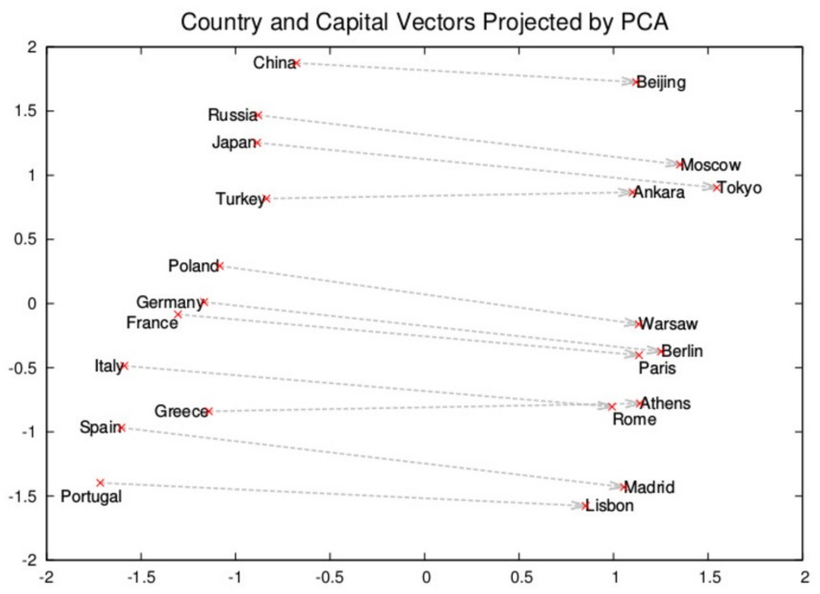


FIGURE 2.1: This image cited from Mikolov et al. (2013a, p. 4) show how relations are present in vector space models with embedding vectors.

Because semantic relations are reflected in these distributional vector space models, these models could possibly also be used to detect hypernym relations. And this has been tried in different ways that will be described in the following sections.

Finding Hypernyms by Projecting embedding vectors

One of the approaches to learn hypernym relations is described in Fu et al. (2014). In this approach, a projection matrix is learned. When an embedding vector of a

[2]The examples given here are all for word2vec. But things that hold for word2vec also hold for GloVe, as word2vec and GloVe practically arrive at very similar model via a different path (Řehůřek, 2015)

hyponym is multiplied with this matrix, the result is close to the vector of it's hyponym. For example, multiplying the vector of *tulip* with this matrix, should result in the vector of *flower*.

Fu et al. (2014) try to learn the projection by solving equation 2.8. In this equation, Φ is a transformation matrix, that, when multiplied with the vector of a hyponym (x) should result in the vector of the hypernym (y). N in the equation is the number of hypernym/hyponym pairs. The optimization results in the optimal matrix Φ. This entails that the difference between $\Phi * \vec{x}$ and the vector of its hyponym ($\vec{y}$) is minimal. In the optimal situation is $\vec{x}\Phi$ equal to $\vec{y}$ meaning that the hyponym vector multiplied with the projection matrix Φ is equal to the hypernym vector.

$$\Phi^* = \underset{\Phi}{\operatorname{argmin}} \frac{1}{N} \sum_{x,y} \|\Phi x - y\|^2 \tag{2.8}$$

An intuitive objection against this method is that not all hypernym relations are the same and thus can not be uniformly represented in the vector space. For example in the flower example, *flower* is a hypernym of *tulip*, *rose* and also *lily*. Because all hyponyms of *flower* have a different meaning, their location in the vector space is different and so their location relative to *flower* is too.

This problem is recognized by Fu et al. (2014) and solved by clustering the offset vectors (i.e. the vectors that are the result of subtracting the hyponym vector from the hypernym vector). They then first learn to assign a hyponym to a cluster and then multiply the hyponym vector with a transformation matrix belonging to that specific cluster to get the vector of the hypernym.

Fu et al. (2014) report results with a maximum precision of .80 and a recall of.62, which are good results in comparison to similar studies

The method of Fu et al. (2014) has been tried by Espinosa-Anke et al. (2016), for a taxonomy system with hypernym relations. They also learn a transformation matrix that should project the vector of a hyponym on the vector of its hypernym. They report F-scores within the range of .22 and .60, depending on the domain (.22 in the physics domain and .60 in the domain of education). These scores are considerably lower than in Fu et al. (2014).

No explanation is given for this difference between the results of both studies, but the difference shows that the approach of learning a projection matrix may be susceptible to many factors like domain, corpus or even language, as the study by Fu et al. (2014) was on a Mandarin Chinese encyclopedia and the study by Espinosa-Anke et al. (2016) was on English Wikipedia.

Supervised methods for hypernym

Apart from the unsupervised approaches presented above, there are also supervised methods for extracting thesauri based on embedding vectors. One of these approaches is described by Baroni et al. (2012). In this experiment a Support Vector Machine (SVM) classifier is trained to predict whether a pair of two words or terms are in a hypernym relation.

The features for this classifier are the embedding vectors for both words concatenated. From two vectors of length 300 one vector of length 600 is created and on these vectors the classifier is trained.

As negative examples (examples of pairs that are not hypernym relations), random pairs of words are used as well as inverted hypernym pairs (tuples of hypernym, hyponym instead of the other way around).

The classifier itself is a binary classifier that only predicts whether given a concatenated vector, can say whether they form a hypernym pair or not. The classifier just answers a yes/no question.

This method of training a binary classifier to predict whether a pair of terms have a hierarchical relation based on embedding vectors has also been used in Roller, Erk, and Boleda (2014). In this paper, the method of Baroni et al. is compared with another method that is mentioned by Baroni et al. (2012) as an idea for future research.

This second method makes use of the phenomenon described in Mikolov et al. (2013a) that relations are signified by adding and subtracting word vectors. In the method by Roller, Erk, and Boleda, a feature vector (f) is created by subtracting the hyponym vector from the hypernym vector. These feature vectors are then enriched by concatenating them with f^2 (itself squared) creating a vector of length 600.

These features are then used to train a logistic regression classifier that that classifies whether a pair of words have a hypernym relation.

In the study by Roller, Erk, and Boleda (2014), the same method for creating negative examples is used as in Baroni et al. (2012), namely, creating pairs of random terms and inverting 33% real pairs to create inverted pairs consisting of hypernym-hyponym instead of hyponym-hypernym.

For their experiments Roller, Erk, and Boleda (2014) report a maximum accuracy of .85 for the difference vectors and a maximum score of .81 for the concatenated vectors. However, the range of accuracy scores for the concatenation vectors goes from .65 to .81, while the lowest value for the difference vector is .80, suggesting that difference vectors are a better and more constant method.

A comprehensive study that compares different vector operations like the ones described above (concatenation, subtraction) has been performed by Weeds et al. (2014). In their study, they compare, subtraction, multiplication, addition, concatenation, and only training on one vector from the pair.

Weeds et al., train different classifiers with on two different datasets using these vector operations. Apart from classifiers that predict hyponymy[3], they also trian classifiers to predict cohyponymy, whether two terms are hyponyms of the same hyponyms (whether they are sisters so to say).

The conclusion of this comparative study is that the results for the different vector operations differ significantly. Weeds et al. also conclude that different operations work well for different tasks, but that subtraction an concatenation work well. Subtraction scored a maximum accuracy for hyponym detection of .75 and a minimum of .74 and concatenation scored accuracies of .74 and .68.

Another remarkable result of Weeds et al. is that by only taking the vector of the hypernym and ignoring the vector of the hyponym as a total an accuracy of .75 can be achieved, which is the same result as the best result for difference vectors. This finding supports the theory of Levy et al. (2015) that the SVM just learns whether a term is more often a hypernym than a hyponym. In other words, the SVM just learns the position of a single word in the hierarchy.

[3]A small difference between the classifiers by Weeds et al. and Roller, Erk, and Boleda is that the classifiers of Weeds et al. predict hyponymy instead of hypernymy. It would be an interesting line of research to investigate hypernym detection is similar to hyponym detection.

Chapter 3

Methodology

As mentioned in the previous sections, thesaurus learning consists of two major steps: term extraction and relation finding. For this reason this chapter is split in two. Section 3.1 is about different approaches towards term extraction and section 3.2 will be about the method for finding relations.

3.1 Term extraction

In this experiment, three measures for termhood will be compared: Log-Likelihood (Rayson and Garside, 2000), Kullback-Leibler Divergence for Informativeness and Phraseness (KLIP)(Tomokiyo and Hurst, 2003) and termhood as used in TExSIS(Macken, Lefever, and Hoste, 2013). All three measures are based on the principle of comparing occurrences of terms in a foreground corpus (legal corpus) with occurrences of terms in a background corpus (general corpus). All of these measure return a score for a term. And sorting terms according to this score ideally leads to a ranked list on which the highest n terms are all legal terms. The ranked lists are evaluated with Average Precision and with Precision and Recall @N.

For the measures Log-Likelihood and KLIP, the ranked lists are filtered with linguistic filters based on Part of Speech. The reason for this is to exclude terms that cannot possibly be legal terms based on their POS-tags. These filters will also be evaluated with Average Precision and with Precision and Recall @N.

On TExSIS no further experiments were done. The reason for this is the limited access to the software of TExSIS. I had no access to the backend of the system. For this reason I could not use the same background corpus with TeXSIS (TeXSIS uses Europarl(Koehn, 2005) as background corpus.) It was also not possible to get POS-tags in the output phrases. This made it difficult to do further experiments. The results of TExSIS have been included for the reason that it is a state of the art system to which I wanted to compare my results.

In this section, the different parts of the term extraction study are described. In section 3.1.1 the foreground corpus and the background corpus are described as well as the different preprocessing steps. In section 3.1.2, a few methodological notes on TExSIS are mentioned In sections 3.1.3 and 3.1.4, the Log-likelihood and KLIP procedures are described respectively. In section 3.1.6 the evaluation methods are described.

3.1.1 Datasets

Foreground Corpus

As a foreground corpus a collection of 300k legal verdicts[1] was used. This dataset contains legal verdicts in all Law areas (e.g. criminal law, tax law or administrative law). This corpus had a total of around 800M tokens of which 650M words and 150M punctuation marks.

The corpus was anonymized with respect to the names of civilians. In most cases these were defendants or witnesses.

The choice for this particular corpus was mainly because of its availability. Because it is public data provided on the website of the Dutch court (www.rechtspraak.nl) it was free to use.

Background Corpus

As background corpus, the newspaper part of OpenSoNaR500 (Reynaert, Camp, and Van Zaanen, 2014) was used. This corpus contains 700k newspaper articles. In total the background corpus consists of 212M tokens.

The main reason to use this corpus was for replication of Verberne et al. (2016) in which a generic newspaper was used. The SoNaR newspaper corpus seemed fit for the purpose of background corpus, since newspaper articles generally take on a wide range of topics. It is true that some newspaper articles are about legal topics, but this is only a subset of the total.

Preprocessing

The SoNaR500 newspaper corpus was lexically preprocessed by the creators of the corpus. (Reynaert, Camp, and Van Zaanen, 2014): The whole corpus was tokenized and POS-tagd with frog (Bosch et al., 2007), a tool for linguistically analyzing Dutch.

The verdict corpus was also preprocessed. It was first tokenized and then POS-tagged with the frog tool (Bosch et al., 2007).

The corpora were preprocessed in such a way that the POS-tags were concatenated to the word they belong to. This way the colibri script from the following part of the pipeline recognized the token and POS-tag as a single word.

Stop words were intentionally not removed from the corpora. The reason for this is that there are legal terms that contain words that are considered stop words. For example: *Officier van justitie* (Eng: prosecutor) contains the word *van* (Eng: of), a preposition that is on most stop word lists. Another example of a term containing stop words is *onderzoek ter terechtzitting* (Eng: examination at the hearing) containing the stop word *ter* (Eng: at the).

3.1.2 TeXSIS

As mentioned before, there was little possibility to influence the TExSIS procedure. The architecture of TExSIS has already been explained in the literature section 2. A few methodological notes will be made here.

[1]Kindly supplied by Legal Intelligence

Macken, Lefever, and Hoste (2013) could not share the source code or compiled code with me. They provided the opportunity to send a corpus and process the corpus with TExSIS themselves[2]. They sent back the term lists with the terms and their termhood values.

The verdict corpus in total was too large to be handled by TExSIS. Because of this, only the part about criminal law (NL: strafrecht) was used. However, also this corpus was too large. Therefore, the corpus was divided into 4 parts that were processed separately. This resulted in 4 separate lists of words with their termhood value. These lists were aggregated according to the following algorithm: The new list was the union of the four lists. If a term occurred in more than one list, the highest termhood value was taken.

3.1.3 Log Likelihood

The log likelihood for each word has been calculated with the algorithm described in (Rayson and Garside, 2000). This means, first computing the log likelihood according to formula 2.3 and then multiply the patterns that have a higher frequency in the background corpus than in the foreground corpus with -1. This algorithm has been implemented in the log-likelihood script that is part of colibri core (Van Gompel and Van Den Bosch, 2016).

The colibri script first creates a pattern model for both of the corpora and then computes the log likelihood for each pattern. A pattern model is a list of patterns with the frequencies of this pattern in the corpus. In this case a pattern was either a 1-, 2- or 3-gram of tokens with their POS-tag concatenated.

The colibri script returns a list of patterns with their log-likelihood value. The list is sorted in such a way that the pattern with the highest log-likelihood is on top and the one with the lowest on the bottom.

The returned list consisted of around 35M patterns. For practical reasons like speed and memory, only the top 1M patterns were taken into account.

3.1.4 Kullback-Leibler divergence for informativeness and phraseness

For each pattern the Kullback-Leibler divergence for Informativeness and Phraseness (KLIP) was computed for all 1-, 2-, and 3-grams in the corpora. The KLIP was computed with formula 2.5 in which $KLI(t)$ is the Kullback-Leibler Divergence for Informativeness (formula 2.1) and $KLP(t)$ the Kullback-Leibler Divergence for Phraseness. The γ (gamma) parameter ranges from 0 to 1 and can be used to control the balance between the KLI and the KLP. A high γ means a high focus on phraseness and a low gamma a high weight for Informativeness. In this experiment, 6 different values for gamma were tried (0.0, 0.2, 0.4, 0.6, 0,8, 1.0).

For this experiment a modified implementation of an existing script[3] has been used. The script was adapted to (1) not remove stop words for reasons described above and (2) not tokenize the corpus. This made for a fairer comparison with the Log Likelihood method. The tokenization was turned off, because the corpus was already tokenized in the preprocessing. Added to that, the Kullback-Leibler experiment must be kept as similar as possible to the Log-Likelihood experiment, which was the second reason to switch of the tokenization module.

[2]Many thanks to Macken, Lefever, and Hoste for providing me with this opportunity.

[3]https://github.com/suzanv/termprofiling

3.1.5 Linguistic Filters

A first analysis of the results of the experiments above (see section 4.2), revealed there were many false positives that could be ascribed to the style differences between the foreground and the background corpus. Function words and punctuation got high Log likelihood values. For this reason, linguistic filters were applied with the intention to only get n-grams that could form a legal term. These filters could be seen as a phraseness measure based on heuristics.

The applied filters are:

1. **Noun**: Selects patterns that exist of one Noun. For example: *rechtbank* (Eng: court)
2. **Verb**: Selects patterns that exist of one verb. For example: *moorden* (Eng: to Murder)
3. **Noun Preposition Noun (NPN)**: For example: *Officier van Justitie* (Eng: prosecutor)
4. **Adjective Noun (Adj. N)**: Selects an adjective followed by a noun. For example: *hoger beroep* (appeal)
5. **SPEC**. A miscellanuous POS-tag. Inspection revealed that some relevant terms had this POS tag. For example: *appellant* (Eng: appellant). This tag is often used for names, but as a result of the anonymization of the corpus, these were not captured by this tag.

3.1.6 Evaluation

The different methods return a list of patterns that are sorted according to the termhood value (Log Likelihood, KLIP or TeXSIS termhood.). The paradigm is that the higher the term is on the sorted list, the better the term represents the domain.

The ideal result would be a list where the top N items are all legally relevant terms and every pattern below N is not.

For this reason, the result is evaluated with precision and recall at N. Precision at n is the precision given the top N terms and recall @ N the recall for every term until rank N. Precision at N punishes for every non legal word high in the ordered list with legally relevant terms. Recall at N evaluates the number of legal terms that have been retrieved at a certain N. The final goal is to find a good balance between precision at N and recall at N so that for a certain top N, as much terms as possible are found (high recall at N) but at the cost of the least number of non legally relevant terms in this top N (high precision at N).

Apart from precision and recall at N, also Average Precision (AP) for the first 1000 terms was calculated. The reason for this is that this measure it is easier to compare the result of the different experiments than with precision for N terms. Average precision is defined as by formula 3.1[4] in which $P(r)$ is the precision at rank r. $rel(r)$ is a binary value depending on whether the term at rank r is relevant in the legal domain (1) or not relevant (0).

$$AP = \frac{\sum_{r=1}^{1000}(P(r) * \mathrm{rel}(r))}{\#relevantdocuments} \tag{3.1}$$

[4]Formula adapted from 'lindabeekeeper' at slideshare.net

As a golden standard an online juridical dictionary[5] was used. This word list contains a total of 4664 legal words. In the evaluation, a word is considered legally relevant if this word is in the list of juridischwoordenboek.

3.2 Relation Extraction

The method for extracting relations is adopted from Roller, Erk, and Boleda (2014). The idea of this method is to classify whether a pair of terms is a hypo- hypernym pair or not. This reduces the problem to a binary classification task.

In this thesis, a method is used that works by training word embeddings with word2vec on a legal corpus and use these features to classify a pair of words have a hypernym relation or not. In the upcoming sections, the method is explained in more detail.

3.2.1 Datasets

For these experiments, two data sets are used: 1) a legal thesaurus for training and evaluating the classifier. 2) a legal text corpus to train the word2vec model.

The Thesaurus

The legal thesaurus [6] that is used in this experiment is manually constructed by specialists in the legal domain. Some examples of pairs in the thesaurus are presented in table 3.1.

TABLE 3.1: A few examples of terms from the legal thesaurus

Pair		**Translation**	
Term	**Hypernym**	**Term**	**Hypernym**
radio-omroep	omroep	radio station	broadcasting station
vader	ouders	father	parents
adoptiekind	kind	adoptive child	child
vuurwapen	wapen	fire arm	weapon

In total, the thesaurus contains 8422 hypernym pairs consisting of a total of 9705 words. This means that there are words that are part of multiple pairs. There are two causes for this: 1) There can be multiple hyponyms of one hypernym. For example: Both rose and tulip are hyponyms of flower, resulting in flower occuring in two pairs (*tulip* - ***flower*** [7] and *rose* - ***flower***). 2) A term that is the hypernym in one pair, can be the hyponym in another pair. For example *flower* is a hypernym of *tulip*, but a hyponym of *plant*, resulting in *flower* being present in the pair *tulip* - *flower* as well as *flower* - *plant*.

Of all the words in the thesaurus, only 5266 could be used in the experiment for only those words were present in the word2vec model. This resulted in a total of usable 3050 pairs.

[5] www.juridischwoordenboek.nl (Eng: juridicaldictionary.nl)

[6] The legal thesaurus used in this experiment was kindly provided by Legal Intelligence

[7] in this thesis, relations are always presented as <hyponym> - <hypernum>

The Corpus

The corpus is the same verdict corpus as used in the term extraction task described above. The only difference was in the preprocessing.

An additional step was added to the preprocessing: All juridical multiword terms in the thesaurus were concatenated with underscores resulting in a term like *monistisch parlementair stelsel* being processed to *monistisch_parlementair_stelsel*. Word2vec only considers unigrams. By concatenating the multiword terms to one string, word 2 vec was therefore able to learn vectors for the multigrams.

Replacing the multiword terms, was done in order from long n-grams to short, meaning that n-grams consisting of more words were first concatenated, in order to always concatenate the longest possible n-gram. For example, the juridical term *monistisch parlementair stelsel* has as a substring the n-gram *parlementair stelsel*. In this case, the n-gram must be concatenated to *monistisch_parlementair_stelsel* and not to *parlementair_centrum*.

Ideally the predictions of the term finding experiment should be used for determining what terms need to be concatenated. However, the results of the term finding experiment (see chapter 4) were from such quality that they could not be used for the follow up.

3.2.2 The Word2Vec Models

Two word2vec models were trained, one skipgram model and one continuous bag of words model. Both models were trained with a window of five and a size of 300.

3.2.3 Features

From the word2vec models two types of features were extracted. Concatenated features and offset features.

The concatenated feature vectors were created by simply concatenating the vector of the hyponym ($\vec{u}$) with the vector of the hypernym ($\vec{v}$) resulting of a vector of length 600. The concatenated vectors then were normalized. The creation of the vector is presented in formula 3.2.

$$\vec{f}_{concat} = \frac{\langle \vec{u}; \vec{v} \rangle}{\|\langle \vec{u}; \vec{v} \rangle\|} \tag{3.2}$$

The offset vectors were created by subtracting the normalized vector of the hyponym($\vec{u}$) from the vector of the normalized hypernym ($\vec{v}$) (eq 3.3). Then the resulting vector ($\vec{d}$) was concatenated with the squared of $\vec{f}$ (eq 3.4 and 3.5).

$$\vec{d} = \|\vec{u}\| - \|\vec{v}\| \tag{3.3}$$

$$\vec{e} = \vec{d}^2 \tag{3.4}$$

$$\vec{f}_{offset} = \langle \vec{d} : \vec{e} \rangle \tag{3.5}$$

Negative Examples

For the training of the classifiers, also negative examples (examples of pairs that do not have a hypernym relation) are necessary. Two types of negative examples were created, random pairs and inverse pairs. The random pairs were two random words from the corpus. Those random pairs were made sure to meet 2 conditions:

1. No two words in a pair are the same. (A pair like *apple - apple* was prevented) [8]
2. A pair must not be an actual hypernym pair.

Some examples of random examples are presented in table 3.2

TABLE 3.2: Examples of random negative examples

Pair		**Translation**	
metaalschroot	claimverzoek	metal scrap	claim request
Koopmanstraat	herinneringsfactuur	Koopmanstraat (street name)	reminder invoice
voedingskosten	dalende	food costs	descending (Adj)
bemiddelingsperiode	stond	Conciliation agreement	stood

The other kind of negative examples were inverse hypernym tuples. These were created by inverting the order of actual hypernym pairs. For example: a pair like *apple - fruit* would be inverted to *fruit - apple*. In total one third of the positive examples was inverted. Some examples of inverted examples are presented in table 3.3.

TABLE 3.3: Examples of inverted negative examples

Pair		**Translation**	
schip	oorlogsschip	ship	war vessel
inkomen	jaarinkomen	income	yearly income
pleegouder	pleegvader	foster parent	foster father
rechterlijk vonnis	mondeling vonnis	judicial verdict	oral verdict

Data Split

The data was randomly split into two parts. A train set (75%), a test set (25%).

Overview Descriptives

An overview of how the dataset is constructed is presented in table 3.4.

TABLE 3.4: An overview of how the training set is compiled

	Positive examples	Random Negative Examples	Inverse Negative Examples	**Total**
Train	1414	2122	708	**4244**
Test	427	640	213	**1280**
Total	**2129**	**2862**	**921**	-

[8] An interesting study would be to investigate whether the addition of such pairs would influence the results.

3.2.4 Classifiers

Following Roller, Erk, and Boleda (2014), two types classifiers were tried. In Roller, Erk, and Boleda (2014), a logistic regression classifier was used for the offset vectors and a support vector machine (SVM) was used for the concatenated vectors. In this experiment, I tried both classifiers for both vectors.

For both the logistic classifier and the SVM the implementation of of scikit learn (Pedregosa et al., 2011) was used.

3.2.5 Evaluation

To evaluate both classifiers, precision, recall and F1-score was used. Apart from those measures, a manual error analysis of the false positives was performed.

Precision and recall were calculated for the positive predictions (a pair has a hypernym relation). Precision was calculated according to formula 3.6, recall with formula 3.7 and precision with formula 3.8.

In the case of the positive predictions, the true positives (TP) are the pairs that are correctly classified as having a hypernym relation and the false positives (FP) are the pairs mistakenly classified as having a hypernym relation. The false negatives (FN) are the pairs that do have a relation, but were classified as not having one.

$$Precision = \frac{TP}{TP+FP} \tag{3.6}$$

$$Recall = \frac{TP}{TP+FN} \tag{3.7}$$

$$F1 = 2 * \frac{Precision*Recall}{Precision+Recall} \tag{3.8}$$

Jaccard Overlap

Another way of evaluating the different classifiers is by measuring the Jaccard overlap between the lists of true positives. The aim of this is to determine to what extend the different classifiers classify the same pairs as having a relation or whether they are able to identify different pairs as hypernym pairs. The Jaccard overlap index (J) is defined in formula 3.9 in which A and B are collections of true positives for different classifiers.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.9}$$

Chapter 4

Term Extraction Results

4.1 TExSIS

The results for TExSIS (Macken, Lefever, and Hoste, 2013) are presented in figure 4.1. The average precision for the results of TExSIS is .0336.

The graph for precision shows a small peak at around N = 220. The height of this peak is about .09, meaning that of the top 220 terms, 9% is a legal term.

Examples of true positives are presented in Table 4.1 and examples of false positives are presented in Table 4.2. The first interesting thing to note is that all termhood values of the top-5 False positives are higher than the termhood value of the highest True Positive. The highest true positive only comes at place 18 in the ranked list of results. This results in the low average precision.

Most of the false positives in Table 4.2 could be considered legal terms. Except for *spoornummer*, which is a railroad term. The term *benadeelde* most of the times co-occurs with other words to form a pattern like *benaldeelde partij* (Eng: disadvantaged party). This is a thing that could be solved with more weight for phraseness.

TABLE 4.1: TExSIS True Positives

Pattern	**Translation**	**Termhood**
bewezenverklaring	statement that charge is proven	98.25
juncto	has a connection with	82.14
gewoonteheling	habitual healing	60.69
dagvaarding	subpoena	59.17
gevangenisstraf	imprisonment	56.63

TABLE 4.2: TExSIS False Positives

Pattern	**Translation**	**Termhood**
bewezenverklaarde	charge of which proof is validated	248.48
parketnummer	district number	183.88
terechtzitting	hearing	177.6
benadeelde	disadvantaged party	176.4
spoornr	track nr	141.62

4.2 Log-Likelihood

The results for the different filters for the Log-Likelihood measure are presented in figure 4.2. In this figure, all filters show the same pattern, namely, a spike in the beginning that then stabilizes at a low value (below .2). This indicates that in the high ranks there is a high concentration of legal words, but that later on, the concentration of legal words drops to about one in five words. In the case of the Noun filter this results in a precision at N = 1000 of about .2.

The shape of the Log-Likelihood results is similar to the shape of the TExSIS results (figure 4.1) and the KLIP results (figure e.g. 4.7)

No filter

When no filter is applied the first few results are False Positives. The first ‘spike’ occurs at N = 16. This means that the 15 patterns with the highest log-likelihood ratio, do not occur in the legal word list form juridischwoordenboek.nl. When observing these 15 patterns it strikes that many patterns consist only of punctuation or punctuation in combination with high frequent words. A few high ranking examples of these patterns are given in table 4.3.

TABLE 4.3: Examples of 'false positives' with high log-likelihood ratios

Pattern	Log-likelihood
.(LET) De(LID)	4208320
](LET)	4012150
[(LET)	3945600
.(LET) "(LET)	2895640

Filter: Noun

For this filter, the first spike occurs at N = 2, meaning that only the single noun pattern with the highest log-likelihood is not present in the word list. The patterns at N=3 and N=4, however, do not occur in the reference word-list. The three highest ranking words that do not occur in the reference list are presented in table 4.4.

TABLE 4.4: Examples of false positives for the Noun filter

Pattern	Translation	Log-likelihood
rechtbank	court	1179910
beroep	appeal	947694
uitspraak	verdict	817633

This is a remarkable top three of false Positives, because these terms are all legally relevant terms. They also seem to have a high phraseness, for the reason that they have a clear meaning as individual entity and are not part of a multiword term. An explanation for this could be that these terms are too *obvious* to be included in this legal word list. These terms could be considered common knowledge within as well as outside of the legal domain in the Netherlands, hence they are not included in the word list. Another reason could be that the creators just did not think about including them.

More specific terms (of which the terms is table 4.4 are a substring) are present in the reference word list. For example *arrondissementsrechtbank* (regional court), *hoger beroep* (synonymous to *beroep*[1] *Beroep can also mean profession, but in the legal domain it is often interchangable with 'hoger beroep')*, and *einduitspraak* (final verdict).

Filter: Verb

The first true positive occurs at N = 6 meaning that the 5 words with the highest log likelihood are not in the reference word list. These words are displayed in table 4.5.

Except from *verdachte (suspect)*, all patterns are function words and therefore are not legally relevant terms. It is therefore remarkable that those words have such a high log-likelihood since you would expect them to have a high frequency as well in the background corpus.

TABLE 4.5: Top 5 verb unigrams. (all false positives)

Pattern	Translation	Log-likelihood
heeft	has (3SG of have)	1972240
zegt	says	700455
verdachte	suspect	686796
werd	became (SG)	430882
worden	become (PL or INF)	364732

Another remarkable thing about the verb unigrams is that many of the top results are actually nominalisations of verbs and therefore nouns instead of verbs. A few examples of this are *gedaagde* (defendant) with a log-likelihood of 364602, and *betrokkene* (involved person) with a log-likelihood of 162656. These mistakes can be explained by the fact that the automatic memory based tagger in frog is not perfect.

Filter: Noun Preposition Noun (NPN)

The precision for the NPN-filter never gets really high. The highest precision is .07 at N=54. The 16 NPN patterns with the highest log-likelihood are all false positives.

Examples of high ranking false positives are presented in table 4.6. The second example in the table is a sub-phrase of a set of larger phrase that typically occur in legal verdicts. This phrase, *application of article* can be followed by any code referring to an article in the Dutch law codification. For example: *toepassing van artikel 1 van het Wetboek van Strafrecht* (Eng: *application of article 1 of the Criminal Code*). This complete phrase does not have a place in a thesaurus, nor does the sub-phrase presented in table 4.6. However, this sub-phrase is an example of a pattern that could be filtered out with a more elaborate measure for phraseness than the linguistic filter used in this experiment, since this sub-phrase is not a complete phrase on its own.

An explanation for why *onderzoek ter zitting* is not present in the word list could be that the phrase *onderzoek ter terechtzitting* is present in the word list. This phrase is synonymous to *onderzoek ter zitting* and also very similar in writing and including both of them could appear redundant for the creators of the word list.

In figure 4.2 the (blue) line corresponding with this filter stops at N = 366, indicating that there are only 366 instances of NPN patterns in the top one million patterns with highest log-likelihood.

[1]

TABLE 4.6: Examples of false positives for the NPN filter

Pattern	Translation	log-likelihood
Artikel(N) zonder(P) titel(N)	article without a title	184221.0
toepassing(N) van(P) artikel(N)	application of law article	58608.6
onderzoek(N) ter(P) zitting(N)	sitting of the court	58301.1

Filter: Adj N

This filter scores relatively well on precision and recall. Figure 4.2 shows that this filter is the third best after the Noun filter en the combined filter.

A few examples of true positives are presented in table 4.7

TABLE 4.7: Examles of true positives consisting of an adjective and a noun

Pattern	Translation	Log Likelihood
hoger beroep	appeal	463601.0
enkelvoudige kamer	single judge	25165.1
Europese Unie	European Union	21925.1
Europese Commissie	European Commission	15280.7
dagelijks bestuur	board	12603.1

Examples of false positives are presented in table 4.8. *Last year* and *last week* are clearly not legal terms, however, *statutory interest* and *provisional provision* are. *General law* is an example of a false positive that could be solved by using a phraseness measure. There are many titles of Dutch laws that start with the phrase *Algemene wet*. For example: ***Algemene wet*** *bestuursrecht* (Eng: General Administrative Law Act) and ***Algemene wet*** *Gelijke Behandeling* (Eng: Equal Treatment Act).

Note that *Algemene wet* starts with a capital letter. This signifies that it is (the start of) a title of a law. This observation justifies the choice of not lowercasing the corpora.

TABLE 4.8: High ranking False Positives after applying the Adjective Noun filter

Pattern	Translation	Log Likelihood
vorig jaar	last year	191207.0
Algemene wet	General law	94727.0
vorige week	last week	82708.5
voorlopige voorziening	provisional provision	59127.4
wettelijke rente	statutory interest	54800.8

Filter: SPEC

The SPEC-filter does not show a clear peak for precision at the beginning like the other examples in figure 4.2. It stabilizes very early at a low precision. A few examples of true positives are presented in table 4.9. All items are nouns but that got the miscellaneous tag. One of the reasons they might have gotten the SPEC-tag, is that (three of them) start with a capital letter [2]

TABLE 4.9: Examles of true positives with a SPEC-tag

Pattern	Tanslation	log-likelihood
appellant	appellant	95040.5
Eiser	plaintiff	13509.9
Appellant	Appellant	13334.4
Verweerder	Defendant	8881.68

Some examples of false positives for this filter are presented in table 4.10. Again, the most important reason for terms being in this list, is that they start with a capital. Some false positives that actually should be in a legal thesaurus are *mr.* and *Uwv*.

TABLE 4.10: Examles of false positives with a SPEC-tag

Pattern	Tanslation	log-likelihood
Van	From/Of	1072430.0
De	The	718478.0
€	€	708809.0
mr.	abbr. master in law	571927.0
Uwv	abbr. Organization for employee insurances	145619.0

Combination of Filters

The results of the combined filters are also present in figure 4.2. Remarkable is that the precision is lower than the results for the only nouns filter.

Comparison of filters

In table 4.11 the Average precision values for all the filters are presented. According to this table, the N-filter has the best results. However, this filter only finds nouns, and in reality, more terms than only noun unigrams need to be found. For this reason, it might be better to use the combined filters or combine even more filters.

TABLE 4.11: Comparison of the Filters of the Log-Likelihood

	No Filter	N	V	N VZ N	Combined
AP	.0523	.1665	.0413	.0496	.1379

4.3 Kullback-Leibler divergence for Informativeness and Phraseness

No filter

The results for the Kullback-Leibler Divergence for Informativeness and Phraseness (KLIP) with no linguistic filter are presented in Figure 4.3. When this figure is compared with the Log-Likelihood results in Figure 4.2 it is clear that for precision as

[2]the corpora were not lower cased on purpose for reasons explained in section 3.2.1

well as recall, the KLIP method performs better. When Figure 4.3 is analyzed on its own, it can be observed that the patternmodels with lower values for γ perform better than those with higher values. This means that when no linguistic filters are applied, it is better to give a low weight to the phraseness (KLP) or to leave it out completely (set γ to 0.0).

Some examples of true positives with the KLIP-method without linguistic filter are presented in table 4.12. Both *besluit* and appellant also occur in the top 3 of the Log-Likelihood method. *Hoger beroep* does not occur high in the results of the Log-Likelihood results. This is likely an example of a term that got a higher score as a result of a high phraseness. The high phraseness is caused by the fact that it has a the term consists of many words.

TABLE 4.12: True Positives found with KLIP and no linguistic filter

Pattern	Translation	KLIP
appellant(N)	appellant	.003137
hoger (ADJ) beroep(N)	appeal	.003050
besluit(N)	decision	.001425

Examples of false positives are presented in table 4.13. Like with the Log-Likelihood method, these false positives with the highest values are no possible legal terms and most likely the result of style differences between the newspaper corpus and the verdict corpus.

Filter: Noun

The results for the noun filter are presented in Figure 4.4. For this filter, KLIP scores better than the Log-Likelihood (see Figure 4.2).

As a result of the filter, all returned terms are have a size of 1 and therefore there is no difference for the different values of γ. There are no results for $\gamma = 1$, as the KLP is by definition equal to 0 for all unigrams.

Examples of false positives are presented in table . The fact that *rechtbank* is not in the word list is curious and has been addressed above. The female words *appellante* and *eiseres* are both not in the golden standard list, but their male versions are. These are nice examples of words that would be desirable to have in a thesaurus at the back-end of a search engine to know that the meaning of both words is essentially the same.

A possible reason for *beroep* not being in the reference word list is that a *beroep* means the same as *hoger beroep*, and most of the times is used in the latter form. This is an example where the phraseness failed to do its job 4.12 since *hoger beroep* only has a KLIP score of 0.00336 when the γ parameter is 1 and thus giving the maximum

TABLE 4.13: False Positives found with KLIP and no linguistic filter

Pattern	Translation	KLIP
van(P)	from	.008590
van(P) de(ART)	from the	.007139
heeft(V)	has (3SG)	.005854

amount of weight to phraseness. In this case one would have expected a higher KLIP score for *hoger beroep*. The fact that the KLIP is not higher means that *beroep* on its own is a much used term and should therefore be part of the thesaurus.

TABLE 4.14: examples of false positives after applying the a filter that selects single nouns

Pattern	Translation	KLIP
rechtbank	court	.00501
appellante	female appellant	.00401
beroep	appeal	.00401
eiseres	female plaintiff	.00366

Filter: Verb

The results for the Verb filter are presented in Figure 4.5. As is the case for the results for no filter and the noun filter, the results are better than for the Log Likelihood.

Compared to the other KLIP-scores, the Verb Filter has rather low values for precision and recall. This is similar to the case with the Log Likelihood scores.

Examples of true positives are presented in table 4.15. First notable fact is that there is a large difference between the KLIP of the highest true positive (*gedaagde*) and the KLIP of the one but highest true positive (*betrokkene*).

Another remarkable result that was also visible in the results for the Log Likelihood is that some of the best results are actually not verbs but derivatives of verbs. This is a result of the fact that the automatic POS-tagger is not perfect.

TABLE 4.15: Examples of true positives after the Verb filter

Pattern	Translation	KLIP
gedaagde(N or ADJ)	defendant	.00284
betrokkene(N)	involved party	.00069
gegrond(ADJ)	justified	.00059
gegeven(N or ADJ or V)	fact (or: given)	.00033

Examples of false positives are in table 4.17. Curious about this is that both *heeft* and *is* are content words and should not have a high KLIP. Both*verdachte* and *bestreden* are words that could be in a legal thesaurus.

TABLE 4.16: My caption

Pattern	Translation	KLIP
heeft (V)	has (2SG of have)	.00439
verdachte (N or ADJ)	suspect or suspicious	.00197
is (V)	is	.00147
bestreden (ADJ or V)	contested	.00115

Filter: Noun Preposition Noun

Results in Figure 4.6. There are slight differences between the different values for γ, but not as large as in the experiment with no filter.

Examples of false positives are in table 4.17. These are examples we already saw above at the Log Likelihood

TABLE 4.17: My caption

Pattern	Translation	KLIP
onderzoek ter zitting	sitting of the court	.00067
toepassing van artikel	application of article	.00052
college van burgermeester	college of mayor	.00024

Filters Combined

Comparison of KLIP

In table 4.18, the average precision is presented for each experiment with KLIP. As expected, there is no differences between the gammas with the unigram filters, because the KLP is zero for every unigram. For the NPN-filter, a high gamma seems beneficiary. At the filters where there is no fixed number of words in an n-gram (No filter and Combined Filters), a low gamma of 0.2, ergo a low but nonzero weight for the phraseness, is the best.

TABLE 4.18: Comparison of Results of KLIP

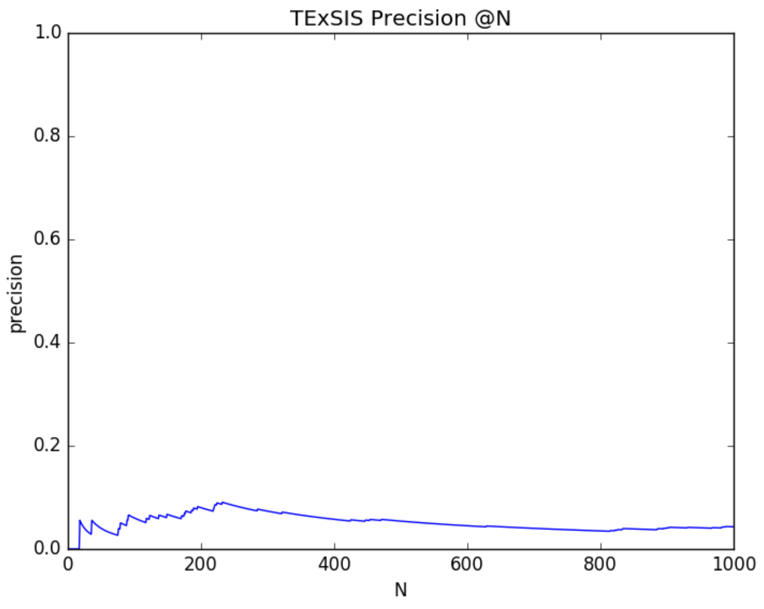
Gamma	No Filter	N	V	N P N	Combined
0.0	.0981	**.2199**	.0705	.0299	.1832
0.2	.0958	**.2199**	.0705	.0335	.1840
0.4	.0931	**.2199**	.0705	.0367	.1810
0.6	.0740	**.2199**	.0705	.0383	.1745
0.8	.0517	**.2199**	.0705	.0394	.1670
1.0	.0333	0	0	**.0395**	**.0395**

4.4 Comparison of TeXSIS, Log Likelihood and KLIP

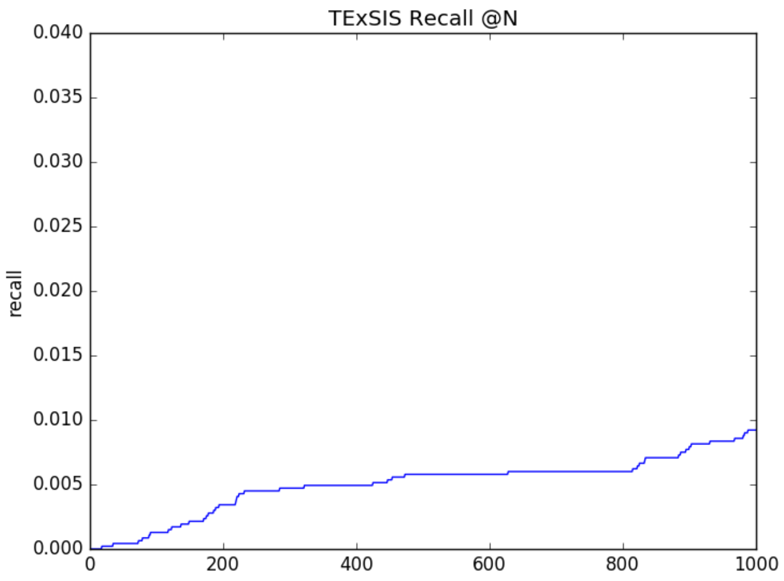
In table 4.19, for each method of term extraction, the Average Precision for no filters and the filters combined is presented. This table shows that the KLIP method scores best.

TABLE 4.19: Table that compares the different termhood measures. in the case of KLIP, the result with the highest AP is presented (hence the different values for γ). With TExSIS, no filters could be applied.

	TExSIS	Log-Likelihood	KLIP
No Filter	.0336	.0523	**.0981** ($\gamma = 0.0$)
Combined Filter	N.A.	.1379	**.1840** ($\gamma = 0.2$)

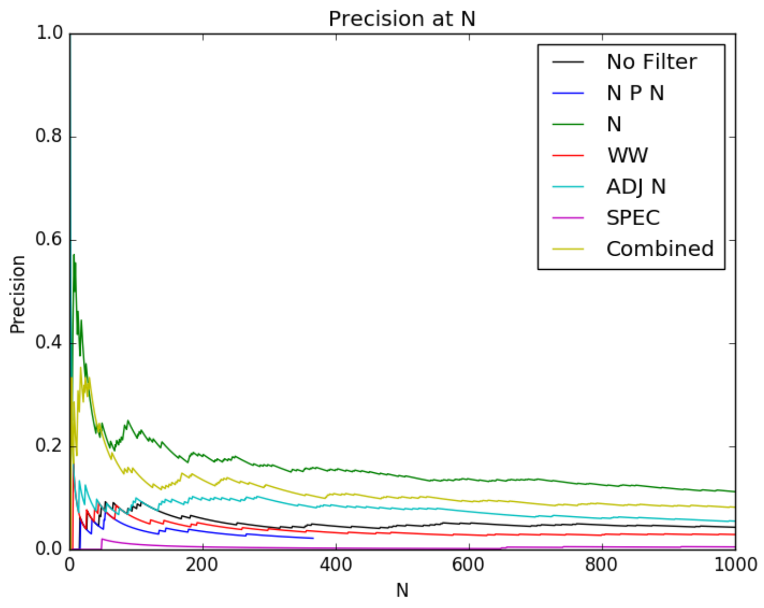


(A) Precision

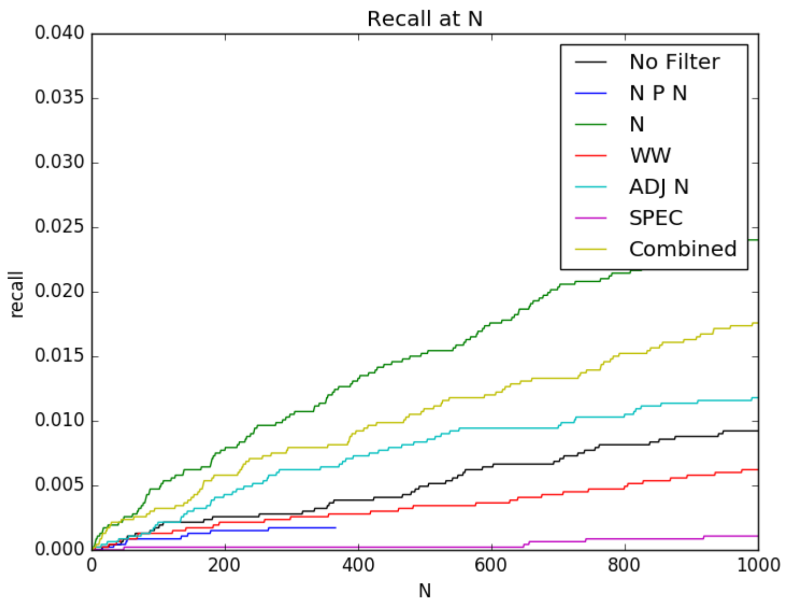


(B) Recall

FIGURE 4.1: Precision and Recall at N for TExSIS. The x-axis is N and the y-axis the precision (A) or recall (B).

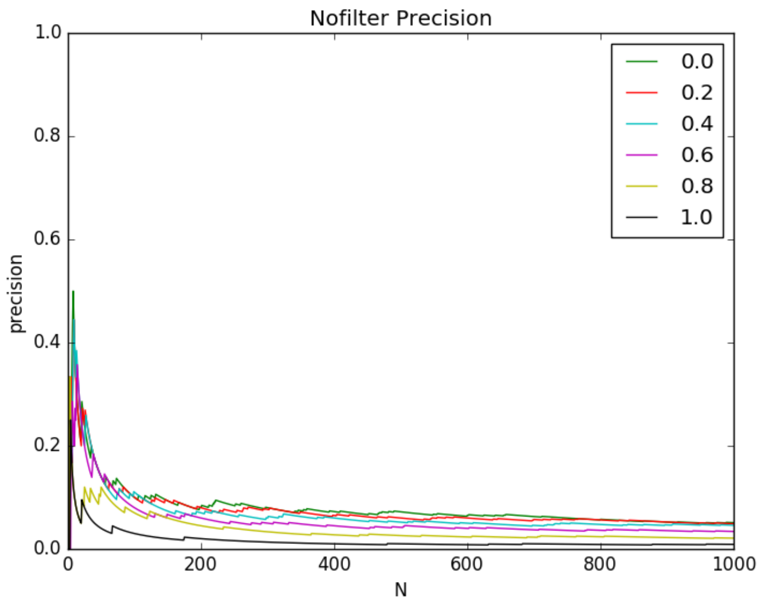


(A) Precision

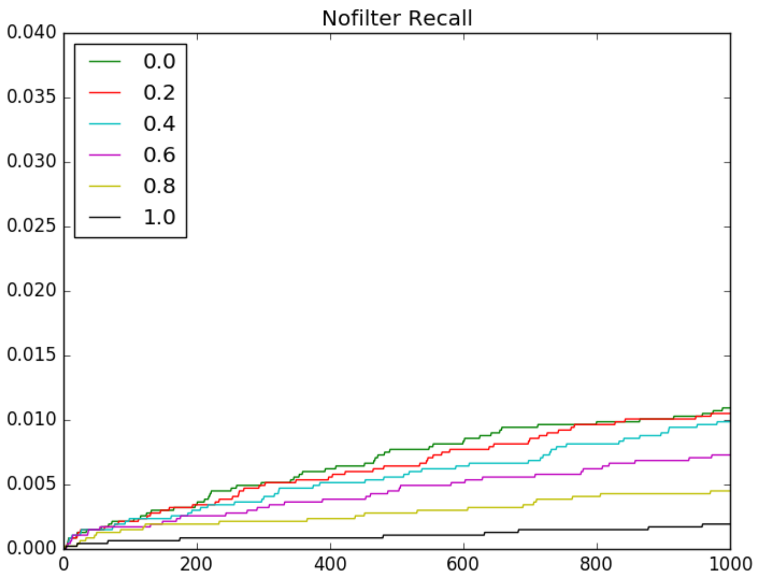


(B) Recall

FIGURE 4.2: The precision at N and recall at N for the different filters on the Log-likelihood data. The x-axis shows N and the y-axis the precision (A) or recall (B).

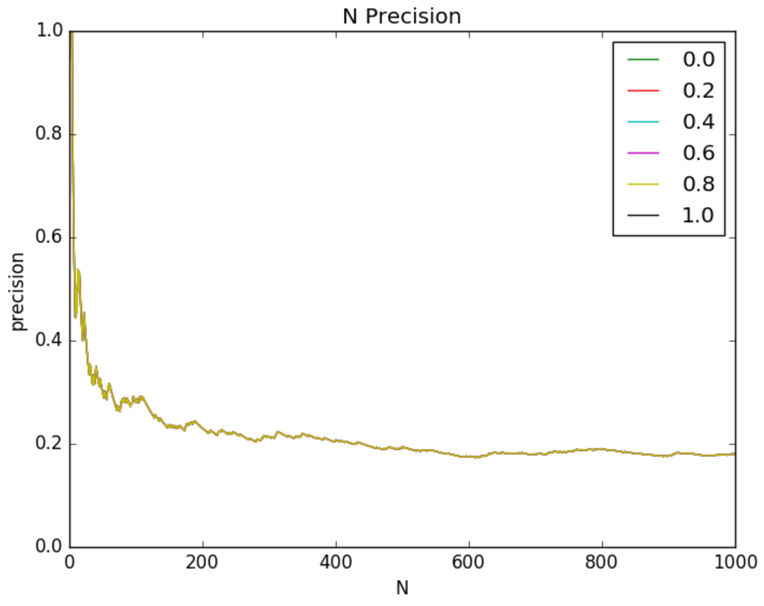


(A) Precision

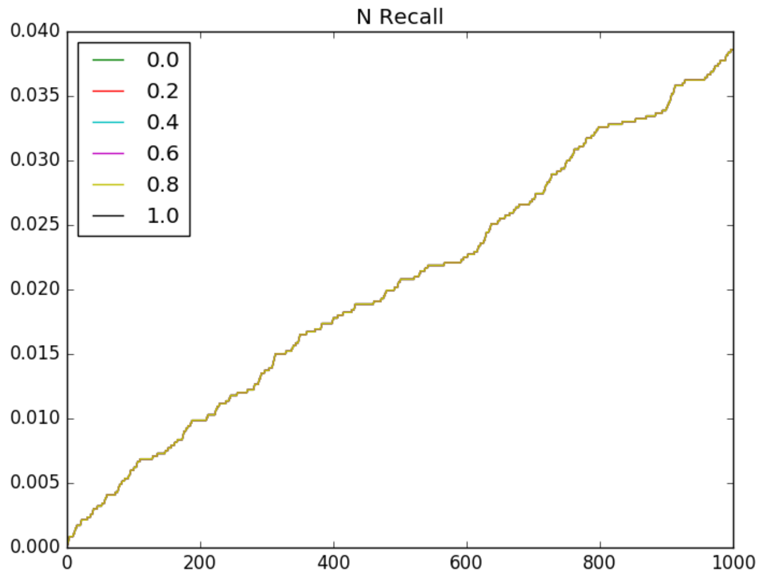


(B) Recall

FIGURE 4.3: Precision and Recall at N for the KLIP-method. This figure shows the results when no filters are applied. The x-axis shows N and the y-axis the precision (A) or recall (B). The different lines represent different values for γ

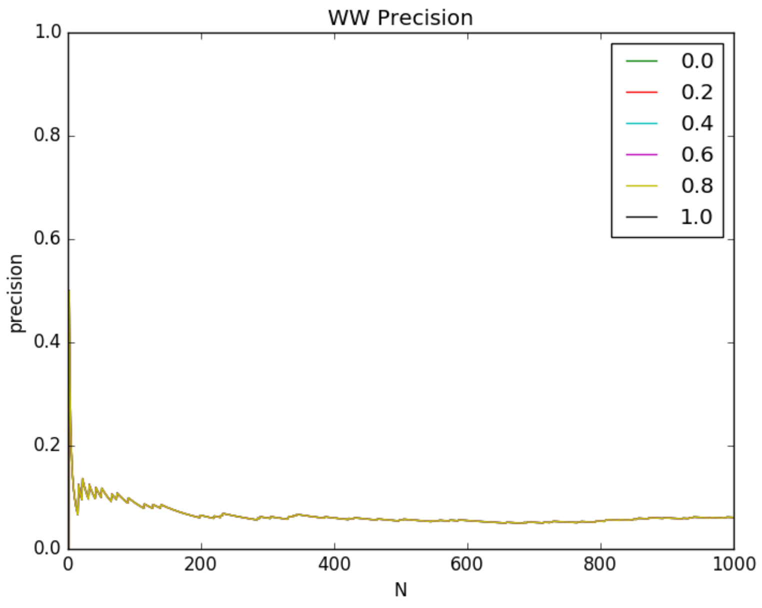


(A) Precision

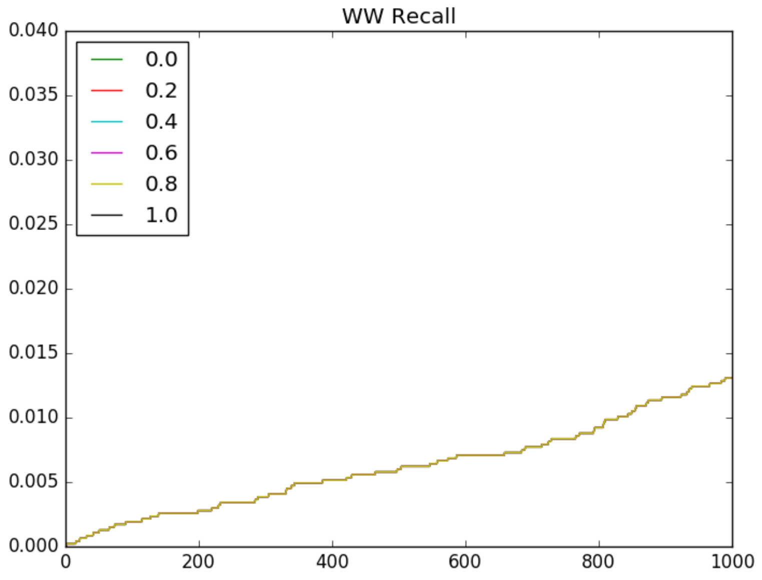


(B) Recall

FIGURE 4.4: Precision and Recall at N when the Noun filter is applied. The x-axis shows N and the y-axis shows precision (A) or recall (B). There are more lines indicated in the legend than visible in the graph. This is because all lines are the same and overlap. The different lines in the legend represent different values for γ

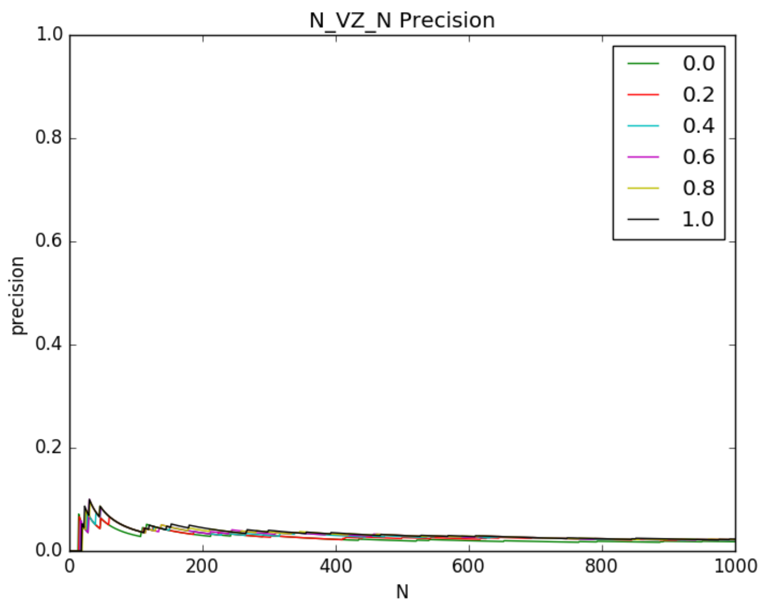


(A) Precision

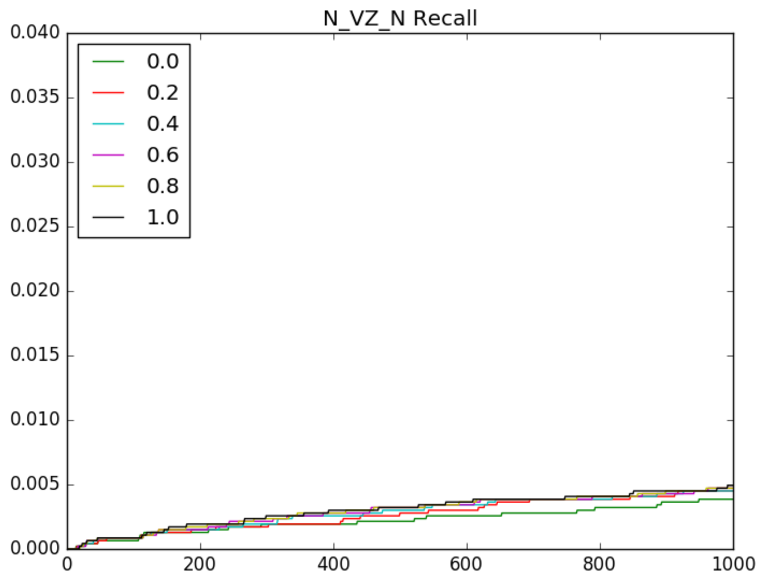


(B) Recall

FIGURE 4.5: Precision and Recall at N when the Verb (Dutch POS-tag = WW) filter is applied. The x-axis shows N and the y-axis shows precision (A) or recall (B). There are more lines indicated in the legend than visible in the graph. This is because all lines are the same and overlap. The different lines in the legend represent different values for γ

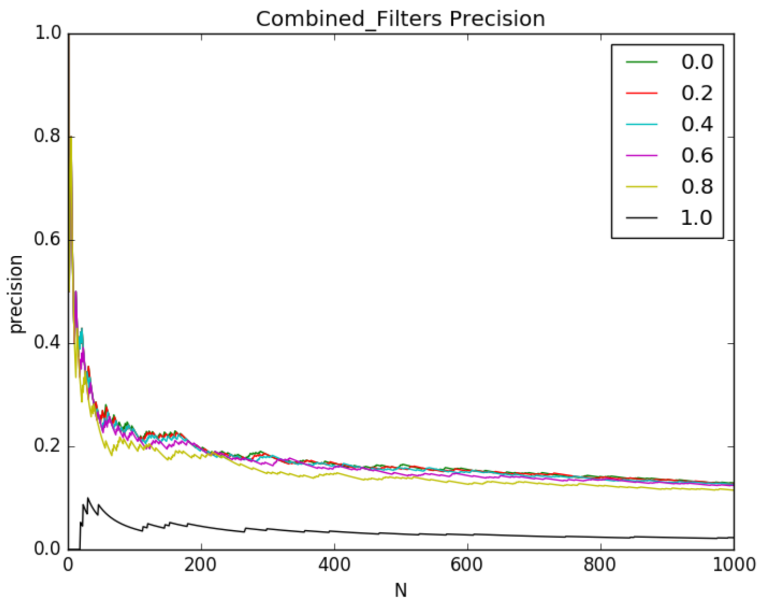


(A) Precision

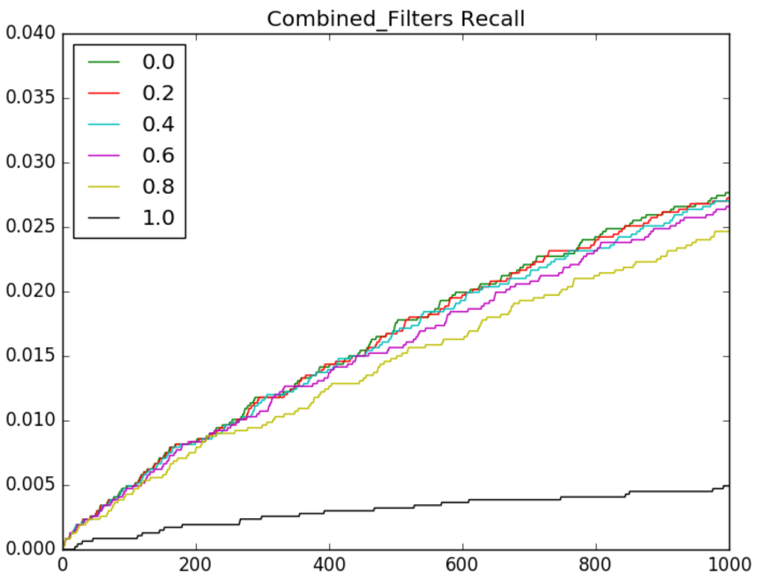


(B) Recall

FIGURE 4.6: Precision and Recall at N when the Noun-Preposition-Noun (Dutch POS-tag for Preposition = VZ) filter is applied. The x-axis shows N and the y-axis shows precision (A) or recall (B). The different lines represent different values for γ



(A) Precision



(B) Recall

FIGURE 4.7: Precision and Recall at N when the filters are combined. The x-axis shows N and the y-axis shows precision (A) or recall (B). The different lines represent different values for γ

Chapter 5

Relation Finding Results

In this experiment different classifiers were tried. The classifiers aimed to predict whether or not term pairs have a hypernym relation. Two types of classifiers (Support Vector Machine and Logistic regression) and two types of vectors (concatenated vectors and offset vectors) are compared. The third tested variable was the type of word2vec model used: skipgram or continuous bag of words.

In tables 5.1 and 5.2 the results for the continuous bag of words model (cbow) and the skipgram model are presented.

TABLE 5.1: Precision Recall and F1-score for the CBOW model vectors.

		P_1	R_1	$F1_1$
Concat	SVM	.83	**.81**	**.82**
	LogReg	.79	.75	.77
Offset	SVM	**.85**	.76	.81
	LogReg	.75	.79	.77

TABLE 5.2: Precision Recall and F1-score for the Skipgram model.

		P_1	R_1	$F1_1$
Concat	SVM	.71	**.83**	.76
	LogReg	.78	.80	.79
Offset	SVM	**.84**	.78	**.81**
	LogReg	.74	.75	.74

Continous bag of words versus skipgram

When comparing the skipgram-model with the cbow-model, it appears that the continuous bag of words (cbow) model performs better than the skipgram model. Although the differences are rather small, it is interesting that the cbow-model outperforms skipgram-model, because previous studies (Baroni et al., 2012; Roller, Erk, and Boleda, 2014) only reported the use of the skipgram model and did not report results for the cbow model.

concatenation vectors versus offset vectors

The concatenation vectors outperform the offset vectors. This is contrary to the experiment of Roller, Erk, and Boleda (2014), in which the offset vectors (or difference vectors as they call it) have better results than the concatenation vectors.

SVM versus logistic regression

For the concatenation vectors, it differed what was the best classifier. For the CBOW model, the SVM-classifier performed best, but for the skipgram-model, the logistic regression classifier was the best for the concatenation vectors.

5.1 Error Analysis

5.1.1 Analysis of False Positives

In tables 5.3 and 5.4 overviews are presented of how the false positives are devided over the random negative examples (random pairs) and the inverse negative examples (hypernym pairs that were inversed). These tables show the percentage of the false positives belonging to both types of negative examples. The percentages always sum to 100, because all false positives are necessarily one of both.

No claims can be made about the difference between the types of False positives. For both skipgram and cbow models, the concat-svm classifier has a majority of the inverse false positives. No other pattern seems to occur. This means that based on these results, you can not say that the classifiers are not particularly better in one type of negative example.

TABLE 5.3: Overview of the composition of the False Positives for the cbow model vectors

		% Random Examples	% Inversion Examples
Concat	SVM	22	78
	LogReg	63	37
Offset	SVM	78	22
	LogReg	14	86

TABLE 5.4: Overview of the composition of the False Positives for the skipgram model vectors

		% Random Examples	% Inversion Examples
Concat	SVM	6	94
	LogReg	17	83
Offset	SVM	51	49
	LogReg	65	35

In tables 5.5 and 5.6 the jaccard similarity between the different lists of false positives is presented. These tables show that the overlap between the lists is not very high. This means that different classifiers recognize different pairs as hypernym pairs.

In table 5.7, some examples of false positives are presented. This table is presented to give an idea of how the false positives look and emphatically not to show a pattern, since no pattern was found.

TABLE 5.5: Jaccard overlap between the different lists of False positives for the cbow models

		Concat		**Offset**	
		SVM	**LogReg**	**SVM**	**LogReg**
Concat	**SVM**	1	.50	.19	.25
	LogReg	.50	1	.23	.25
Offset	**SVM**	.19	.23	1	.28
	LogReg	.25	.25	.28	1

TABLE 5.6: Jaccard overlap between the different lists of False positives for the skipgram models

		Concat		**Offset**	
		SVM	**LogReg**	**SVM**	**LogReg**
Concat	**SVM**	1	.48	.31	.18
	LogReg	.48	1	.27	.24
Offset	**SVM**	.31	.27	1	.27
	LogReg	.18	.24	.20	1

TABLE 5.7: Examples of false positives.

	Pair		**Translation**		**Experiement**
Random Examples	spoorverkeer	hittebron	railway traffic	heat source	CBOW-LogReg-Concat
	ommeloop	exploiteerde	circulation	exploited	CBOW-SVM-Concat
	rits	jacht-overeenkomst	zipper	hunting agreement	CBOW-LogReg-Offset
	servicekosten	roltrappen	service charge	escalators	Skipgram-LogReg-Concat
Inverted Examples	computer	spelcomputer	computer	game console	CBOW-LogReg-Concat
	gezondheids-schade	letselschade	health damage	injury	CBOW-SVM-Concat
	luchtvaartuig	vliegtuig	aircraft	airplane	CBOW-LogReg-Offset
	pleegouder	pleegvader	foster parent	foster father	Skipgram-LogReg-Concat

Chapter 6

Discussion and Conclusion

6.1 Discussion: Term Extraction

Over all, the best method for extracting terms is the Kullback Leibler Divergence for Informativeness and Phraseness (KLIP). This method reached an average precision of .1840. This is lower than the results reported in Verberne et al. (2016). The result of the KLIP was better than the result of the Log Likelihood (.1379) and TExSIS (.0336). KLIP outperforming the Log Likelihood is in line with the results in Verberne et al. (2016).

The TExSIS tool performing this poorly is remarkable. In Macken, Lefever, and Hoste (2013) a precision of .58 is reported for term extraction in Dutch texts. One of the reasons for this low performance could be that the background corpus in TExSIS is the Europarl corpus. And as Europarl consists of transcriptions of the law making process in the European Union, there might be quite some juridical terms in this corpus.

Another disadvantage of TExSIS, was that it could not handle the size of the full corpus, which required selecting a subcorpus and even that subcorpus had to be split into 4 parts. This splitting and later the aggregating of the results is not good for the result.

It would be interesting to see how the results would improve if those problems could be overcome and TExSIS would be optimized for term extraction in the legal domain. I expect that plugging another background corpus in TExSIS would not be a problem, and a more memory friendly implementation of TExSIS could help the corpus size problem. Or otherwise, TExSIS could be installed on a computer with more memory.

The problem is that the TExSIS software is not open source, which makes it hard to experiment with it. The results in this thesis were acquired by sending the owners of TExSIS the corpus and let them run it through TExSIS.

Recall

There are a lot of False negative in all the experiments. This leads to a low recall. The main cause for this was that many of the terms did not occur in the foreground corpus. This might have to do with the fact that only verdicts and no other text types were used. As Van Opijnen and Santos (2017) notes, there are many different types of legal documents. Some words are just not likely to appear in some text types. Verdicts are especially a type of text that needs to be readable by people that are not legal experts. As a result of this, jargon can be avoided.

The solution to this problem is using a larger foreground corpus containing a larger variety of legal texts. For example, also add text types academic literature, blogs and textbooks. By using a larger variety of texts more legal terms will be present automatically. The size of the background corpus should not really impact the quality of the term extraction methods (Knoth et al., 2009). However, I think it would be sound to increase the size of the background corpus accordingly to the increase of the foreground corpus.

Precision

Also the high number of false positives influenced the result. The high number of false positives caused a low precision. One of the causes for the high number of false positives was the quality of the golden standard list. Manual inspection of the false positives showed that many of the terms marked as false positives were actually legal terms, but were not present on the list of juridischwoordenboek.nl.

One thing that this shows is that it is indeed difficult to make a complete word list by hand, since there are always therms that are forgotten. This means that an automatic unbiased system could be of good assistance when creating a word list and therefore there is a legitimate need for a term extraction engine.

To better evaluate a term extraction system however, a larger word list should be used. Combining legal word lists for different sources might be a good way to accomplish this. Furthermore, it is advisable to always do a manual inspection of the results for the precision and recall measures do not always accurately reflect the true quality of a system for the reason described above.

Apart from the incorrect false positives, there were also a large amount of actual false positives. Part of these could be solved by using larger corpora and fine tuning parameters. However, I suspect that only that is not enough. For this reason I would propose a more thorough study to the filters. The aim of this study would then be to find the best filters for filtering out the false positives. Such filters could either be linguistic filters or something like a binary classifier that classifies whether or not a term is a legal term. This classifier could work on features like occurrence, length or even on embedding features.

6.2 Discussion Relation Extraction

The results of the relation extraction match the results in previous literature. Compared to the most similar studies that also train binary classifiers, the F1 score of .81 is higher than the .76 from Baroni et al. (2012) and also close to Roller, Erk, and Boleda (2014) who report a maximum average accuracy of .84. (Which is more or less comparable with the precision of .85 in this thesis.)

A big problem with the system presented in this thesis is that it is not a relation extraction system: It is a classifier. The classifier could be used to extract relations, but it will perform worse than the classifier on its own. A system using this classifier to find hypernyms would work like this: To find a hypernym for a given legal term, this term must be paired up with all other legal terms and for every pair individually, it must be determined whether or not it is a hypernym pair. This theoretically means that, when one pair is paired up with 1000 other pairs and classified with a precision of .84, there are 160 false positives for this term. This means that the system will return a list of about 160 terms, of which only one[1] is the correct hypernym. A recall

[1]assuming that every term has only one hypernym.

of .81 would mean that in 19 per cent of the terms, the correct hypernym is not even present in this list of 160 terms.

This is why systems that search directly for hypernyms (Hearst, 1992; Agichtein and Gravano, 2000; Fu et al., 2014, a.o.) might be better options. A downside of these systems is that they tend to have a low recall (Hearst, 1992; Agichtein and Gravano, 2000) or that they are really complex in the case of Fu et al. (2014). Which one of the the reasons why I preferred the classifier for this thesis.

6.3 Future Research

In the first place, future research could focus on optimizing the main two parts of a thesaurus learning engine described in this paper. Further directions of future research could be studies in how the current systems could be effectively used in the industry.

6.3.1 Optimizing Term Extraction

For optimizing the term extraction, experiments can be done with adding more corpora to include more juridical terms and this way improving the recall. The precision could be improved by experimenting with more filters. Instead of the linguistic filters, a classifier can be used to classify whether or not a term is a legal term. Possible features in this classifier could be embedding features, but also less abstract features like occurrence count, POS-tag, length or words in the context.

This way the first (extraction) stage could be optimized for recall and the second (filter) stage could be optimized for precision.

6.3.2 Optimizing Relation Extraction

A possible way to improve the classifiers presented in this thesis could be to combine them. As the Jaccard indices showed, there was not much overlap between the true positives of the different classifiers. So by combining the results of the classifiers, the recall could be improved.

A more subtle way of combining the classifiers would be in a voting system where every system can vote on a solution for each pair and pairs with the highest votes are presented first. This might result in a better balance between precision and recall.

Other improvements that could be investigated are using a larger corpus to train the word2vec models. This will improve the quality of the vectors. Furthermore, studies could be done to the effect of the different hyper parameters of word2vec on the results.

A more explorative study would be looking whether the training material should be legal in nature. It would be an interesting experiment to look whether hypernym-relations in the hyperspace of word2vec could be generalized: whether the relations in the legal domain are similar to relations in the biomedical or the general domain. If this would indeed be the case, corpora and thesauri from other domains than the legal domain could be used in training a legal relation classifier. The benefit of this is that multiple corpora from different domains could be combined. This would lead to a sudden increase of the amount of data available to train the classifier.

6.3.3 Thesaurus Learning in Practice

The systems presented in this paper are not good enough to work autonomously. Let alone, to form an autonomous thesaurus learning engine. For this reason I would propose a system that has a human in the loop. Such a system would assists a human in making a thesaurus.

In the case of the term extraction, the system would propose a list of words and the human expert only has to say which terms in the list are legal terms. This way the computer speeds up the work of the thesaurus maker for judging words takes less effort than coming up with legal terms oneself.

One thing that must not be forgotten is that in real life, we would not have to start at zero. There are already many word lists around. For this reason, the system only would have to suggest terms that are not in existing word lists yet. This would significantly reduce the amount of human effort that has to be done. Therefore it would be even better to have a system that helps with updating an existing word list than creating an entire new one. Only in some rare cases when a word list must be created for a total new (sub) area, it would be desirable to start from scratch.

The same strategy can be applied to the relation extraction. For finding relations autonomously, the system performs too poor. However, a system could be devised that makes suggestions to a thesaurus maker. The classifier should be trained on existing thesauri, but as the thesaurus maker adds more pairs to the thesaurus, the classifier could be retrained with the new pairs added to the training data. This way the automatic assistant gets better as the thesaurus grows in size.

6.4 Conclusion

In this thesis, two components of a thesaurus learning system where studied; term extraction and relation classification. For term extraction, terms were scored with their termhood value and then sorted accordingly. Kullback Leibler Divergence for Informativeness and Phraseness (KLIP) appeared to be the best termhood measure. However, lots of improvements should be made before it is practically useful for term extraction.

Also for relation extraction, further research is necessary: In this thesis I was able to train a classifier that could classify whether to terms have a hypernym relation. This system scored as good as previous studies on this task. The classifier as such, however is not yet a relation extraction system. Therefore, the classifier is not ready to be used in practice. Future research should reveal how the system best could be used for relation extraction.

The main two research questions in this thesis were:

1. What is the best method for extracting legal terms from a Dutch verdict corpus?
2. What is the best method for relation classification on Dutch legal terms?

The answer to the first question is: *The best method for extracting legal terms from a Dutch verdict corpus is via the KLDIV method.*

The answer to the second research question is: *The best method for classifying relations between Dutch legal terms is concatenating word vectors trained from a cbow-model and classifying them with an SVM-classifier.*

Bibliography

Adam Wyner Raquel Mochales-Palau, Marie-Francine Moens and David Milward (2010). "Approaches to Text Mining Arguments from Legal Cases". In: *Semantic processing of legal texts: Where the language of law meets the law of language*. Ed. by Enrico Francesconi et al. Berlin: Springer. Chap. 4, pp. 60–79.

Agichtein, Eugene and Luis Gravano (2000). "Snowball: Extracting relations from large plain-text collections". In: *Proceedings of the fifth ACM conference on Digital libraries*. ACM, pp. 85–94.

Aitchison, Jean, Alan Gilchrist, and David Bawden (2000). *Thesaurus construction and use: a practical manual*. Psychology Press.

Aletras, Nikolaos et al. (2016). "Predicting judicial decisions of the European Court of Human Rights: A natural language processing perspective". In: *PeerJ Computer Science* 2, e93.

Aronson, Alan R, Thomas C Rindflesch, and Allen C Browne (1994). "Exploiting a large thesaurus for information retrieval". In: *Intelligent Multimedia Information Retrieval Systems and Management-Volume 1*. Le Centre de Hautes Etudes Internationales D'Informatique Documentaire, pp. 197–216.

Baroni, Marco et al. (2012). "Entailment above the word level in distributional semantics". In: *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 23–32.

Bhogal, Jagdev, Andrew MacFarlane, and Peter Smith (2007). "A review of ontology based query expansion". In: *Information processing & management* 43.4, pp. 866–886.

Bosch, Antal van den et al. (2007). "An efficient memory-based morphosyntactic tagger and parser for Dutch". In: *LOT Occasional Series* 7, pp. 191–206.

Coulthard, Malcolm and Alison Johnson (2009). *An introduction to Forensic Linguistics: Language in Evidence*. Routledge.

Deerwester, Scott et al. (1990). "Indexing by latent semantic analysis". In: *Journal of the American society for information science* 41.6, p. 391.

Dozier, Christopher et al. (2010). "Named Entity Recognition and Resolution in Legal Text". In: *Semantic processing of legal texts: Where the language of law meets the law of language*. Ed. by Enrico Francesconi et al. Berlin: Springer. Chap. 2, pp. 27–43.

Espinosa-Anke, Luis et al. (2016). "Supervised distributional hypernym discovery via domain adaptation". In: *Proceedings of EMNLP*, pp. 424–435.

Francesconi, Enrico et al. (2010). *Semantic processing of legal texts: Where the language of law meets the law of language*. Vol. 6036. Springer.

Frantzi, Katerina T and Sophia Ananiadou (1999). "The C-value/NC-value domain-independent method for multi-word term extraction". In: *Journal of Natural Language Processing* 6.3, pp. 145–179.

Fu, Ruiji et al. (2014). "Learning Semantic Hierarchies via Word Embeddings." In: *ACL (1)*, pp. 1199–1209.

Gelbukh, Alexander et al. (2010). "Automatic term extraction using log-likelihood based comparison with general reference corpus". In: *Natural Language Processing and Information Systems*, pp. 248–255.

Hearst, Marti A (1992). "Automatic acquisition of hyponyms from large text corpora". In: *Proceedings of the 14th conference on Computational linguistics-Volume 2*. Association for Computational Linguistics, pp. 539–545.

Hiemstra, Djoerd, Stephen Robertson, and Hugo Zaragoza (2004). "Parsimonious language models for information retrieval". In: *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pp. 178–185.

Hoge Raad (2012). *Runescape Arrest*. Online. URL: `\url{http://deeplink.rechtspraak.nl/uitspraak?id=ECLI:NL:HR:2012:BQ9251}`.

Jarmasz, Mario (2012). "Roget's thesaurus as a lexical resource for natural language processing". In: *arXiv preprint arXiv:1204.0140*.

Jing, Yufeng and W Bruce Croft (1994). "An association thesaurus for information retrieval". In: *Intelligent Multimedia Information Retrieval Systems and Management-Volume 1*. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, pp. 146–160.

Jungiewicz, Michał and Michał Łopuszyński (2014). "Unsupervised keyword extraction from Polish legal texts". In: *International Conference on Natural Language Processing*. Springer, pp. 65–70.

Kageura, Kyo, Keita Tsuji, and Akiko N Aizawa (2000). "Automatic thesaurus generation through multiple filtering". In: *Proceedings of the 18th conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, pp. 397–403.

Katz, Daniel Martin, Michael J Bommarito II, and Josh Blackman (2017). "A general approach for predicting the behavior of the Supreme Court of the United States". In: *PloS one* 12.4, e0174698.

Knoth, Petr et al. (2009). "Towards a framework for comparing automatic term recognition methods". In: *Conference Znalosti*.

Koehn, Philipp (2005). "Europarl: A parallel corpus for statistical machine translation". In: *MT summit*. Vol. 5, pp. 79–86.

Lenci, A, S Montemagni, and V Pirrelli (2006). "NLP based ontology learning from legal texts". In: *Proceedings of LOAIT '07 II Workshop on Legal Ontologies and Artificial Intelligence Techniques*, pp. 113–129.

Levy, Omer et al. (2015). "Do Supervised Distributional Methods Really Learn Lexical Inference Relations?" In: *HLT-NAACL*, pp. 970–976.

Liu, Yi-Hung and Yen-Liang Chen (2017). "A two-phase sentiment analysis approach for judgement prediction". In: *Journal of Information Science*, p. 0165551517722741.

Luo, Bingfeng et al. (2017). "Learning to Predict Charges for Criminal Cases with Legal Basis". In: *arXiv preprint arXiv:1707.09168*.

Macken, Lieve, Els Lefever, and Veronique Hoste (2013). "TExSIS: Bilingual terminology extraction from parallel corpora using chunk-based alignment". In: *Terminology. International Journal of Theoretical and Applied Issues in Specialized Communication* 19.1, pp. 1–30.

Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig (2013). "Linguistic regularities in continuous space word representations." In: *hlt-Naacl*. Vol. 13, pp. 746–751.

Mikolov, Tomas et al. (2013a). "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*, pp. 3111–3119.

Mikolov, Tomas et al. (2013b). "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781*.

Moens, Marie-Francine (2001). "Innovative techniques for legal text retrieval". In: *Artificial Intelligence and Law* 9.1, pp. 29–57.

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Pennington, Jeffrey, Richard Socher, and Christopher D Manning (2014). "Glove: Global vectors for word representation." In: *EMNLP*. Vol. 14, pp. 1532–1543.

Quaresma, Paulo and Teresa Gonçalves (2010). "Using Linguistic Information and Machine Learning Techniques to Identify Entities from Juridical Documents". In: *Semantic processing of legal texts: Where the language of law meets the law of language*. Ed. by Enrico Francesconi et al. Berlin: Springer. Chap. 3, pp. 44–59.

Rayson, Paul and Roger Garside (2000). "Comparing corpora using frequency profiling". In: *Proceedings of the workshop on Comparing Corpora*. Association for Computational Linguistics, pp. 1–6.

Reynaert, Martin, Camp, and M Van Zaanen (2014). "OpenSoNaR: user-driven development of the SoNaR corpus interfaces". In:

Roller, Stephen, Katrin Erk, and Gemma Boleda (2014). "Inclusive yet Selective: Supervised Distributional Hypernymy Detection." In: *COLING*, pp. 1025–1036.

Salton, Gerard and Christopher Buckley (1988). "Term-weighting approaches in automatic text retrieval". In: *Information processing & management* 24.5, pp. 513–523.

Tomokiyo, Takashi and Matthew Hurst (2003). "A language model approach to keyphrase extraction". In: *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment-Volume 18*. Association for Computational Linguistics, pp. 33–40.

Van Gompel, Maarten and Antal Van Den Bosch (2016). "Efficient n-gram, skipgram and flexgram modelling with Colibri Core". In: *Journal of Open Research Software* 4.1.

Van Opijnen, Marc and Cristiana Santos (2017). "On the concept of relevance in legal information retrieval". In: *Artificial Intelligence and Law* 25.1, pp. 65–87.

Velardi, Paola, Stefano Faralli, and Roberto Navigli (2013). "Ontolearn reloaded: A graph-based algorithm for taxonomy induction". In: *Computational Linguistics* 39.3, pp. 665–707.

Venturi, Giulia (2010). "Legal Language and Legal Knowledge Management Applications". In: *Semantic processing of legal texts: Where the language of law meets the law of language*. Ed. by Enrico Francesconi et al. Berlin: Springer. Chap. 1, pp. 3–26.

Verberne, Suzan et al. (2016). "Evaluation and analysis of term scoring methods for term extraction". In: *Information Retrieval Journal* 19.5, pp. 510–545.

Verheugt, J.W.P. (2013). *Inleiding in het Nederlandse recht, zeventiende druk*. Boom Juridische Uitgevers.

Voorhees, Ellen M (1994). "Query expansion using lexical-semantic relations". In: *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*. Springer-Verlag New York, Inc., pp. 61–69.

Řehůřek, Radim (2015). *Word2vec & friends*. `https://youtu.be/wTp3P2UnTfQ`.

Weeds, Julie et al. (2014). "Learning to distinguish hypernyms and co-hyponyms". In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin City University and Association for Computational Linguistics, pp. 2249–2259.

Wong, Wilson, Wei Liu, and Mohammed Bennamoun (2007). "Determining termhood for learning domain ontologies using domain prevalence and tendency". In: *Proceedings of the sixth Australasian conference on Data mining and analytics-Volume 70*. Australian Computer Society, Inc., pp. 47–54.

Yang, Christopher C and Johnny Luk (2003). "Automatic generation of English/Chinese thesaurus based on a parallel corpus in laws". In: *Journal of the Association for Information Science and Technology* 54.7, pp. 671–682.