Technische Universität München
Department of Informatics

Master's Thesis in Information Systems

Prototypical Implementation and Assessment of Relatedness Search in Laws, Judgments and Commentaries

Philipp Pickel

Technische Universität München
Department of Informatics

Master's Thesis in Information Systems

Prototypical Implementation and Assessment of Relatedness Search in Laws, Judgments and Commentaries

Prototypische Implementierung und Bewertung einer Ähnlichkeitssuche über Gesetze, Urteile und Kommentare

Author:	Philipp Pickel
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Bernhard Waltl
Submission date:	15.12.2016

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Garching b. München, 15.12.2016	__________________________
Place, Date	Signature

Abstract

The number of legal documents is rapidly growing and most of them are available in digital format. These documents are often the basis of a legal expert’s daily work. He often needs to find certain documents, but cannot oversee the huge amount of data without computer aid. The starting point of such a search is frequently an already found document.

Therefore, this thesis provides a prototypical implementation of a relatedness search for legal documents. Beforehand different approaches to identify similarity in texts are discussed by looking at related work and their approaches. These approaches use various techniques from natural language processing and machine learning. They differ in complexity and underlying models.

To ease the effort of implementation, a concept for the architecture of the similarity search is developed. The main goal of the architecture is to allow adding further similarity methods as easily as possible. Also approaches for persistence and visualization are described in this thesis.

Three of the techniques discussed in the literature review are prototypically implemented. Additionally, a similarity search provided by the database elasticsearch is integrated as further point of reference. The described concepts for architecture, persistence and visualization are also realized.

After the implementation is done, the results provided by the different similarity methods are assessed. This is done via a questionnaire, in which the participants rank the search results according their relevance towards a source document. Based on these relevance rankings the performances of the relatedness methods are compared.

In the end a summary of the findings of this thesis is given. Also starting points for further research in this field are shortly mentioned.

Table of Contents

List of Figures

List of Listings

List of Tables

List of Abbreviations

BGB	Bürgerliches Gesetzbuch
CBOW	Continuous Bag-of-Words
DAG	Directed Acyclic Graph
EDA	Exploratory Data Analysis
LDA	Latent Dirichlet Allocation
NER	Named Entity Recognition
NLP	Natural Language Processing
NP	Noun Phrase
POS	Part-Of-Speech
Ruta	Rule-based Text Annotation
sebis	Software Engineering for Business Information Systems
StGB	Strafgesetzbuch
UIMA	Unstructured Information Management Architecture

1 Introduction

1.1 Motivation

In Germany about 1.6 million civil cases are heard every year [1]. The lawyers of accuser and accused try to add weight to their perspective of things with sound arguments. In the Anglo-American region most of these arguments refer to past cases, since there is a case law in place. The legal system in Continental Europe is not based on case law, but rather on the interpretation of the applicable laws. Nevertheless, the influence of past judgments on the legal practice in Continental Europe and therefore also in Germany must not be neglected.

Especially judgments issued by High Courts, such as the Bundesgerichtshof in Germany, often have an announcement effect. The Bundesgerichtshof sometimes adds its decision an interpretation of the law. Such fundamental judgments can become common law, and therefore have a huge impact on future decisions. However, also other past cases can be used as guidance by lawyers.

Finding such similar cases and other related legal documents is consequently an important task in a legal expert's daily work. The number of available documents is enormous and a considerable part of them can already be retrieved in a digital format. The already mentioned Bundesgerichtshof, for example, publishes every month about 300 judgments on its website[1]. In addition to the websites of single courts, there are databases that aggregate most of the legal documents. Regarding German legal documents, the most important databases are juris and beck-online. Juris, which is majoritarian owned by the Federal Republic of Germany, grants access to more than 2.5 million documents from the legal domain [2]. The database of beck-online, which is part of the publishing company C.H. Beck, contains over four million legal documents [3].

This enormous amount of available documents indicates that a legal expert cannot possibly be able to search them all without computer aid. Therefore, the goal of this thesis is to find an approach to help finding related legal documents given a document as a starting point of the search. This will be referred to as relatedness or similarity search throughout this thesis, while the two terms are seen as interchangeable.

Legal documents are a combination of structured and unstructured data. The structured part is the often similar format. A judgment, for example, consists in most cases of the tenor, the facts describing the case, information about courts of lower instances, guiding principles and the reasons for the decision. The unstructured part is the text itself. This needs to be prepared, so that a computer system can work with it. Approaches from machine learning and natural language processing (NLP) can and will be used in this thesis to address this problem. The use of these methods is especially promising because legal documents have only few grammatical or spelling mistakes in general.

[1] www.bundesgerichtshof.de

There is a multitude of possible useful techniques available to estimate the relatedness of documents. Therefore, this thesis will give an overview over promising approaches with different technical background. Three of these methods will be then prototypically implemented and their results will be assessed. The assessment will be conducted using a questionnaire. Therein the participants rank the results provided by the similarity methods according their relevance towards a source document. This hopefully helps to understand what makes a good similarity search regarding legal documents.

The similarity methods will be embedded in some kind of recommender system that provides a suitable architecture for the methods. Also things such as persistence and visualization are considered in the thesis.

1.2 LEXIA

This thesis will not be greenfield development but will be further development of the LEXIA system. As stated by Waltl et al., LEXIA is a “Data Science Environment for Semantic Analysis of German Legal Texts” [4]. It has been developed by the chair of *Software Engineering for Business Information Systems*[2] (sebis) at the TU München as part of the interdisciplinary research program *Lexalyze*[3]. This project wants to make use of synergies between the legal domain and computer science.

LEXIA already provides some capabilities, from which this thesis will make use throughout its progress. To get a first overview, the different components and subcomponents of LEXIA are shown in Figure 1. The interactions of these components are denoted by arrows connecting them. The illustration presents the state of LEXIA at the beginning of this thesis. A short

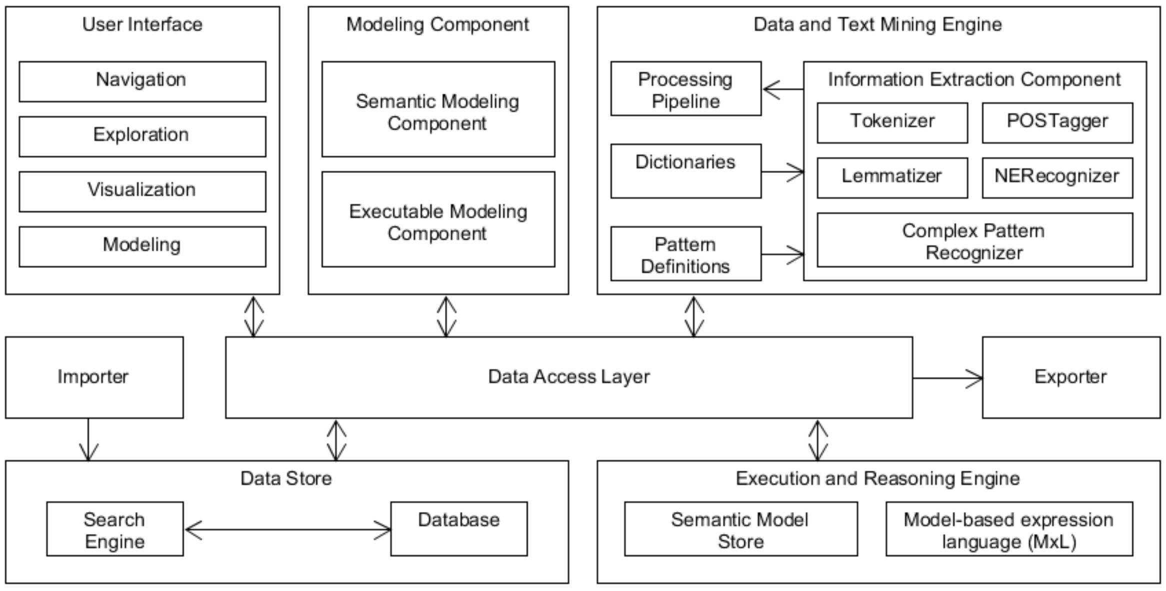


Figure 1: LEXIA components
Source: Waltl et al. [4]

description of the most important components regarding this thesis will be given in the following.

With the help of the *Importer*, legal documents in different file formats such as PDF or XML can be loaded into the system. This is done by different importer classes tailored to the file format and the structure of the text, which differs greatly between judgments, laws and contracts. The imported data are then stored in an elasticsearch database. This schema-free full-text search database already provides a search engine. Schema-free means that the schema of the data does not need to be defined a priori. The schema is derived from the objects that are

[2] wwwmatthes.in.tum.de
[3] www.lexalyze.de

indexed to the database. This allows more flexibility on changing data models in the application.

The capabilities of the database are capsulated in the *Data Store* component. Between the *Data Store* and other components there is the *Data Access Layer*. Its purpose is to transform the query results of the database into java objects the other components can work with.

The data model for the imported legal documents can be seen in Figure 2. The different types of legal documents such as *Law*, *Judgment* and *Contract* are all derived from one abstract class *LegalDocument*. These *LegalDocuments* consist of *LegalDocumentContents* which can be *SectionContainers* and *Sections*. The *SectionContainer* and *Section* represent the inherent structure of legal documents, which was mentioned in chapter 1.1. For example, a *Section* can be an article of a law or the reasoning in a judgment. For each *Section* there can be *Annotations*. These *Annotations* are the result of NLP, which can be performed by the component *Data and Text Mining Engine*.

The *Data and Text Mining Engine* is one of the central components of LEXIA. It is based on the Apache UIMA (Unstructured Information Management Architecture) framework. This framework provides a reference implementation for processing unstructured data [5]. LEXIA uses the architecture for NLP tasks. These tasks use a pipeline model by which it is possible to easily add or remove steps from the NLP task. As indicated in the graphic, possible steps are, for example, the use of a *Tokenizer*, *POSTagger*, *Lemmatizer* or *NERecognizer*. Additionally, the *Data and Text Mining Engine* makes use of the Apache Ruta (Rule-based Text Annotation). This part of the UIMA enables the LEXIA component to find complex semantic patterns [6]. These patterns are defined in so-called Ruta scripts which can be added to the previously mentioned pipelines. The prototypical implementation described within this thesis will repeatedly make use of the *Data and Text Mining Engine*.

Since this system should not only be used by computer scientists but rather also by experts from the legal domain, LEXIA provides a graphical user interface. This user interface com-

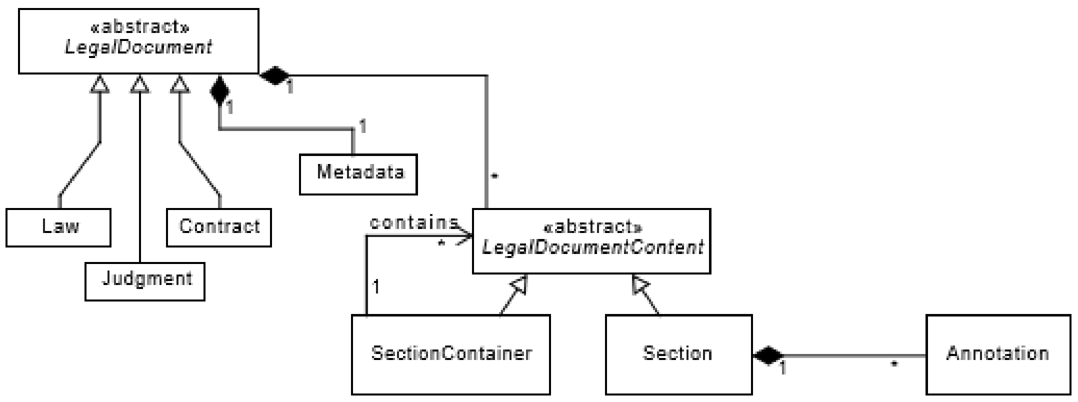
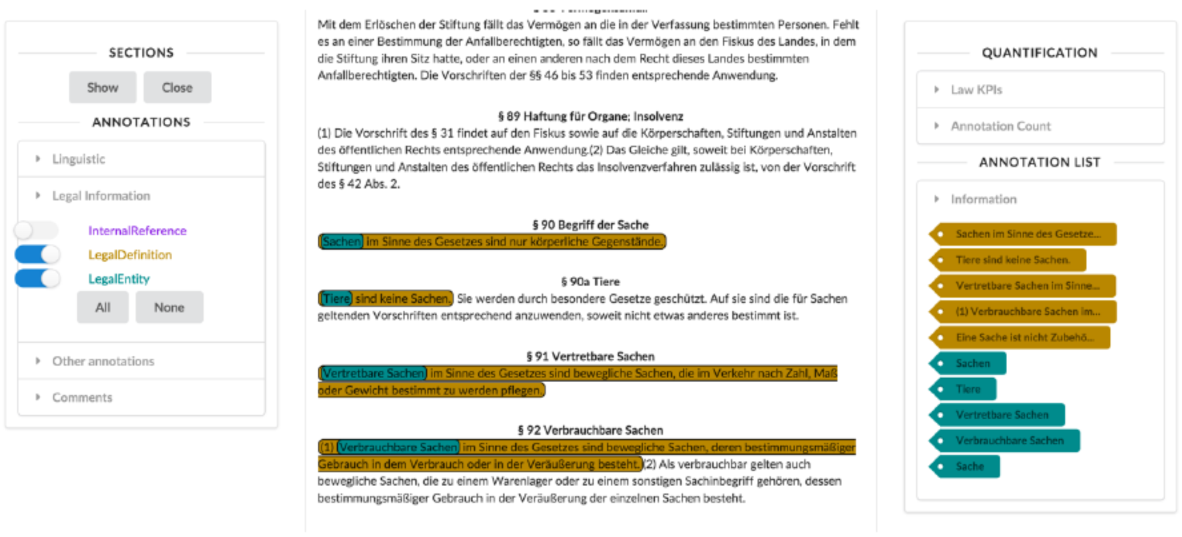


Figure 2: LEXIA data model
Source: Based on [4]

prises views to import legal data, start NLP processes or define Ruta scripts, but also the results of these processes are visualized by the user interface. For example, a user has the possibility to look at the imported legal documents and have the annotations, resulting from the NLP, highlighted. An example of this is shown in Figure 3. Since this thesis will extend the possibilities to process the legal documents, the prototypical implementation will also include an expansion of the graphical user interface.

The components which have not been mentioned so far, i.e. the *Modelling Component*, the *Execution and Reasoning Engine* and the *Exporter*, are of no or only little interest to this thesis. Therefore, they will not be described in detail.



Mit dem Erlöschen der Stiftung fällt das Vermögen an die in der Verfassung bestimmten Personen. Fehlt es an einer Bestimmung der Anfallberechtigten, so fällt das Vermögen an den Fiskus des Landes, in dem die Stiftung ihren Sitz hatte, oder an einen anderen nach dem Recht dieses Landes bestimmten Anfallberechtigten. Die Vorschriften der §§ 46 bis 53 finden entsprechende Anwendung.

§ 89 Haftung für Organe; Insolvenz

(1) Die Vorschrift des § 31 findet auf den Fiskus sowie auf die Körperschaften, Stiftungen und Anstalten des öffentlichen Rechts entsprechende Anwendung.(2) Das Gleiche gilt, soweit bei Körperschaften, Stiftungen und Anstalten des öffentlichen Rechts das Insolvenzverfahren zulässig ist, von der Vorschrift des § 42 Abs. 2.

§ 90 Begriff der Sache

Sachen im Sinne des Gesetzes sind nur körperliche Gegenstände.

§ 90a Tiere

Tiere sind keine Sachen. Sie werden durch besondere Gesetze geschützt. Auf sie sind die für Sachen geltenden Vorschriften entsprechend anzuwenden, soweit nicht etwas anderes bestimmt ist.

§ 91 Vertretbare Sachen

Vertretbare Sachen im Sinne des Gesetzes sind bewegliche Sachen, die im Verkehr nach Zahl, Maß oder Gewicht bestimmt zu werden pflegen.

§ 92 Verbrauchbare Sachen

(1) Verbrauchbare Sachen im Sinne des Gesetzes sind bewegliche Sachen, deren bestimmungsmäßiger Gebrauch in dem Verbrauch oder in der Veräußerung besteht.(2) Als verbrauchbar gelten auch bewegliche Sachen, die zu einem Warenlager oder zu einem sonstigen Sachinbegriff gehören, dessen bestimmungsmäßiger Gebrauch in der Veräußerung der einzelnen Sachen besteht.

Figure 3: LEXIA user interface
Source: Waltl et al. [4]

2 Research Objectives

The introduction chapter gave some insights on the reasons why research is needed in this field and in what environment this thesis will take place. The reminder of this chapter shortly describes the four research questions on which this thesis will orient itself.

What approaches can be used to measure similarity between two legal documents or parts of it?

The processing of unstructured information made great progress in the last decade. NLP and machine learning are to be named in this context. Several approaches have been used to identify similarity of texts. Chapter 5 introduces seven of them. Therefore, this chapter will describe how the basic approaches work based on related literature. Special attention will be given to papers already dealing with legal documents. This will then be concluded by a comparison of the different methods.

What is a suitable approach to provide such similarity methods?

It is important for every application to have a consistent architecture. This ensures easier maintainability and expandability. Best practices and proven concepts help to achieve a good fundamental structure of a recommender system enclosing the similarity methods. The concept of the architecture and further design decisions described in chapter 6.1 try to meet this goal and answer the above question. While the concept is more universally applicable, chapter 7.1 explains the concrete prototypical implementation added to LEXIA.

How can the results of a similarity search be presented to a user?

The user interface is one of the most crucial points of an application. It often plays an important role in the user's decision to work with an application or not. Therefore, there is a need to not only provide the right information but also to present it in an easily comprehensible format. Chapter 6.1.3 discusses two possible approaches with slightly different use cases. The implementation of them is then described in chapter 7.4 with the aid of screenshots.

How well do selected similarity methods perform?

The results of the implemented relatedness methods need to be assessed. This can give an insight into which method delivers the best results regarding legal documents. The assessment can also be used as a starting point for further optimization of the methods or as comparable figure. The process of the assessment is described in chapter 8.2.

3 Research Method

This thesis explores the usage of NLP and machine learning to get insights into the similarity of legal documents. To accomplish this goal some techniques of exploratory research will be used. “Exploratory research is a first step, conducted with the expectation that additional research will be needed to provide more conclusive evidence” [7]. So, regarding LEXIA, this thesis is a starting point to provide a relatedness search.

The data from the legal field is quite broad. An example is the test data for this thesis described in chapter 4. Data itself can be the starting point for a hypothesis. This approach is named exploratory data analysis or short EDA. The term was primarily coined by John Tukey. In his book he claims that statistical hypothesis testing is emphasized too much in the field of statistics [8]. He wanted data to be used not only to confirm existing hypothesis but also to derive new hypothesis. So EDA “is used to identify systematic relations between variables when there are no (or not complete) a priori expectations as to the nature of those relations” [9].

The variables in this thesis are unstructured data in the form of texts from the legal domain, which are then prepared by NLP and machine learning. The relation to be found is which similarities in the prepared data suggest similarity of the whole texts. Just by reading the texts one can make assumptions about what makes them similar or not. Examples of what could indicate similarity are the use of the same words or the use of similar syntactical constructions. Also the position of a word seems to have an impact on the similarity.

To have these ideas in a more formalized way, they are embedded in a literature review. Since others have already researched the field of text similarity, their findings and suggested methods are an important ingredient to this thesis. The key outcomes of the literature review are an overview over several similarity methods in general, insights into methods that have already been used in the legal domain and a comparison of the methods.

So, as indicated in Figure 4, the associations seen in the data, the approaches of NLP and the literature review lead to a set of similarity methods. The descriptions of them are in turn the starting point for the further research in this thesis.

As also shown in the graphic, the implementation of some of these methods is the next step. Therefore, first a concept for the architecture must be created. This concept has to allow an easy integration and adaption of further methods as one central design issue.

As a last step, the implemented methods are assessed, which is also shown in the illustration of Figure 4. While the literature review to derive suitable similarity methods uses a more qualitative approach, the assessment of the implemented methods happens in a more quantitative process. Each result of each method gets evaluated to receive an estimate of the quality for each suggestion.

This might lead to the result that one method always provides better suggestions than the others. Then one could conclude that this is probably the best method for a similarity search. But since this is exploratory research, such a finding would need to be evaluated in further research.

Even if none of the implemented methods can constantly perform better than the others the research done would still be valuable as a starting point for other investigations. The data of the assessment might be useful for improving the implemented methods. They could also be used as comparable figure for further methods.

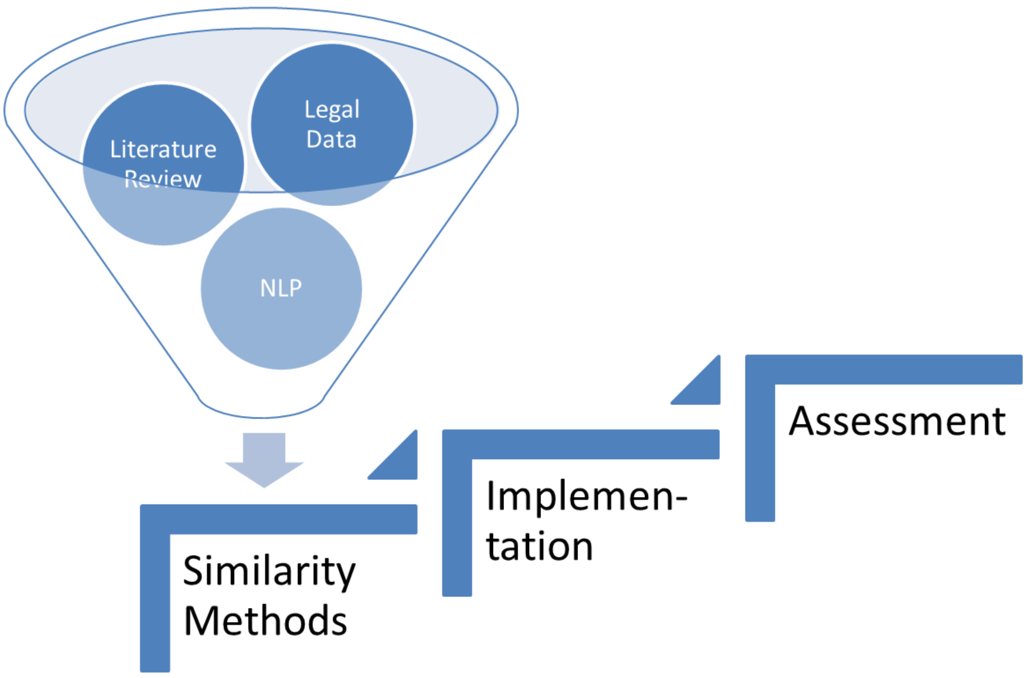


Figure 4: Research Method
Source: Own illustration

4 Data

As written in the motivation (see chapter 1.1), there is a huge amount of legal data in digital format. For this thesis, only a subset of these data can be used as sample data. The documents used originate from different sources. These sources are the already mentioned beck-online (see chapter 1.1) and the DATEV, which is a German IT service provider for tax consultants and lawyers dealing with tax law [10]. Additionally, all German federal laws are available.

It is important to get a first insight into the data corpus to get an understanding of what kind of documents one is dealing with. Also, this can give first hints what makes a document similar to another one.

4.1 Beck-online dataset

The documents of this dataset are an export from two searches in the beck-online database. For the first search the preferences were set to judgments referring to the stock corporation law, which were pronounced by the Bundesgerichtshof. The judgments for the second search refer to tenancy law and were also issued by the Bundesgerichtshof. The exports are given in HTML format.

The dataset consists of more than 900 judgments regarding stock corporation law and over 700 judgments referring to tenancy law. Therefore, the dataset is quite small, but since these judgments were issued by the highest civil court in Germany, they have a high relevance for these two fields of law.

The creation dates of the documents span a time between the years 1951 and 2015. About three quarters of the judgments in this dataset were issued after the year 2000 and therefore represent the latest interpretation of the norms, but also the changes over time can be traced with this distribution.

A problem of this dataset is the inconsistent use of HTML, which makes it hard to parse the documents. Thus, it takes quite some effort to get these data into a format one can work with in further steps.

Additionally, some of the documents represent only parts of the judgments. In some cases, for example, only the facts or extracts of the reasons for the decision are available. This needs to be kept in mind when working with this dataset.

Despite the problems with inconsistency and fragmentariness, the dataset is useful as sample data for this thesis. It provides documents of high relevance over a long period of time. Furthermore, dealing with such problems is close to the real use case.

4.2 DATEV dataset

This dataset is provided by the DATEV and is a mix of different kinds of legal documents. The set consists of more than 132,000 documents, which are saved in an easy to parse XML format. In the XML there are some metadata besides the text itself. These metadata include, among other things, the document type, the creation date and the institution this document was created by. As this is a huge amount of data, it is worth taking a deeper look into the provided metadata.

As seen in Figure 5, the majority with 47,359 documents are judgments which are of special interest for this thesis. Also present in high quantity are essays (28,966), notes (15,506) decrees (7,509), resolutions (7,336) and short articles (6,203). The remaining 19,802 documents are divided into 45 further document types. This shows the great variety in document types which needs to be considered when working with this dataset because similarity between different types might be harder to measure.

Another important piece of information about one document is the date it was issued. Since laws and also the interpretation of one law changes over time, the more recent documents might have a higher relevance than older ones. Figure 6 shows this distribution over time. Not surprisingly, the number of documents created each year rises until the year 2000. Since the year 2000 about 4000 documents have been added to the dataset each year. This corresponds to the fact that documents were consistently saved in digital format starting in the late 90s. The drop for the year 2016 results from the circumstance that the year was not over when the data was provided for this thesis.

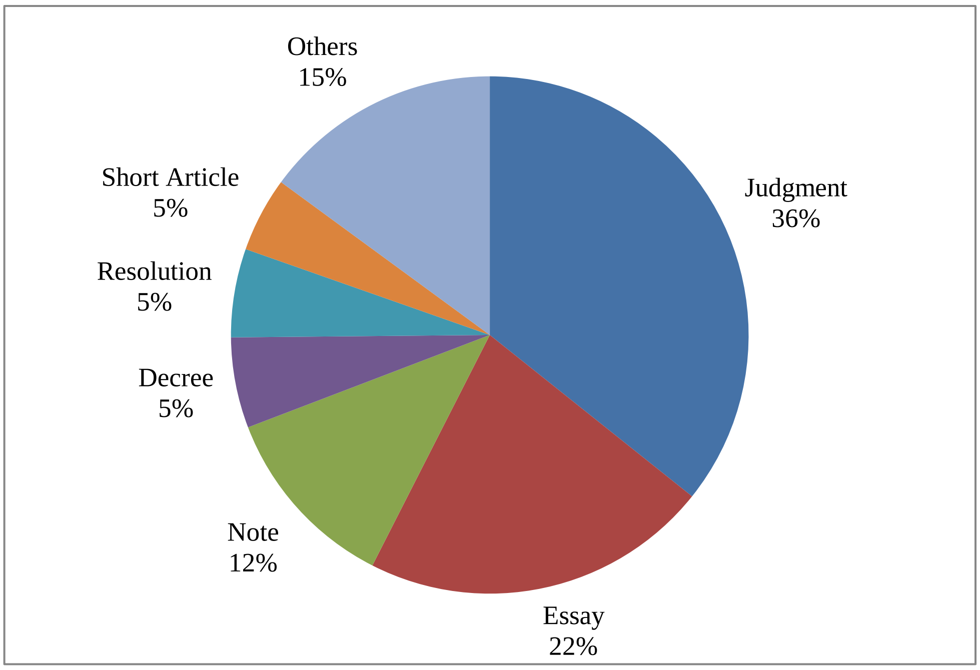


Figure 5: Document types of DATEV dataset
Source: Own illustration

As already mentioned the judgments from this dataset are of special interest. Therefore, Figure 7 shows which institutions issued the judgments. Most of them (17,323) were pronounced by the highest court for tax law in Germany, the Bundesfinanzhof. The high courts for tax law of states like Niedersachsen or Baden-Württemberg are also often present as well as courts for tax law in big cities such as Munich. The remaining 10,771 judgments show a broad variety in the institution they were issued by. They spread themselves over 161 further institutions. Therefore, all levels of courts regarding tax law are covered by this dataset. However, the main focus is still on the judgments pronounced by high courts, which provide high relevance and are used as guidance for lower legal authorities.

A look on which norms the legal documents refer to shows the expectable results. Since the DATEV is specialized in tax law, most norms mentioned come from the *Einkommenssteuergesetz*, the German income tax act. But also the *Grundgesetz* (constitution), *Aktiengesetz* (stock corporation act), *Handelsgesetzbuch* (commercial code) or *Abgabenordnung* (fiscal code) are repeatedly referred to in the dataset.

All in all, the dataset provides decent sample data for this thesis because of its variety in document types, institutions and dates the documents were issued. Additionally, the XML format allows an easy handling of the data. The only small point of criticism is the concentration on only one field of law, but as this thesis only wants to provide a prototypically implementation, this does not seem to be a relevant problem.

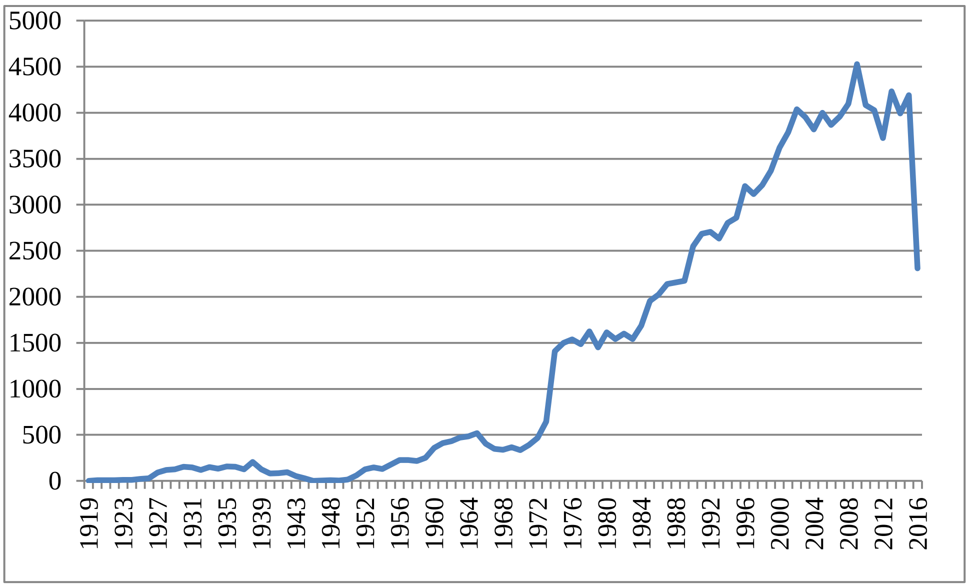


Figure 6 Document distribution over time of DATEV dataset
Source: Own illustration

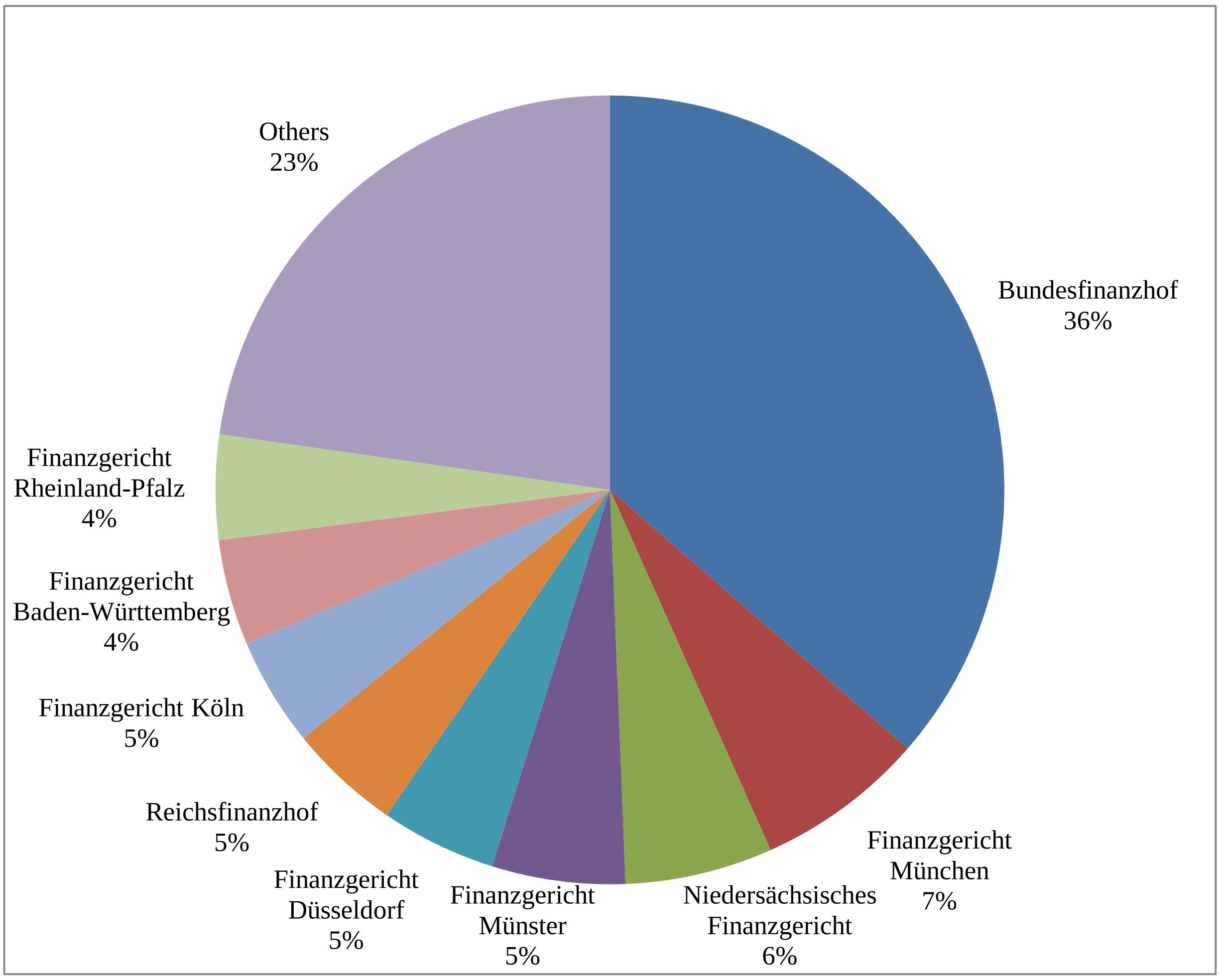


Figure 7: Institution that issued judgments in DATEV dataset
Source: Own illustration

4.3 German Federal Laws

There are more than 1,800 German federal laws[4], all of which can be found on the internet. As an example, the website gesetze-im-internet.de, which is associated with already mentioned juris (see chapter 1.1), provides most of them in an easy to parse XML format.

This dataset only consists of laws, while the two previously described mainly contained judgments. These two document types differ fundamentally. Laws have much shorter passages of continuous text, since they are much more divided into paragraphs and subparagraphs. This leads to only 50 to 60 words per paragraph on average.

The number of paragraphs can be very different from law to law. The *Hopfengesetz* for example consists of only six paragraphs, while the *Bürgerliche Gesetzbuch* (BGB) contains 2402 legal norms. In such huge laws many different fields of law are touched. So many paragraphs are not related to each other. This is the reason why a similarity search to a whole law does not make sense in most cases.

Although some sections of one law are not related to each other, there are often explicit references. These explicit references indicate a connection between two paragraphs, which leads to the assumption that they are similar in some ways.

All in all the good availability and data format make German federal laws an excellent testing data set for this thesis. Also, some of these laws are well known to every legal expert, whereby an assessment of a similarity search is made much easier.

[4] Search on juris.de for document type "Gesetz", federal and title documents only

5 Related Work

After the look into the test data in the previous chapter, which hinted some possible factors indicating similarity, this chapter deals with the already existing literature regarding text similarity. Seven different approaches are discussed. In the process each approach is described and some of its usage in earlier research is highlighted. A special attention is paid to usage in the legal domain.

5.1 Bag of Words

The consideration that the occurrence of the same words in different texts can indicate similarity of those texts is quite obvious. The approach to count the occurrences of words is called bag of words. The term can be traced back to Zellig Harris [11] who used it in a linguistic context.

The basic procedure is as follows: all words of the two texts that want to be compared are put into one set. According to this set and the frequency a word occurs in each text, the texts are transformed to vectors. To make this approach a similarity measure one can calculate the similarity of the vectors. The similarity of this approach is symmetric.

For an example the two texts in Listing 1 are used.

```
(1) John likes to play football. Mary likes to watch movies.
(2) John likes to watch football games on TV.
```

Listing 1: Example texts
Source: Own illustration

These two texts lead to the following set of words shown in Listing 2.

```
[
        “John”,
        “likes”,
        “to”,
        “play”,
        “football”,
        “Mary”,
        “watch”,
        “movies”,
        “games”,
        “on”,
        “TV”
]
```

Listing 2: Word set
Source: Own illustration

The vector representation of the example texts is shown in Listing 3. The position in the vector indicates which word was counted by referring to the position in the set shown in Listing 2. So, for example, the first position in the vector represents how often the word “John” occurs in each text.

```
(1) [1, 2, 2, 1, 1, 1, 1, 1, 0, 0, 0]
(2) [1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1]
```

Listing 3: Word vectors
Source: Own illustration

The similarity of these two vectors can be calculated using the cosine similarity but also other similarity measures for vectors would be applicable. The formula for the cosine similarity *cos(θ)* between two vectors *A* and *B* is defined as follows:

$$cos(\theta) = \frac{A * B}{\|A\| * \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} * \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Using this formula the similarity value of the example texts would be 0.66.

The representation of texts as vectors is commonly known as vector space model (VSM). The VSM using a bag of words approach was strongly influenced by Salton & McGill [12]. They also suggested using the cosine similarity of the vectors to determine the overall similarity of the texts.

A visualization for such a VSM is shown in Figure 8. Each document D_1 to D_3 is described by

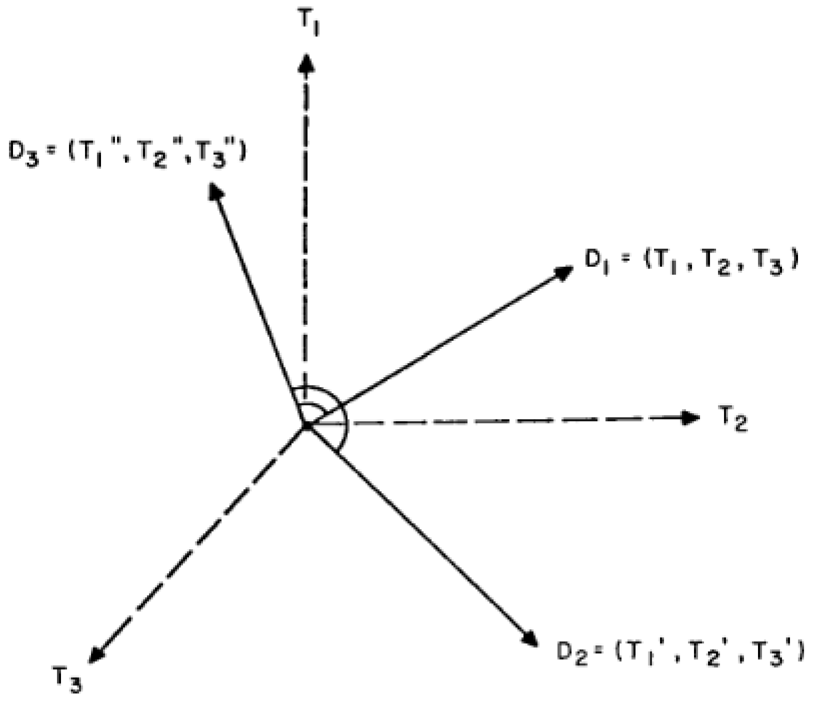


Figure 8: Example of a VSM
Source: Salton, Wong, Yang [44]

its terms T_1 to T_3. The angles between these vectors indicate the similarity of the documents shown.

The bag of words approach is often used as basis for further processing in classification and information retrieval applications. So, a lot of extensions to this approach exist.

The most common extensions include stopword lists, stemming and lemmatization. A stopword list usually contains words that are very common in a language but do not transfer meaning on their own. For English, such a list would for example contain words like “and”, “the” and “a”. The words from the stopword list then are not considered for the set of words and therefore are not part of the vectors representing the text.

Stemming and lemmatization pursue a similar goal. They want to eliminate different forms of the same word. An example for different forms of the same word would be “go”, “goes”, “went” and “going”. In a naïve bag of words approach these would be treated as four different words, even though they have almost the same meaning.

Stemming tries in a heuristic process to reduce the different forms to one base form by mostly chopping off suffixes. These heuristic approaches often have problems with irregular word forms like “went”. A well-known algorithm for stemming was introduced by Porter [13].

Lemmatization has a cleaner approach. It tries to return the base dictionary form, which is called lemma. This is done by the use of a vocabulary and a morphological analysis of the words. By this technique also “went” should be detected as form of “go”. But this is a more expensive process regarding performance than stemming [14].

One interesting further extension was introduced by Corley & Mihalcea [15]. They did not want to rely only on identical words but also make use of semantic similar words. Therefore, they first identify similar words in the texts they want to compare. The word similarity is expressed by a score. For example the words “fill” and “complete” achieve a score of 0.86. This example is taken from their paper. Identical words would have a score of 1.0. These word similarity scores then are used to compute the overall similarity of the two texts.

In the research regarding texts from the legal domain, bag of words is repeatedly used to represent these texts as vectors. Based on such vectors, a clustering for huge data corpora was done [16]. The researchers could successfully visualize the clusters and concluded that their “system is very helpful for data analysis offering quick insight into the structure of the visualized corpus” [16].

Clustering and similarity search using a bag of words approach are very alike. While the similarity search determines a similarity value based on the angle between two vectors, the clustering uses all vectors to find similar ones to put them in one cluster. A clustering based on vectors can be seen in Figure 9. An “x” represents a text vector, while the dots indicate a cluster centroid. The vectors in one cluster have a small angle between them and therefore have a high cosine similarity.

All in all, the bag of words approach is widely used to represent texts. But the set approach and the vectorization lead to one of the methods biggest shortcomings: the word order is lost. This valuable information also cannot be conserved by any extension of this approach. The consideration of the information provided by the word order can only be done by other techniques which will be discussed later.

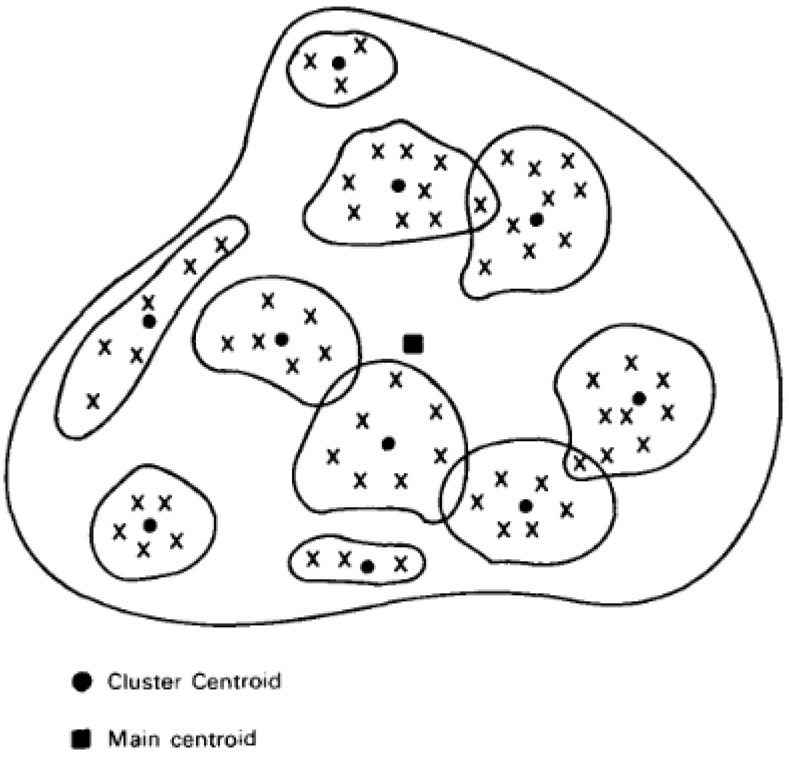


Figure 9: Clustering based on vectors
Source: Salton, Wong, Yang [44]

5.2 N-gram

The approaches of n-grams and bag of words to determine text similarity have many parallels. However, by using n-grams the loss of word order, which is a big shortcoming of bag of words, can be partly overcome.

In general the procedure is to divide texts into fragments. The sequence of *n* consecutive fragments then is called n-gram. A fragment can be a letter, a word, a morpheme, a syllable or something similar [14].

After the division into fragments the similarity of the texts can be expressed by the Dice coefficient *d*, for example. For two texts *a* and *b* the formula is the following:

$$d(a, b) = \frac{2 * |T(a) \cap T(b)|}{|T(a)| + |T(b)|}$$

while *T(x)* is the set of n-grams contained in the text *x* [17].

Another regularly used similarity measure is the Jaccard index *J*, which has the following formula [18]:

$$J(a, b) = \frac{|T(a) \cap T(b)|}{|T(a) \cup T(b)|}$$

In the following example a word is considered a fragment and the value of *n* is set to three. An n-gram with the value of three for *n* is often also referred to as “trigram”. Again both texts from Listing 1 (see chapter 5.1) are used for the following example.

By splitting the texts the trigrams shown in Listing 4 arise. The “*” indicates an empty space in the example.

```
T(1) = {
        “* * John”, “* John likes”, “John likes to”, “likes to play”, “to play football”,
        “play football Mary”, “football Mary likes”, “Mary likes to”, “likes to watch”,
        “to watch movies”, “watch movies *”, “movies * *”
        }

T(2) = {
        “* * John”, “* John likes”, “John likes to”, “likes to watch”, “to watch football”,
        “watch football games”, “football games on”, “games on TV”, “on TV *”, “TV * *”
        }

T(1) ∩ T(2) = {
                “* * John”, “* John likes”, “John likes to”, “likes to watch”
                }
```

Listing 4: Trigrams
Source:Own illustration

According to the Dice coefficient the similarity of these two texts is 0.36. Of course, this value cannot be compared to the similarity value calculated in chapter 5.1 but only to other Dice coefficients. The similarity value also differs greatly depending on the size of *n*. For *n* = 1, which is almost identical to bag of words, the Dice coefficient would have a value of 0.56. The Jaccard index for this example has a value of 0.22.

As long as a word is considered as a fragment one might have problems with the different forms of one word. Since this problem is identical to the one discussed for bag of words in the previous chapter, also the solutions are the same. Therefore, stemming and lemmatization are also often utilized when using n-grams.

The problem of different forms of one word has a much smaller impact when letters are considered as fragments. That is the case because the division of words into short sequences of letters performs implicitly some kind of stemming. Therefore, this approach works throughout most languages without adaption [14].

Cavner & Trenkle also use n-grams of letters to perform a text categorization [19]. To achieve this they do not put the number of n-grams contained in both texts into relation to the total number of n-grams in the text like in the example before, but rather use the n-gram frequency distribution of the texts. The approach is based on the idea that if one is “comparing documents from the same category they should have similar N-gram frequency distributions” [19]. To get the frequency distribution they compute all n-grams that occur in a text and count the frequency of each n-gram. Then the n-grams are ranked from the most to the least frequent one. As distant measure an *out-of-place* value is calculated. For each n-gram the rank in one text is subtracted from the rank in the other text. The absolute values of these differences are then summed up for the *out-of-place* value.

With some adaptions this idea could also be used as the basis of a similarity measure for two texts. One adaption would be a standardization of this *out-of-place* measure. Also, the value for *n* would need good consideration.

5.3 NLP Annotations

In contrast to the approaches of bag of words and n-grams described in the previous chapters, with NLP annotations one can make use of the texts' semantics to compare them. Therefore, the text is not only tokenized, but part-of-speech (POS) tagging, named entity recognition (NER) and pattern finding are, for example, also used.

POS tagging allows assigning a particular part of speech to each word in a text, as the name already suggests. POS tagging is mostly used as a basis to find more complex syntactical constructions. These more complex syntactical constructions then often give also insights into the semantics of the text.

Lame uses NLP to find legal terms in the French Codes [20]. She defines legal terms "as terms labeling world objects apprehended by law and artifacts created by law" [20]. These terms are found by using syntactical analysis. This approach is paired with statistical analysis to make a step towards a legal ontology relying on text-based NLP.

There are also examples that use NER to understand documents from the legal domain. One of them is introduced by Dozier et al. [21]. NER tries to recognize named entities like persons, places, companies, etc. and classify them to the according semantic type. Dozier et al. apply their implementation of NER to judgments from the American case law. They are able to extract amongst others the names of judges, attorneys and courts. In a second step they link these found entities to each other, which they describe as named entity resolution. The link between two entities is found by the co-occurrence in one document. Named entity resolution can help to understand how the found entities interact with each other.

NLP provides a multitude of ways to get a better understanding of texts. Often various concepts are combined to create a new application. Schweighofer for example envisions a "dynamic electronic legal commentary" [22] based on NLP. This should be achieved with the help of powerful world and legal ontologies derived by NLP. Also the linking between legal documents should be executed by computer systems and periodically updated.

While the vision of Schweighofer seems to be bit more far off, NLP already provides today a lot of useful tools. These tools can also be used as the basis of a similarity search. By using NLP it is possible to find occurrences of syntactical constructions like the previously mentioned legal terms across multiple texts. In a second step the frequency of occurrence can be transformed into a VSM. Based on the similarity of these vectors representing the texts the relatedness of the texts can be calculated. The possibilities of which syntactical construction underlies such a VSM are almost endless and also multiple constructions can be considered for the text similarity. These different constructions can even be weighted differently to get a widely adjustable method that determines the similarity of two texts.

For each syntactical construction that is used the creator has a certain semantic in mind that is expressed by this construction. Therefore, this method implicitly takes semantics into account for the comparison.

Also the approach of NER and named entity resolution can be turned into a similarity measure. The resolution produces a graph showing the interactions of different entities. The edges of this graph can be weighted according the frequency of co-occurrence of two entities. Such a graph would additionally allow considering transitive relations of two entities for the similarity score.

All in all, NLP annotations provide a wide range of possibilities to get insights into texts, which then can be transferred into an estimation of the similarity of texts. To obtain such a similarity value suitable syntactical patterns can be used, as well as the approaches of NER.

5.4 Word Embeddings

The basic idea of word embeddings is to represent a word as a vector of real numbers. The vectors are calculated based on the idea that similar words are used in similar contexts. For the vector calculation a training dataset is needed. The approach of word embeddings must not be confused with the approach described in chapter 5.1, where a whole text is expressed as a vector of natural numbers.

The number of dimensions of the vectors plays an important role for the quality and the computing time. The higher the number of dimensions, the higher the quality but also the processing time. The increase in quality becomes more marginal with every dimension added. Typically, a number of dimensions between 100 and 1000 is used.

The calculation of the word vectors relies on a neural network architecture. While neural natural languages models are a quite old idea that dates back to the year 1986 [23], word embeddings gained new traction in recent years because of more efficient algorithms. Important work in this field was done by Bengio et al. [24] and even more influential was the work of Mikolov et al. [25].

Mikolov et al. introduced two architectures for the training phase. One is the continuous bag-of-words (CBOW) model and the other one is the continuous skip-gram model. Using CBOW the current word is predicted from a window of surrounding context words. The window size can be chosen arbitrarily. This architecture is shown in Figure 10 on the left side. On the right side of the graphic skip-gram is visualized. Using this architecture the model predicts the sur-

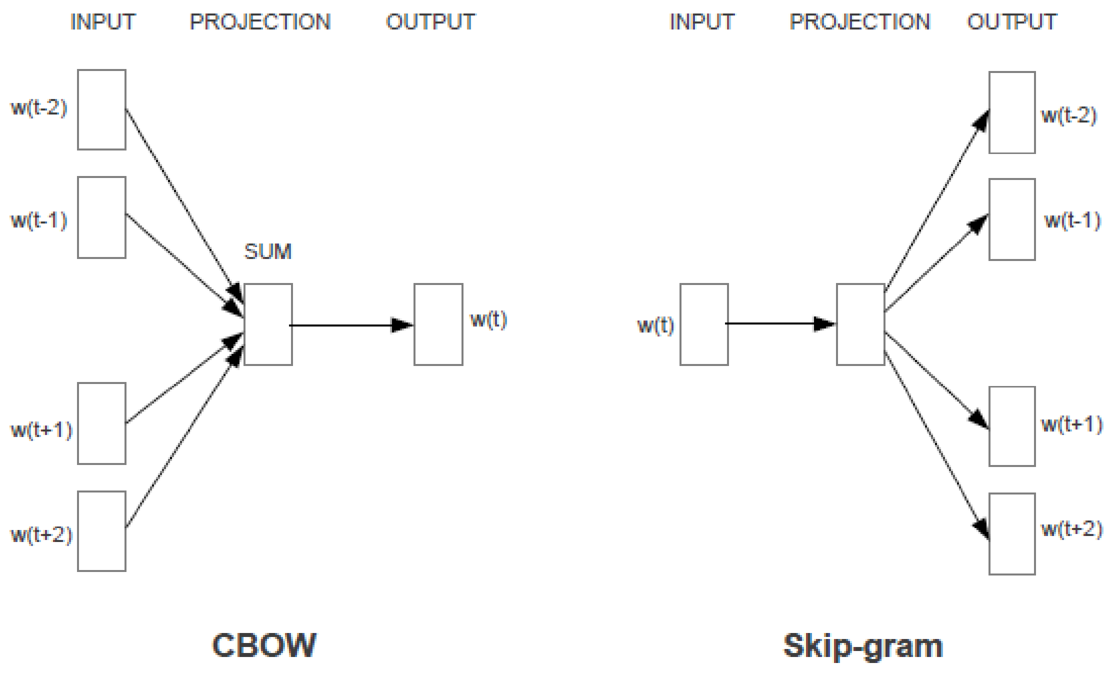


Figure 10: CBOW and Skip-gram
Source: Mikolov et al. [25]

rounding window of context words based on the current word.

In a further paper Mikolov et al. proofed that the skip-gram model outperforms earlier approaches [26]. The training dataset contained about 30 billion words and the dimensionality was set to 1000. Using the skip-gram model the training phase was completed in a shorter amount of time and the vectors provided better results than all other tested approaches.

The implementation of the algorithms suggested by Mikolov et al. is commonly referred to as *word2vec*. They made their code available as an open source project[5] for further research in this field.

In the word vectors computed by the before mentioned techniques the meanings of these words are somehow encoded. Therefore, similar words have similar vectors. So the similarity of words can be expressed by the cosine similarity of the vectors, which was already mentioned in chapter 5.1. But the representation as mathematical vector even allows using basic algebraic operations like addition and subtraction.

One can for example test that *vector*(”King”) - *vector*(”Man”) + *vector*(”Woman”) results in a vector close to the representation of “Queen” [25]. Also the term *vector*(“Prince”) – *vector*(“Man”) + *vector*(“Woman”) results in a vector near the vector of the expected word “Princess”. A simplified illustration of this phenomenon can be seen in Figure 11.

So word embeddings provide the ability to find similar words based on their vectors and to perform algebraic operations on them. But this thesis does not want to compare only words or find relationships between them but rather find the similarity of whole texts. Since a text con-

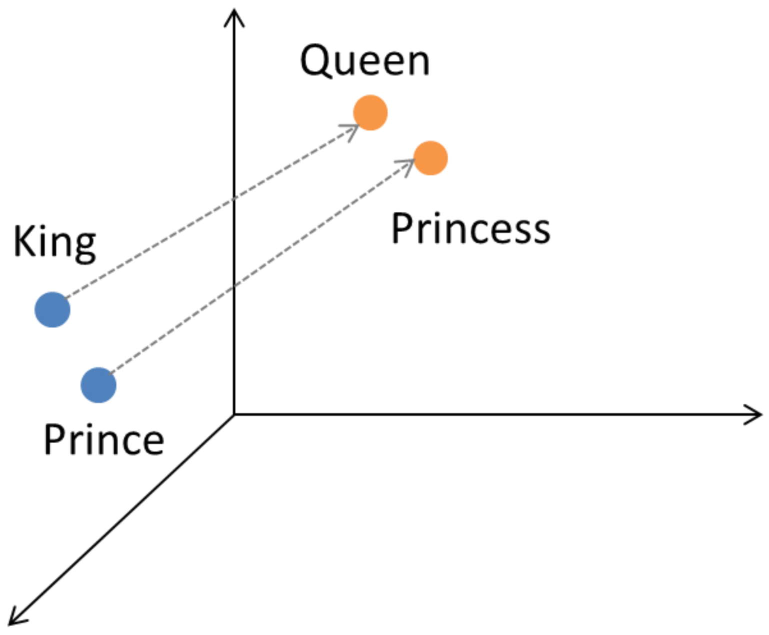


Figure 11: Word vectors
Source: Based on [25]

[5] code.google.com/p/word2vec

sists of multiple words, it seems to be a viable option to represent the text as the sum of the word vectors.

The vector sum representing a text then can be compared to the vector sum representing another text for example via cosine similarity to obtain a similarity measure for those two texts. This approach has some advantages over the already discussed methods using vectors and their similarity.

One advantage is that the vectors of similar words are similar by definition. This leads to fewer problems with different forms of the same word. Another advantage is that the word order is considered at least implicitly when the texts that shall be compared are also the training dataset. This is the case because the context is considered when calculating the word vectors.

5.5 Citation Network

Texts often explicitly refer to other texts. Especially in the legal domain this is very common. In the reasoning of a judgment often the laws and paragraphs the decision was based on are mentioned. Also similar cases are sometimes stated. References are as well given within one law. An example is given in Listing 5 showing the section BGB §437. The explicit references are: "§ 439", "§§ 440, 323 und 326 Abs. 5", "§ 441", "§ 440, 280, 281, 283 und 311a" and "§ 284".

Such explicit references can be found with the help of pattern matching done by regular expressions or using Apache Ruta (see chapter 1.2). As one can see from the example in Listing 5 the explicit references can look quite differently. There can be references to only one other paragraph or to multiple paragraphs. Then the multiple section numbers can be divided by commas or "und". Additionally, the subsection can be given like in "326 Abs. 5". Therefore, the search patterns can get quite complex and this example does not even consider references across different documents.

§ 437 Rechte des Käufers bei Mängeln

Ist die Sache mangelhaft, kann der Käufer, wenn die Voraussetzungen der folgenden Vorschriften vorliegen und soweit nicht ein anderes bestimmt ist,

1. nach § 439 Nacherfüllung verlangen,
2. nach den §§ 440, 323 und 326 Abs. 5 von dem Vertrag zurücktreten oder nach § 441 den Kaufpreis mindern und
3. nach den §§ 440, 280, 281, 283 und 311a Schadensersatz oder nach § 284 Ersatz vergeblicher Aufwendungen verlangen

Listing 5: BGB § 437
Source: Bürgerliches Gesetzbuch

When all references are found with a suitable pattern, the references then can be merged into a citation network. For a similarity search all outgoing references would be considered. If one document is mentioned multiple times it would be ranked higher than a document that is only referred to once. Also transitive references could be considered. As an example, if document *A* references document *B* and document *B* references document *C*, then document *C* might be also relevant to document *A*.

Winkels et al. use a similar approach to create a citation network over 13,311 documents from the Dutch immigration law [27]. Not only do they use the texts themselves, but they also utilize some of the given metadata. The used metadata include the date of the decision, the field of law and the court it was issued. To find the references they use regular expressions and achieve a recall of 87% and a precision of 99%. In the paper it is stated that the resolving of references over document boundaries is not that easy as already suspected in this chapter. It is for example especially hard when not the name of the law is mentioned but only "that law" referring to the last mentioned law. Another problem is the usage of abbreviations for laws.

After they overcame these problems Winkels et al. achieved a recall of 85% and a precision of 95% for their reference resolving. The resolved references then add up to a network.

To indicate that one case can refer multiple times to one law they add weights to edges of their network. The weight W is calculated as: $W = \frac{1}{n}$ where n is the number of occurrences. Therefore, the "lower the weight, the stronger the impact on the network is" [27].

The use of a citation network is especially in the legal domain an interesting approach. That is the case because references are very common in legal documents and the naming of documents is quite consistent. Judgments for example have file numbers and laws have in most cases only one name they are referred to. Additionally, metadata is often given, which can be used. The court and the date a judgment was issued are often provided. Then the higher the court is, the higher the relevance of the judgment, holds probably true. The same applies to the date: the newer, the better.

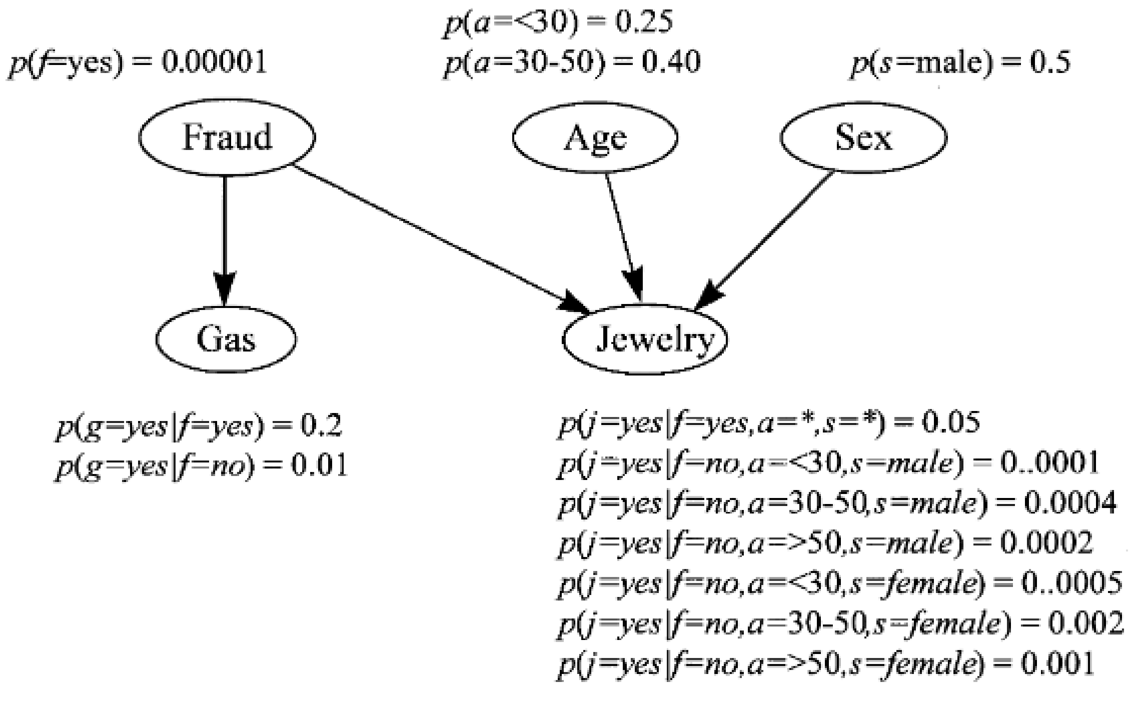
5.6 Bayesian Network

A Bayesian network is a directed acyclic graph (DAG). The nodes of this graph represent random variables in the Bayesian sense. An edge indicates a conditional dependency between the two nodes. Each node possesses a probability function. This function takes a set of values represented by the node's parents as an input and outputs the probability of the variable represented by the node.

An example for such a Bayesian network can be seen in Figure 12. This example is taken from a publication of Heckerman [28]. The network models the detection of credit-card fraud. *Fraud*, *Age* and *Sex* are conditionally independent in this example, while *Gas* and *Jewelry* are dependent. The exact construction of such a network and the underlying mathematical considerations can be learned from the paper mentioned above. The explanation of those is out of scope for this thesis.

Turtle & Croft use a Bayesian network for information retrieval [29]. They also call it "inference network". The basic idea is that "given a set of prior probabilities for the roots of the DAG, these networks can be used to compute the probability or degree of belief associated with all remaining nodes" [29].

Their basic model is shown in Figure 13. It consists of two component networks: a document network and a query network. While the document network is only built once for the document corpus, the query network is built for each information need. The nodes d_1 to d_i repre-



p(f=yes) = 0.00001
p(a=<30) = 0.25
p(a=30-50) = 0.40
p(s=male) = 0.5
Fraud
Age
Sex
Gas
Jewelry
p(g=yes|f=yes) = 0.2
p(g=yes|f=no) = 0.01
p(j=yes|f=yes,a=*,s=*) = 0.05
p(j=yes|f=no,a=<30,s=male) = 0..0001
p(j=yes|f=no,a=30-50,s=male) = 0.0004
p(j=yes|f=no,a=>50,s=male) = 0.0002
p(j=yes|f=no,a=<30,s=female) = 0..0005
p(j=yes|f=no,a=30-50,s=female) = 0.002
p(j=yes|f=no,a=>50,s=female) = 0.001

Figure 12: Bayesian Network
Source: Heckerman [28]

sent the documents, while t_1 to t_j represent the texts. There is a one-to-one relationship between these two set of nodes. The text nodes point to several representation nodes r_1 to r_k. These representation nodes contain information extracted from the texts in the text nodes, for example phrases or metadata. Since different texts can include the same phrase, there are links from different text nodes to the same representation node.

"The query network is an "inverted" DAG with a single leaf that corresponds to the event that an information need is met" [29]. While q_1 and q_2 divide the information need I, they are themselves divided into query concept nodes c_1 to c_m. The document and the query network are joined by the links between the document representation nodes and the query concept nodes.

The whole network represents the dependence of the belief that the information need is met mediated by the document representations and query concepts. So if all documents are equally likely to be observed one gets as a result the probability that the document corpus fulfills the information need. To gain the probability that a certain document d_l meets the information need, one needs to attach evidence to the network by asserting $d_l = true$. By repeating this

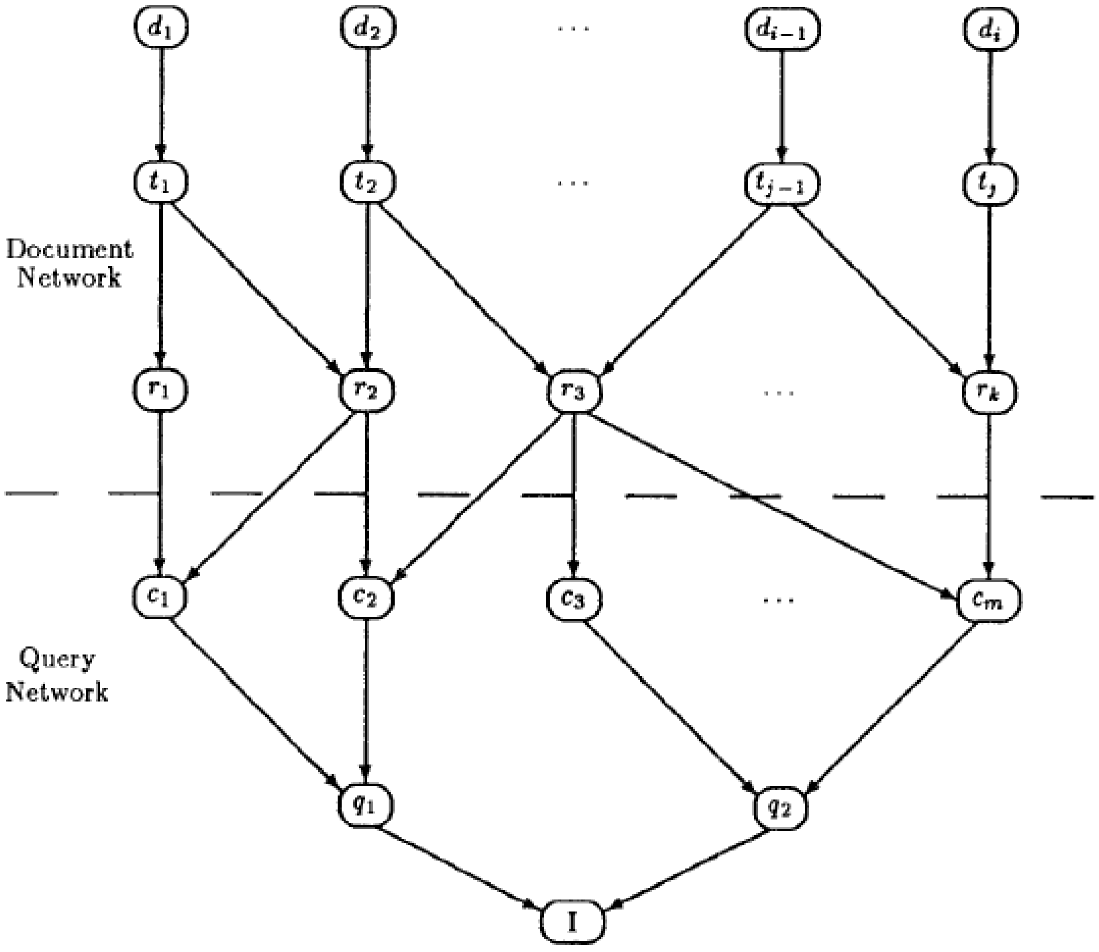


Figure 13: Inference Network
Source: Turtle & Croft [29]

step for each document one can calculate the probability for each document to meet the information need. The documents then can be ranked according their probability to answer the user's information need [30].

The previously described idea of "inference networks" can be quite easily transformed into a method to determine the similarity of documents. For the similarity method the document network can be created in the same way. The basis for the information need and therefore the query network is one of the documents. After each document of the corpus has been the basis of the query network, the similarities between every pair of documents has been determined.

The method implies great space for adaption. The selection of the representation nodes can be adjusted to the document corpus. As a representation single words, phrases or NLP annotations (see chapter 5.3) could be used. Additionally, also metadata can be part of the representation nodes.

The big flexibility in a concrete implementation of this concept justifies its also high complexity.

5.7 Topic Modeling

Topic modeling is a probabilistic method like the already discussed word embeddings (see chapter 5.4) and Bayesian network (see chapter 5.6). The basic idea is that every text is about one or more topics. If a text is about a certain topic can be determined by the frequency certain words are used. In a text that can be assigned to the topic “computer” for example it is more likely that the terms “computer”, “cpu” or “application” occur than in texts that do not belong to this topic. Each topic can be seen as a probability distribution over words. One document can have more than one topic.

The topics are seen as hidden structure of a text. For a corpus of documents the words describing a topic can be extracted. Figure 14 shows the results of such an extraction for the Yale Law Journal. In total 20 topics were extracted whereof the top eight are shown with their most frequent words. The words’ positions along the x-axis denote the specificity to the documents. For example “earnings” is more specific than “tax” in the topic shown in the upper left [31].

There are several approaches to determine topics in text corpora. One of the most well-known ones is *Latent Dirichlet Allocation* (LDA) which was described by Blei, Ng and Jordan in 2003 [32]. This approach uses a bag of words assumption and therefore does not include the order of words in its considerations. Wallach suggested an extension of this technique using n-grams instead of bag of words to include the word order [33].

Another assumption of LDA is that the ordering of the documents does not matter. Therefore, the change of topics over time cannot be understood by this approach. Dynamic topic modeling tries to overcome this shortcoming. Blei & Lafferty described one possible implementa-

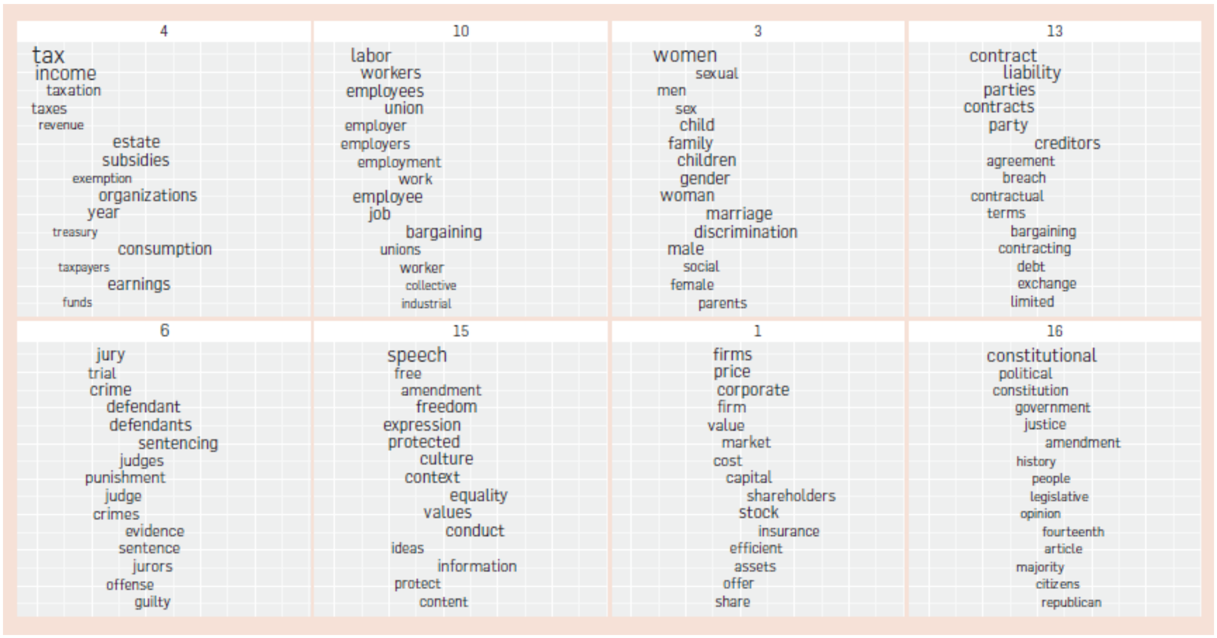


Figure 14: Topics extracted from Yale Law Journal
Source: Blei [31]

tion of a dynamic topic model in 2006 [34].

Lin & Wilbur proposed a document similarity ranking for PubMed[6] based on topic models in 2007 [35]. PubMed is database for biomedical literature with over 26 million entries. Their approach based on the consideration “that the relatedness of documents is mediated through topics” [35]. They describe the relatedness of two documents as the product of the probability that one is interested in document c given topic s_j, the probability that one is interested in document d given topic s_j and the probability of topic s_j, summed across all topics. This idea is then turned into a formula to calculate the similarity of two documents. After some evaluation they state that their system “is able to effectively retrieve related articles” [35].

Another approach to use topic modeling to determine text similarity was suggested by Quan et al. [36]. It was designed for short texts but might also be applicable for longer ones. The basic idea is that especially in short texts there are only few word co-occurrences and therefore the similarity scores using a VSM are low. Their suggestion is to use topic models to find words that are with a certain probability in the same topic. For the words in the same topic the value in the feature vector is increased which leads to a higher similarity score. In experiments they could prove that this approach performs well.

Topic modeling is one of the more complex approaches discussed in this thesis but can be used as the basis of a similarity method of its own as described by Lin & Wilbur. But it can also be used to overcome shortcomings of other similarity methods as shown by Quan et al.

[6] www.ncbi.nlm.nih.gov/pubmed

5.8 Method comparison

In the previous chapters different possible methods to obtain the similarity of two texts were discussed. Table 1 provides an overview of these different approaches and compares them in different categories. For the assessment only the basic approaches without extensions were taken into account

The different categories are: *Preprocessing*, *Text Coverage*, *Word Order*, *Set or Probabilistic*, *Symmetric* and *Complexity*.

Preprocessing describes what needs to be done before the according technique can be made use of. While bag of words and n-grams only need a very simple preprocessing with a Tokenizer, the NLP annotations use a POS tagger, NER or Ruta. Ruta can also be used for the Citation network as well as regular expressions. Word embeddings, Bayesian network and topic modeling all have their specific algorithms to construct their basic data.

Text Coverage deals with how much of the text is taken into consideration by each technique. As one can see from the table the methods bag of words, n-gram, word embeddings and topic modeling make use of the full text. Using the full text states that the maximum of the information given is used, but this might not always be useful because of words that transport no meaning of their own. So these approaches except topic modeling often use stopword lists, which were described in chapter 5.1 and then do not use the full text. Topic modeling sorts out such meaningless words without further guidance. For NLP annotations and Bayesian

Method	**Preprocessing**	**Text Coverage**	**Word Order**	**Set or Probabilistic**	**Symmetric**	**Complexity**
Bag of Words	Tokenizer	Full	No	Set	Yes	Low
N-gram	Tokenizer	Full	Partly	Set	Yes	Low
NLP Annotations	POS, NER, Ruta	Varying	Yes	Set	Yes	Medium
Word Embeddings	Word vectors, e.g. word2vec	Full	Implicit	Probabilistic	Yes	High
Citation Network	Regex or Ruta	Low	No	Set	No	Low
Bayesian Network	Network creation	Varying	No	Probabilistic	No	High
Topic Modeling	Topics, e.g. LDA	Full	No	Probabilistic	Yes	High

Table 1: Method comparison
Source: Own illustration

network the concrete implementation decides how much of the text is used. Therefore, they are described with “varying”. The citation network only uses a very small part of the text since there are normally only few references in a text.

The category *Word Order* wants to give an overview if the word order is considered in the computation of a method. While bag of words, citation network, Bayesian network and topic modeling do not include the word order at all, n-gram does this at least partly. Of course, this only applies when at least bigrams are used. The higher the *n* of the n-grams is, the higher the impact of the word order. The “implicit” stated for word embeddings refers to the fact that the word order is considered while computing the vectors, but has no impact when the word vectors are summed up to get a vector representing the whole text. NLP annotations is the only method out of the described ones that can take the word order fully into account. But the concrete implementation then needs to take care of that.

With *Set or Probabilistic* it is stated whether the similarity is computed by leaning on a set or a probabilistic approach. For word embeddings this is answered with “probabilistic” because the calculation of the vectors is based on a probabilistic model. For the other two probabilistic techniques a probability value is directly taken to describe the similarity of two documents. The set approaches on the other hand all rely on counting how often a certain pattern occurs and calculate based on that a relatedness value.

Symmetric states if the similarity value from document d_1 towards document d_2 is the same as the value from d_2 towards d_1. This is only for citation network and Bayesian network not the case. This characteristic should be kept in mind when using one of the methods. For asymmetric methods the processing time might increase because the similarity value between two documents must be calculated twice. On the other hand this asymmetry provides further insights into the document corpus.

The last category is *Complexity*. This refers to both the complexity of understanding the concept and the complexity of the computation, since these two positively correlate. With higher complexity one hopes to get also a higher quality in the output. But this relation does not always hold true. Therefore, it is an important part of consideration to think about the quality needed for the output and take an according method.

The low complexity of bag of words, n-gram and citation network can already be sensed by the comparatively straightforward preprocessing methods being a tokenizer or regular expressions. This makes the concept easy to understand and leads to fast implementations. Also the absence of complex probabilistic models reduces complexity.

While the method using NLP annotations does not have any probabilistic model as well, it is nevertheless classified as medium complex. This is due to the higher effort to understand the concepts of POS tagging or NER as well as their higher computations costs. But NLP is a well-developed field of research and there are numerous reference implementations to carry out the previously mentioned tasks.

The three probabilistic approaches fall into the group of high complexity. Understanding the underlying mathematics of these methods takes some effort. These approaches could also fill a thesis on their own. This complexity of concept probably also leads to a higher processing complexity. Therefore, they all have specialized algorithms, which try to reduce the processing times. Especially in the case of word embeddings this was achieved in recent years.

This method comparison does not want to argue that one technique is superior to the other, but help to get a feeling when what method might be a good fit. This is heavily influenced by the texts at hand as well as the needed quality in the outcome. Also combinations of the discussed method are possible and sometimes a reasonable alternative.

6 Concepts

6.1 Recommendation System

In order to get an easy to use and easy to extend recommender system it is important to think about the key concepts upfront. In the following these concepts are described which will later be implemented (see chapter 7).

6.1.1 Architecture

The main goal of the architecture is to make adding additional methods that determine the similarity between documents as straightforward as possible. Therefore, one needs to think about which processing steps all similarity methods have in common. Thereof the later program flow can be derived.

The process of computing the similarity can be roughly divided into three steps for each similarity method. These steps can be described as preprocessing, actual comparison and postprocessing.

The first phase is the preprocessing. Some of the already mentioned methods (see chapter 5) need for example NLP to later make the comparison. Others for instance need the transformation into word vectors. Also part of this phase is the deletion of output of earlier runs of the comparison method. All these tasks which have to be done before the actual comparison can take place are part of the step preprocessing.

The comparison of documents itself is the second step. In this phase each document is compared to each other document according to the chosen method. Also for the subparts, called *Section* in the data model (see chapter 1.2), the similarity needs to be computed.

In the third step a postprocessing takes place. Possibly needed actions are standardization of the resulting values or the clean-up of temporary data. Also part of the postprocessing is the persistence of the results. This topic is further discussed in chapter 6.1.2.

Due to this division into three phases an architecture where each step is easily interchangeable seems to provide the best solution for the previously stated requirements. The program flow stays then always the same from the perspective of the invoking instance. This can be achieved by a pipeline paradigm.

The broadest definition of the pipeline concept states that the output of one part of the pipeline is the input of the next part of the pipeline [37]. Figure 15 illustrates this basic structure each similarity method should have. Using this paradigm each method can have its own implementation of each step without looking different from the outside. The three processing steps might be triggered from the outside.

Since the number of documents that need to be compared can be huge, it is reasonable to write the results of the preprocessing to disc, which is usually not intended in the pipeline pattern. However, this approach allows reusing the outcomes of the preprocessing and keeps the occupied main memory reasonably small.

Another advantage of such a pipeline like architecture is that it makes the program easier to be executed in parallel. To use parallelization, a synchronization point after each step would be needed. This assures that for example the preprocessing of a document has already been done before the document gets compared.

Parallelization would be a must in a real life application to be able to handle huge amounts of data. But for the implementation of this thesis it is out of scope and gets only theoretical attention in the concept.

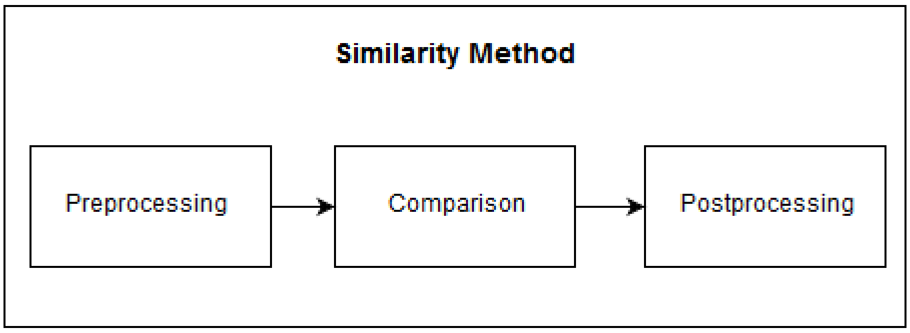


Figure 15: Similarity Method
Source: Own illustration

6.1.2 Persistence

In order to persist the results of the comparison of documents there are several possible alternatives. They will be discussed in the following.

The easiest alternative is not writing the similarities to disc but to calculate them on the fly when requested. However, this approach has some serious drawbacks. The computation of the similarity can take quite some time especially when expensive preprocessing like NLP is needed. Long waiting times are a severe threat for the user experience. Additionally, the work load for the server is unnecessarily increased.

Another alternative is saving the results of the comparison in one huge adjacency matrix. This solves the problem of computing the same similarities again and again. But also in this approach the waiting time can be long, since this matrix grows with the factor n^2 and always needs to be loaded as a whole. This can also lead to running out of memory. Furthermore this matrix is probably only sparsely populate which makes this approach inefficient.

Adding the similarities as an attribute is a further possible approach. The attribute would most likely be an array of the similarity value for each document in the corpus. In this case the classes *LegalDocument* and *LegaDocumentContent* of the data model (see chapter 1.2) would be extended. This alternative raises similar problems like the approach discussed before. The array would only be sparsely populated and can only be searched after it has been loaded as a whole. This is faster than loading the whole matrix, but still probably not the best solution for this kind of problem.

To save each similarity as an own object is another possible solution. This alternative avoids the downsides of the before considered approaches. The objects can be saved in a database in a way that they can be queried. Therefore, not the whole data must be loaded to find the most similar documents. So the waiting time for the user is reduced. Another advantage is that additional attributes that describe the relation between the two compared documents can be easily added. This gives more room for further development.

	Loading Time	**Main Memory Usage**	**Searchable**	**Additional Attributes**	**Disc Usage**
No Persistence	High	Low	No	Yes	None
Matrix	Medium	High	No	No	Medium
Attribute	Medium	Medium	No	No	Medium
Object	Low	Low	Yes	Yes	High

Table 2: Persistence alternatives
Source: Own illustration

A comprehension of each persistence alternative is given in Table 2. Based on this comprehension the best solution is evaluated.

The option of no persistence must be rejected because of the high loading time, although the low main memory usage, the possibility to add additional attributes to the document relation and no disc usage plead for this approach.

A medium loading time is denoted for the adjacency matrix and the alternative to save the similarity values as attribute. This is paid with high or medium main memory usage, medium disc usage and the loss of adding additional attributes. This tradeoff, however, does not make these options viable.

The remaining option to save the similarity values in own objects provides a low loading time, which makes it preferable above the others. Also the low main memory usage, the possibility to search the adequate relations directly in the database and the possibility to add further attributes plead for this alternative. The only drawback of high disc usage can be neglected because disc space is not an expensive good.

So, the similarity values will be saved in objects in the later implementation.

6.1.3 Visualization

The user interface is an important part of every application. Therefore, the recommender system should fit into the existing design of LEXIA. Furthermore it should be easy to use and intuitive.

The user needs on the one hand an interface to start the process of computing the similarities between the given documents. This can be kept quite plain only providing the most important information to the user.

On the other hand, which is much more important, the results of the similarity search need to be displayed. For this problem there are two solutions that come quickly to mind. A simple table that shows the most similar documents is one proven solution for displaying search results. A fancier way of visualizing the results is a network graph.

The network graph can illustrate different similarity values by different distances to the starting node. Additionally, it allows seeing the similarities between the results of the search or displays the most similar documents to one of the result documents. Such a network structure gives the user more insight into the analyzed dataset. It can display the connections between a lot of documents in a denser fashion than a table.

6.2 Similarity of Documents

After the basic concept of the recommendation system was discussed in the previously chapter, this chapter deals with the concept of the similarity methods that get implemented in this thesis. First of all it needed to be decided which of the possible methods discussed in chapter 5 should be implemented. It was decided that a variation of bag of words, an approach relying on NLP annotations and word embeddings should be realized. The decision was based on the different complexities of the techniques and the already existing capabilities provided by LEXIA.

6.2.1 Bag of Words

As described in chapter 5, bag of words is a quite simple technique. This makes it straightforward to implement. Therefore, it can be used as a baseline for the more complex methods. In this thesis not the basic approach will be implemented but a version with some modifications.

One modification is that only nouns will be counted. This decision resolves some of the issues attached to the bag of words approach that were shortly mentioned in chapter 5.1. One of these issues is the handling of particles and similar words which contain no information without their context. This is often tackled by list of words that should be ignored. But this way the next problem arises, that is, which words should be part of the list. Another problem is the appearance of the same word in different conjugations or declinations. Since the test data are in German and show therefore more conjugations and declinations than in English, this is even more serious. Also the obvious solution using stemming or lemmatization does not always work well on German texts. The declination of the noun on the other hand provides valuable information because there is, for example, a great difference between acting as the subject or the object of a sentence.

This decision was encouraged by the fact that LEXIA already provides a POS tagger. So the implementation effort increases only slightly by this decision, while the expected quality of output rises clearly.

Another modification is that not all nouns present in the document corpus are used to describe the vectors but only the ones that are contained in the two texts which are compared. This shortens the vectors by leaving out some zeros and does not change the output of the similarity measure.

Not a real modification but a specification is that the cosine similarity is used to determine the similarity of the vectors in the VSM. This is due to the wide spread usage of this measure for bag of words approaches.

6.2.2 NPChunk

The second method, which will be implemented, leans even more on NLP than the previously discussed version of bag of words. This corresponds to the capabilities in this field provided by LEXIA. It can be put in the category of methods using NLP annotations described in chapter 5.3. As also noted there, this approach has a medium complexity.

The method uses instead of single nouns so-called noun phrases to measure the similarity between texts. Noun phrases are commonly abbreviated as NP and defined as a group of words with a noun as its head [38]. Since this method uses chunks of words that represent NPs, it is henceforth referred to as *NPChunk*.

Based on the NPs found in the texts, vectors are created for a VSM similar to the bag of words approach. Also the similarity value is calculated by using the cosine similarity again. So, the first two approaches differ only in the words used to create vectors.

This method will give in the assessment a good indication whether it is worth to seek out for more complex syntactical constructions to determine relatedness.

6.2.3 Word Embeddings

As third method an approach using word embeddings will be implemented. Unlike the two previously described techniques, this one has an underlying probabilistic model. This was already mentioned in chapter 5, as well as the higher complexity of this approach.

This method will not be built from scratch but uses an implementation developed at the sebis chair of the TU München. The implementation uses the approach of *word2vec* that has already been mentioned in chapter 5.4. It was developed using Python.

It provides the possibilities to upload text documents and to compute the word vectors based on them. Then one can retrieve the most similar words to an input word. Much more interesting for this thesis is that one can also request the similarity of two documents in the uploaded collection. The relatedness of two documents is calculated by summing up the word vectors for each document and comparing the resulting vectors by using the cosine similarity. All of this functionality can be accessed via a RESTful API.

Therefore, the implementation done in this thesis will mainly deal with using the API via HTTP. Additionally, the results need to be transformed into a format LEXIA can work with. This must happen using the architecture proposed in chapter 6.1.1.

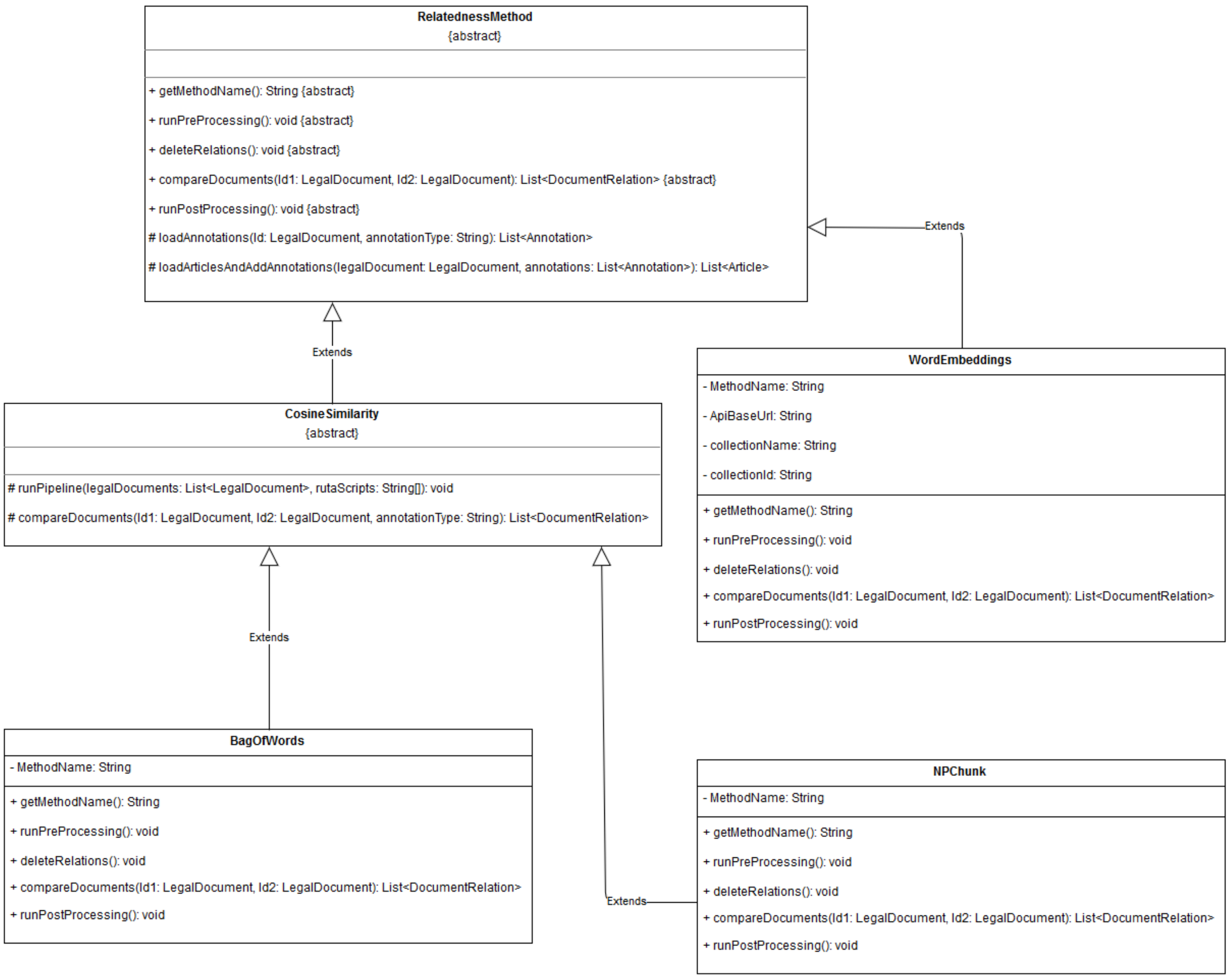
For the assessment this method is interesting to get a comparison between set and probabilistic based techniques. Also, it represents some of the newer trends in research in the fields of NLP.

7 Implementation

7.1 Architecture

As described in chapter 6.1 the expandability of the recommender system is a central design issue. To address this requirement all methods to compute the similarity inherit from one abstract super class. This can be seen in the class diagram in Figure 16.

The super class *RelatednessMethod* defines that each child class needs to implement a few methods. In order to identify each similarity technique a child class must implement the method *getMethodName*. The methods *runPreProcessing*, *compareDocuments* and *runPostProcessing* map to the three steps – preprocessing, comparison, postprocessing - identified in chapter 6.1.1. This abstraction allows having always the same program flow no matter what similarity method needs to be executed. Therefore, further similarity techniques can be easily added.



RelatednessMethod
{abstract}
+ getMethodName(): String {abstract}
+ runPreProcessing(): void {abstract}
+ deleteRelations(): void {abstract}
+ compareDocuments(Id1: LegalDocument, Id2: LegalDocument): List<DocumentRelation> {abstract}
+ runPostProcessing(): void {abstract}
# loadAnnotations(Id: LegalDocument, annotationType: String): List<Annotation>
# loadArticlesAndAddAnnotations(legalDocument: LegalDocument, annotations: List<Annotation>): List<Article>
Extends
Extends
WordEmbeddings
- MethodName: String
- ApiBaseUrl: String
- collectionName: String
- collectionId: String
+ getMethodName(): String
+ runPreProcessing(): void
+ deleteRelations(): void
+ compareDocuments(Id1: LegalDocument, Id2: LegalDocument): List<DocumentRelation>
+ runPostProcessing(): void
CosineSimilarity
{abstract}
# runPipeline(legalDocuments: List<LegalDocument>, rutaScripts: String[]): void
# compareDocuments(Id1: LegalDocument, Id2: LegalDocument, annotationType: String): List<DocumentRelation>
Extends
BagOfWords
- MethodName: String
+ getMethodName(): String
+ runPreProcessing(): void
+ deleteRelations(): void
+ compareDocuments(Id1: LegalDocument, Id2: LegalDocument): List<DocumentRelation>
+ runPostProcessing(): void
Extends
NPChunk
- MethodName: String
+ getMethodName(): String
+ runPreProcessing(): void
+ deleteRelations(): void
+ compareDocuments(Id1: LegalDocument, Id2: LegalDocument): List<DocumentRelation>
+ runPostProcessing(): void

Figure 16: Similarity methods class diagram
Source: Own illustration

As an addition there is a *deleteRelations* method required by the super class. This method should check if there are already similarities computed with the specified similarity approach and delete them. Thereby no duplicates can emerge by running the same similarity method several times.

The techniques implemented in *BagOfWords* and *NPChunk* have some resemblance. So there is another abstract class called *CosineSimilarity* they inherit from. This approach makes some methods accessible to both of them and thereby avoids the duplication of code. The implementations of these similarity techniques are more precisely described in chapter 7.2.1 respectively chapter 7.2.2.

For the persistence an additional class was introduced as described in chapter 6.1.2. Its properties can be seen in Figure 17. The mentioned advantage of adding attributes that describe the relationship is used to specify the document part that is compared. That allows distinguishing between the comparison of whole documents or only parts of it. So it is possible to use the inherent structure of legal documents to improve the results of the similarity search. These inherent structures where described in chapter 1.1.

In order to allow saving asymmetric similarity measures as well as symmetric ones the attribute *value* always refers to the similarity from *ld1* respectively *article1* towards *ld2* respectively *article2*. This means on the other hand that also for symmetric similarity measures two *DocumentRelations* need to be saved for one comparison.

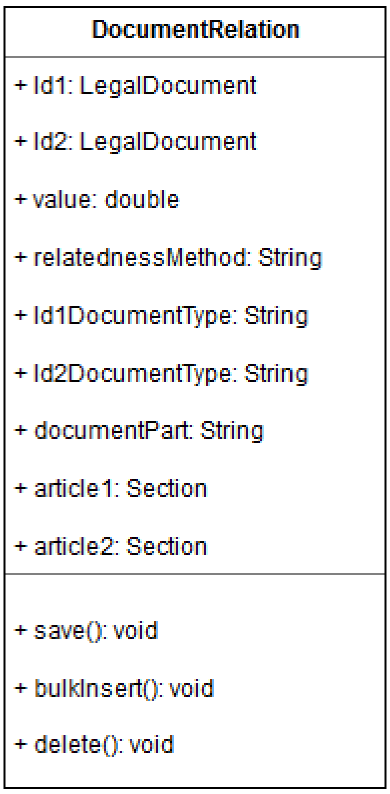


Figure 17: DocumentRelation class
Source: Own illustration

The *DocumentRelation* class offers a *bulkInsert* method. This method makes use of the *Bulk API* of elasticsearch [39]. Since every document part is compared to every other document part, the number of *DocumentRelations* rapidly grows with each document imported to the system. Therefore, the insert into the database can only be handled in reasonable time with such a bulk operation. The maximum bulk size was set to 10,000 after tests showed that bigger bulk sizes lead to a performance decrease.

As mentioned before the program flow stays always the same regardless of which similarity technique is executed. The process is shown in Figure 18 exemplarily by the execution of *BagOfWords*.

The starting point of the program flow is the user who makes a call to the *RelatednessController* via the angular frontend of LEXIA. In the call the desired similarity technique is specified by a string. In this example case it is 'BagOfWords'. According to the transferred string a new instance of the class *BagOfWords* is created.

Inside this instance first the *deleteRelations* method is called to remove already existing *DocumentRelations* from earlier runs. This method itself queries the elasticsearch database to de-

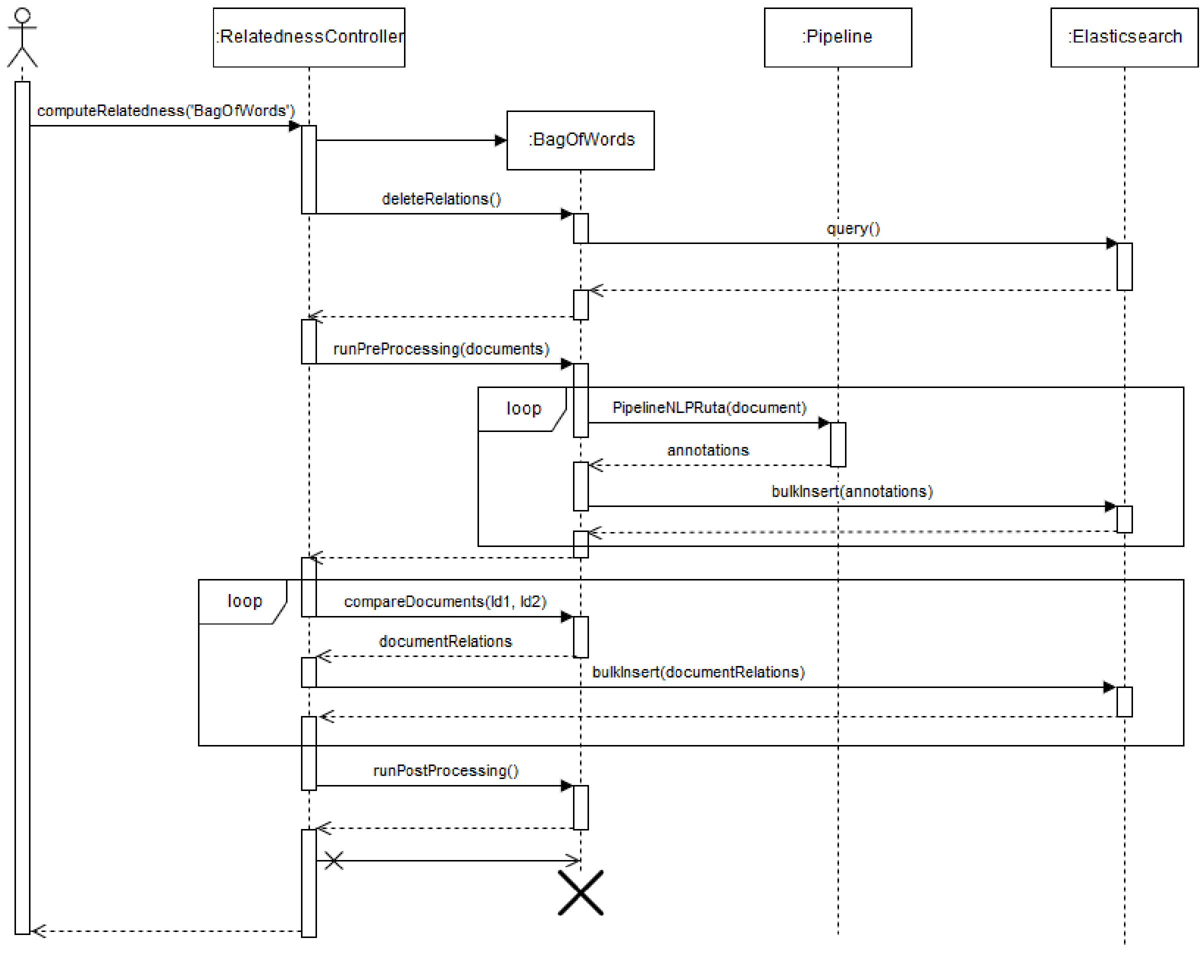


Figure 18: Bag of words execution example
Source: Own illustration

lete these entries. It makes use of the *Delete by Query Plugin* of elasticsearch [40]. Using this plugin one can avoid loading and deleting each database entry individually. As the name suggests the entries to delete are specified by a query. The deletion of all matching entries is done in one operation; however, it can take some time according to how many *DocumentRelations* are present in the database.

After the old *DocumentRelations* have been deleted the preprocessing for the comparison can start. In the case of *BagOfWords* and *NPChunk* the preprocessing consists of NLP. This step utilizes LEXIA's capabilities in this field. The resulting annotations are then saved in the database for usage in the next step.

WordEmbeddings does not use NLP in its preprocessing and so there are also no annotations to store. However, this difference does not change the program flow seen from the perspective of the *RelatednessController* because it just calls the *runPreProcessing* functionality without needing to know what happens there.

In the next step for each two documents the similarity value is computed while using the results of the preprocessing. Not only the similarity of the whole documents but also of its sub parts like articles in laws or statement of facts in judgments is determined. The similarity is expressed by a score between zero and one while zero is no similarity and one is identity. The result of the *compareDocuments* method is a list of *DocumentRelation* objects which then are saved to the database.

After the *DocumentRelations* have been saved the method executing the postprocessing is called. In the case of *BagOfWords* no actions need to be performed. Afterwards the instance of *BagOfWords* is discarded. If everything ran without an error a success message is send to the user.

Since this is a long running process the user also has the possibility to track the process on a periodically updated view of the LEXIA frontend. Every before mentioned method updates the status displayed in this view.

7.2 Similarity Methods

The underlying concept of each method was described in chapter 6.2. The general program flow to compute the similarity between documents has been described in the previous chapter. However each concrete implementation of the similarity concepts has some noteworthy features, which will be described in the following.

7.2.1 Bag of Words

The approach of counting word occurrences to determine the similarity of two texts is quite straightforward and easy to implement. The only questions to answer are: which words one counts and how the score is calculated.

As described in chapter 6.2.1 the choice was made to count only nouns. These nouns then get transformed into a VSM. According to the vectors representing the nouns the text similarity is calculated using the cosine similarity.

In order to find all nouns in a document the present implementation performs POS tagging. The functionality to perform such an action was already implemented in LEXIA before the start of this thesis. The method *runPipeline* in the class *CosineSimilarity* shown in Figure 17 in chapter 7.1 serves as wrapper to access the NLP capabilities of LEXIA. As also described in chapter 7.1 the found nouns are saved to the database in the preprocessing step. Therefore, this expensive operation does not need to be executed multiple times and the resulting annotations also do not need to be held in memory which could otherwise lead to out of memory exceptions.

In the comparison step the nouns of the documents that are to be compared are loaded and accordingly assigned. This is done by the methods *loadAnnotations* and *loadArticlesAndAddAnnotation* which are defined in the super class *RelatednessMethod*. By defining them in the super class other similarity methods using NLP annotations can also access them.

After the nouns are loaded, the number of occurrences of each noun in each document and each part of the documents is counted. The result is retained in a vector for each document and document part. Then the cosine similarity between these vectors is computed to get the similarity score. This is done as mentioned before by using the cosine similarity. The formula for the cosine similarity is given in chapter 5.1.

Since the vectors are all in positive space due to their creation process, the resulting similarity values are in the desired range between zero and one. These values are then saved in *DocumentRelation* objects in the database.

In the *runPostProcessing* method no action is taken. A possible action in this step would have been to delete the annotations produced in the preprocessing. It was decided not to do that because creating the POS tags is an expensive task. So if only one document is added to the document corpus it is possible to only compute the POS tags for this added document and then compare the documents. Otherwise the POS tags for all documents would need to be computed again.

7.2.2 NPChunk

The implementations of *NPChunk* and *BagOfWords* have a lot in common. That is why they share the common parent class *CosineSimilarity*. *NPChunk* also uses the NLP capabilities provided by LEXIA. In this case not only a POS tagging is used but also the UIMA Ruta component [6].

“The UIMA Ruta language is an imperative rule language extended with scripting elements” [6]. UIMA Ruta therefore allows finding more complex syntactical constructions such as NPs. As described in chapter 6.2.2 this method wants to find NPs and calculate a text similarity based on them.

The definition of a NP includes a lot of syntactical constructions. For the sake of simplicity and hopefully higher precision this thesis only takes noun phrases with adjectives into consideration. Therefore, the rule used in the implementation searches for an arbitrary number of adjectives followed by one noun.

Similar to the implementation of bag of words the occurrences of each noun phrase then are counted. The results are converted to vectors afterwards. To determine the similarity score the cosine similarity is used again.

7.2.3 Word Embeddings

As described in chapter 6.2.3 this similarity technique uses an implementation developed at the sebis chair of the TU München. The implementation is based on *word2vec* and its RESTful API is used to add word embeddings as a similarity method to LEXIA.

The API provides methods to upload texts, get the similarity score between two texts and to delete texts via HTTP calls. So in the *runPreProcessing* method the texts of the documents that are to be compared are uploaded in bulk via HTTP POST. In this step it can also be specified how many dimensions the vectors should have, if the words should be stemmed and if a stopword list should be used. For this thesis it was decided to use 200 dimensions and to forgo using stemming or a stopword list. After this upload is done the word embeddings implementation computes the vectors and determines the similarity.

The *compareDocuments* method queries each similarity value via HTTP GET. The results are then saved in the database. In *runPostProcessing* the before uploaded texts are deleted to clean up the remote application.

7.2.4 More like this

In contrast to the methods mentioned before this method is not implemented using the architecture described in chapter 7.1. That is the case because it uses the capabilities provided by elasticsearch itself. This part of elasticsearch is called *More Like This Query* and is part of its *Query DSL* [41]. This similarity method is also used as comparable figure for the methods described before, since this is developed by full text search specialists.

This similarity method is initiated by a query to the elasticsearch database that is formatted in JSON. One part of the JSON object is the text to which similar texts should be found. Inside the database the following happens. For the input text the top terms with the highest tf-idf score are determined. The top n of these terms are then taken as input for a search on the elasticsearch index. The terms are connected with a logical OR. The number of terms to use can be stated as one parameter of the *More Like This Query*. For this thesis the n was set to 25. The most similar documents are the documents with the highest score regarding the OR-Query. Elasticsearch is based on Lucene and therefore the scoring model of Lucene is used to determine the score. This scoring mechanism also works with tf-idf [42].

Beneath the number of top terms that should be used there are a few other parameters to tweak the quality of the results. One is the minimum term frequency below which the term from the input text will be ignored. This is set to one. Also set to one is the minimum document frequency. As in the other methods no stopword list was used which is a further option of the query.

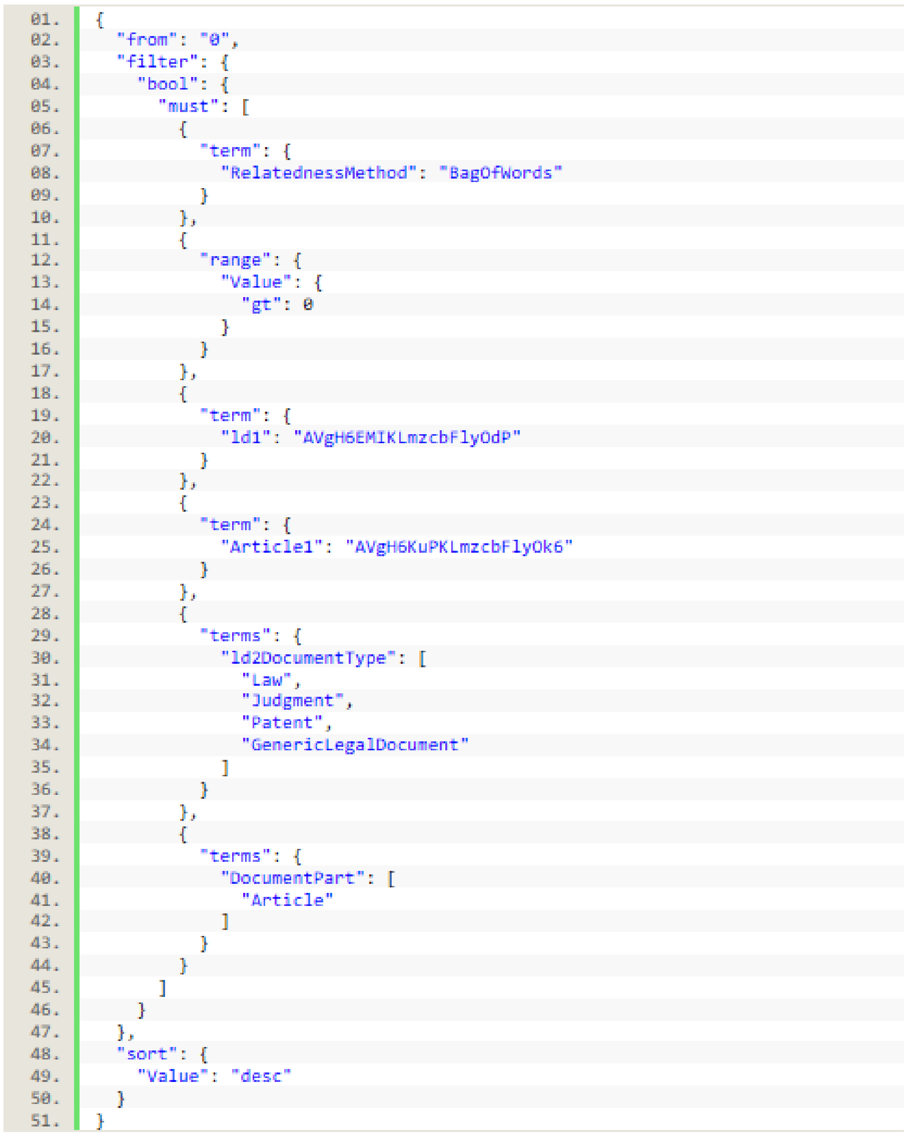
An important difference to the techniques described before is that this one generates the results and the score on the fly. It does not need to compute the score a priori for every document to have a reasonable response time. This is possible because of the indexing done by elasticsearch while saving one document.

7.3 Data retrieval

After describing how the similarity data is computed and saved it is also important to know how it is retrieved. Since the data is saved as objects with multiple attributes, each of these attributes can be specified in the query.

The attributes that always need to be part of the query are *ld1*, *relatednessMethod* and *ld1DocumentType*. Since elasticsearch is not an object database only the *IDs* of the *LegalDocuments* and *Sections* are written to the database and are therefore queried.

Specifying further attributes gives the user the possibility to narrow down the results according to his preferences. One example might be that he only wants to see judgments in the results. So he would set the field *ld2DocumentType* to *Judgment*.



```
{
  "from": "0",
  "filter": {
    "bool": {
      "must": [
        {
          "term": {
            "RelatednessMethod": "BagOfWords"
          }
        },
        {
          "range": {
            "Value": {
              "gt": 0
            }
          }
        },
        {
          "term": {
            "ld1": "AVgH6EMIKLmzcbFlyOdP"
          }
        },
        {
          "term": {
            "Article1": "AVgH6KuPKLmzcbFlyOk6"
          }
        },
        {
          "terms": {
            "ld2DocumentType": [
              "Law",
              "Judgment",
              "Patent",
              "GenericLegalDocument"
            ]
          }
        },
        {
          "terms": {
            "DocumentPart": [
              "Article"
            ]
          }
        }
      ]
    }
  },
  "sort": {
    "Value": "desc"
  }
}
```

Figure 19: Data retrieval query
Source: Own illustration

As touched on in chapter 7.2.4 the more complex queries in elasticsearch are expressed as JSON objects. In Figure 19 one can see such a complex query.

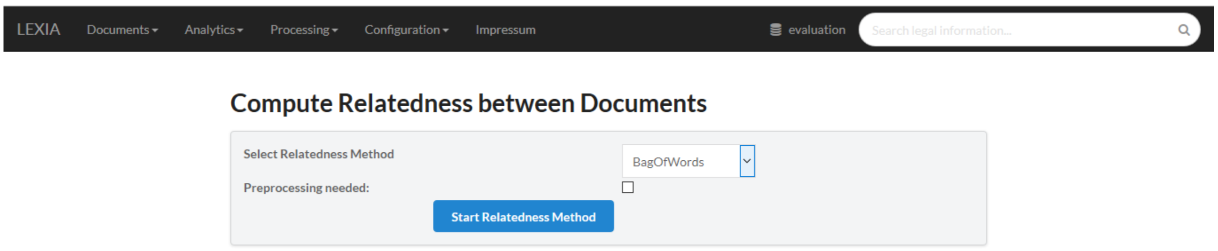
In this query first of all the offset for the result set is defined by the field *from* (line 02). The more interesting part happens afterwards. The array *must*, starting in line 05, contains all conditions that must be fulfilled to match. Each condition is represented by one object and they are connected by a logical AND. The *relatednessMethod* is set to 'BagOfWords' in line 08. For *ld1* (line 20) and *article1* (line 25) ids are specified. These ids identify the document and its section to which similar paragraphs are searched. In this query only articles are of interest. So the *documentPart* is set to *Article* in line 41. The requested articles should be part of documents of the type *Law*, *Judgment*, *Patent* or *GenericLegalDocument*. This is stated in the lines 30 to 35. In the end the results are sorted by *value* in descending order which is defined in line 49.

In order to get java objects the result that is a JSON object needs to be interpreted. Additionally, the ids for *LegalDocuments* and *Sections* in each result must be resolved to the object they refer to. Then the application can work with them again and send them as response to the user.

7.4 Visualization

For the visualization part there was on the one hand the need to have an input screen to start the process of computing the similarities. On the other hand also the results need to be displayed. As described in the chapter before the user can make some adjustments to the query. These options need also to be displayed.

As shown in Figure 20 the site to start the computing process is kept very simple. In the dropdown the user can choose which similarity method shall be executed. The checkbox *preprocessing needed* gives the user the option to skip the preprocessing step. This might be useful to save some time if the annotations for *BagOfWords* or *NPChunk* are already in the database. The submit button leads to an asynchronous call of the *RelatednessController*. Therefore, the user can move on to other sites of the application without interrupting the similarity computing process.



Compute Relatedness between Documents

Select Relatedness Method: BagOfWords

Preprocessing needed: ☐

Figure 20: Start similarity computing process
Source: Own illustration

To view the results the user has two possibilities as described in the concept in chapter 6.1.3. The first one is a list representation where the user can look through the paginated results. The second possibility is a network view. Starting with one node the graph can be further expanded according to the interests of the user.

Figure 21 shows the list view. On the left side the user can select the query options described in chapter 7.3. In the upper part, that is named *Views*, the different similarity methods are displayed which include *Bag of Words*, *NPChunk*, *Word Embeddings* and *More Like This*. The other menu points named *Plain*, *Semantics* and *References* provide information about the selected document respectively document part that are not related to similarity and therefore not part of this thesis. Beneath the method selection the user can include or exclude certain document types and document parts. These options are displayed below the heading *Recommendations*. In the picture the user is only interested in documents of the type *Judgment* and only the document part *Statement of Facts*.

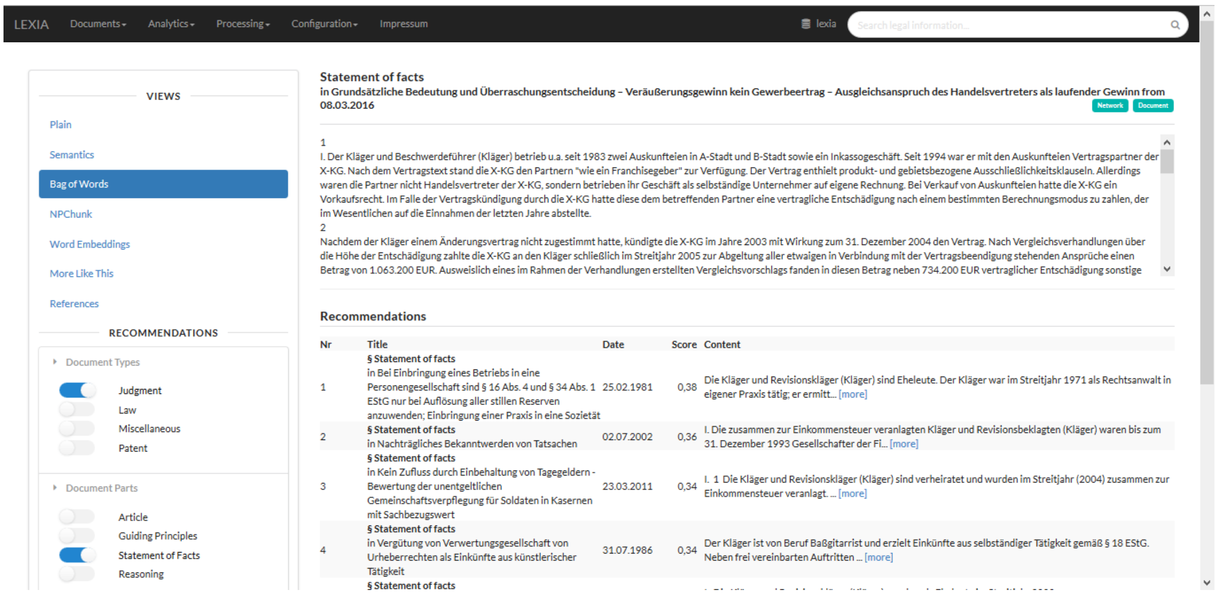
The upper part of the content on the right side is occupied by the selected text. Also the paragraph name, the document name and the date of the text are shown there. Below this the results of the similarity search are displayed starting with the heading *Recommendations*. They are shown in a table ordered from higher to lower similarity score. For each result the name of the paragraph, the document name, the promulgation date, the score and a text preview are shown. The *[more]* link directly leads to the paragraph of the document. A maximum of ten results are shown at a time. An arrow below the table that cannot be seen in the image allows moving to the next ten results.

An example of the network view is shown in Figure 22. The screen in this view is divided into three parts. On the left there are statistics and filter options, in the middle the graph itself is displayed and on the right side the text of a paragraph can be shown.

The statistics on the left side give an overview of how many nodes are present in the graph. Additionally, the average, the maximum and the minimum similarity score of the edges is displayed. Beneath these statistics the user can choose filters for the graph nodes. These filters are similar to the ones in the list view. So the user can decide which document types and which document parts should be shown in the graph as nodes. In contrast to the list view also a similarity threshold can be stated. Then only documents with a similarity score higher than the threshold will appear in the network graph.

The graph in the middle shows the nodes and their connections to each other. Each node has its paragraph number displayed as identifier. On mouseover the whole paragraph name is shown in the top left corner of the graph box.

In the network graph the following applies: the smaller the distance between two connected nodes is the higher is their similarity score. But twice the distance does not mean that it is half



Statement of facts
in Grundsätzliche Bedeutung und Überraschungsentscheidung – Veräußerungsgewinn kein Gewerbeertrag – Ausgleichsanspruch des Handelsvertreters als laufender Gewinn from 08.03.2016

1
I. Der Kläger und Beschwerdeführer (Kläger) betrieb u.a. seit 1983 zwei Auskunfteien in A-Stadt und B-Stadt sowie ein Inkassogeschäft. Seit 1994 war er mit den Auskunfteien Vertragspartner der X-KG. Nach dem Vertragstext stand die X-KG den Partnern "wie ein Franchisegeber" zur Verfügung. Der Vertrag enthielt produkt- und gebietsbezogene Ausschließlichkeitsklauseln. Allerdings waren die Partner nicht Handelsvertreter der X-KG, sondern betrieben ihr Geschäft als selbständige Unternehmer auf eigene Rechnung. Bei Verkauf von Auskunfteien hatte die X-KG ein Vorkaufsrecht. Im Falle der Vertragskündigung durch die X-KG hatte diese dem betreffenden Partner eine vertragliche Entschädigung nach einem bestimmten Berechnungsmodus zu zahlen, der im Wesentlichen auf die Einnahmen der letzten Jahre abstellte.
2
Nachdem der Kläger einem Änderungsvertrag nicht zugestimmt hatte, kündigte die X-KG im Jahre 2003 mit Wirkung zum 31. Dezember 2004 den Vertrag. Nach Vergleichsverhandlungen über die Höhe der Entschädigung zahlte die X-KG an den Kläger schließlich im Streitjahr 2005 zur Abgeltung aller etwaigen in Verbindung mit der Vertragsbeendigung stehenden Ansprüche einen Betrag von 1.063.200 EUR. Ausweislich eines im Rahmen der Verhandlungen erstellten Vergleichsvorschlags fanden in diesen Betrag neben 734.200 EUR vertraglicher Entschädigung sonstige

Recommendations

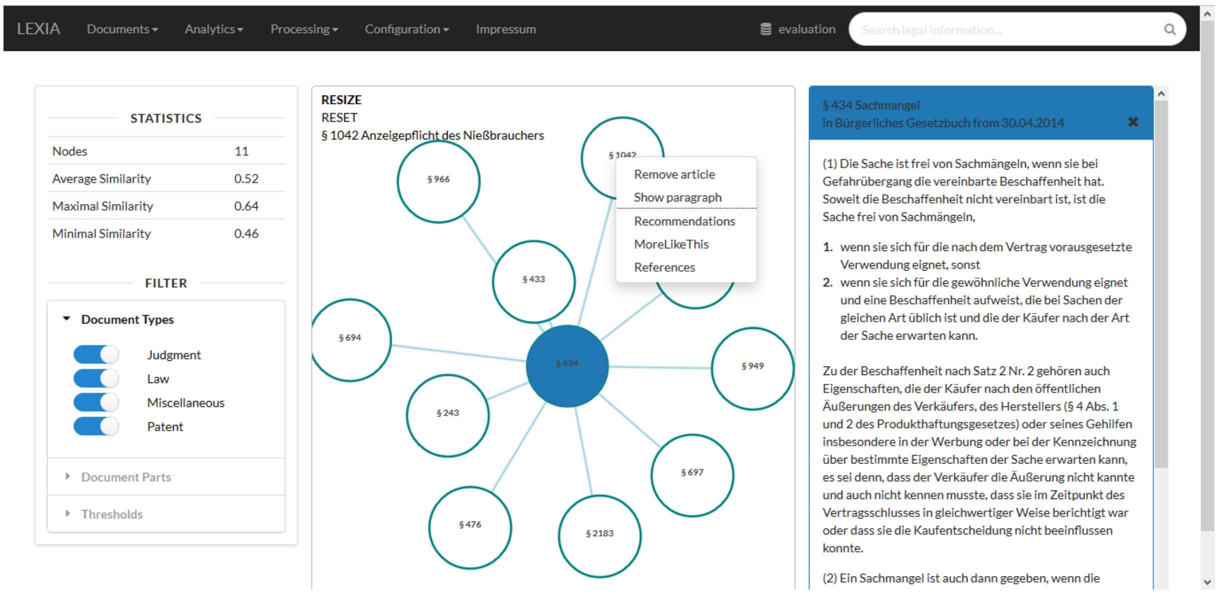
Nr	Title	Date	Score	Content
1	**§ Statement of facts** in Bei Einbringung eines Betriebs in eine Personengesellschaft sind § 16 Abs. 4 und § 34 Abs. 1 EStG nur bei Auflösung aller stillen Reserven anzuwenden; Einbringung einer Praxis in eine Sozietät	25.02.1981	0,38	Die Kläger und Revisionskläger (Kläger) sind Eheleute. Der Kläger war im Streitjahr 1971 als Rechtsanwalt in eigener Praxis tätig; er ermitt... [more]
2	**§ Statement of facts** in Nachträgliches Bekanntwerden von Tatsachen	02.07.2002	0,36	I. Die zusammen zur Einkommensteuer veranlagten Kläger und Revisionsbeklagten (Kläger) waren bis zum 31. Dezember 1993 Gesellschafter der Fi... [more]
3	**§ Statement of facts** in Kein Zufluss durch Einbehaltung von Tagegeldern - Bewertung der unentgeltlichen Gemeinschaftsverpflegung für Soldaten in Kasernen mit Sachbezugswert	23.03.2011	0,34	I. 1 Die Kläger und Revisionskläger (Kläger) sind verheiratet und wurden im Streitjahr (2004) zusammen zur Einkommensteuer veranlagt. ... [more]
4	**§ Statement of facts** in Vergütung von Verwertungsgesellschaft von Urheberrechten als Einkünfte aus künstlerischer Tätigkeit	31.07.1986	0,34	Der Kläger ist von Beruf Baßgitarrist und erzielt Einkünfte aus selbständiger Tätigkeit gemäß § 18 EStG. Neben frei vereinbarten Auftritten ... [more]
	§ Statement of facts			

Figure 21: List view of results
Source: Own illustration

the similarity score. Otherwise nodes would often overlap and the graph would not be readable at all.

On right click of a node a context menu appears. It gives the user various possibilities to work on the graph. *Remove article* deletes the selected node from the network. With *Show paragraph* the user can choose to display the text of the selected paragraph on the right side. Then the node and the corresponding text get the same background color. *Recommendations* and *More Like This* expand the graph. The first one uses the results of the method *BagOfWords* while the other one uses the method *MoreLikeThis*. The ten documents with the highest similarity score nodes are then added if they are not already part of the graph. Then connections to the selected node are drawn.

These two described views support different use cases. The list view gives the user an exact analysis on one document. The network view on the other hand has a more explorative approach. The user can get a quick overview of his data and see the interconnections of the documents.



STATISTICS

Nodes	11
Average Similarity	0.52
Maximal Similarity	0.64
Minimal Similarity	0.46

§ 434 Sachmangel
in Bürgerliches Gesetzbuch from 30.04.2014

(1) Die Sache ist frei von Sachmängeln, wenn sie bei Gefahrübergang die vereinbarte Beschaffenheit hat. Soweit die Beschaffenheit nicht vereinbart ist, ist die Sache frei von Sachmängeln,

1. wenn sie sich für die nach dem Vertrag vorausgesetzte Verwendung eignet, sonst
2. wenn sie sich für die gewöhnliche Verwendung eignet und eine Beschaffenheit aufweist, die bei Sachen der gleichen Art üblich ist und die der Käufer nach der Art der Sache erwarten kann.

Zu der Beschaffenheit nach Satz 2 Nr. 2 gehören auch Eigenschaften, die der Käufer nach den öffentlichen Äußerungen des Verkäufers, des Herstellers (§ 4 Abs. 1 und 2 des Produkthaftungsgesetzes) oder seines Gehilfen insbesondere in der Werbung oder bei der Kennzeichnung über bestimmte Eigenschaften der Sache erwarten kann, es sei denn, dass der Verkäufer die Äußerung nicht kannte und auch nicht kennen musste, dass sie im Zeitpunkt des Vertragsschlusses in gleichwertiger Weise berichtigt war oder dass sie die Kaufentscheidung nicht beeinflussen konnte.

(2) Ein Sachmangel ist auch dann gegeben, wenn die

Figure 22: Network view of results
Source: Own illustration

8 Assessment

The assessment of the previously described implementation is divided into two parts. One part deals with the technical aspect of the implementation and will be more quantitative. The other part revolves around the question how good the results of the similarity search are. This will be done via a questionnaire.

8.1 Technical assessment

The in chapter 6.1.1 stated objective of the architecture to be easily expandable is fulfilled by the implemented architecture described in chapter 7.1. While it took more than four weeks to set up the architecture and implement the first similarity method, the following methods only took about two working days each to be added to the system.

An important piece of technical data for a software application is always the performance and therefore the responsiveness of the application. Since it was decided in the concept (see chapter 6.1) to compute the similarities a priori and then retrieve the results from the database, these two steps will also be independently assessed.

For the performance assessment four German laws with different numbers of paragraphs were used. These laws were the *Bürgerliches Gesetzbuch* (BGB) with 2402 paragraphs, the *Strafgesetzbuch* (StGB) with 532 paragraphs, the *Verwaltungsgerichtsordnung* (VwGO) with 205

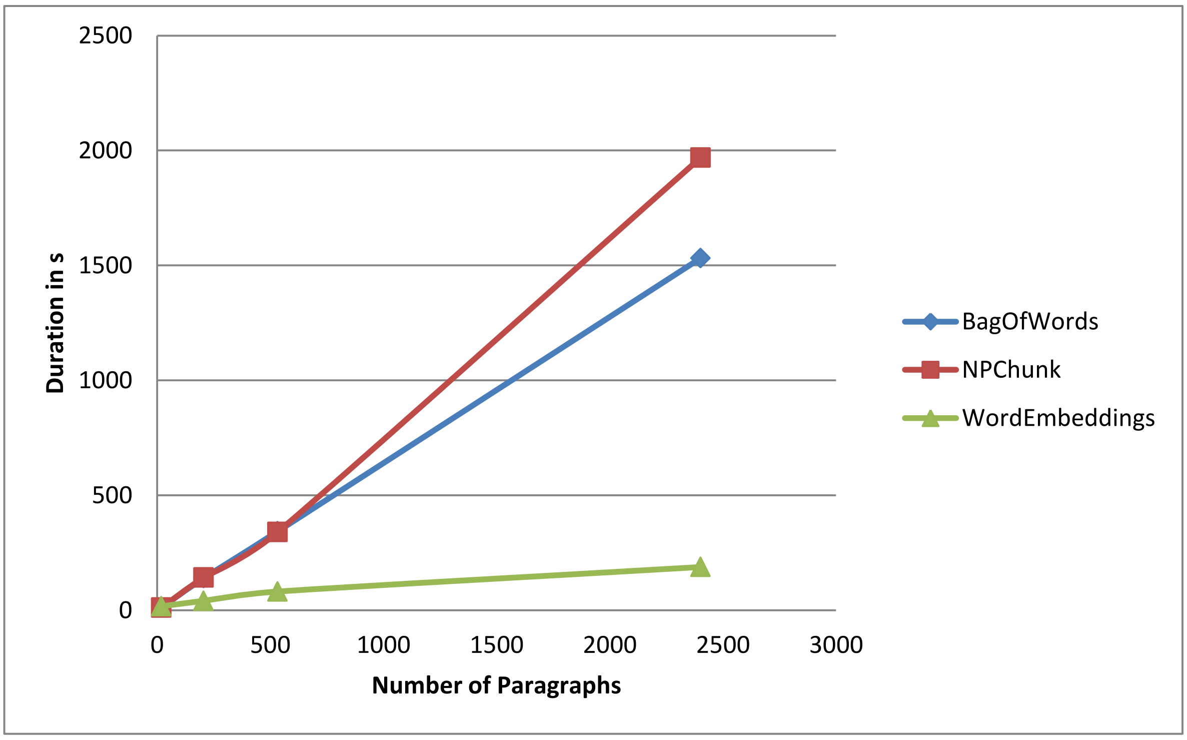


Figure 23: Preprocessing time for different number of articles
Source: Own illustration

paragraphs and the *Produkthaftungsgesetz* (ProdHaftG) with 19 paragraphs. One paragraph contains between 50 and 60 words on average.

Before the comparisons themselves can take place, the preprocessing needs to run. Figure 23 shows the time needed to run the preprocessing relating to the number of paragraphs in the document. For all three implemented methods the correlation seems to be linear. The preprocessing of *WordEmbeddings* does take the least time because it does not need POS tagging. *NPChunk* runs longer than *BagOfWords* because on top of the POS tagging a Ruta script is executed. All methods take quite some time in the preprocessing step, *NPChunk* for example over half an hour for the BGB. This justifies the decision to save the results of the preprocessing.

In the comparison step each paragraph of a law is compared to each paragraph of the same law. This leads to $\frac{n*(n-1)}{2}$ comparisons of document parts for symmetric methods and $n*(n-1)$ comparisons for assymetric ones. So for *BagOfWords* processing the BGB 2,883,601 comparisons of paragraphs need to take place. In Figure 24 the quadratic relationship of processing time and number of paragraphs is shown exemplarily for *BagOfWords*. But it is also true for *NPChunk* and *WordEmbeddings*.

In order to get a better understanding of how the number of words in each paragraph impacts the performance also parts of the DATEV dataset described in chapter 4.2 were tested. Therefore, different numbers of judgments were randomly chosen for test datasets. The four sets consisted of 30, 150, 300 and 1498 paragraphs. One paragraph contained between 500 and

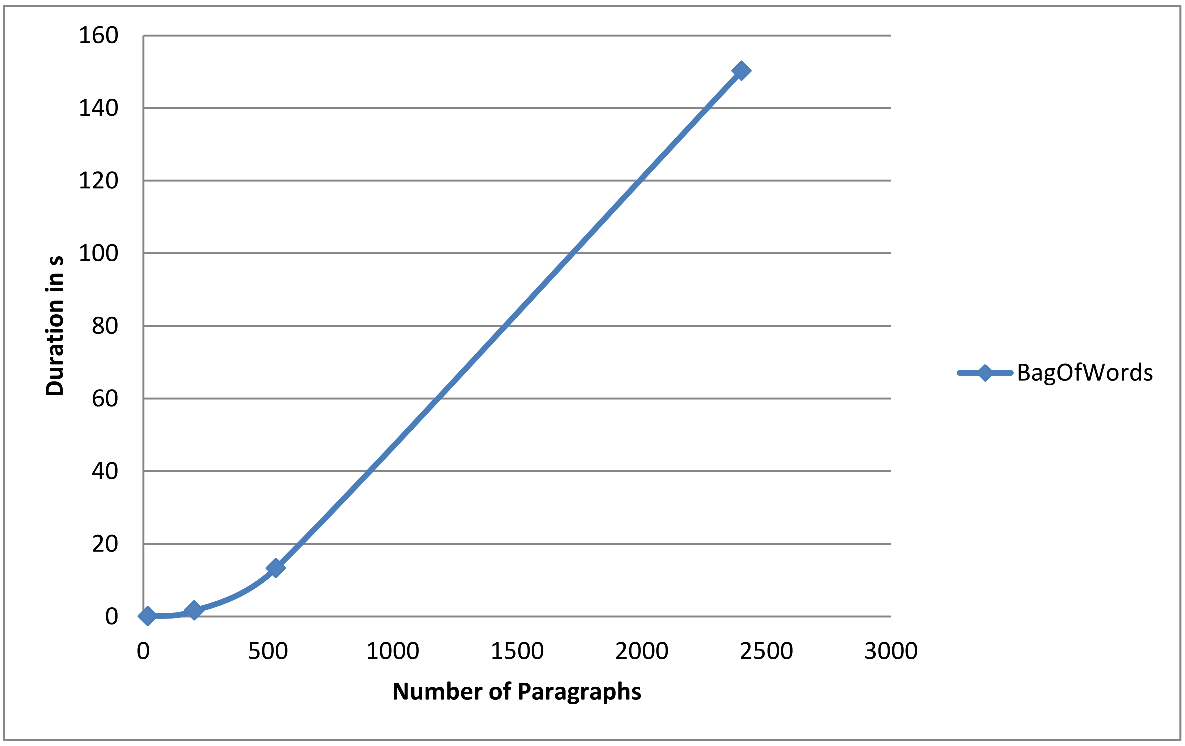


Figure 24: Comparison time for different number of paragraphs
Source: Own illustration

600 words on average. The average paragraph is therefore ten times longer than the average paragraph of the laws tested before.

For each of the test datasets the similarity method *BagOfWords* was computed. The preprocessing times are shown in Figure 25 by the blue line. The data is extrapolated which is displayed as black line. So it can be better compared to the already known processing times of the German laws which are shown in red. One can see that the increase of words per paragraph leads to an increase in preprocessing time. But this happens in a much smaller magnitude than the increase in the number of words. While the word count was increased by a factor of about ten, the preprocessing time only increased by a factor of about 1,3.

The impact of increased paragraph length on the comparisons themselves is for *BagOfWords* and *NPChunk* tremendously bigger. The time for the comparison of two paragraphs is increased by a factor of about 50. This is due to the increased number of annotations that need to be loaded from the database. This in turn leads to longer vectors for each paragraph and therefore more time is needed to create those. Also the calculation of the cosine similarity of those vectors takes longer because of their increased length. In the end this adds up to a factor that is much higher than the factor the word count grew with.

All processing times were measured with a single machine setup. The machine was equipped with an Intel Core i7-6500U processor, eight GB of DDR3 RAM and a 500 GB SSD.

Watching the resource monitor of the machine one could see that alternately the processor and the disc were under heavy load. This is due to the fact that elasticsearch has its index divided into segments. These segments need to be merged from time to time so that all segments have

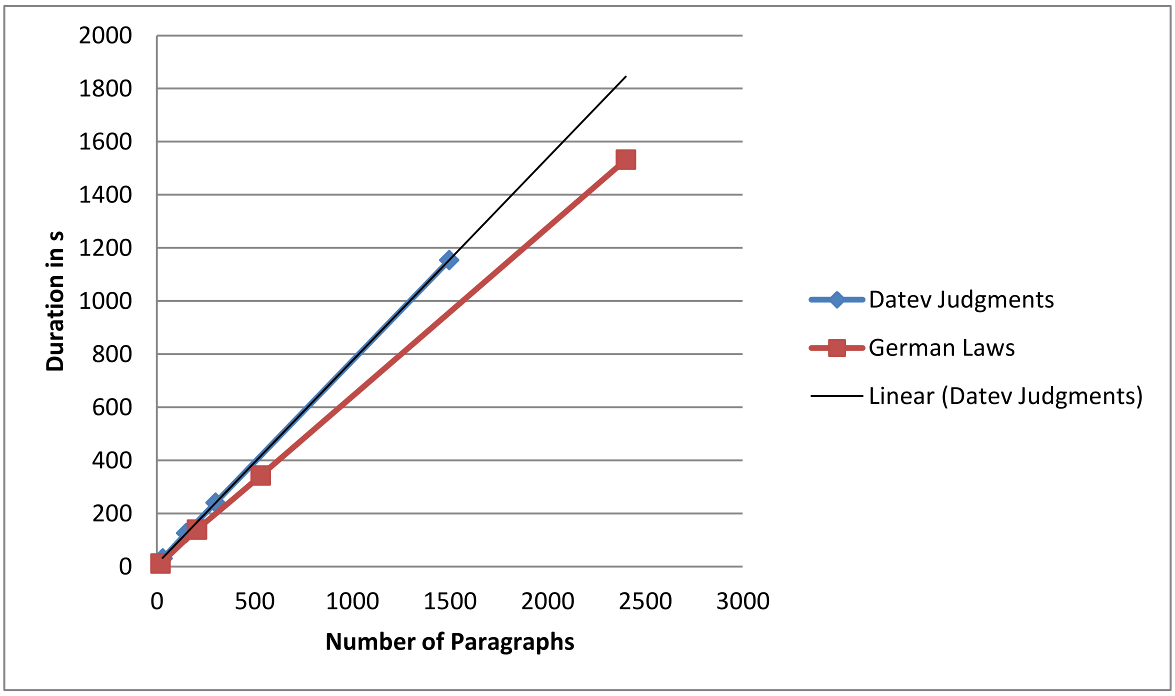


Figure 25: Impact of number of words on preprocessing time
Source: Own illustration

an equal size and only a maximum number of segments exists. The merge of two segments implies that the whole data of the two segments need to be rewritten [43]. This explains the heavy load on the disc in periodic intervalls. Since a lot of *DocumentRelation* objects need to be inserted in a row, this characteristic of elasticsearch slows the process down. This indicates that another database system storing the *DocumentRelation* data might speed up the process of computing the similarities between documents, especially when the number of documents and document parts is high.

Also from a retrieval perspective a relational database system storing the similarities might have been a better choice. Due to the conception of elasticsearch as full-text search database it is not as fast as a relational database when a query filters on exact matches in different fields. Another problem of elasticsearch is handling such queries on a huge amount of data. Figure 26 shows this. The query times for less than two million *DocumentRelations* are in a reasonable range with less than a second as maximum. But a further increase of objects in the database leads to long loading times with over seven seconds at about 7.2 million entries in the database.

On the other hand, elasticsearch handles the *More Like This* queries even for bigger datasets in a small amount of time. Therefore, it is a good idea to store the texts themselves in such a full-text search engine.

In conclusion, one can say that for a real life application with a huge document corpus the load of computing the similarities needs to be distributed to several machines. Otherwise this process would take ages. Also there must be some thinking of which kind of data should be stored in which kind of database. That is important to minimize query times.

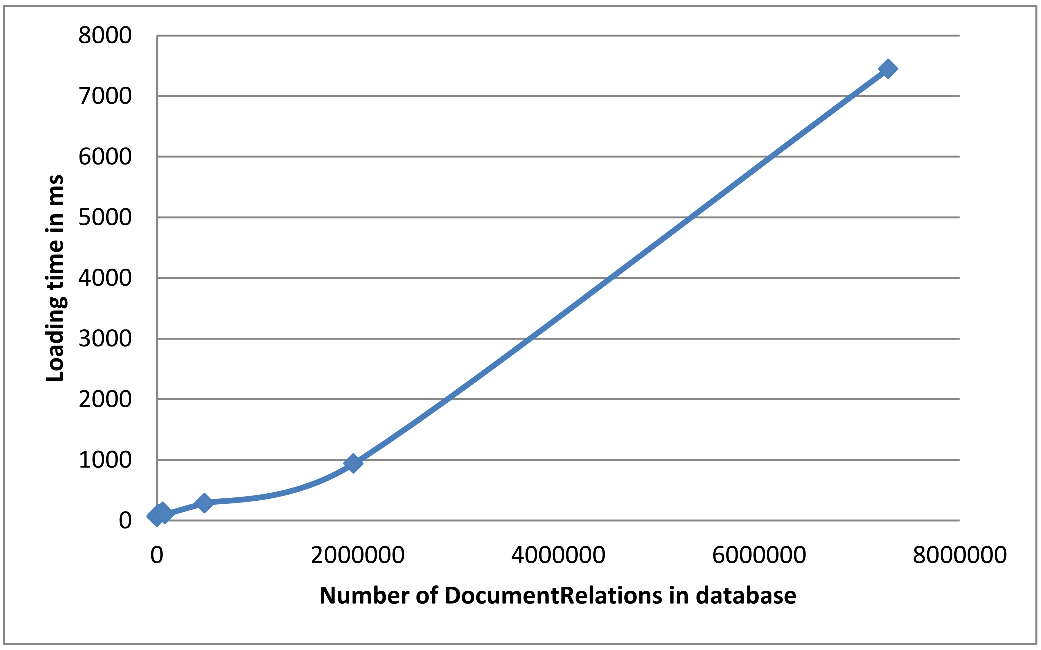


Figure 26: Query time for DocumentRelations
Source: Own illustration

8.2 Functional assessment

The objective of the functional assessment was to get a perception of how good the suggestions of the different similarity methods were. Therefore, a questionnaire was created using google forms[7]. The questionnaire could be filled out and submitted online which made it more convenient for the participants.

8.2.1 Survey design

In the questionnaire the participants should evaluate how relevant a legal norm is in relation to another norm on a scale from one to four. The rating one stated that the norm was not relevant, while four indicated that the norm was very relevant. If a participant was not familiar with one of these norms he could also give no answer.

The norms for which the suggestions were evaluated were BGB §§ 280, 434, 985 and StGB § 242. For each of these source paragraphs and each method the ten most similar paragraphs within the same law were taken to be assessed. In some cases there were less than ten suggestions made by a method which lead to fewer paragraphs that needed to be evaluated. All in all there were 93 norms that needed to be evaluated against a base norm. The participants did not know which suggestion was produced by which method. The whole questionnaire can be found in the appendix.

It was decided to use these four before mentioned methods because they are quite well known by most people working or studying in the legal domain. So the participants did not need to read long texts before they could give a relevance estimate. This in turn leads to a shorter time needed to fill out the questionnaire which increases the chance of getting answers. These advantages could not be provided by judgments. That is why they were left out in the process of the functional assessment.

In order to get reliable answers but also answers from different points of views the link to the questionnaire was mailed to several experts from the legal domain. Additionally, it was posted in forums and groups for law students.

At the end there were twelve answers submitted. For the further assessment only eleven of them were consider because one answer rated every legal norm relation with a relevance of three. Since this cannot be considered as a serious opinion, this one is left out. Otherwise it would have led to a biased analysis.

The questions on further relevant paragraphs were only answered by two participants. Since this is not significant, this part of the questionnaire is left out in the further assessment.

[7] www.google.de/intl/de/forms/about/

8.2.2 Results

The first thing to evaluate is how much the participants agreed on the relevance of each norm towards the source norm. Therefore, the standard deviation for each relevance rating was calculated. In Figure 27 a boxplot of the standard deviations for the 93 questions can be seen. It shows that there was broad consensus with standard deviations as low as 0.3 on some norms. But the boxplot also shows that there was great disagreement on other norms with a standard deviation up to 1.14.

The median has a value of 0.62 and the interquartile range is 0.34. This suggests that in the majority of cases the agreement on the relevance of one norm was quite broad. Apart from the outliers with a high standard deviation, one can say that the ratings given by the participants are reliable and reproducible.

Figure 28 presents the average relevance ratings. They are divided by source norm, similarity method and how many suggestions were taken into account. *Top 3* means that only the results of the similarity search with the three highest similarity scores were taken. *Top 5* and *Top 10* work accordingly with the five respectively ten highest scoring suggestions.

NPChunk delivered for the source norms BGB § 434 and StGB § 242 only three results and for BGB § 985 no result at all. Therefore, these averages are left blank.

A positive outcome that can be seen in Figure 28 is that the average relevance ratings of the *Top 3* are in most cases better than those of the *Top 5* and the *Top 5* in turn is on average better than the *Top 10*. This means that the scoring of the similarity methods from better to worse seems to work. However a look into the raw data, which is provided in the appendix, shows that also in the *Top 3* there are sometimes suggestions with an average relevance rating below

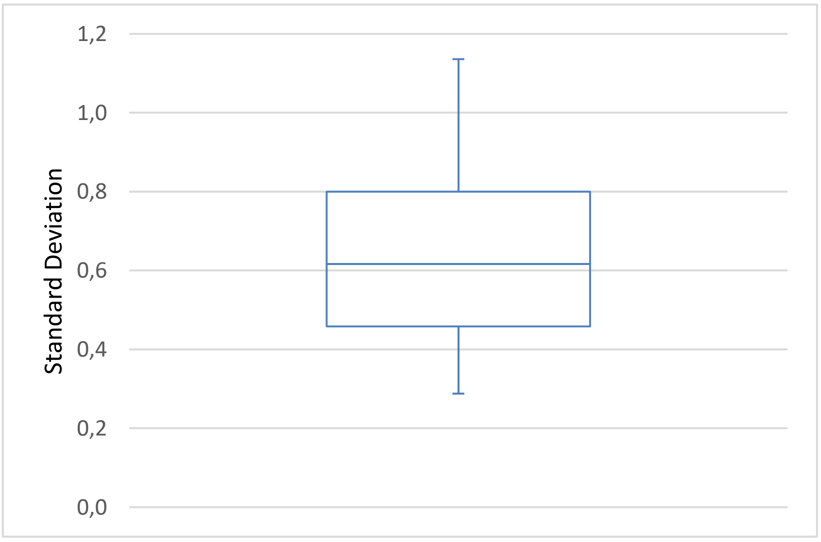


Figure 27: Boxplot of standard deviations
Source: Own illustration

1.5. Nevertheless it can be stated that the probability to find a highly relevant suggestion in the *Top 3* is higher than in the *Top 10*.

One finding that can also be derived from the graphs in Figure 28 is that there is no method implemented that constantly performs better than the others. In addition, which method has the best average relevance rating often depends on whether one is considering the *Top 3*, *Top 5* or *Top 10*.

BagOfWords outscores all other methods with its results for BGB §§ 434 and 985 regardless of how many suggestions are taken into account. But it is outperformed by *MoreLikeThis* and *NPChunk* on the results for §StGB 242 where *WordEmbeddings* has similar average ratings like *BagOfWords*. Also the *Top 3* and *Top 5* suggestions of *MoreLikeThis* on BGB § 280 receive higher average ratings than the ones of *BagOfWords*. Then again the *Top 10* average

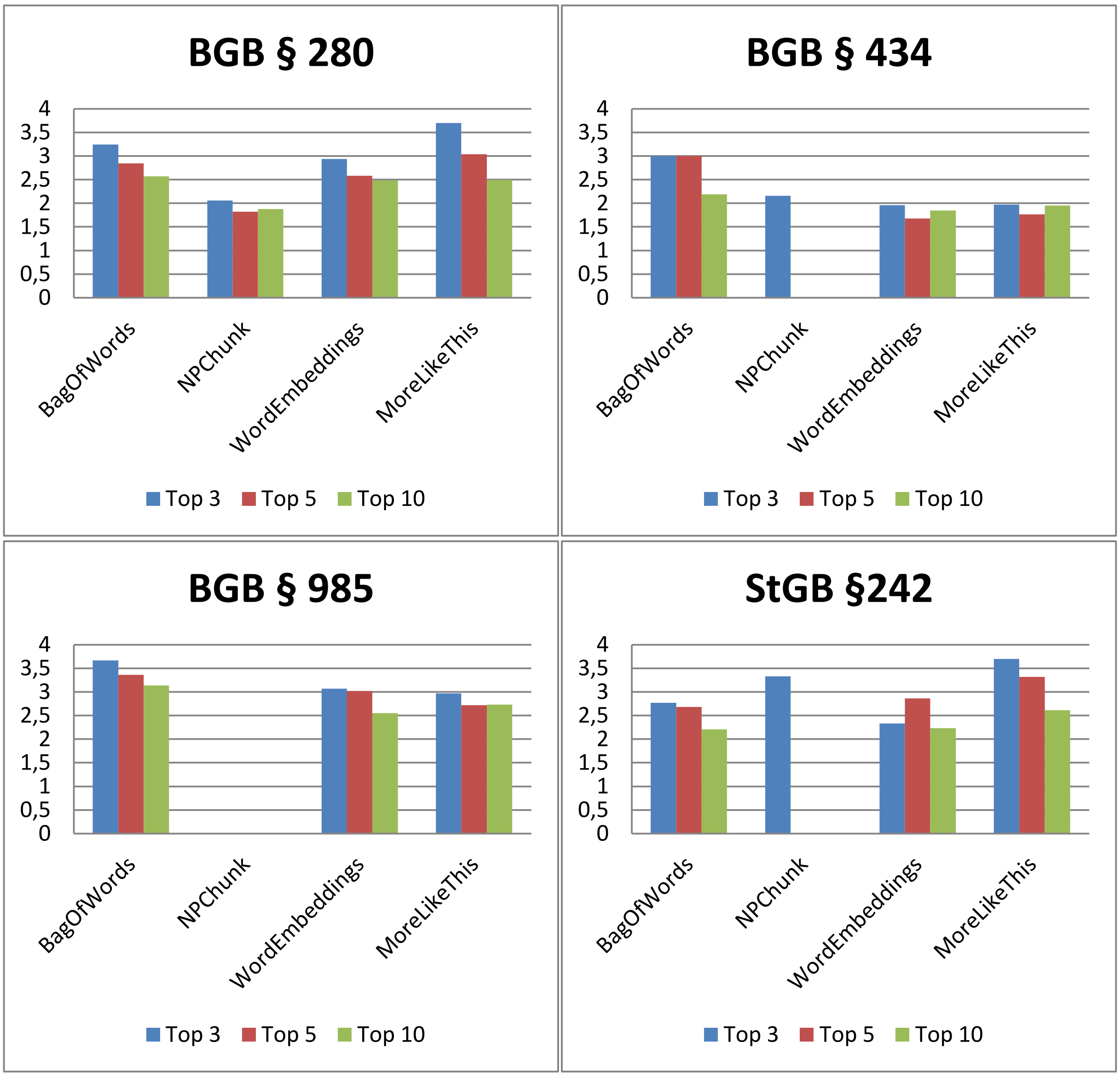


Figure 28: Average relevance ratings
Source: Own illustration

rating on the search results for this norm is matched by *WordEmbeddings*.

NPChunk delivers no or only few results apart from BGB §280 as source norm. The few suggestions that are made are also in many cases not relevant. The only exception are the three suggestion made for StGB § 242 with an average relevance rating of 3.33. The bad overall performance of *NPChunk* might be traced back to the small number of words per paragraph. The low amount of words leads to only a little number of noun phrases that can be found. This in turn makes finding equal noun phrases in other norms more unlikely which leads to fewer and worse results.

There is always a method that performs better than *WordEmbeddings*. But on the other hand, there is also always a method that has an average relevance rating as bad as or worse than *WordEmbeddings*. For this method there might be the most potential for improvement with only little effort. With adjusting the various parameters that can be set for this method the performance might increase drastically.

MoreLikeThis seems to be quite unreliable. It has the best ratings for its suggestions for BGB §280 and StGB §242. On the other hand, its performance is the worst of the four methods for BGB §§ 434 and 985. Also for this method an adjustment of the parameters might help. However, since this is a built-in feature of elasticsearch, one has the least control over what is happening.

All in all for every source norm at least one method achieves an average rating for its *Top 3* and *Top 5* of 3.0 or higher. This means that the suggestions are at least slightly relevant on average which is an encouraging result. The *Top 3* are in three out of four cases even above 3.5 indicating a really good performance.

But the averages do not show the whole picture. The boxplots in Figure 29 give an impression of how wide spread the relevance estimates of the suggestions were. For the graphs all given suggestions by each method were included. So it is not surprising that for *NPChunk*, which mostly did not provide ten suggestions, the ranges are smaller.

In most cases the whiskers span nearly the whole rating scale. This means that almost always the search results include very relevant but also irrelevant suggestions. Looking at the for the most part high interquartile range one can see that these are not only outliers. Therefore, the suggestions are mostly a broad mix of relevant and irrelevant ones.

A desirable look of the boxplot represents the one for *BagOfWords* for BGB § 985. The lowest rating is still above 2.0 and the median is above 3.0. Also the interquartile range is low with 0.9. This means that the average is high while there are only few downward outliers which are also in still acceptable scope. So it is not surprising that these suggestions have the highest average for a *Top 10* in this assessment.

But a high average in the relevance rating does not show another important fact: what very relevant norm is not suggested by a method in its *Top 10*. For this purpose one needs to have again a look at the raw data. There one can see that no method has all paragraphs that are highly relevant, i.e. has an average relevance rating above 3.5, in its *Top 10* suggestions. Even *BagOfWords* for BGB § 985 misses the norm BGB § 994 which has a relevance rating of 3.6. This paragraph is only found on position 44 of the search results.

Another interesting finding that can be derived from the results is the fact that all methods perform better on short paragraphs than on long ones. This is quite surprising since one could think that more input leads to a higher precision in the output. But BGB § 985 consists of only eleven words and the average relevance rating of the methods' *Top 10* suggestions is always above 2.5. BGB § 434, on the other hand, contains 218 words and the suggestions do not reach a higher relevance rating than 2.2. BGB § 280 and StGB § 242 lie somewhere in between consisting of 73 respectively 36 words. Also the average ratings for these norms are between the ones of BGB § 985 and BGB § 434. Getting a deeper understanding why this is

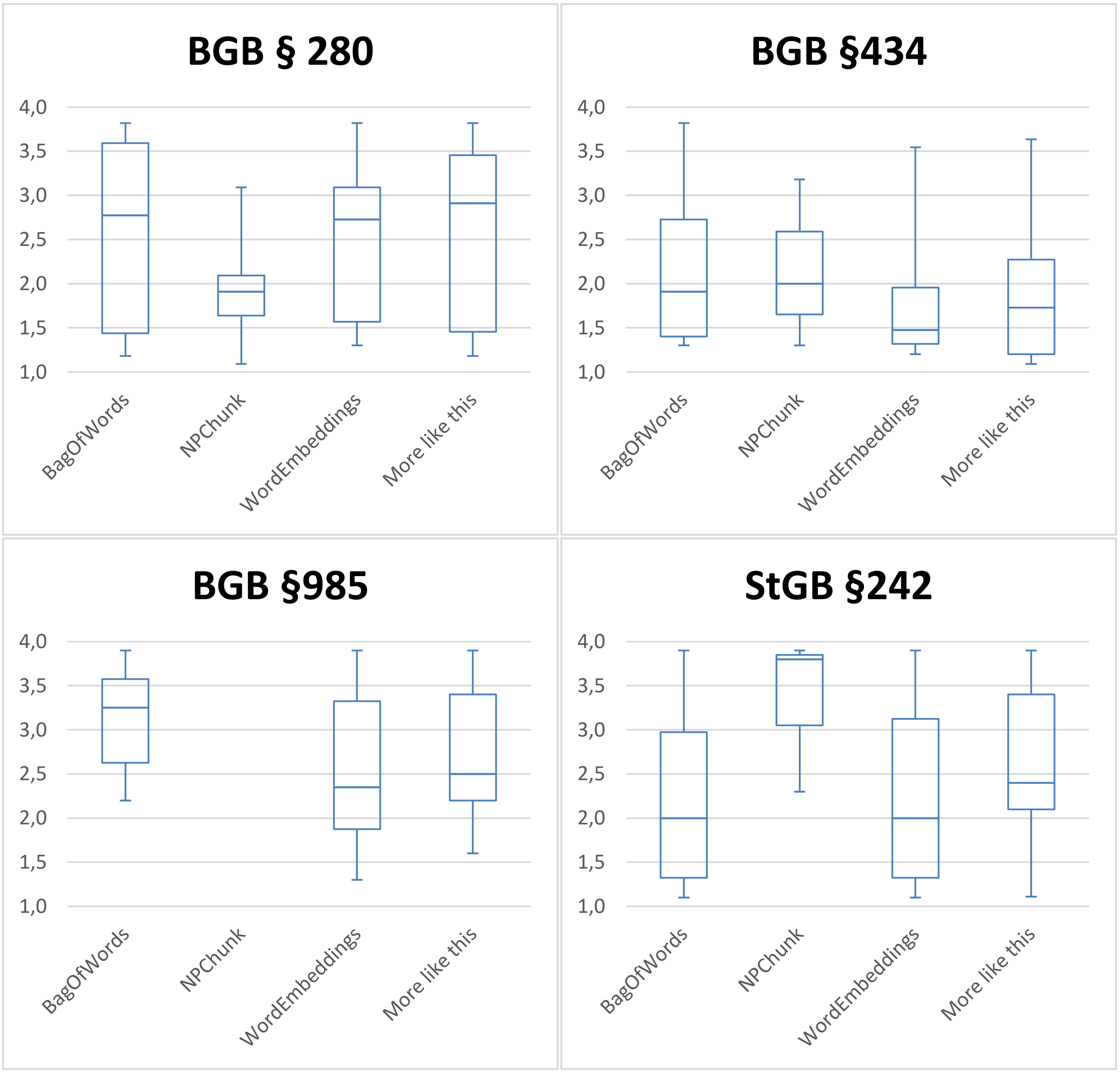


Figure 29: Boxplots for relevance ratings
Source: Own illustration

the case might be a first step for further improvement of the methods. Also a further investigation if this correlation holds true for other paragraphs or is just a coincidence in the test data would be needed.

In conclusion the results of the assessment show that it is possible to find relevant norms given a source norm with the implemented similarity methods. The quality of the results varies from source norm to source norm and from method to method. Unfortunately no method could prove itself superior to the others.

The more naïve approaches of *BagOfWords* and *MoreLikeThis* performed better than the more sophisticated ones of *NPChunk* and *WordEmbeddings*. As already mentioned *NPChunk* might have suffered from the short paragraphs while the search results of *WordEmbeddings* might be dramatically improved by adjusting the parameters. This could even be made on the basis of the results of this assessment.

9 Conclusion and outlook

In this thesis a prototypical implementation and assessment of a relatedness search for legal documents was described. Therefore, a literature review of related work was done to find techniques that can determine similarity of texts. Special attention was paid to papers that already dealt with documents from the legal domain. A comparison of these methods concluded this part of the thesis.

The prototypical implementation was a further development of LEXIA. For this purpose three approaches of the before discussed techniques were selected. Additionally, it was made use of a feature of the elasticsearch database to get another comparison method. Attention was also given to the architectural design within these techniques were implemented. The design was first described in a more general manner. Then the concrete implementation in LEXIA followed.

From a technical point of view the architectural design provided the expected benefits and eased the implementation of further similarity methods. However there were also some technical shortcomings. Especially the time needed to compute the similarities for a huge data corpus needs to be named.

For the functional assessment a questionnaire was designed to evaluate the search results of the different similarity methods. The participants filling out this questionnaire work or study in the legal domain. The results of the assessment showed that the implemented methods were able to find similar texts. But the methods also suggested not relevant texts. In the assessment no implemented method could constantly outperform the other methods.

Figure 30 shows the components of LEXIA at the end of this thesis. In comparison to the state at the beginning of this thesis (see chapter 1.2) the *Similarity Engine* was added. It interacts

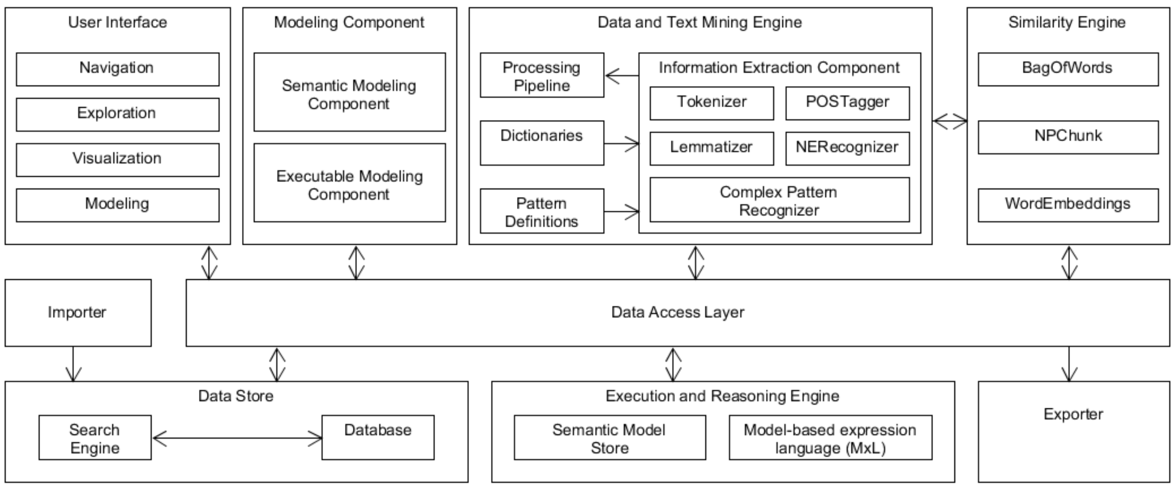


Figure 30: New LEXIA components
Source: Based on [4]

with the *Data and Text Mining Engine* and the *Data Access Layer*.

Further progress for LEXIA could be made into different directions. One possibility is to tackle the issues with the processing time. Faster processing could be achieved by introducing parallelization and distributing tasks across several machines. The implemented architecture already divides the process into tasks that are independent apart from a few needed synchronization points. Another way to boost performance might be the use of another database system for certain data. Reducing the processing time would enable the users of LEXIA to compute the similarities for bigger datasets which might lead to more meaningful results.

Another way of advancing LEXIA lies in the implementation of further similarity methods. Maybe more sophisticated techniques deliver better results than the ones implemented in this thesis. These new methods would need to be evaluated again. But also with the results of the assessment in this thesis one could improve the already existing methods. By adjusting parameters one could try to find all relevant texts. But one must be careful that no overfitting occurs.

So hopefully this thesis can provide a useful foundation for further development of a relatedness search in the legal domain.

Appendix

Functional assessment questionnaire

Evaluation einer Ähnlichkeitssuche über Gesetze

Im Rahmen von Forschungsarbeiten an der TU München & Lexalyze wird die Leistungsfähigkeit computergestützter Methoden für den Bereich semantischer Analyse von Gesetzen, Urteilen und Kommentare untersucht. Ein Teilaspekt ist die Analyse von Ähnlichkeiten und Abhängigkeiten zwischen Normen. Hierzu wurden verschiedene computergestützte Verfahren entwickelt. Diese berechnen, basierend auf unterschiedlichen analytischen und linguistischen Verfahren, Ähnlichkeiten und semantische Abhängigkeiten zwischen Gesetzesnormen (Paragraphen).

Um die Güte der Verfahren bewerten und vergleichen zu können, bitten wir Sie, sich kurz für deren Evaluation Zeit zu nehmen. Es werden Ihnen Vorschläge zu 4 Normen (BGB §§ 280, 434, 985, StGB § 242) angezeigt, deren Relevanz Sie in Bezug auf die Ausgangsnorm bewerten sollen.

Die Evaluation dauert ca. 15 - 20 Minuten. Falls Sie sich zu einer Norm nicht äußern wollen, können Sie diese auch einfach freilassen.

1. BGB §280 Schadensersatz wegen Pflichtverletzung

Bewertung von 1 (nicht relevant) bis 4 (sehr relevant)

	1	2	3	4	k.A.
§ 282 Schadensersatz statt der Leistung wegen Verletzung einer Pflicht nach § 241 Abs. 2	□	□	□	□	□
§ 281 Schadensersatz statt der Leistung wegen nicht oder nicht wie geschuldet erbrachter Leistung	□	□	□	□	□
§ 326 Befreiung von der Gegenleistung und Rücktritt beim Ausschluss der Leistungspflicht	□	□	□	□	□
§ 1243 Rechtswidrige Veräußerung	□	□	□	□	□
§ 634 Rechte des Bestellers bei Mängeln	□	□	□	□	□
§ 1165 Freiwerden des Schuldners	□	□	□	□	□
§ 283 Schadensersatz statt der Leistung bei Ausschluss der Leistungspflicht	□	□	□	□	□
§ 437 Rechte des Käufers bei Mängeln	□	□	□	□	□
§ 78 Festsetzung von Zwangsgeld	□	□	□	□	□
§ 312a Verhältnis zu anderen Vorschriften	□	□	□	□	□
§ 284 Ersatz vergeblicher Aufwendungen	□	□	□	□	□
§ 289 Zinseszinsverbot	□	□	□	□	□
§ 867 Verfolgungsrecht des Besitzers	□	□	□	□	□
§ 825 Bestimmung zu sexuellen Handlungen	□	□	□	□	□
§ 678 Geschäftsführung gegen den Willen des Geschäftsherrn	□	□	□	□	□
§ 536c Während der Mietzeit auftretende Mängel; Mängelanzeige durch den Mieter	□	□	□	□	□
§ 823 Schadensersatzpflicht	□	□	□	□	□
§ 675 Entgeltliche Geschäftsbesorgung	□	□	□	□	□
§ 628 Teilvergütung und Schadensersatz bei fristloser Kündigung	□	□	□	□	□
§ 341 Strafversprechen für nicht gehörige Erfüllung	□	□	□	□	□
§ 340 Strafversprechen für Nichterfüllung	□	□	□	□	□
§ 323 Rücktritt wegen nicht oder nicht vertragsgemäß erbrachter Leistung	□	□	□	□	□
§ 285 Herausgabe des Ersatzes	□	□	□	□	□
§ 267 Leistung durch Dritte	□	□	□	□	□
§ 275 Ausschluss der Leistungspflicht	□	□	□	□	□
§ 380 Nachweis der Empfangsberechtigung	□	□	□	□	□
§ 311a Leistungshindernis bei Vertragsschluss	□	□	□	□	□
§ 808 Namenspapiere mit Inhaberklausel	□	□	□	□	□
§ 432 Mehrere Gläubiger einer unteilbaren Leistung	□	□	□	□	□
§ 1281 Leistung vor Fälligkeit	□	□	□	□	□

Gibt es weitere Paragraphen, die Sie in Verbindung mit BGB §280 für relevant halten?

____

____

____

2. BGB §434 Sachmangel

Bewertung von 1 (nicht relevant) bis 4 (sehr relevant)

	1	2	3	4	k.A.
§ 433 Vertragstypische Pflichten beim Kaufvertrag	☐	☐	☐	☐	☐
§ 243 Gattungsschuld	☐	☐	☐	☐	☐
§ 445 Haftungsbegrenzung bei öffentlichen Versteigerungen	☐	☐	☐	☐	☐
§ 2183 Haftung für Sachmängel	☐	☐	☐	☐	☐
§ 476 Beweislastumkehr	☐	☐	☐	☐	☐
§ 697 Rückgabeort	☐	☐	☐	☐	☐
§ 949 Erlöschen von Rechten Dritter	☐	☐	☐	☐	☐
§ 1042 Anzeigepflicht des Nießbrauchers	☐	☐	☐	☐	☐
§ 694 Schadensersatzpflicht des Hinterlegers	☐	☐	☐	☐	☐
§ 966 Verwahrungspflicht	☐	☐	☐	☐	☐
§ 633 Sach- und Rechtsmangel	☐	☐	☐	☐	☐
§ 997 Wegnahmerecht	☐	☐	☐	☐	☐
§ 312b Fernabsatzverträge	☐	☐	☐	☐	☐
§ 957 Gestattung durch den Nichtberechtigten	☐	☐	☐	☐	☐
§ 1044 Duldung von Ausbesserungen	☐	☐	☐	☐	☐
§ 702a Erlass der Haftung	☐	☐	☐	☐	☐
§ 932 Gutgläubiger Erwerb vom Nichtberechtigten	☐	☐	☐	☐	☐
§ 443 Beschaffenheits- und Haltbarkeitsgarantie	☐	☐	☐	☐	☐
§ 934 Gutgläubiger Erwerb bei Abtretung des Herausgabeanspruchs	☐	☐	☐	☐	☐
§ 648a Bauhandwerkersicherung	☐	☐	☐	☐	☐
§ 357 Rechtsfolgen des Widerrufs und der Rückgabe	☐	☐	☐	☐	☐
§ 2109 Unwirksamwerden der Nacherbschaft	☐	☐	☐	☐	☐
§ 1905 Sterilisation	☐	☐	☐	☐	☐
§ 438 Verjährung der Mängelansprüche	☐	☐	☐	☐	☐
§ 495 Widerrufsrecht	☐	☐	☐	☐	☐
§ 81 Stiftungsgeschäft	☐	☐	☐	☐	☐
§ 309 Klauselverbote ohne Wertungsmöglichkeit	☐	☐	☐	☐	☐
§ 595 Fortsetzung des Pachtverhältnisses	☐	☐	☐	☐	☐
§ 308 Klauselverbote mit Wertungsmöglichkeit	☐	☐	☐	☐	☐

Gibt es weitere Paragraphen, die Sie in Verbindung mit BGB §434 für relevant halten?

____

____

____

3. BGB §985 Herausgabeanspruch

Bewertung von 1 (nicht relevant) bis 4 (sehr relevant)

	1	2	3	4	k.A.
§ 986 Einwendungen des Besitzers	□	□	□	□	□
§ 1007 Ansprüche des früheren Besitzers, Ausschluss bei Kenntnis	□	□	□	□	□
§ 988 Nutzungen des unentgeltlichen Besitzers	□	□	□	□	□
§ 1000 Zurückbehaltungsrecht des Besitzers	□	□	□	□	□
§ 850 Ersatz von Verwendungen	□	□	□	□	□
§ 1001 Klage auf Verwendungsersatz	□	□	□	□	□
§ 1006 Eigentumsvermutung für Besitzer	□	□	□	□	□
§ 1005 Verfolgungsrecht	□	□	□	□	□
§ 931 Abtretung des Herausgabeanspruchs	□	□	□	□	□
§ 935 Kein gutgläubiger Erwerb von abhanden gekommenen Sachen	□	□	□	□	□
§ 2185 Ersatz von Verwendungen und Aufwendungen	□	□	□	□	□
§ 869 Ansprüche des mittelbaren Besitzers	□	□	□	□	□
§ 1100 Rechte des Käufers	□	□	□	□	□
§ 904 Notstand	□	□	□	□	□
§ 2185 Ersatz von Verwendungen und Aufwendungen	□	□	□	□	□
§ 1002 Erlöschen des Verwendungsanspruchs	□	□	□	□	□
§ 2362 Herausgabe- und Auskunftsanspruch des wirklichen Erben	□	□	□	□	□
§ 994 Notwendige Verwendungen	□	□	□	□	□

Gibt es weitere Paragraphen, die Sie in Verbindung mit BGB §985 für relevant halten?

____

____

____

4. StGB §242 Diebstahl

Bewertung von 1 (nicht relevant) bis 4 (sehr relevant)

	1	2	3	4	k.A.
§ 303 Sachbeschädigung	☐	☐	☐	☐	☐
§ 289 Pfandkehr	☐	☐	☐	☐	☐
§ 246 Unterschlagung	☐	☐	☐	☐	☐
§ 259 Hehlerei	☐	☐	☐	☐	☐
§ 293 Fischwilderei	☐	☐	☐	☐	☐
§ 249 Raub	☐	☐	☐	☐	☐
§ 324 Gewässerverunreinigung	☐	☐	☐	☐	☐
§ 223 Körperverletzung	☐	☐	☐	☐	☐
§ 317 Störung von Telekommunikationsanlagen	☐	☐	☐	☐	☐
§ 167a Störung einer Bestattungsfeier	☐	☐	☐	☐	☐
§ 265 Versicherungsmißbrauch	☐	☐	☐	☐	☐
§ 248c Entziehung elektrischer Energie	☐	☐	☐	☐	☐
§ 223 Körperverletzung	☐	☐	☐	☐	☐
§ 107c Verletzung des Wahlgeheimnisses	☐	☐	☐	☐	☐
§ 253 Erpressung	☐	☐	☐	☐	☐
§ 108 Wählernötigung	☐	☐	☐	☐	☐

Gibt es weitere Paragraphen, die Sie in Verbindung mit StGB §242 für relevant halten?

____

____

____

Functional assessment results

BGB § 280

Method: *BagOfWords*

Similarity Search Result	**Average Relevance Rating**
§ 282 Schadensersatz statt der Leistung wegen Verletzung einer Pflicht nach § 241 Abs. 2	3,45
§ 281 Schadensersatz statt der Leistung wegen nicht oder nicht wie geschuldet erbrachter Leistung	3,82
§ 326 Befreiung von der Gegenleistung und Rücktritt beim Ausschluss der Leistungspflicht	2,45
§ 1243 Rechtswidrige Veräußerung	1,40
§ 634 Rechte des Bestellers bei Mängeln	3,09
§ 1165 Freiwerden des Schuldners	1,18
§ 283 Schadensersatz statt der Leistung bei Ausschluss der Leistungspflicht	3,82
§ 437 Rechte des Käufers bei Mängeln	3,64
§ 78 Festsetzung von Zwangsgeld	1,27
§ 312a Verhältnis zu anderen Vorschriften	1,55

Method: *NPChunk*

Similarity Search Result	**Average Relevance Rating**
§ 284 Ersatz vergeblicher Aufwendungen	3,09
§ 289 Zinseszinsverbot	2,00
§ 867 Verfolgungsrecht des Besitzers	1,09
§ 825 Bestimmung zu sexuellen Handlungen	1,18
§ 678 Geschäftsführung gegen den Willen des Geschäftsherrn	1,73
§ 536c Während der Mietzeit auftretende Mängel; Mängelanzeige durch den Mieter	2,09
§ 823 Schadensersatzpflicht	2,18
§ 675 Entgeltliche Geschäftsbesorgung	1,91
§ 628 Teilvergütung und Schadensersatz bei fristloser Kündigung	1,64

Method: *WordEmbeddings*

Similarity Search Result	**Average Relevance Rating**
§ 282 Schadensersatz statt der Leistung wegen Verletzung einer Pflicht nach § 241 Abs. 2	3,45
§ 281 Schadensersatz statt der Leistung wegen nicht oder nicht wie geschuldet erbrachter Leistung	3,82
§ 341 Strafversprechen für nicht gehörige Erfüllung	1,55

§ 326 Befreiung von der Gegenleistung und Rücktritt beim Ausschluss der Leistungspflicht	2,45
§ 340 Strafversprechen für Nichterfüllung	1,64
§ 323 Rücktritt wegen nicht oder nicht vertragsgemäß erbrachter Leistung	3,09
§ 285 Herausgabe des Ersatzes	3,00
§ 267 Leistung durch Dritte	1,45
§ 275 Ausschluss der Leistungspflicht	3,09
§ 380 Nachweis der Empfangsberechtigung	1,30

Method: *MoreLikeThis*

Similarity Search Result	**Average Relevance Rating**
§ 283 Schadensersatz statt der Leistung bei Ausschluss der Leistungspflicht	3,82
§ 282 Schadensersatz statt der Leistung wegen Verletzung einer Pflicht nach § 241 Abs. 2	3,45
§ 281 Schadensersatz statt der Leistung wegen nicht oder nicht wie geschuldet erbrachter Leistung	3,82
§ 311a Leistungshindernis bei Vertragsschluss	2,91
§ 808 Namenspapiere mit Inhaberklausel	1,18
§ 285 Herausgabe des Ersatzes	3,00
§ 267 Leistung durch Dritte	1,45
§ 432 Mehrere Gläubiger einer unteilbaren Leistung	1,45
§ 1281 Leistung vor Fälligkeit	1,36

BGB § 434

Method: *BagOfWords*

Similarity Search Result	Average Relevance Rating
§ 433 Vertragstypische Pflichten beim Kaufvertrag	3,82
§ 243 Gattungsschuld	2,82
§ 445 Haftungsbegrenzung bei öffentlichen Versteigerungen	2,36
§ 2183 Haftung für Sachmängel	2,45
§ 476 Beweislastumkehr	3,55
§ 697 Rückgabeort	1,45
§ 949 Erlöschen von Rechten Dritter	1,36
§ 1042 Anzeigepflicht des Nießbrauchers	1,30
§ 694 Schadensersatzpflicht des Hinterlegers	1,40
§ 966 Verwahrungspflicht	1,40

Method: *NPChunk*

Similarity Search Result	Average Relevance Rating
§ 633 Sach- und Rechtsmangel	3,18
§ 997 Wegnahmerecht	1,30
§ 312b Fernabsatzverträge	2,00

Method: *WordEmbeddings*

Similarity Search Result	Average Relevance Rating
§ 957 Gestattung durch den Nichtberechtigten	3,82
§ 633 Sach- und Rechtsmangel	2,82
§ 1044 Duldung von Ausbesserungen	2,36
§ 702a Erlass der Haftung	2,45
§ 997 Wegnahmerecht	3,55
§ 932 Gutgläubiger Erwerb vom Nichtberechtigten	1,45
§ 443 Beschaffenheits- und Haltbarkeitsgarantie	1,36
§ 934 Gutgläubiger Erwerb bei Abtretung des Herausgabean-spruchs	1,30
§ 648a Bauhandwerkersicherung	1,40
§ 357 Rechtsfolgen des Widerrufs und der Rückgabe	1,40

Method: *MoreLikeThis*

Similarity Search Result	**Average Relevance Rating**
§ 2109 Unwirksamwerden der Nacherbschaft	1,18
§ 1905 Sterilisation	1,09
§ 438 Verjährung der Mängelansprüche	3,64
§ 495 Widerrufsrecht	1,73
§ 81 Stiftungsgeschäft	1,20
§ 309 Klauselverbote ohne Wertungsmöglichkeit	2,27
§ 633 Sach- und Rechtsmangel	3,18
§ 595 Fortsetzung des Pachtverhältnisses	1,20
§ 308 Klauselverbote mit Wertungsmöglichkeit	2,09

BGB § 985

Method: *BagOfWords*

Similarity Search Result	**Average Relevance Rating**
§ 986 Einwendungen des Besitzers	3,90
§ 1007 Ansprüche des früheren Besitzers, Ausschluss bei Kenntnis	3,40
§ 988 Nutzungen des unentgeltlichen Besitzers	3,70
§ 1000 Zurückbehaltungsrecht des Besitzers	3,60
§ 850 Ersatz von Verwendungen	2,20
§ 1001 Klage auf Verwendungsersatz	3,10
§ 1006 Eigentumsvermutung für Besitzer	3,50
§ 1005 Verfolgungsrecht	2,50
§ 931 Abtretung des Herausgabeanspruchs	2,50
§ 935 Kein gutgläubiger Erwerb von abhanden gekommenen Sachen	3,00

Method: *WordEmbeddings*

Similarity Search Result	**Average Relevance Rating**
§ 986 Einwendungen des Besitzers	3,90
§ 1001 Klage auf Verwendungsersatz	3,10
§ 850 Ersatz von Verwendungen	2,20
§ 931 Abtretung des Herausgabeanspruchs	2,50
§ 1007 Ansprüche des früheren Besitzers, Ausschluss bei Kenntnis	3,40
§ 988 Nutzungen des unentgeltlichen Besitzers	3,70
§ 2185 Ersatz von Verwendungen und Aufwendungen	1,80
§ 869 Ansprüche des mittelbaren Besitzers	2,10
§ 1100 Rechte des Käufers	1,30
§ 904 Notstand	1,50

Method: *MoreLikeThis*

Similarity Search Result	**Average Relevance Rating**
§ 986 Einwendungen des Besitzers	3,90
§ 2185 Ersatz von Verwendungen und Aufwendungen	1,60
§ 1007 Ansprüche des früheren Besitzers, Ausschluss bei Kenntnis	3,40
§ 850 Ersatz von Verwendungen	2,20
§ 1002 Erlöschen des Verwendungsanspruchs	2,50

§ 2362 Herausgabe- und Auskunftsanspruch des wirklichen Erben	1,80
§ 1001 Klage auf Verwendungsersatz	3,10
§ 931 Abtretung des Herausgabeanspruchs	2,50
§ 994 Notwendige Verwendungen	3,60

StGB § 242

Method: *BagOfWords*

Similarity Search Result	Average Relevance Rating
§ 303 Sachbeschädigung	2,10
§ 289 Pfandkehr	2,30
§ 246 Unterschlagung	3,90
§ 259 Hehlerei	3,20
§ 293 Fischwilderei	1,90
§ 249 Raub	3,80
§ 324 Gewässerverunreinigung	1,30
§ 223 Körperverletzung	1,40
§ 317 Störung von Telekommunikationsanlagen	1,10
§ 167a Störung einer Bestattungsfeier	1,10

Method: *NPChunk*

Similarity Search Result	Average Relevance Rating
§ 246 Unterschlagung	3,90
§ 289 Pfandkehr	2,30
§ 249 Raub	3,80

Method: *WordEmbeddings*

Similarity Search Result	Average Relevance Rating
§ 249 Raub	3,80
§ 265 Versicherungsmißbrauch	1,30
§ 293 Fischwilderei	1,90
§ 248c Entziehung elektrischer Energie	3,40
§ 246 Unterschlagung	3,90
§ 303 Sachbeschädigung	2,10
§ 289 Pfandkehr	2,30
§ 223 Körperverletzung	1,40
§ 107c Verletzung des Wahlgeheimnisses	1,10
§ 317 Störung von Telekommunikationsanlagen	1,10

Method: *MoreLikeThis*

Similarity Search Result	**Average Relevance Rating**
§ 249 Raub	3,80
§ 246 Unterschlagung	3,90
§ 248c Entziehung elektrischer Energie	3,40
§ 289 Pfandkehr	2,30
§ 259 Hehlerei	3,20
§ 265 Versicherungsmißbrauch	1,30
§ 303 Sachbeschädigung	2,10
§ 253 Erpressung	2,40
§ 108 Wählernötigung	1,11

Bibliography

[1] Statistisches Bundesamt, Statistisches Jahrbuch 2015, Wiesbaden.

[2] juris GmbH, "juris.de," [Online]. Available: http://www.juris.de. [Accessed 20 06 2016].

[3] U. Wesel and H. D. Beck, 250 Jahre rechtswissenschaftlicher Verlag C.H.Beck: 1763-2013, C.H.Beck, 2015.

[4] B. Waltl, F. Matthes, T. Waltl and T. Grass, "LEXIA - A Data Science Environment for Semantic Analysis of German Legal Texts," in *IRIS: Internationales Rechtsinformatik Symposium*, Salzburg, 2016.

[5] The Apache Software Foundation, "Apache UIMA," [Online]. Available: https://uima.apache.org. [Accessed 15 09 2016].

[6] The Apache Software Foundation, "Apache UIMA Ruta," [Online]. Available: https://uima.apache.org/ruta.html. [Accessed 06 11 2016].

[7] W. Zikmund, B. Babin, J. Carr and M. Griffin, Business Research Methods, Cengage Learning, 2013.

[8] J. W. Tukey, Exploratory Data Analysis, Addison-Wesley Publishing Company, 1977.

[9] T. Hill, P. Lewicki and P. Lewicki, Statistics: methods and applications: a comprehensive reference for science, industry, and data mining, StatSoft, Inc., 2006.

[10] DATEV eG, "DATEV," [Online]. Available: www.datev.de. [Accessed 23 August 2016].

[11] Z. S. Harris, "Distributional structure," *Word,* pp. 146-162, 1954.

[12] G. Salton and M. McGill, Introduction to modern information retrieval, New York: McGraw-Hill, 1983.

[13] M. Porter, "An algorithm for suffix stripping," *Program,* pp. 130-137, 1980.

[14] W. Stock, Information Retrieval, München: Oldenbourg, 2007.

[15] C. Corley and R. Mihalcea, "Measuring the semantic similarity of texts," *Proceedings of the ACL workshop on empirical modeling of semantic equivalence and entailment,*

pp. 13-18, 2005.

[16] B. Fortuna, M. Grobelnik and D. Mladenic, "Visualization of text document corpus," *Informatica,* 2005.

[17] G. Kondrak, "N-gram similarity and distance," in *International Symposium on String Processing and Information Retrieval,* Springer, 2005, pp. 115-126.

[18] M. Levandowsky and D. Winter, "Distance between sets," *Nature,* pp. 34-35, 1971.

[19] W. Cavnar and J. Trenkle, "N-gram-based text categorization," *Ann Arbor MI,* pp. 161-175, 1994.

[20] G. Lame, "Using NLP techniques to identify legal ontology components: concepts and relations," in *Law and the Semantic Web,* Springer, 2005, pp. 169-184.

[21] C. Dozier, R. Kondadadi, M. Light, A. Vachher, S. Veeramachaneni and R. Wudali, "Named entity recognition and resolution in legal text," in *Semantic Processing of Legal Texts,* Springer, 2010, pp. 27-43.

[22] E. Schweighofer, "Semantic indexing of legal documents," in *Semantic Processing of Legal Texts,* Springer, 2010, pp. 157-169.

[23] G. Hinton, "Learning distributed representations of concepts," *Proceedings of the eighth annual conference of the cognitive science society,* 1986.

[24] Y. Bengio, R. Ducharme, P. Vincent and C. Jauvin, "A neural probabilistic language model," *journal of machine learning research,* pp. 1137-1155, Februar 2003.

[25] T. Mikolov, K. Chen, G. Corrado and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781,* 2013.

[26] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems,* pp. 3111-3119, 2013.

[27] R. Winkels, A. Boer, B. Vredebregt and A. van Someren, "Towards a Legal Recommender System," in *Jurix,* 2014.

[28] D. Heckerman, "Bayesian networks for data mining," *Data mining and knowledge discovery,* pp. 79-119, 1997.

[29] H. Turtle and B. Croft, "Inference networks for document retrieval," in *Proceedings of the 13th annual international ACM SIGIR conference on Research and development in*

information retrieval, 1989, pp. 1-24.

[30] H. Turtle and B. Croft, "Evaluation of an inference network-based retrieval model," *ACM Transactions on Information Systems (TOIS),* pp. 187-222, 1991.

[31] D. Blei, "Probabilistic topic models," *Communications of the ACM,* pp. 77-84, 2012.

[32] D. Blei, A. Ng and M. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research,* pp. 993-1022, 2003.

[33] H. Wallach, "Topic modeling: beyond bag-of-words," in *Proceedings of the 23rd international conference on Machine learning,* ACM, 2006, pp. 977-984.

[34] D. Blei and J. Lafferty, "Dynamic topic models," in *Proceedings of the 23rd international conference on Machine learning,* ACM, 2006, pp. 113-120.

[35] J. Lin and J. Wilbur, "PubMed related articles: a probabilistic topic-based model for content similarity," *BMC bioinformatics,* 2007.

[36] X. Quan, G. Liu, Z. Lu, X. Ni and L. Wenyin, "Short text similarity based on probabilistic topics," *Knowledge and information systems,* pp. 473-491, 2010.

[37] J. S. Greenfield, Distributed programming paradigms with cryptography applications, Springer Science & Business Media, 1994.

[38] N. Chomsky, Syntactic structures, Walter de Gruyter, 2002.

[39] Elasticsearch, "BulkAPI | Elasticsearch," [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/guide/current/bulk.html. [Accessed 28 10 2016].

[40] Elasticsearch, "Delete by Query | Elasticsearch," [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/plugins/2.0/delete-by-query-usage.html. [Accessed 03 11 2016].

[41] Elasticsearch, "More Like This Query | Elasticsearch," [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/2.0/query-dsl-mlt-query.html. [Accessed 06 11 2016].

[42] The Apache Software Foundation, "TFIDF Similarity (Lucene API)," [Online]. Available: https://lucene.apache.org/core/4_9_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html. [Accessed 07 11 2016].

[43] Elasticsearch, "Segment Merging | Elasticsearch," [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/guide/current/merge-process.html. [Accessed 10 11 2016].

[44] G. Salton, A. Wong and C.-S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, pp. 613-620, 1975.