TECHNISCHE UNIVERSITÄT DRESDEN

Dep. of Computer Science Institute for System Architecture, Database Technology Group

Study Thesis

WORD EMBEDDING-BASED PATENT RETRIEVAL

Hendrik Cziommer

Matr.-Nr.: 3917922

Supervised by:

Prof. Dr.-Ing. Wolfgang Lehner

and:

Dr.-Ing. Maik Thiele

Submitted on 11 May 2019

CONFIRMATION

I confirm that I independently prepared the thesis and that I used only the references and auxiliary means indicated in the thesis.

Dresden, 11 May 2019

ABSTRACT

Information Retrieval (IR) is for companies more relevant than ever. The aim of the study was to investigate the use of word embddings in the domain of IR, in particular, patent retrieval. Word embedding methods such as Sent2Vec and Word2Vec were compared to traditional IR methods. A prototype using Sent2Vec was presented. The experiments demonstrated the improved performance of Sent2Vec in several tasks. This indicates an approach based on semantics yields better results than traditional tf-idf based methods.

CONTENTS

1 INTRODUCTION

Information Retrieval (IR) [MRS08] is for companies more relevant than ever as more and more data is collected in the form of e-mails, issue tickets, internal documents, presentations, etc.... For an efficient work process of employees, it is indispensable to find relevant text documents within these large collections. Hence, search systems have to be easy to interact with, fast and the retrieved documents have to be highly relevant. Traditional IR systems such as tf-idf [SJ72] or BM25 [RZ$^+$09] have served well in the past to cope with these large amounts of data, but as they rely on matching words, they come with natural limitations.

For example, queries for documents about "airplane" should also return documents containing synonymous terms like "aircraft" or "airliner". Query expansion methods [Eft96] such as a thesaurus dictionaries or elaborately created ontologies improve the quality of search results but may be incomplete and hard to maintain due to their manual creation. A promising approach are *Word Embeddings* which come with very desirable attributes, including natural handling of synonyms.

Word Embeddings have become popular lately due to their good performance in natural language processing tasks like automatic speech recognition and machine translation [Sch07, Mik12]. They model semantic relationships hidden in natural language by representing words as dense vectors. Word Embeddings are generated by training a shallow neural network which results in dense representations of each word's natural context as multi-dimensional feature vectors. Therefore, words with similar contexts have a shorter cosine distance in the vector space to each other then words in different contexts. This makes sense because e.g. synonyms appear in the same context. Two popular trainings methods for Word Embeddings are Word2Vec [1] and Fasttext [BGJM17] which also includes sub words to improve the quality of Word Embeddings.

For our purpose of retrieving documents, we have to expand the idea Word Embeddings because they only embed single words and no documents. A more generalized version of Word2Vec is Doc2Vec [LM14], as it was designed to model the meaning of whole documents. Unfortunately, the vector space of Doc2Vec is limited to documents and therefore, not applicable.

Another approach to embed documents is to average all word vectors of a document to obtain a document vector resulting in a vector space with word and document vectors. Queries can now

[1]https://code.google.com/archive/p/word2vec/

contain both words documents, which is desirable for an IR system. This concept can be further enhanced by using Sent2Vec [PGJ18], that is based on training special word vectors, which are optimized for averaging over sentences or documents.

The target of IR is to find documents consisting of text that satisfies an information need within large collections [MRS08]. The approach of word and document embeddings using e.g. Sent2Vec can be applied intuitively to IR tasks on account of its modeling of concepts behind the words and documents in vector space where similarity is measured by the cosine distance.

Our system will briefly look like this: A query in form of a sequence of words and documents has to be translated into the embedding space. To obtain the final query vector all query vectors are averaged. In the last step, the nearest vectors have to be calculated, and the corresponding documents have to be returned, respectively.

In this work we will outline major benefits compared to traditional IR systems such as standard *Vector Space Model* [Sal62] or *BM25* [RZ$^+$09]. These improvements are based on the decoupling of exact string matching as it searches for concepts and does not rely on query expansion methods to include synonyms in the search and therefore, increases the quality of the search results.

Background

This part will provide a short background for our thesis, which was created in cooperation with the Dresden, Germany based company interface:projects.[2] interface:projects main product is an enterprise search engine called integator [3].

Primary value proposition of intergator is convenient access to internal company data. While selling additionally web search solutions, strong focus is on optimal support of enterprise search in all its specifics. The software is composed of multiple components, partially known also from other application areas of IR such as generic web search, digital library or e-commerce search facilities.

In order to achieve better search results, an approach based on Word Embedding has been under development since 2017. This is necessary due to new customers with new requirements. The major drive for this approach is the German Patent and Trade Mark Office[4] (DPMA). One of DPMA main task is the processing of patent applications. Patents come with special problems compared to other domains like news articles or Wikipedia's articles which we will outline in the following chapters in more detail.

[2]interface projects GmbH, Zwinglistrasse 11/13, 01277 Dresden, Germany
[3]https://www.intergator.de/
[4]https://www.dpma.de/

Contribution

In this work we compare a word embedding based IR approach to a traditional IR system and show the following improvements:

1. **Semantic Component** We show that our system introduces a semantic component to the IR system. This means that the system will recognize relevant documents as relevant even when there is no exact string matching between the query and the document.
2. **Improved Recall** Improving recall is always a key goal in IR. Therefore, we will evaluate the word embeddings system against a tf-idf-like retrieval with the standard metric of recall.
3. **Disambiguity** Queries sometimes can be ambiguous. For example, "wing" could refer to a "political wing" or a "bird wing". In this work, we show an approach to make it easy for a user to give feedback and disambiguate such queries.

Outline

In the following chapter, we will give an overview of our foundations, including an introduction to Word Embeddings, traditional IR and dimensionality reduction techniques. Afterwards, we will outline the problems related to patents. In Chapter 4 we describe the concept of our system. Next, we talk about the implementation in detail and explain all used libraries. In Chapter 7, we evaluate the system and check our hypothesis described above. Finally, we conclude our work and give an outlook for further improvements.

2 FOUNDATIONS

The *Vector Space Model (VSM)* [Sal62] is a popular IR technique for document retrieval, which is based on the idea of representing documents as vectors. To retrieve documents, a query is translated into the vector space and the n most cosine similar document vectors are retrieved as depicted in Figure 2.1. In this chapter, we introduce algorithms that have been proven over many years to create VSMs, as well as new ones that have attracted a lot of attention lately, in order to improve recall in the domain of patents. Furthermore, we present techniques to reduce VSMs with several hundred dimension for visualizations that can be grasped by humans.

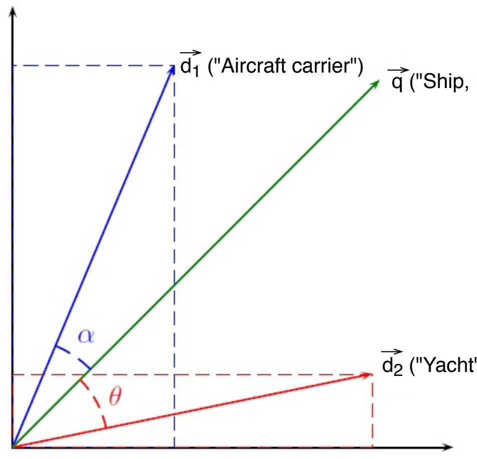


Figure 2.1: Vector space model with two patent document vectors and a query vector: The document "Aircraft Carrier" is more relevant for the query of "Ship, Aircraft"' than "Yacht", which is expressed by the smaller angle of α compared to θ. Returned documents are sorted by angle in ascending order.

2.1 DISTRIBUTED REPRESENTATION OF WORDS AND DOCUMENTS IN VECTOR SPACE

Word Embeddings

Word Embeddings have recently attracted a lot of attention due to their good performance in a variety of natural language processing (NLP) and IR tasks. The basic idea is to project each individual word in a text corpus into a low-dimensional feature vector of fixed size, so that vectors with semantically similar meanings have a higher cosine similarity compared to vectors representing words with different meanings resulting in a VSM.

For example, we can expect that the vector representation of the word "cat" is more similar to the vector of "kitten" than "dog" or "car". Furthermore, this representation performs well in analogical reasoning tasks as shown by Mikolov et al. [MCCD13]. This task includes to complete analogies like King:Queen=Man:? To determine the missing word we calculate: $vector('Queen') - vector('King') + vector('Man') \approx vector('Woman')$ which means that the resulting vector is closer to "Woman" than to any other vector. The vectors and relationships are depicted in Figure 2.2.

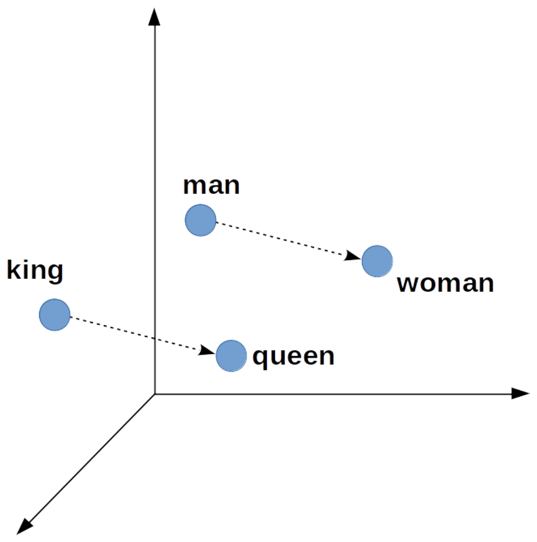


Figure 2.2: Word Embeddings

Various methods can be applied to determine the VSM, but recently, neural networks have become appealing. The popular algorithm Word2Vec was introduced by Mikolov et al. [MCCD13] in 2013. Word2Vec uses a shallow neural network for training, which is simpler and has a lower computational complexity compared to other neural network architectures such as recurrent and feed-forward neural networks.

Two proposals of architectures for training, Skip-Gram and CBOW, have received much attention due to their good performance in semantic and syntactic word relationship tests.

Skip-Gram Architecture

Skip-gram [MSC+13] is an architecture which approaches the task of learning semantic word vectors by their ability to predict vectors of their surrounding words.

Given a word w_i and a sequence of words $w_{i-c}, ..., w_{i-3}, w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2}, w_{i+3}, ..., w_{i+c}$ where i is position of the word in the text and c is the context size.

Formally, the objective of the Skip-gram model is to maximize the following average logarithmic probability.

$$\frac{1}{ws} \sum_{i=1}^{ws} \sum_{-ws \leq j \leq ws, j \neq 0} \log p(w_{t+j}, w)$$

Please note that a hyper-parameter for the window size (ws) is given and context size (c) is randomly chosen with $1 \leq c \leq ws$. By doing so, we give more distant words less weight, which makes intuitively sense.

According to Mikolov et al. [MSC+13] this approach is best suited for small amounts of training data and represents even rare words or phrases well.

Continuous Bag-of-Words Architecture

On the other hand the idea of Continuous Bag-of-Words Mode (CBOW) [MCCD13] tackles the task by using the context words to predict the vector of the target word. Opposed to Skip-Gram CBOW predicts a word from its context.

Given the context of a word as a sequence of words: $w_{i-c}, ..., w_{i-3}, w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2}, w_{i+3}, ..., w_{i+c}$ with context size (c) and the target word w_i. The objective is to maximize the likelihood of correctly predicting the target word w_i. According to Mikolov et al. [MSC+13] this approaches is particularly suitable for large corpora due its faster training and also better accuracy for the frequent words. We choose the CBOW over the Skip-gram architecture because we are dealing with large corpora.

Sentence and Document Embeddings

Sentence embeddings (Sent2Vec) [PGJ18] are used to capture the meaning of sentences in one feature vector. Sent2Vec can conceptually be seen as a natural extension to word embeddings, focusing on the efficient embedding of word sequences instead of single words. Therefore, we train a word embedding model, which is especially optimized towards additive combination over the sentence. Every sentence embedding is defined as the average of the source embeddings of its individual words. The model training is further augmented by not only using single words (unigrams) as in Word2Vec's CBOW for predicting a word given a context but also n-grams present in each sentence and averaging the n-gram embeddings along with the words. A sentence embedding v_S given a sentence S is calculated as follows:

$$v_S := \frac{1}{R(S)} \sum_{w \in R(S)} v_w$$

where $R(S)$is the list of n-grams (including unigrams) present in sentence S. Negative sampling is used for the training due to its efficiency [LG14].

2.2 DIMENSIONALITY REDUCTION

Word embedding VSMs usually have several hundred dimensions. Due to the fact that humans cannot comprehend vectors with several hundred dimensions and there is not an simple way to visualize them in an appealing way, we have to lower their dimensionality. Fortunately, there are various approaches for reducing high-dimensional vectors in order to preserve as much information as possible. In this section we introduce the well-known PCA algorithm as well as the novel t-SNE algorithm briefly and explain the usage of PCA over t-SNE.

Principal Component Analysis

First, we give a short introduction to Principal Component Analysis (PCA) which was first introduced in 1901 by Karl Pearson [Pea01] and evolved by Harold Hotelling [Hot33] in 1933. PCA was first developed for the field of biology but can be used in any field to reduce vector dimensionality. PCA's objective is to reduce the dimensionality in such a way that the variance in the resulting space is the highest in the first component, second highest in the second component and so on. The intuition is to capture the data points with the largest distance between each other in the first component, because this provides the most information. For illustration we show in Figure 2.3 the mean variance (e.g. the informativeness) of the first 50 components of 1000 randomly sampled reduced vectors from Google's news word embeddings model[1]. We note that the variance decreases exponentially. In essence PCA is an orthogonal linear transformation which transforms the vector space such that the components of the vectors decrease in their variance as shown in Figure 2.3.

Time complexity as implemented in scikit-learn[2] is $O(n_{max}^2 \times n_{min})$ with $n_{max} = max(n_{samples}, n_{features})$ and $n_{min} = min(n_{samples}, n_{features})$ for the randomized truncated SVD [HMT09] which results in calculations in millisecond range on modern systems.

t-distributed Stochastic Neighbor Embedding

t-distributed Stochastic Neighbor Embedding (t-SNE) [MH08] was introduced by van der Maaten and Hinton in 2008 with the goal in mind to visualize high-dimensional vector spaces by creating a low-dimensional embedding. In essence it is a nonlinear dimensionality reduction algorithm

[1]https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit
[2]http://scikit-learn.org/stable/modules/decomposition.html

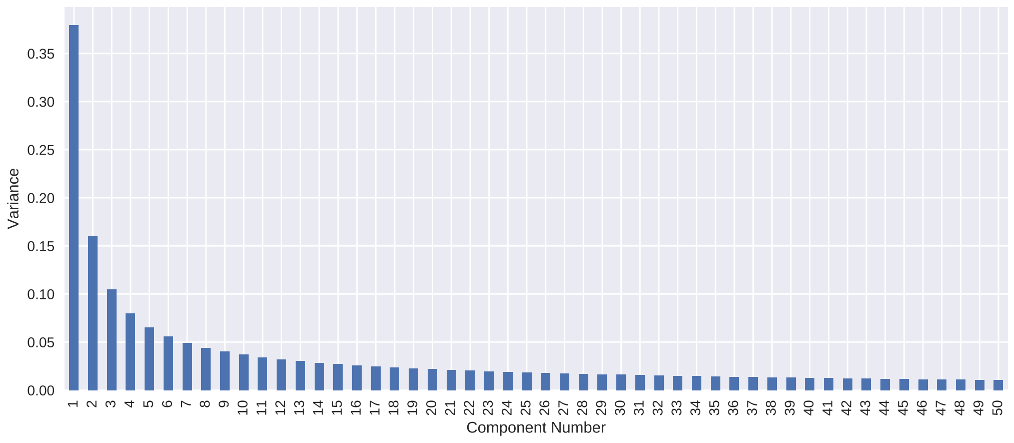


Figure 2.3: Variance of components of a 50 component PCA created from 400 vectors of size 300 vectors from a Word2Vec model

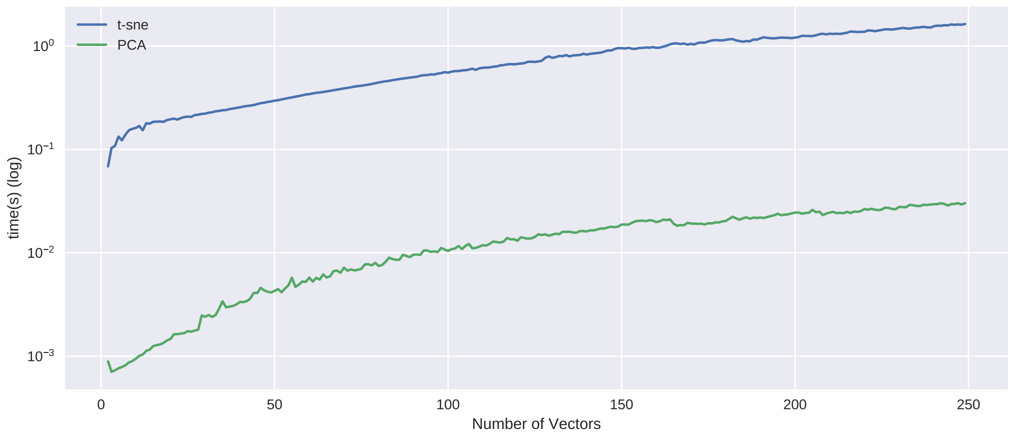


Figure 2.4: Performance of t-SNE and PCA

with the objective to model similar vectors by nearby vectors and dissimilar data by distant vectors with high probability.

To achieve this it converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence [KL51] between the joint probabilities of the low-dimensional embedding and the high-dimensional data.

t-SNE was developed for a high-quality modeling of the higher-dimensional vector's space in the reduced vectors, but not for computing speed. We compared the performance of t-SNE and PCA in Figure 2.4 by measuring the time to reduce a given number of vectors (displayed on the x-axis). We find that PCA is more than 50 times faster than t-SNE when reducing 250 vectors of size 300 to 2-dimension. We must also transform the reduced vector back into the original space, which is not supported by t-SNE but by PCA. On this account we choose PCA to reduce dimensionality, despite the better results of t-SNE.

2.3 TRADITIONAL INFORMATION RETRIEVAL

Using computers to retrieve relevant information has a long history dating back to 1945. It was popularized by Vannevar Bush in the article "As We May Think" [B+45]. We present two approaches that have long been state-of-the-art in many applications.

TF-IDF

tf-idf (Term Frequency-Inverse Document Frequency) was introduced 1972 by Karen Spaerck Jones by publishing a paper about the specificity of terms and its application in information retrieval [SJ72]. It is the most common weighting method used to describe documents in the VSM [SM05]. For each document in the text collection, a vector is calculated where each component relates to one word in the corpus which takes the weighted term frequency as well as the inverse document frequency into account. The term frequency measures how often a term occurs in one particular document where as the inverse document frequency determines the informativeness of a term.

There are slightly different formulas of tf-idf but here we use the common formula of Yang et al. [YL+99]. As the name suggests the formula consists of a tf and an idf part which we will describe first.

Weighted Term Frequency applies the logarithm scale to the term frequency $tf_{t,d}$ of a term t within a document d. We add one to the term frequency because $log_{10}(1) = 0$ and $log_{10}(0)$ is undefined.

$$w_{t,d} = log_{10}(tf_{t,d} + 1)$$

Inverse Document Frequency measures the informativeness of a term t by dividing the number of documents in the set N by document frequency df_t which is number of documents where t occurs in.

$$idf(t) = log_{10}(\frac{N}{df_t})$$

We use a logarithmic scale in both parts to dampen the effect of terms with high frequencies i.e. increasing frequency of terms with an already high frequency does have little significance for the relevance.

This concludes the final formula 2.1 of weight of one vector component of one in term in a document.

$$tf\text{-}idf_{t,d} = \begin{cases} log_{10}(tf_{t,d} + 1)log_{10}(\frac{N}{df_t}), & \text{if } tf_{t,d} > 0. \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

BM25

Okapi BM25 [RZ+09] (Best Matching 25) is a tf-idf-like retrieval model and has maintained its state-of-the-art status of information retrieval for almost 20 years. As tf-idf it relies on document frequencies of the terms of a query over all documents in a collection as well as the term frequency within one document. Originally, it was designed for short catalog records and abstracts of fairly consistent length. But today it is one of the most accepted and widely used retrieval models in all kinds of application areas.

The retrieval model Okapi BM25 [RW94] has the following formula:

$$BM25_{d,q} = \sum_{w \in Q} idf(w) \times \frac{(k_1 + 1) \times tf(w,d)}{k_1 \times K(d) + tf(w,d)} \times \frac{(k_3 + 1) \times tf(w,q)}{k_3 + tf(w,q)}$$

with $Q = d \cap q$ as a set of common terms between the document d and the query q.

This expression comprises four parts: the inverse document frequency (idf) component, the term frequency (tf) component, two saturation functions controlled by parameters k_1 and k_3 and the document length normalization factor $K(d)$.

The *idf* formula is formulated as follows:

$$idf_w = log\left(\frac{|D| - df(w) + 0.5}{df(w) + 0.5}\right)$$

and the document-length-normalization $K(d)$

$$K(d) = (1 - b) + b \times \frac{len(d)}{avdl}$$

Here $len(d)$ is the document length and $avdl$ is the average document length for the collection.

BM25 like tf-idf dampens effects of terms with high frequency but takes this a step further: The saturation functions asymptotically approaches a limit for high term frequencies opposed to the logarithmic function of tf-idf which has no limit. BM25 is used in the popular *elastic-search*[3] search engine which we will use as a baseline for tests.

2.4 RELEVANCE FEEDBACK

Relevance Feedback [XC17] (RF) is based on using information from the initial query and performing an enriched new query. Traditional RF is an iterative process which works in the following way:

1. The user enters a query and a set of documents is returned.
2. The user marks documents as relevant and non-relevant.

[3]https://www.elastic.co/

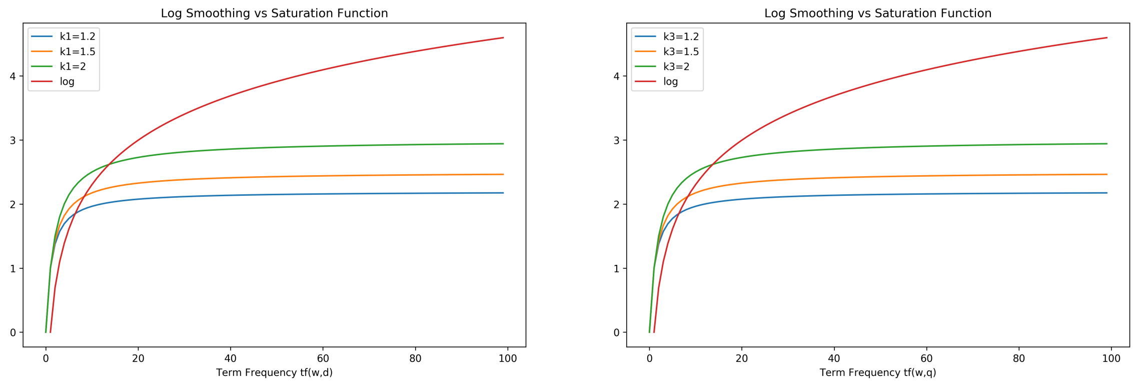


Figure 2.5: Saturation Functions

3. The IR system computes a new representation of the information need as a new query and returns new results.

To calculate a new query, the system can use the Rocchio algorithm [Roc71], which utilizes the VSM. The idea is to move the initial query vector towards the relevant documents and away from the non-relevant documents. This is done by calculating the mean also known as the centroid of the relevant and non-relevant documents and adding the difference of the centroid of the relevant documents and the centroid of the non-relevant documents to centroid of the relevant documents which is listed in Figure 2.6. We depicted the formal in equation 2.2.

$$\vec{q}_{new} = \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j + \left[\frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j \right] \tag{2.2}$$

Where D_r is the set of relevant documents and D_{nr} is the set of non-relevant documents.

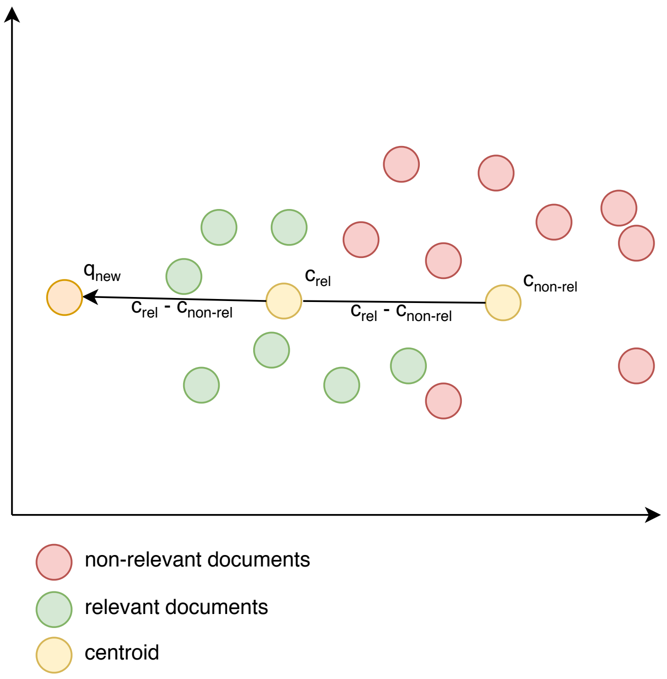


Figure 2.6: Performing a RF based vector shift in VSM

Besides the explicit approach the system can monitor user behavior to mark documents as relevant in an implicit way e.g. if the user interacts with a document the system can assume it is relevant.

It is also possible to use pseudo RF [BSMS95], which does not require user interaction [CNGR08]. There are different types of pseudo RF which we will detail. We can assume that the *n* documents are relevant and perform a new query e.g. using Rocchio. We can take this method a step further by only selecting the top *k* terms with the highest *tf-idf* weight in this selection. So we can just add these *k* terms to the initial query and perform a new search.

3 PROBLEM OUTLINE

In this chapter, we describe the problems that arise in the domain of patents and provide background information. German Patent and Trade Mark Office (German: Deutsches Patent- und Markenamt; abbreviation: DPMA) employees approx. 1000 patent examiner and every day around 300 patent applications are filed. There are more and more applications to process, e.g. in 2010, there were 148,860 pending patent applications whereas in 2016 the number has increased by over one-third to 201,718 pending applications.

Deciding on patent applications is a difficult and lengthy process. The most time consuming part is to research the state-of-the-art for an application that is looking at all patent and non-patent literature and comparing it to the application. This also includes patents in other countries and therefore, in different languages. In order to severely limit the number of patents, the examiner conducts the following steps:

Die Erfindung betrifft ein **elektronisch** gesteuertes **Federungssystem** für ein Fahrrad (1), enthaltend zumindest einem **Federelement** (3, 4), welches zwischen einem ersten Teil (10) des **Fahrrades** und einem zweiten Teil (14, 15) des **Fahrrades** (1) angeordnet ist, welche beweglich miteinander **verbunden** sind, wobei zumindest eine Kenngröße des **Federelementes** veränderbar ist, und zumindest einen Aktor (431), welcher auf das **Federelement** (3, 4) einwirkt, um die zumindest eine Kenngröße zu verändern, und ein **Elektronikmodul** (6), mit welchem ein Ansteuersignal für den zumindest einen Aktor (431) erzeugbar ist, wobei weiterhin ein **Steuerelement** (2, 63, 66) vorhanden ist, mit welchem das vom **Elektronikmodul** (6) erzeugte Ansteuersignal beeinflussbar ist, wobei das **Steuerelement** (2, 63) mit dem **Elektronikmodul** (6) über ein **Funksignal** (64) verbindbar ist und/oder der Aktor (431) mit dem **Elektronikmodul** (6) über ein Funksignal (64) verbindbar ist. Weiterhin betrifft die Erfindung ein entsprechendes Verfahren zur **Steuerung** eines **Federungssystems** für ein **Fahrrad** und ein Computerprogramm zu dessen Durchführung. [...]

Figure 3.1: Patent text from "Stoßdämpfer für Fahrrad" with highlighted relevant keywords

Registration of the patent application The examiner has to read through the patent to get an overview and understand the ideas and concepts the applicant wants to patent. Then, the examiner can reject the patent application if it is not industrially applicable. Otherwise, further steps must be taken.

Extraction of relevant keywords Next, the examiner must look precisely at the patent text and mark all the important keywords. This is crucial as it is the basis for the future search. We depicted an example of a patent text in Figure 3.1.

Transformation into a search query The examiner has to express a search query for the in-house search system, which is currently based on a boolean search model. Examiners link search terms using `AND` and `OR` and can use wildcards (*) which indicates only partial string overlapping. This step requires a high amount of well developed expertise and intuition, but is still time-consuming as well as error-prone. Queries can become very long as depicted in Figure 3.2. As mentioned above patents of other countries have to be searched as well which means that queries have to be phrased also in other languages. Moreover, queries can take up to hours to complete.

```
(elektronisch OR elektrisch OR elektronik) AND
(fahrrad OR elektrorad OR e-bike OR pedelec OR zweirad) AND
(feder* OR dämpf*) AND steuer* AND (*modul OR *element)

(electronic OR electric) AND (bicycle OR bike OR e-bike
OR pedelec OR two-wheeler) (suspension OR damping OR shock
absorption)
```

Figure 3.2: Example search query in German and English for "Stoßdämpfer für Fahrrad"

Review of the result documents The last step is to examine all retrieved documents and lists and review all citations which represent the state-of-the-art. Citations are characterized by the similarity of the application and the citation. Now, the examiner has to decide whether the invention is new by comparing the filed patent with the citations.

As shown in table 3.1 very similar ideas can be expressed by different words with very little to no string overlap.

Patent Application	Citation
Elektronikmodul, elektronisch	Elektronikeinheit, elektrisch
Steuerelement, Steuerung, gesteuert	gesteuert Steuereinrichtung, steuerbar
Federungssystem, Federelement	Stoßdämpfer, Dämpfereinrichtung, Dämpferkammer

Table 3.1: Similar terms from patent "Stoßdämpfer für Fahrrad" and its citations

Besides the inherent problems in the patent domain patent filers have certain goals in mind, which make the search process even harder. One goal is to increase the coverage of their patents and increases the applicability of patent. Therefore, they use generalizations of words, e.g. "vehicle" instead of "car". This way, the patent is not only applicable for cars but for all kinds of vehicles. Furthermore, they try to deceive the subject or idea of the application by usage of rare, uncommon or non state-of-the-art terms or paraphrases, like "move in circles" instead of "rotates" so that the intrinsic ideas can be hidden. Another way is to invent new words that work particularly well in German using compound words such as “Müllentsorgungsdrohne” (garbage disposal drone).

To sum up, patents are a daunting domain. Boolean IR methods are hard to handle, tedious, error-prone and deliver perfectible quality of results.

3.1 REQUIREMENTS

After understanding the problem domain we can derive requirements which we will present here. The prototype should include the following basic non-functional requirements and basic functionalities:

Non-Functional Requirements

1. **Improved Recall:** The central aspect of the prototype is to improve recall as it is highly relevant for patent examiners to find citations in a feasible amount of citation suggestions.
2. **High Performance:** Queries must complete processing within seconds so that examiners can work continuously on an application.

Functional Requirements

1. **Combined Search of Words and Documents** The user interface provides a search slot where a sequence of words and/or document IDs can be entered. Words and documents are present in two-dimensional plot and users can trigger new searches by clicking near relevant documents/words.
2. **Simple Query Formulation** Query formulation needs a lot of experience and is nevertheless error-prone. The new query should consists of the patent application text. Examiners can enter further terms which are weighted more than words in the patent text.
3. **Semantic Component:** Usage of synonyms, different notations and potential spelling errors should not affect the search results. Queries including “vehicle” have to return documents about cars without specific mentioning of the term “vehicle”.

4 CONCEPTION

In this chapter we describe the conception of the prototype. This includes the presentation of the trainings pipeline and the work flow of the prototype.

4.1 WORD EMBEDDING PREPROCESSING AND TRAINING

Before we discuss the actual work flow, we describe the training of our model which we use for our prototype. Our goal is to create a model which projects documents and words in the same vector space. The full process is depicted in Figure 4.1.

First documents (*docs*) have to be loaded from the elastic search *store* and prepared. Duplicates have to be removed (e.g. different versions of the same patent) using *docs_dedup*. To dissolve cluster (e.g. documents are sorted by topic beforehand) we use *docs_shuffle* to shuffle the documents. We can only use a subset of the documents which shortens training time (e.g. for when running experiments on subsets) with *docs_slice*, but for the full model we use all documents. Next, we use a *tokenizer* and a *analyzer* to remove all numbers and stop-words in the corresponding language from text as well as to demarcated words.

Furthermore, we split all documents by the dot character resulting in the *corpus*. Hereafter, we assign every document an ID (*texts_and_labels*) which is necessary for later reference. To create the Sent2Vec model we use to *sents_and_labels* to first split every document into its sentences and then shuffle them using *sents_and_labels_shuffle*. We create a *word2vec* model using Fasttext which we eventually use as a pre-trained model for the *sent2vec* training.

Next, we convert the model into a memory mapped format which allows fast access without loading the model into memory using *sent2vec_convert*.

After creating the *sent2vec_model* which includes all words from our text corpus we create the document vectors by averaging all documents resulting in *doc2vec*.

Finally, we combine the *doc2vec* which holds all documents and the *sent2vec_model* into one single *doc2vec_model*. This model allows us access to all document and word vectors as well as returning k-nearest neighbors given a words, documents and vectors.

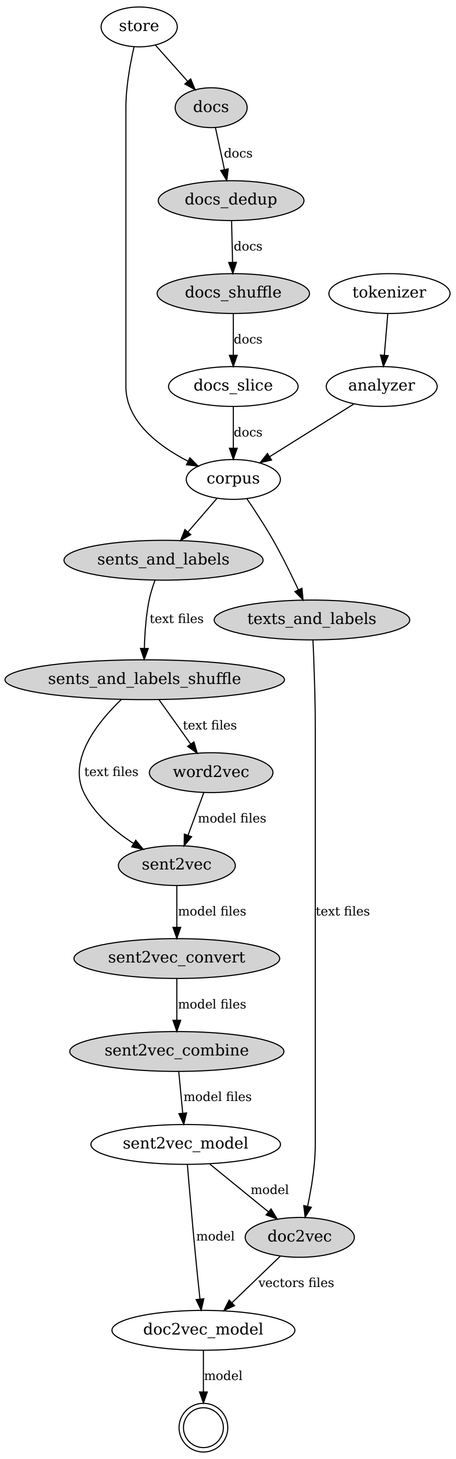


Figure 4.1: Training Pipe

4.2 WORD EMBEDDING ENHANCED DOCUMENT RETRIEVAL

The Sent2Vec model which was acquired in the previous section is used in the prototype below. This section details its interacting components and the workflow.

Components

Our system consists of the following components.

1. **User Interface:** Allows the user to enter search terms and navigate in the VSM.
2. **Document Collection:** Stores all meta data for the documents such as titles, patent applicants or patent classes.
3. **Dimensionality Reducer:** Transforms vectors from the original space into two-dimensional vectors.
4. **Sent2Vec Model:** Stores the VSM, i.e., vectors with related words/vectors and can perform function such as most similar words/documents given a word, document or vector

Work Flow

Next, we explain how the individual components interact with each other in order to explain the processing of a query. We depicted the work flow visual in Figure 4.2.

1. The user enters a query consisting of words and/or document IDs and submit.
2. The query is translated into a vector representation by the Sent2Vec model by looking up each embedding vector for every word/document and averaging them.
3. A fixed number of similar documents (IDs) and words together with their embedding vectors and similarity to the query vector are returned ordered by their similarity respectively.
   (a) For every document ID, a title and a set of meta data is retrieved from the document collection.
4. All returned vectors are reduced to a two-dimensional representation by the dimensionality reducer and passed together with their respective documents and words and to the user interface.
5. The user interface displays all two-dimensional vectors together with their respective words and document titles. The set of meta data provides additional information on demand. A draggable marker is shown in the center of the plot. All documents are listed next to the plot in order of their relevance. The user has the opportunity to drag the marker in a direction if he or she wants to explore the space more.
6. If the user decides to do so the two dimensional position is passed to the dimensionality reducer.

7. The dimensionality reducer back transforms the new two dimensional query vector into the embedding vector space resulting in a new query vector which is passed to the Sent2Vec model.

8. The Sent2Vec model will now use the vector instead of words/documents for calculating the nearest words/documents. The process continues at at step 2.

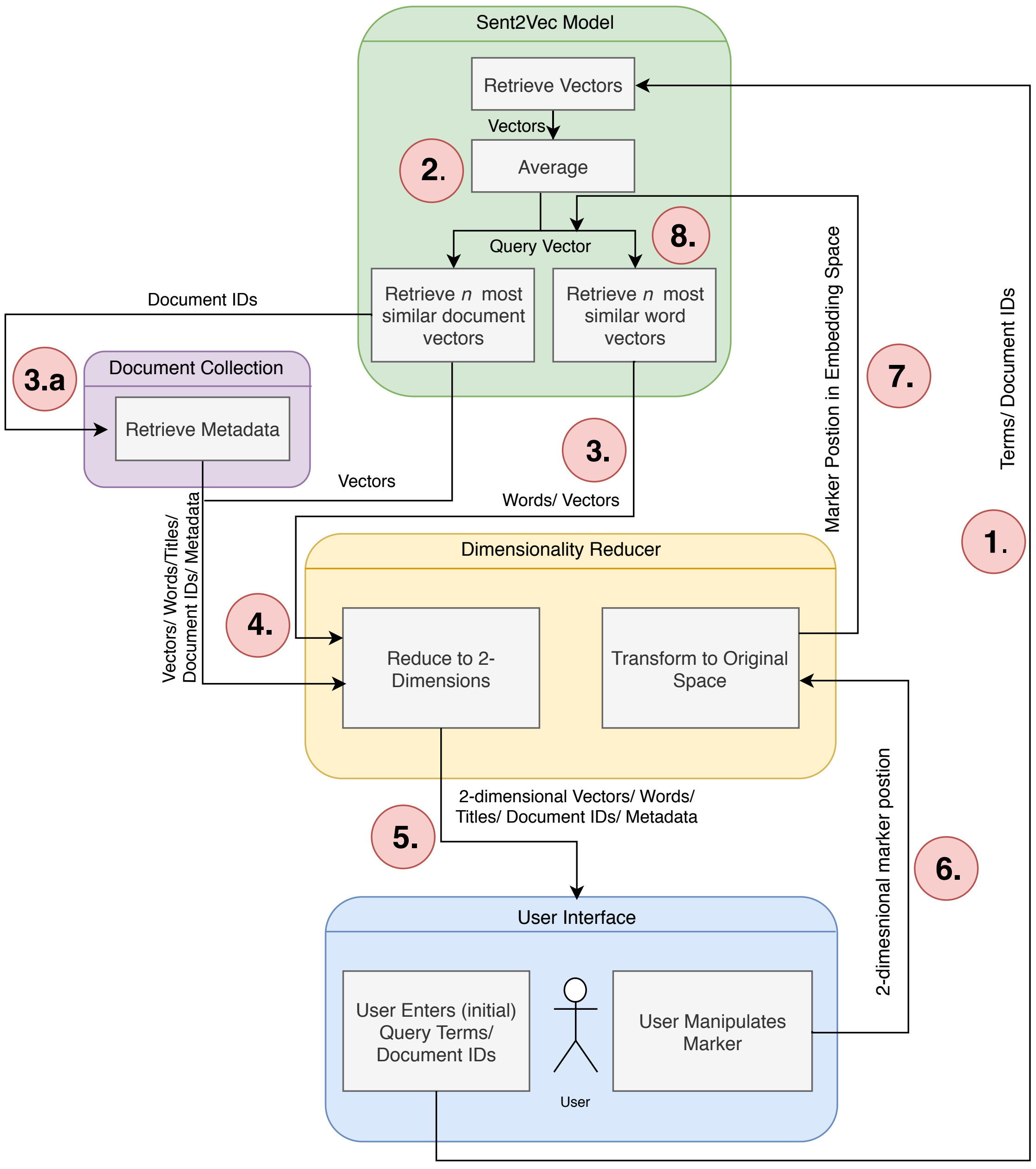

Sent2Vec Model
Retrieve Vectors
Vectors
2.
Average
Query Vector
8.
Retrieve *n* most similar document vectors
Retrieve *n* most similar word vectors
Document IDs
3.a
Document Collection
Retrieve Metadata
Vectors
3.
Words/ Vectors
Marker Postion in Embedding Space
7.
Terms/ Document IDs
1.
Dimensionality Reducer
Vectors/ Words/Titles/ Document IDs/ Metadata
4.
Reduce to 2-Dimensions
Transform to Original Space
2-dimensional Vectors/ Words/ Titles/ Document IDs/ Metadata
5.
2-dimesnional marker postion
6.
User Interface
User Enters (initial) Query Terms/ Document IDs
User
User Manipulates Marker

Figure 4.2: Flow Chart

5 IMPLEMENTATION

In this chapter, we present the implementation details of the architecture described, including the libraries used.

Frontend

Value proposition of the user interface (UI) is to be integrated in interface:project's integator and therefore, has to run in web browsers. Due to this fact, the UI was implemented in JavaScript (JS).

For visualization, we use the recently introduced Vega-toolkit[1] which is based on the popular interactive data visualization library D3[2] for JS. Vega is a declarative visualization grammar for creating, saving, and sharing interactive visualization designs. Vega plots are defined by a JSON configuration which includes the visual appearance and interactive behavior of a visualization. Vega consists of basic building blocks for data loading and transformation, scales, map projections, axes, legends, and graphical marks used in plots. To interact with input event streams such as mouse clicks or drag-and-drop actions Vega offers reactive signals, which modify the plot accordingly. Data can be loaded via an internal data loader or loaded into the Vega via a provided API. This API also provides access to the reactive signals on which we can listen and register event handlers. Due to the data-driven approach, plots are customizable in an easy way by manipulating properties in the input data.

Our interactive Vega plot, depicted in Figure 5.1 provides zooming, moving the view window by drag-and-drop and moving a "search marker" by double-click or drag-and-drop. Every word is marked as a labeled circle and every document as a rectangle and labeled with its title. To make the plot more appealing Vega assigns a color to each mark according to its cluster provided by the backend.

We included the not official feature for automatically label arrangement *vega-label*[3], which aligns and hides labels according to the space available. We found this particularly useful as hidden

[1]https://vega.github.io/
[2]https://d3js.org/
[3]https://github.com/vega/vega-label

labels will emerge while zooming in because fewer labels have to be displayed. To trigger a new search the search marker must be moved by double click or drag-and-drop. This will change a reactive signal which will trigger a previously registered event handler. This handler will execute JS code and will make a REST request to the server using the JS library jQuery[4].

On the response, another handler is executed and will update the UI. The response contains a fixed number of documents and words together with their two-dimensional vectors ordered descending by their relevance.

Words and documents are put back into Vega by using the API. Moreover, the documents are listed next to the plot. When hovering the documents the corresponding mark will highlight in the plot to make comprehension easier.

Alternative UI Manipulation

Another available approach for manipulating the UI and trigger new search queries is to select the documents directly. In this way, the user can express more clearly what he is interested in. In contrast the search marker is more vague.

After clicking all interesting documents the user submits them to the server where all corresponding vectors are averaged resulting in the new query vector.

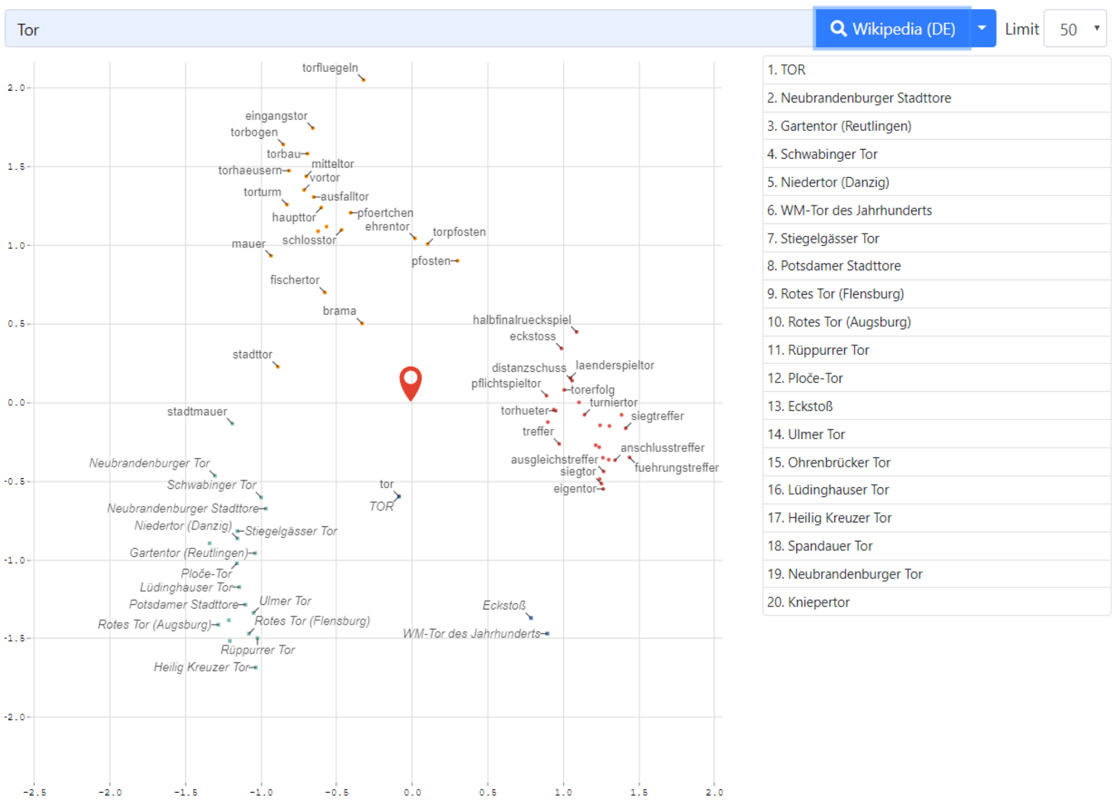
Backend

The server has to receive and send data as well as doing of scientific computing, e.g. handling word embeddings, calculating nearest neighbors, dimensionality reduction and clustering. For this use case Python[5] is an obvious choice.

As a general-purpose programming language, it has become adored in recent years especially in the machine learning community due to its support for deep leaning frameworks such as tensorflow [Gir16] as well as traditional artificial intelligence libraries such as scikit-learn [PVG+11].

On the other hand, popularity for web development in Python is also on the rise. In particular, we use the micro web framework flask[6] to build our web server and accept REST requests from the frontend. When a two-dimensional vector is revived we use sklearn's implementation of PCA to transform it to the original vector space. The PCA was trained beforehand on the embedding vectors. We use the C++ based implementation with a Python API of Sent2Vec from authors of "Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features" [PGJ18] from Github, which is based on a popular word embedding implementation fastText [JGB+16]. The implementation can perform the most-similar-function which retrieves the top *n* words or documents given a list of words and documents or vectors. As mentioned above we use DBSCAN [EKS+96] for clustering, which is a state-of-the-art density-based clustering non-parametric algorithm. For storage and easy retrieval of documents, we use the elastic-search storage[7].

[4]https://jquery.com/
[5]https://www.python.org/
[6]http://flask.pocoo.org/
[7]https://www.elastic.co/de/blog/found-dive-into-elasticsearch-storage



Limit 50

1. TOR
2. Neubrandenburger Stadttore
3. Gartentor (Reutlingen)
4. Schwabinger Tor
5. Niedertor (Danzig)
6. WM-Tor des Jahrhunderts
7. Stiegelgässer Tor
8. Potsdamer Stadttore
9. Rotes Tor (Flensburg)
10. Rotes Tor (Augsburg)
11. Rüppurrer Tor
12. Ploče-Tor
13. Eckstoß
14. Ulmer Tor
15. Ohrenbrücker Tor
16. Lüdinghauser Tor
17. Heilig Kreuzer Tor
18. Spandauer Tor
19. Neubrandenburger Tor
20. Kniepertor

Figure 5.1: UI after querying the word "Tor"

6 EVALUATION

In this chapter, we present our experiment execution and evaluate our findings. First, we introduce the used dataset. Afterwards, we describe the setup of our experiments and the results.

6.1 DATASETS

In order to conduct the experiments, we choose two datasets which are different in nature and allow us to achieve broader coverage and make results more stable. For one side, we use a German patent corpus provided by the German Patent and Trade Mark Office (DPMA) with long and taxing documents, then again, we use a news article corpus made by Reuters with shorter texts in English language.

DPMA Patents

The corpus contains around 300,000 documents in German language, which were filed between 2010 and 2015. Each document contains a title, a description and claims. The claims are an enumeration of ideas and concepts to be patented, whereas the description provides a detailed explanation of the claims. The data comes in XML format and was processed and stored in our elastic search environment. For training we combine all parts into a single text document.

Some patents come with citations, which are gathered by patent experts. As explained in Chapter 3 citations are semantically similar to the patent application and are expected as results when a query is conducted. There are around 2,500 patents, which have one to 20 citations. We have listed the distribution of citation count in Figure 6.1 and notice that the majority only has one citation.

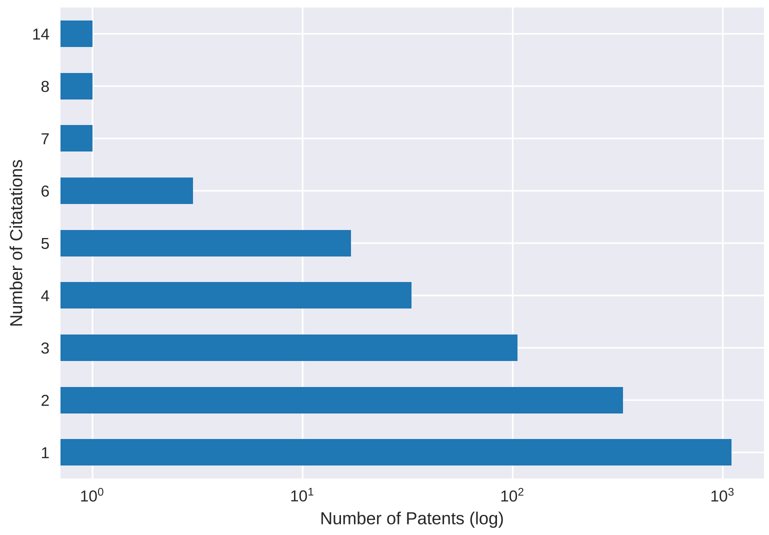


Figure 6.1: Distribution of citations

Reuters Corpus Volume I

The Reuters Corpus Volume I (RCVI) [LYRL04] is an archive of English newswire stories made available by Reuters, Ltd. for research purpose. It contains over 800,000 manually categorized articles produced by Reuters journalists between August 20, 1996, and August 19, 1997. They are categorized by topics, industries, and regions. The data is distributed by the National Institute of Standards and Technology (NIST) and can be requested free of charge at the NIST home page[1]. The data is also delivered in the form of XML and was processed and stored in the same way as DPMA.

For our research, we were especially interested in the category "topic", which we use to determine the relevance ranking. The corpus contains 126 different topics, and each article may have multiple. When a query is conducted we except articles which deal with the same subject. Hence, we mark all documents as relevant for a particular document if the document has a superset of topics of query document. We use 10,000 randomly sampled documents for our test to compensate outliers.

6.2 QUALITY MEASUREMENTS

For our experiments, we test our system against the traditional metrics of recall, precision and F1-score, which are defined as follows:

$$precision = \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|}{|\{retrieved\ documents\}|}$$

[1]https://trec.nist.gov/data/reuters/reuters.html

$$recall = \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|}{|\{relevant\ documents\}|}$$

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

These metrics are popular in IR to measure the quality of an IR system.

6.3 SETUP

After the presentation of our datasets and quality measurements, we detail our experimental setup. For evaluation, we have carried out three tests. First, we perform an experiment we call *Title Test*. Second, we test the BM25 and the word embedding based approaches with document queries and precision, recall and F1-score metrics. Third, we apply explicit and pseudo relevance feedback to these systems and evaluate them with same metrics. We listed the exact results in Appendix A. Finally, we test the word embeddings based system with ambiguous query terms in graphical user interface.

Below, we use the following shorthands:

- *BM25* refers to the Okapi BM25 implementation of the elastic search engine.
- *Word2Vec* refers to a VSM of document and word vectors trained on the basis of a fastText model. To determine a document vector all terms of a document are averaged.
- *Sent2Vec* is the similar to Word2Vec except we use word embeddings trained with the Sent2Vec method. The full process is depicted in Chapter 4.

6.4 TITLE TEST

The *Title Test* is very potent for showing how the system can also capture the semantics of the query and not just the syntax. It is based on the idea that the title of a document describes the semantics of the document, i.e., it roughly summarizes the document. On the other hand, the semantics should differ only slightly if we remove the terms present in the title from the document. For this experiment, we randomly choose 10,000 documents from each corpus. The execution is carried out as follows:

1. Choose a query document
2. Split the title into its words
3. Remove every title word from the query document
4. Determine the new document vector by averaging all words from the document

After that we retrieve the documents that are most similar to the query consisting of the title words. We evaluate this test using our Word2Vec model as well as our Sent2Vec model on RCV1 and DPMA. We exclude BM25 as it requires matching terms between query and document, which is impossible in this setup. We calculate the recall for the first n documents with $1 \leq n \leq 1000$ denoted by @n. We mark the original document as relevant and all else as non-relevant.

We first look carefully at the results on base of the RCV1 corpus. We notice that Sent2Vec performs significantly better than Word2Vec as depicted in Table 6.1 (a) and Figure 6.2 which is in line with our expectations as Sent2Vec enhances word embeddings with focus on averaging over documents. Both converge to 100% so the recall gap decreases in the long run.

Recall	Sent2Vec	Word2Vec
@1	24.38%	10.82%
@5	49.13%	26.72%
@10	59.68%	35.97%
@50	78.51%	58.20%
@100	84.66%	67.46%
@500	95.32%	87.22%
@1000	97.65%	92.83%

(a) RCV1

Recall	Sent2Vec	Word2Vec
@1	64.5%	11.79%
@5	87.1%	25.50%
@10	91.10%	32.67%
@50	97.5%	52.58%
@100	97.89%	62.61%
@500	99.6%	83.25%
@1000	100.0%	90.28%

(b) DPMA

Table 6.1: Recall for Title Test

Next, we observe that the recall was on DPMA also better when using Sent2Vec than Word2Vec but the gap is larger than on the RCV1 corpus, as depicted in Table 6.1 (b). We notice a similar performance for Word2Vec on both corpora but a significantly better performance for Sent2Vec on DPMA for myriad reasons. First, there are many articles that are very similar as they report on same events, persons or countries. Second, articles are relatively short and the most important words which also appear in text are removed. Hence, meaning of the article shifts significantly more than when applying this method to patent texts. Overall, we observe a recall exceeding 90% after 10 retrieved document for Sent2Vec on DPMA which shows the capability of the system to encode semantics into the query.

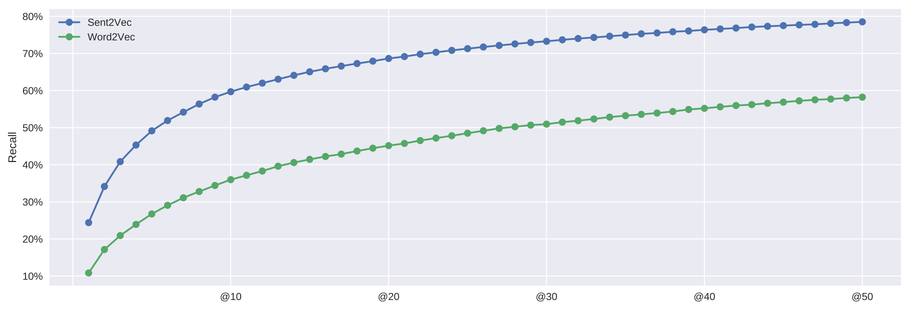


Figure 6.2: Recall for Title Test on the basis of RCV1

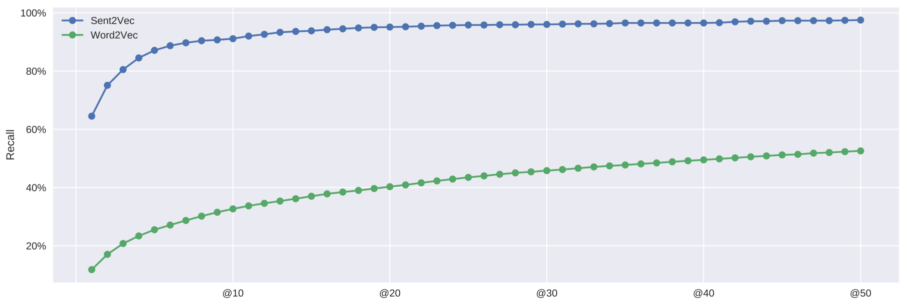
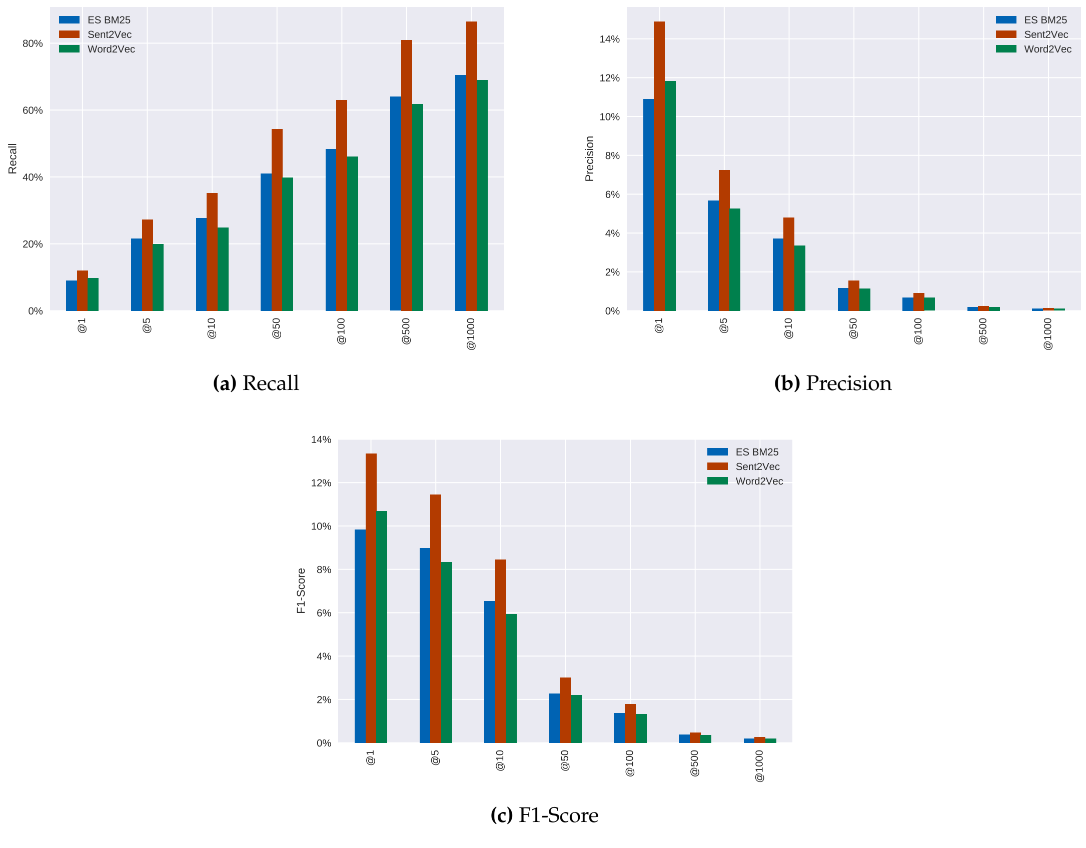


Figure 6.3: Recall for Title Test on basis of DPMA

6.5 COMPARISON WITH BM25

In this experiment, we evaluate the tf-idf-like retrieval method BM25 against Sent2Vec and Word2Vec on the basis of recall, precision and F1-score. The experiment is carried out as follows: First, we perform a query using one document at a time and look at the first 1,000 retrieved documents. Then, we calculate the metrics described above for the first n documents with $1 \leq n \leq 1000$ denoted by @n. The results are averaged for all queries conducted.



(a) Recall

(b) Precision

(c) F1-Score

Figure 6.4: Quality metrics on DPMA

Using the DPMA corpus, we notice a significantly better performance in terms of all metrics using Sent2Vec compared to Word2Vec and BM25 as depicted in Figure 6.4, Figure 6.5, Figure 6.6

and Figure 6.7. Word2Vec results are similar to BM25 results indicating no performance improvement. This indicates a good modeling of the vector space which is no less than BM25's. This is astonishing since Word2Vec's goal was only a good modeling of single words. Furthermore, we notice a drop in precision since most of the time a patent is only associated with one relevant document. On the other hand, recall is expected to be higher, but overall low, for the same reason. This indicates that the modeling of the vector space for patents is hard, which we detail in Section 6.7.

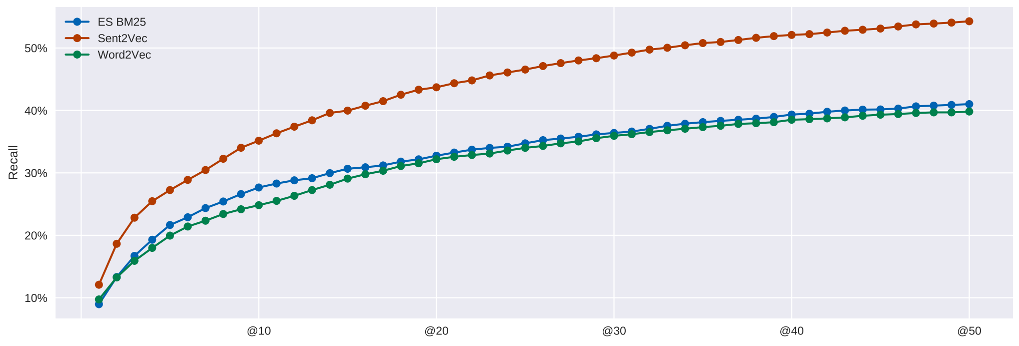


Figure 6.5: Recall on the basis of DPMA

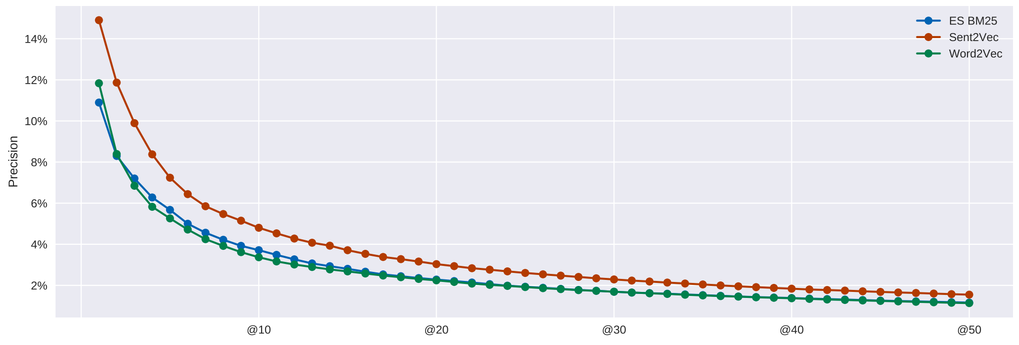


Figure 6.6: Precision on the basis of DPMA

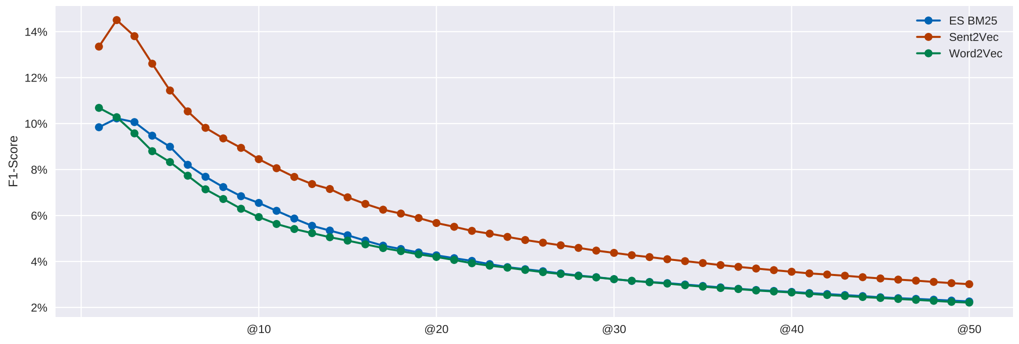
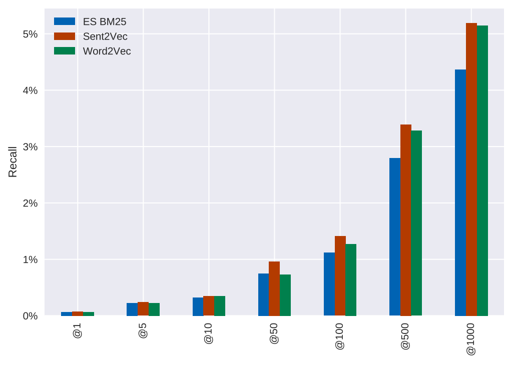


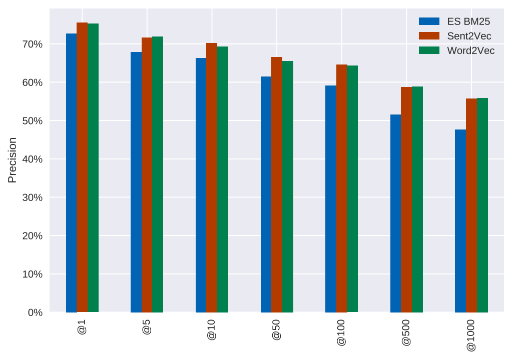
Figure 6.7: F1-Score on the basis of DPMA

Next, we discuss the results using the RCV1 corpus. Sent2Vec's precision is first on par with Word2Vec's until @5 but than is higher as illustrated in Figure 6.10 and Figure 6.8 (b). Later @100 the difference is almost 0 and @500 and @1000 Word2Vec's precision is even marginally better.

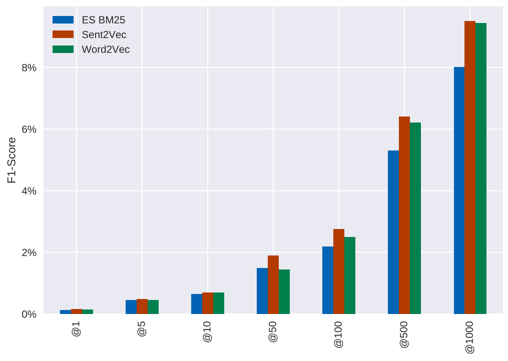
ES BM25's precision is always inferior than the word embedding based approaches. We note a similar behaviors for the other metrics recall and F1-score listed in Figure 6.11, Figure 6.9 and Figure 6.4 (a)/(c). We find that the naïve approach of averaging Word2Vec's word embeddings is better than BM25. Sent2Vec is better in the beginning but later converges to Word2Vec's curve.



(a) Recall



(b) Precision



(c) F1-Score

Figure 6.8: Quality metrics on RCV1

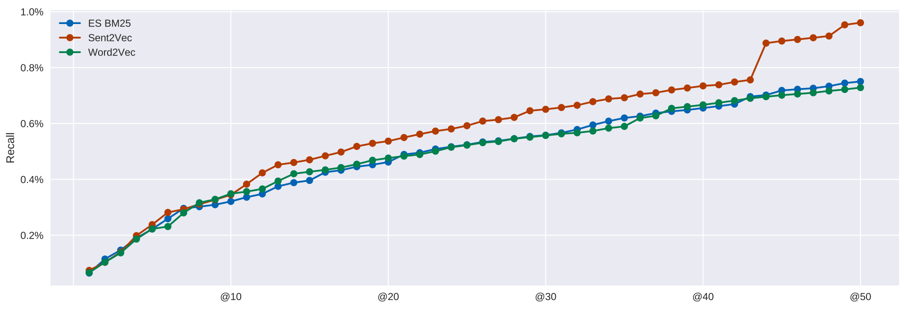


Figure 6.9: Recall on the basis of RCV1

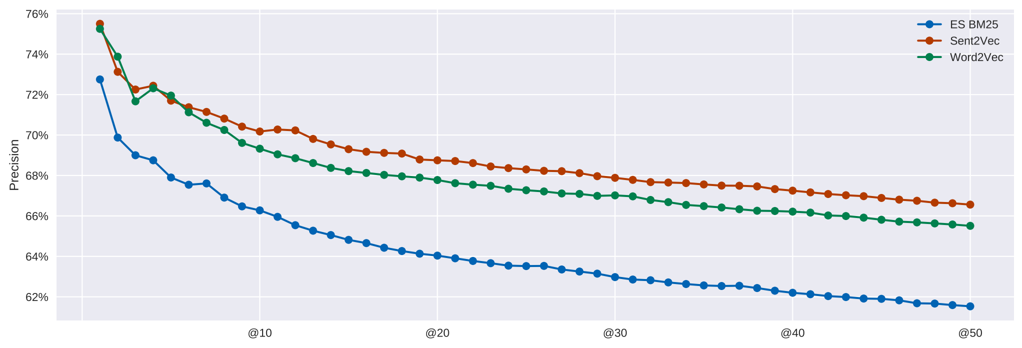


Figure 6.10: Precision on the basis of RCV1

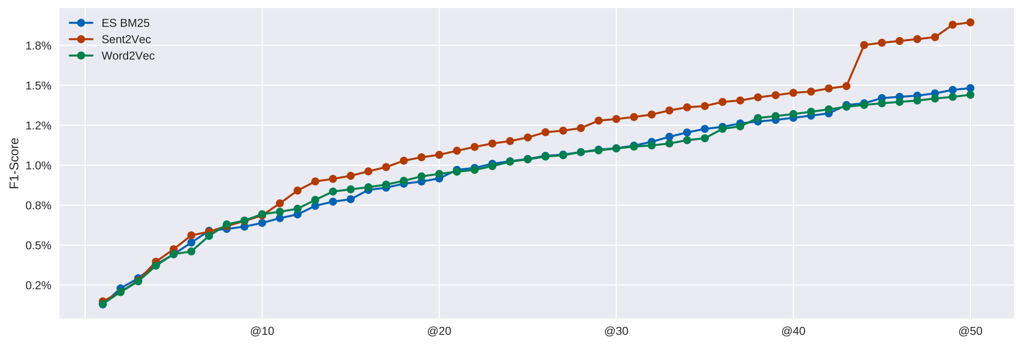


Figure 6.11: F1 score on the basis of RCV1

6.6 RELEVANCE FEEDBACK

We apply explicit relevance feedback (RF) and pseudo relevance feedback (PRF) to the search process which is done in the following way:

1. RF: Explicit relevance feedback usually requires user interaction. Their input is simulated by looking at the first 10 retrieved documents. In order to emulate user errors, we mark relevant document with a probability of 0.9 as relevant and non-relevant documents with a probability of 0.1 as relevant. We perform a new query using all relevant marked documents as well as the initial document.

2. PRF: For pseudo relevance feedback we select from the top n documents the top k terms with the highest tf-idf score. We chose $n = 10$ and $k = 20$, due to a grid search. To get the 20 key words, we average the first 10 tf-idf vectors and take the words with the highest value from the resulting vector.

After performing the new queries, we evaluate the quality metrics again. Please note that we only use RF and PRF for the RCV1 corpus since it requires a decent number of relevant documents and the DPMA patents mostly have only one document as shown in Figure 6.1.

Data in Figures 6.12 to 6.17 suggests that RF has significantly improved all quality metrics. We notice that the lead of RF gets smaller the more documents are received but is still present @1000.

RF improves the centroid vector but does not change the VSM, so this behavior is expected. We observe that recall for BM25 with RF is higher in the beginning than Sent2Vec without RF but is equal between @10 and @11 and then lower. This underlines a better modeling of VSM for this IR task.

In contrast, PRF's improvement is not significantly. It should, however, be noted that PRF yields inferior metrics to some extend. Averaging the key words of the first 10 documents is similar to the first document. Hence, it does not shift the query as the initial document is already the average of the most important key words. Furthermore, as PRF also includes non-relevant documents, on average around 30% as shown in Figure 6.13 for the first 10 documents, the centroid of the query may slide in the wrong direction.

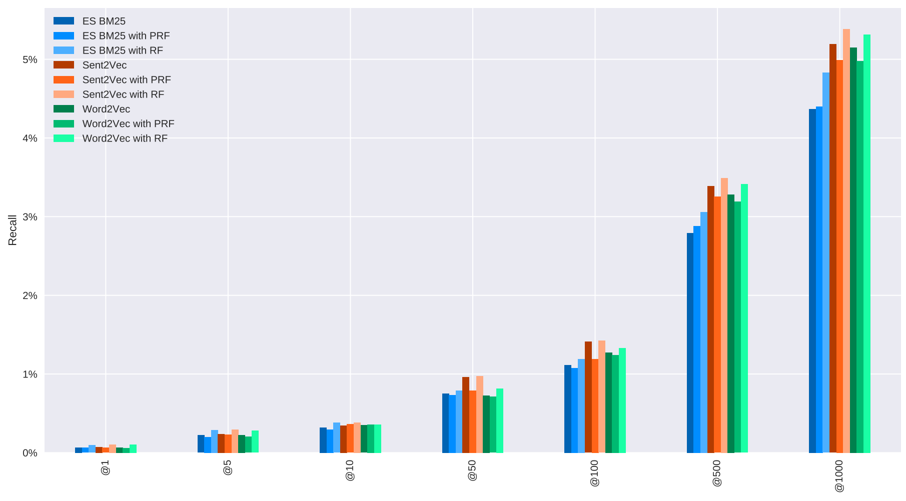


Figure 6.12: Recall on the basis of RCV1

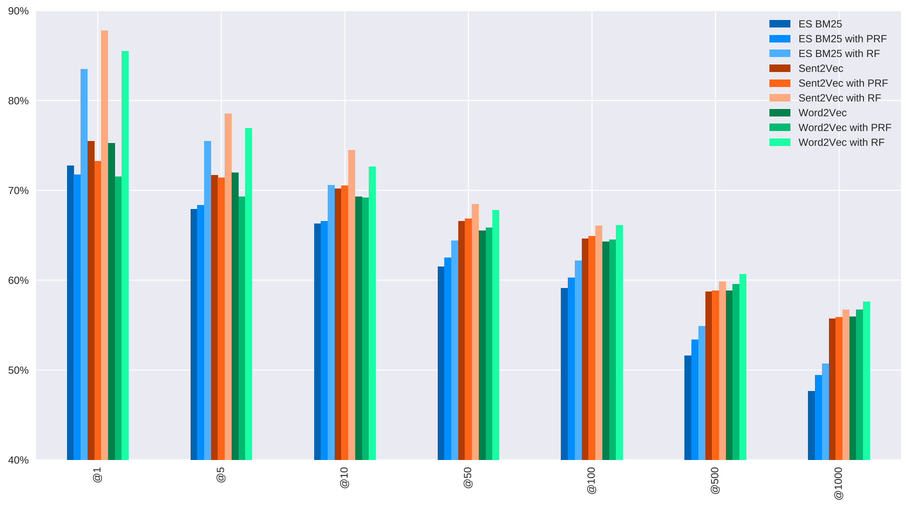


Figure 6.13: Precision on the basis of RCV1

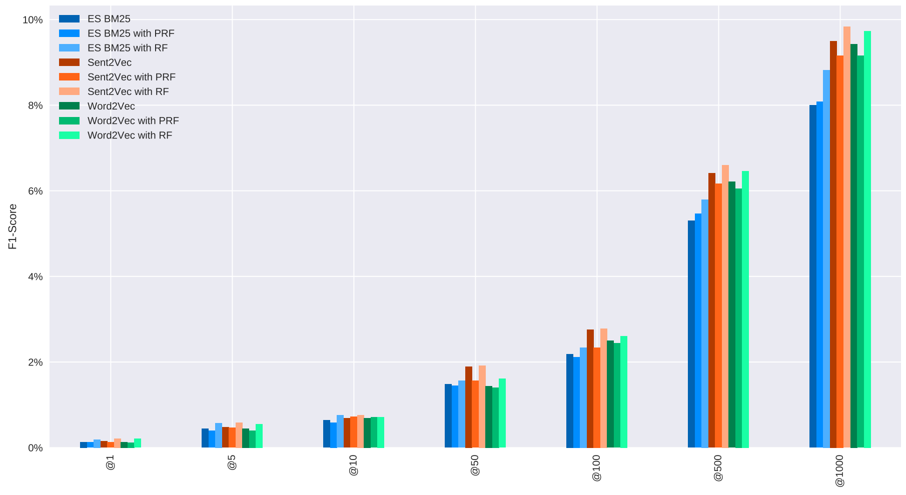


Figure 6.14: F1-Score on the basis of RCV1

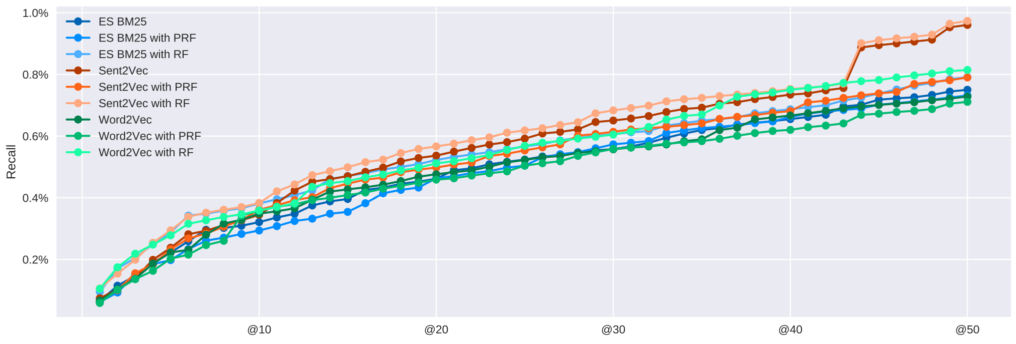


Figure 6.15: Recall on the basis of RCV1

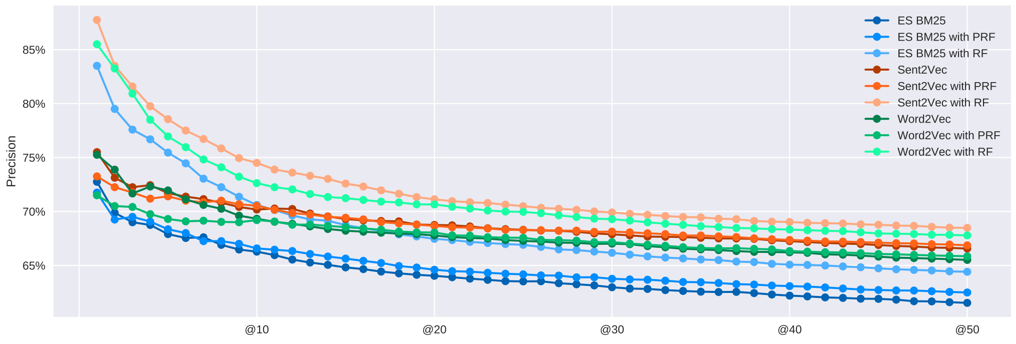


Figure 6.16: Precision on the basis of RCV1

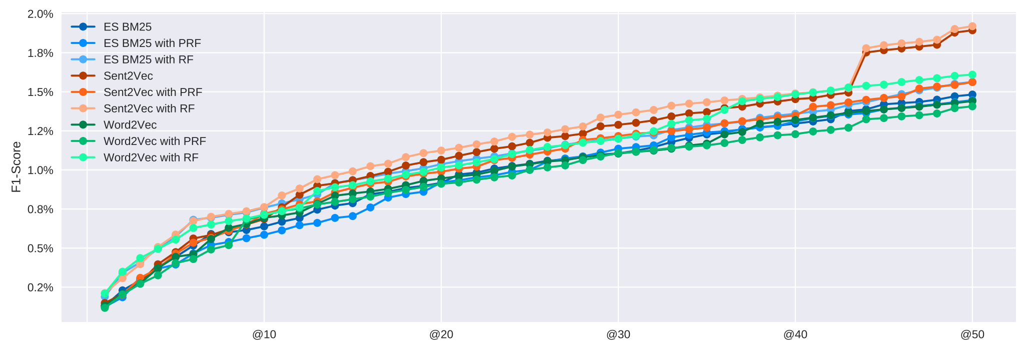


Figure 6.17: F1 score on the basis of RCV1

6.7 DETAILED ANALYSIS OF KEY WORD SIMILARITIES

While examining the results, we found that the precision of the DPMA patents is significantly lower than for RCV1. That can be justified by the fact that most patents have only one citation as shown in Figure 6.1. On the other hand, recall is also relatively low. As most DPMA patents only have one citation, this means that it is found only in 35% of cases within the first ten hits and only in 54% within the first 50 documents. In Chapter 3 we already outlined the inherent problems of patents, but we want to give more insights and compare to RCV1, which has a significantly better recall.

Therefore, we analyze a news article from RCV1 and patent from DPMA to get an intuition. For illustration, we look at the abstract of the German patent "Müllentsorgungsdrohne"[2] listed in Figure 6.18 and an excerpt from a news article from RCV1 in Figure 6.19 for comparison.

A **drone** equipped with a **gripping device** which is able to fish out objects floating in **water**. These **drones** are attached to **ships** and can be operated manually or by means of a **camera** which detects and locates the objects floating in the **water**. This procedure ensures the decontamination of the **oceans** and serves as an attraction on **ships**. In addition, flight hours and a corresponding **drone pilot's license** can be offered and completed.

Figure 6.18: Abstract of "Müllentsorgungsdrohne"

Mexico's top **anti-drug** official, **jailed** on **corruption** charges, has admitted knowing a **drug** henchman but said he made the contact hoping it would lead to the arrest of a **cocaine king**, newspapers reported on Friday. The newspapers said Gen. Jesus Gutierrez Rebollo, fired on Monday after two months as head of the National Institute for Drug Combat (INCD), had pleaded innocent in an initial statement to prosecutors. Gutierrez Rebollo's arrest provoked the most serious drug corruption scandal ever made public in Mexico and triggered rebukes from Washington, where officials worry that secret drug intelligence shared with the general may now be in **cartel** hands.[...]

Figure 6.19: Text of article "Mexico: Anti-drug Official Arrested"

[2]https://register.dpma.de/DPMAregister/pat/register?AKZ=1020140028046

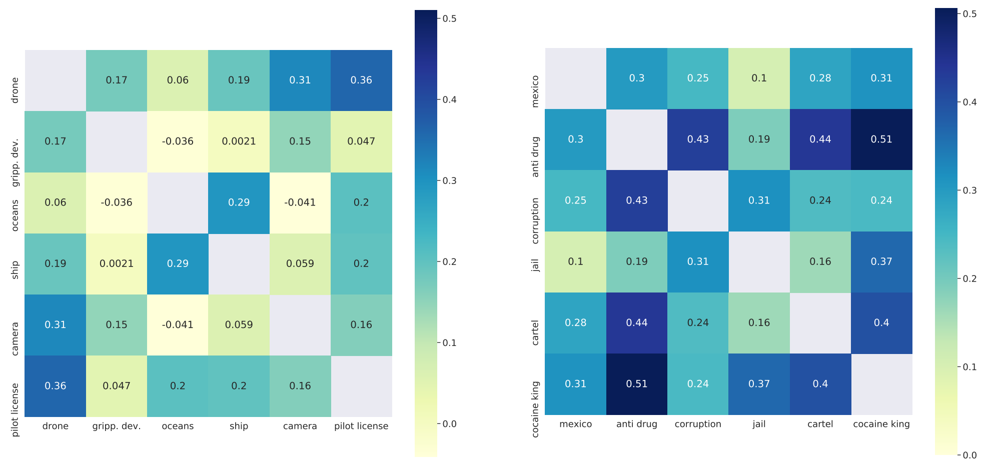
- Drone
- Gripping device
- Water/oceans
- Ship
- Camera
- Pilot's license

Figure 6.20: Key word list of the patent

- Mexico
- Anti drug
- Corruption
- Jail
- Cartel
- Cocaine king

Figure 6.21: Key word list of the article

For the sake of clarity, we mark all relevant key words in both excerpts and list them in Figure 6.20 and Figure 6.19, respectively. As we analyze the key words, we notice that the topics of the patent are manifolded, and every key word is more or less its own topic. It is crucial for patents to combine different concepts in a non trivial way, as it is a prerequisite to get approval.



(a) Patent Abstract **(b)** News Article

Figure 6.22: Similarity Matrix of Key words

On the other hand, the content of the Reuter's article is coherent and sufficiently described in the title which holds true for most of the documents in the corpora. In Figure 6.22 we depict a similarity matrix of the most important terms in both documents. As we can see the cosine similarity in the terms in the article is higher (on average 0.302 for RCV1 vs. 0.142 for DPMA) which underscores our claim. It is not easy to determine which concepts are relevant for the patent and which not.

6.8 DISAMBIGUATION USING GRAPHICAL RELEVANCE FEEDBACK

In this experiment, we show how we can use our search system to disambiguate search terms. We conducted the experiment in the following steps: First, we use a German homonym, which is listed in Appendix B to formulate a query. Second, we examine the plot and look closely for the different meanings. If we identify a meaning of the homonym we move the marker close to the words representing the meaning and perform a new query. If the words do not relate to the meaning or not any matching terms were found in the first place, we mark the meaning as "not found" otherwise as "found" (marked by bold type in the appendix). We use the Sent2Vec model trained on German Wikipedia as it covers a wide range of topics.

We found that per homonym on average 64% of its meanings were found. A histogram listing the distribution of recalls for all homonyms can be found in Figure 6.23.

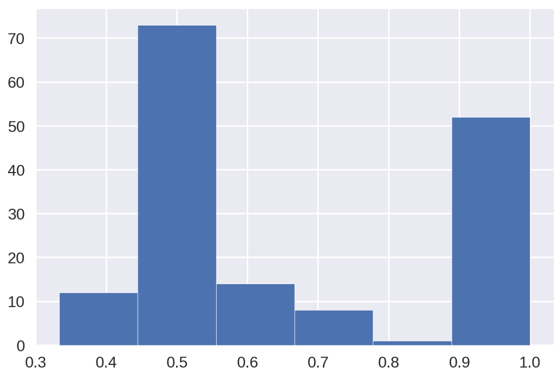


Figure 6.23: Distribution of recalls of homonyms

We found that the result is very dependent on the number of Wikipedia articles about this specific meaning and is therefore, one-sided. Let us take the Geman word "bank" meaning a financial institute or a bench. Unfortunately, our prototype only returned financial institutions as a result.

There are a lot of financial institutions (JPMorgan Chase, Deutsche Bank, Goldman Sachs, etc.) each with one Wikipedia article. On the other hand, the (garden-, wood-, stone-, etc.) bench meaning has a few articles where all different types of benches are listed. Therefore, there are much fewer sentences where "bank" is used in this context of "bench". This results in a strong bias towards the financial institutions since word embeddings take into account the number of times a word appears the in specific context.

This behavior is perfectly fine for a search system if we use the right corpus, i.e. the same corpus for training and IR. If we are using a corpus about finance, we are only interested in financial institutions and not benches. In the domain of patents, we are mostly interested in technical terms and can exclude other non-technical meanings.

6.9 SUMMARY

In summary, we showed that the word embedding based IR approach yields promising results in a real-world scenario as well as in constructed use cases. The Title-Test indicates the word embedding models' ability to detect semantic similarities between queries and documents. We showed an improved recall, precision and F1-Score of the Sent2Vec approach compared to BM25 as well as the problems using Sent2Vec. Since word embeddings are a novel field of study, these results are promising and future improvements can further enhance them. Furthermore, applying RF to the word embedding based approach increases all quality metrics measured. Finally, our prototype was able to detect multiple meanings of German homonyms.

Findings suggest that a word embeddings based approach is better suited than a tf-idf approach to span a VSM in the domain of IR, because it relies on semantics rather than exact string matching.

7 CONCLUSION AND OUTLOOK

Effective training of neural networks using GPUs and massive amounts of data has become possible in the last decade. Hence, neural networks are used for more and more tasks and replace traditional algorithms. In the course of this thesis, we showed that a word embedding is indeed better than traditional IR methods. We have shown that our Sent2Vec model has surpassed the traditional IR methods by far in various metrics, even the naïve approach of averaging all terms of a document using the Word2Vec model can occasionally deliver better results than BM25. Recall, precision and F1-score showed the superiority of Sent2Vec model in comparison to BM25.

We also demonstrated that we can use word embeddings to disambiguate homonyms using our user interface but there is still room for improvement. We also presented a working prototype for word embedding based IR system using Sent2Vec to model the VSM. The prototype allows the user to give direct relevance feedback by navigating the VSM. To make the VSM appealing to the user, we used PCA to transform vectors from the original space to two-dimensions and transformed back user feedback in form of a two-dimensional marker back to the original space.

We executed a set of experiments to testify the hypothesis state in the introduction.

1. **Semantic Component** The Title-Test showed that a word embedding based search truly encodes semantic aspects of the terms. We demonstrated decent results when removing all matching terms from a document.
2. **Improved Recall** We showed that the naÃŕve approach of averaging word vectors using fastText was mostly on par with the long-established Okapi BM25 ranking function and occasionally even superior. Furthermore, we showed that the improved approach for embedding documents Sent2Vec outpaced BM25 in all quality measurements measured. Also, we showed that explicit relevant feedback can be applied to the Sent2Vec approach.
3. **Disambiguity** We also showed that we can use word embeddings to disambiguate homonyms using our user interface. We also detailed the limitation in this regard as they are depending on the training corpus.

In this work, we presented an approach for the usage of word embeddings in an IR context. We showed how we can overcome traditional IR ranking algorithms, but also demonstrated limitations of our approaches. Overall, the concept taken looks promising and can be improved through further research.

Outlook

Neuronal network-based algorithms are currently a highly investigated topic and are constantly being further developed.

BERT [DCLT18] and LASER [SD17] are new word embedding techniques, which produce embeddings based on the context of the word. In order to do this, they take whole sentences as input to weave in the context of each word into the embedding. Both are based on neural network architectures where BERT uses *Transformers* [VSP$^+$17] and LASER Long short-term memory (LSTM) [HS97] cells. We showed the problem of homonyms, which could be resolved with these new approaches, e.g. the word "mean" will be translated into a different vector in the sentence "I calculate the mean" than in "He is a mean person".

Auto-encoders [BK88] are another neural network based approach for reducing dimensionalities. They are based on an three-layer architecture with an encoding, hidden and decoding layer. It is also possible to use more complex architectures with several encoding and decoding layers shown in Figure 7.1. They are trained on word embeddings and use the encoder layers to compress vectors to a two-dimensional representation. Back transformation to the original space can be done by using only the decoder layers. Shallow auto-encoders can approximate PCA orthogonal linear transformation but also can further improve it, by introducing non-linear layers.

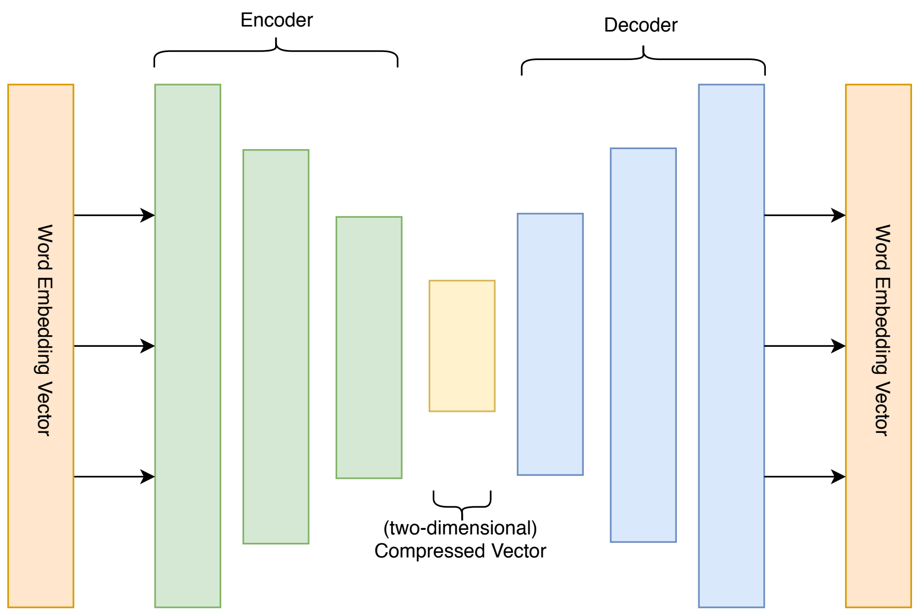


Figure 7.1: Deep Auto-Encoder

BIBLIOGRAPHY

[B^{+}45] Vannevar Bush et al. As we may think. *The atlantic monthly*, 176(1):101–108, 1945.

[BGJM17] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[BK88] Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4-5):291–294, 1988.

[BSA93] Chris Buckley, Gerard Salton, and James Allan. Automatic retrieval with locality information using smart. In *Proceedings of the First Text REtrieval Conference TREC-1*, pages 59–72, 1993.

[BSMS95] Chris Buckley, Amit Singhal, Mandar Mitra, and Gerard Salton. New retrieval approaches using smart: Trec 4. In *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pages 25–48, 1995.

[CNGR08] Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. Selecting good expansion terms for pseudo-relevance feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 243–250. ACM, 2008.

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[Eft96] Efthimis N Efthimiadis. Query expansion. *Annual review of information science and technology (ARIST)*, 31:121–87, 1996.

[EKS^{+}96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[Gir16] Sanjay Surendranath Girija. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. 2016.

[HMT09] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions. 2009.

[Hot33] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[JGB$^+$16] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

[KL51] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[LG14] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 302–308, 2014.

[LM14] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.

[LYRL04] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397, 2004.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[MH08] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[MHM18] L. McInnes, J. Healy, and J. Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv e-prints*, February 2018.

[MHSG18] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.

[Mik12] Tomáš Mikolov. Statistical language models based on neural networks. *Presentation at Google, Mountain View, 2nd April*, 2012.

[MRS08] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. Evaluation in information retrieval. *Introduction to information retrieval*, 1:188–210, 2008.

[MSC$^+$13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[Pea01] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[PGJ18] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. In *NAACL 2018 - Conference of the North American Chapter of the Association for Computational Linguistics*, 2018.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[RES16] Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. Ultradense word embeddings by orthogonal transformation. *arXiv preprint arXiv:1602.07572*, 2016.

[Roc71] Joseph John Rocchio. Relevance feedback in information retrieval. *The SMART retrieval system: experiments in automatic document processing*, pages 313–323, 1971.

[RW94] Stephen E Robertson and Steve Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIRâĂŹ94*, pages 232–241. Springer, 1994.

[RZ+09] Stephen Robertson, Hugo Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

[Sal62] Gerard Salton. Some experiments in the generation of word and document associations. In *Proceedings of the December 4-6, 1962, fall joint computer conference*, pages 234–250. ACM, 1962.

[Sch07] Holger Schwenk. Continuous space language models. *Computer Speech & Language*, 21(3):492–518, 2007.

[SD17] Holger Schwenk and Matthijs Douze. Learning joint multilingual sentence representations with neural machine translation. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 157–167, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[SJ72] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

[SM05] Pascal Soucy and Guy W Mineau. Beyond tfidf weighting for text categorization in the vector space model. In *IJCAI*, volume 5, pages 1130–1135, 2005.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[WYZ16] Yasi Wang, Hongxun Yao, and Sicheng Zhao. Auto-encoder based dimensionality reduction. *Neurocomputing*, 184:232–242, 2016.

[XC17] Jinxi Xu and W Bruce Croft. Quary expansion using local and global document analysis. In *Acm sigir forum*, volume 51, pages 168–175. ACM, 2017.

[YL+99] Yiming Yang, Xin Liu, et al. A re-examination of text categorization methods. In *Sigir*, volume 99, page 99, 1999.

APPENDICES

A Evaluation Results

We detail the the exact numbers of our experiments.

Recall	ES BM25	ES BM25 *	ES BM25 †	Sent2Vec	Sent2Vec *	Sent2Vec †	Word2Vec	Word2Vec *	Word2Vec †
@1	0.065%	0.062%	0.095%	0.074%	0.063%	0.106%	0.066%	0.059%	0.105%
@5	0.222%	0.198%	0.288%	0.238%	0.233%	0.294%	0.223%	0.202%	0.277%
@10	0.321%	0.294%	0.381%	0.345%	0.361%	0.383%	0.349%	0.358%	0.359%
@50	0.75%	0.732%	0.791%	0.96%	0.79%	0.974%	0.728%	0.711%	0.815%
@100	1.115%	1.078%	1.192%	1.409%	1.191%	1.423%	1.274%	1.242%	1.329%
@500	2.793%	2.881%	3.058%	3.389%	3.253%	3.493%	3.280%	3.19%	3.412%
@1000	4.367%	4.401%	4.829%	5.192%	4.989%	5.384%	5.149%	4.979%	5.317%

Table 1: Recall using RCV1. † with Relevance Feedback, *with Pseudo Relevance Feedback

Precision	ES BM25	ES BM25 *	ES BM25 †	Sent2Vec	Sent2Vec *	Sent2Vec †	Word2Vec	Word2Vec *	Word2Vec †
@1	72.75%	71.75%	83.5%	75.5%	73.25%	87.75%	75.25%	71.5%	85.5%
@5	67.9%	68.35%	75.44%	71.7%	71.39%	78.55%	71.95%	69.3%	76.95%
@10	66.27%	66.57%	70.6%	70.17%	70.52%	74.5%	69.32%	69.19%	72.62%
@50	61.52%	62.49%	64.41%	66.56%	66.86%	68.47%	65.51%	65.84%	67.78%
@100	59.13%	60.27%	62.16%	64.59%	64.90%	66.08%	64.29%	64.51%	66.11%
@500	51.58%	53.35%	54.89%	58.71%	58.84%	59.83%	58.85%	59.58%	60.67%
@1000	47.63%	49.42%	50.71%	55.73%	55.85%	56.71%	55.91%	56.73%	57.58%

Table 2: Precision using RCV1. † with Relevance Feedback, *with Pseudo Relevance Feedback

F1	ES BM25	ES BM25 *	ES BM25 †	Sent2Vec	Sent2Vec *	Sent2Vec †	Word2Vec	Word2Vec *	Word2Vec †
@1	0.129%	0.124%	0.189%	0.148%	0.126%	0.211%	0.133%	0.117%	0.209%
@5	0.443%	0.395%	0.573%	0.474%	0.464%	0.586%	0.444%	0.402%	0.553%
@10	0.639%	0.585%	0.758%	0.685%	0.719%	0.762%	0.694%	0.712%	0.714%
@50	1.482%	1.447%	1.563%	1.894%	1.562%	1.92%	1.440%	1.407%	1.609%
@100	2.189%	2.119%	2.34%	2.758%	2.338%	2.786%	2.498%	2.437%	2.606%
@500	5.3%	5.466%	5.795%	6.407%	6.165%	6.600%	6.214%	6.056%	6.461%
@1000	8.001%	8.083%	8.819%	9.499%	9.16%	9.834%	9.429%	9.154%	9.736%

Table 3: F1 using RCV1. † with Relevance Feedback, *with Pseudo Relevance Feedback

Recall	Sent2Vec	ES BM25	Word2Vec
@1	12.08%	8.972%	9.736%
@5	27.25%	21.65%	19.95%
@10	35.15%	27.66%	24.82%
@50	54.27%	41.01%	39.82%
@100	62.95%	48.30%	46.05%
@500	80.95%	63.97%	61.82%
@1000	86.53%	70.53%	69.02%

(a) Recall

Precision	Sent2Vec	ES BM25	Word2Vec
@1	14.90%	10.89%	11.83%
@5	7.239%	5.673%	5.26%
@10	4.803%	3.713%	3.368%
@50	1.549%	1.162%	1.137%
@100	0.907%	0.691%	0.668%
@500	0.237%	0.187%	0.182%
@1000	0.128%	0.103%	0.102%

(b) Precision

F1	Sent2Vec	ES BM25	Word2Vec
@1	13.34%	9.841%	10.68%
@5	11.43%	8.991%	8.325%
@10	8.451%	6.547%	5.933%
@50	3.012%	2.26%	2.211%
@100	1.789%	1.362%	1.318%
@500	0.474%	0.373%	0.364%
@1000	0.255%	0.206%	0.203%

(c) F1-score

Table 4: Recall, Precision and F1-score on basis of DPMA

B German Homonyms

We present our list of German homonyms on which we evaluated the disambiguation. All hits are marked fat.

Word	First Meaning	Second Meaning	Third Meaning	Fourth Meaning	Fifth Meaning
Absatz	**am Schuh**	**Textunterbruch**	**Unternehmensumsatz**	-	-
Abzug	**bei Waffen**	Dampfabzug	**Truppenabzug**	-	-
Akt	**Teil eines Theaterstücks**	nackter Menschen	-	-	-
Angel	Türscharnier	die Angel zum Fischen	**Engel (englisch)**	**Nachname**	-
Apfel	Augapfel	**Obst**	Pferdeapfel	-	-
Atlas	Gebirge	**Kartenwerk**	**Mond**	griechische Mythologie	**-**
Aufzug	**Fahrstuhl**	Teil eines Theaterstücks (Akt)	-	-	-
Bahn	**Eisenbahn**	Schlittschuhbahn	-	-	-
Ball	**zum Spielen**	Tanzveranstaltung	-	-	-
Band	**Musikgruppe**	Klebeband	-	-	-
Bande	**Spielfeldbegrenzung**	**Gruppe von Einbrechern**	-	-	-
Bank	zum Sitzen	**Geldinstitut**	-	-	-
Bar	**Druckmesseinheit**	**Gaststätte**	-	-	-
Barren	**Turngerät**	**Goldbarren**	**Pferdetraining**	-	-
Bart	am Schlüssel	**im Gesicht**	**Bart Simpson**	-	-
Bau	**halbfertiges Gebäude**	Gefängnis	Tierhöhle	-	-
Bauer	**Landwirt**	Vogelkäfig	**Schachfigur**	**Nachname**	-
Becken	**Musikinstrument**	**Wassergefäß**	**anatomisch**	**Landschaft**	-
Bein	Tischbein	**Glieder bei Mensch und Tier**	-	-	-
Berliner	süßes Gebäck	**Einwohner der dt. Hauptstadt**	-	-	-
Bett	Flussbett	**Schlafgelegenheit**	-	-	-
Bienenstich	**Gebäck**	**Einstich des Insekts**	-	-	-
Birne	**Obst**	Glühbirne	-	-	-
Blatt	am Baum	**im Buch**	Spielkarte	**Zeitung**	-
blau	betrunken	**Farbe**	-	-	-
Blinker	**zum Angeln**	**am Auto**	**Zeitschrift**	-	-
Blüte	Falschgeld	**Blüte der Blume**	-	-	-
Bogen	**Briefbogen**	**Pfeilbogen**	Geigenbogen	-	-
Bock	**Bockbier**	**das Tier**	Lust	**Turngerät**	**Nachname**
Boxer	Hunderasse	**Sportler**	-	-	-
Brause	**die Brause zum Trinken**	Dusche	**Unternehmen**	-	-
Bremse	**am Fahrzeug**	das Insekt	-	-	-
Bruder	Mönch	**männl. Geschwister**	-	-	-
Bruch	**Knochenbruch**	**Bruch beim Rechnen**	**Bruch mit jemandem**	**Geografisch**	-
Brücke	**über den Fluss**	der Zahnersatz	Turnübung	-	-
Bulle	Polizist	männl. Rind	**die Bulle**	**Gemeinde in Frankreich**	-
Bund	**Staatenbund**	Hosenbund	-	-	-
Clip	Ohrring	**Musikclip**	**Werbefilm**	-	-
Dame	**Frau**	**Brettspiel**	**Schachfigur**	**Spielkarte**	**Ritterwürde**
Decke	Bettdecke	**Decke des Zimmers**	-	-	-

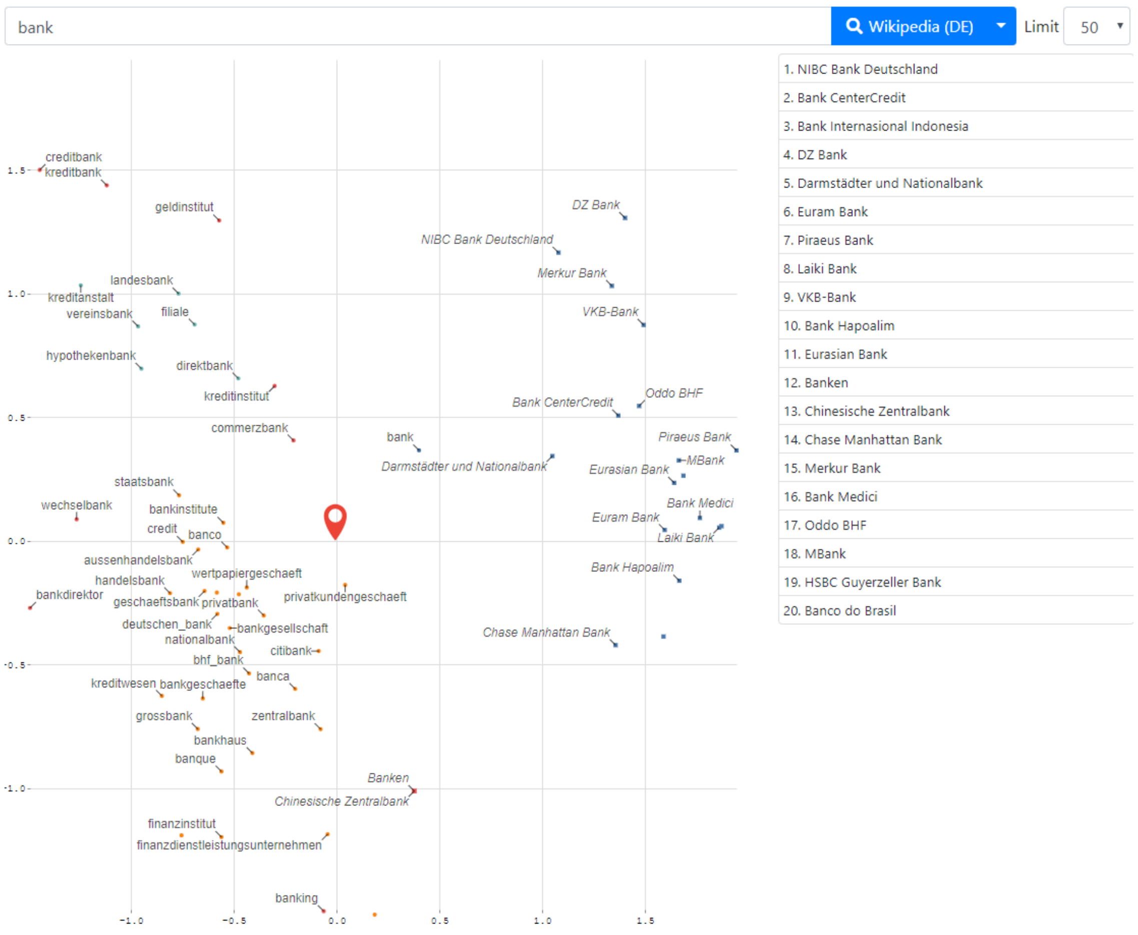
Table 5: German homonym table 1

Word	First Meaning	Second Meaning	Third Meaning	Fourth Meaning	Fifth Meaning
Dichtung	**Gedicht**	Abdichtungsring	-	-	-
Diele	Fußbodenbrett	**Hauseingang**	**Ortsteil von Weener**	-	-
Dietrich	Vorname	**Nachschlüssel**	-	-	-
Dosen	**Mehrzahl von Dose**	**Mehrzahl von Dosis**	-	-	-
Drache	**Ungeheuer**	Papierdrache	-	-	-
Eis	**gefrorenes Wasser**	Eis zum Schlecken	-	-	-
Elle	Ellenbogen	**Maßeinheit**	**Zeitschrift**	-	-
Ente	**das Tier**	falsche Zeitungsnachricht	-	-	-
Erde	Gartenerde	**Welt**	-	-	-
Eselsohr	beim Tier	**im Buch**	**Pilz**	**Zeitschrift**	-
Feder	**Schreibfeder**	**Vogelfeder**	**Sprungfeder**	-	-
Felge	**Reifen**	Turnübung	-	-	-
Fingerhut	**Pflanze**	**der Fingerhut beim Nähen**	-	-	-
Flasche	Versager	**Glasgefäß**	Seilrolle	-	-
Flecken	**ganz kleiner Ort**	Verunreinigung	**Auge**	**Farbtupfer**	-
Fliege	**Tier**	**Krawattenart**	-	-	-
Flügel	Klavier	**Teil des Vogels**	**Gebäudeteil**	-	-
Fuchs	**Raubtier**	rötliches Pferd	-	-	-
Fuge	**Musikstück**	Fuge zwischen Bauteilen	-	-	-
Gabel	**Besteckteil**	**Fahrradteil**	-	-	-
Gang	**langer Flur**	Speisefolge	**wie jemand geht**	**Gruppe, Bande**	-
Gehalt	**Lohn**	**Inhalt**	-	-	-
Geist	**Gespenst**	**Seele**	-	-	-
Gericht	**Mahlzeit**	Rechtswesen	-	-	-
Geschirr	**Tassen und Teller**	Pferdezaum	-	-	-
Golf	**das Auto**	**der Sport**	**die Meeresbucht**	-	-
Grund	**Grund und Boden**	**Ursache**	-	-	-
Hahn	der Wasserhahn	**das Tier**	-	-	-
Harz	**Mittelgebirge**	Baumharz	-	-	-
Heide	**Nichtchrist**	**Landschaft**	-	-	-
Hering	**beim Zelt**	der Fisch	-	-	-
Hose	**Kleidungsstück**	Wirbelwind	-	-	-
Hut	**Kopfbedeckung**	Vorsicht	-	-	-
Kamm	**zum Frisieren**	Berggrat	-	-	-
Kapelle	**kleine Kirche**	Musikergruppe	-	-	-
Karte	**Spielkarte**	**Landkarte**	-	-	-
Kater	**männl. Katze**	Kopf nach einer Feier	-	-	-
Kerze	**Wachskerze**	Turnübung	-	-	-
Kiefer	**Baum**	**Kiefer im Mund**	-	-	-
Kiwi	**Frucht**	**Vogel**	Einwohner Neuseelands	-	-
Koks	**Kohlenart**	ugs. für Kokain	-	-	-
Krebs	**Krankheit**	Tier	Sternzeichen	-	-
Kreuz	**das Symbol**	der Rücken	-	-	-
Krone	**Königsornat**	**Baumteil**	-	-	-
Kunde	**Käufer**	Nachricht	-	-	-
Laster	LKW	**schlechte Angewohnheit**	-	-	-
Läufer	**Sportler**	Teppich	**Schachfigur**	-	-
Leiter	Leiter zum Klettern	**Leiter einer Gruppe**	-	-	-
Linse	**Teil der Kamera**	**Hülsenfrucht**	Teil des Auges	-	-
Löffel	**Besteckteil**	Hasenohr	-	-	-
Lösung	**die Lösung eines Problems**	**Mixtur aus der Chemie**	-	-	-
Mangel	Gerät zum Wäscheplätten	**Fehlen von etw.**	-	-	-
Mark	Knochenmark	alte dt. Währung	-	-	-
Maus	**das Tier**	**das Computereingabegerät**	-	-	-
Messe	**Gottesdienst**	**Ausstellung**	-	-	-
Mine	**Bergwerk**	Schreibstift	-	-	-
Morgenstern	Venus	Waffe	-	-	-
Mühle	**Spiel**	die Getreidemühle	-	-	-
Mutter	**die Mama**	das Schraubenteil	-	-	-
Nadel	**Tannennadel**	**Nähnadel**	-	-	-

Table 6: German homonym table 2

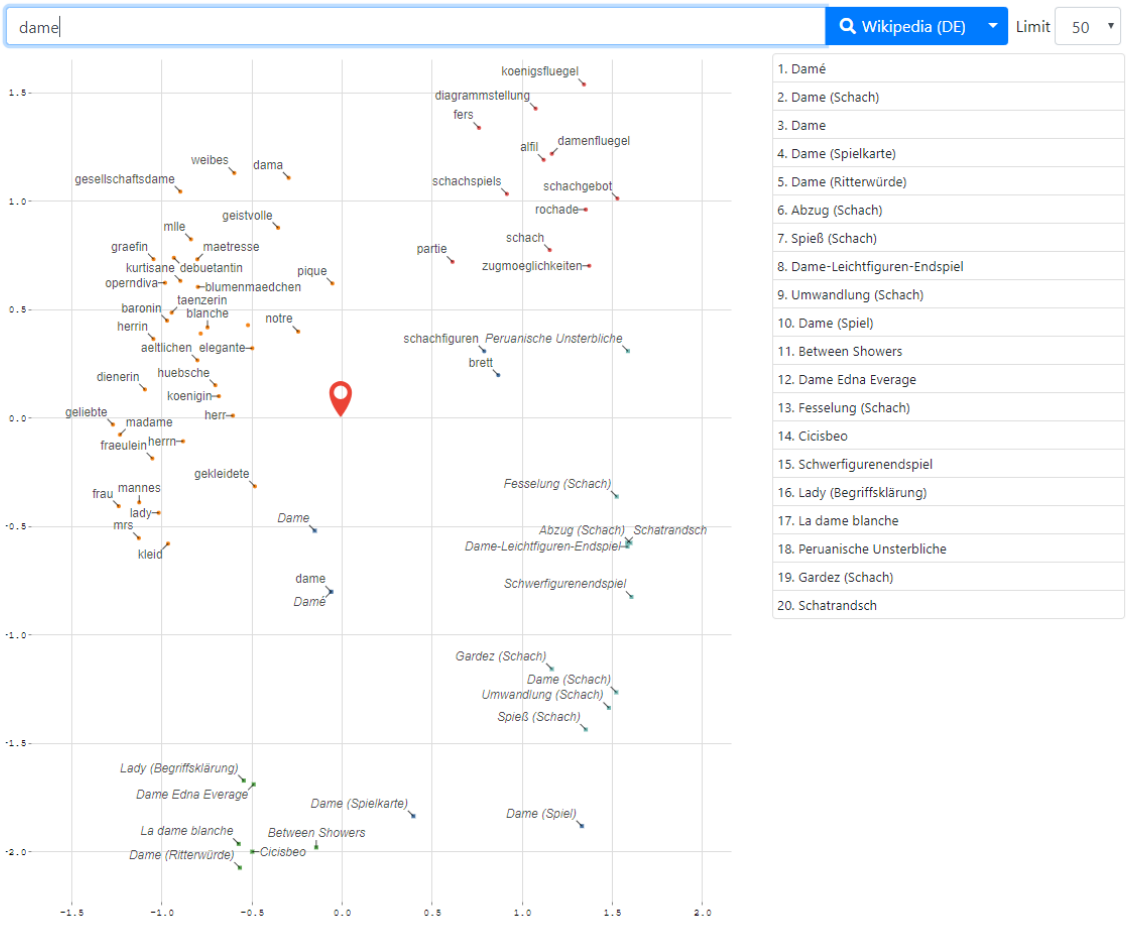
Word	First Meaning	Second Meaning	Third Meaning	Fourth Meaning	Fifth Meaning
Nagel	**Stift**	Fingernagel	-	-	-
Niete	am Gürtel	Fehllos	**Verbindungselement**	-	-
Note	**Schulnote**	**Musiknote**	-	-	-
Orange	**Farbe**	**Frucht**	-	-	-
Otter	**Marder**	Schlangenart	-	-	-
Pass	**Ausweis**	**Gebirgsübergang**	-	-	-
Pfeife	**zum Rauchen**	**Trillerpfeife**	-	-	-
Pflaster	Wundbehandlung	**Straßenbelag**	-	-	-
Pickel	**am Körper**	Eispickel	-	-	-
Platte	**Schallplatte**	**Tischplatte**	Computers	**Kontinentalplatte**	-
Rad	**Fahrrad**	Turnübung	-	-	-
Rasen	**Grünfläche**	**Geschwindigkeitsüberschreitung**	-	-	-
Raupe	**Insekt**	Teil einer Baumaschine	-	-	-
Ring	**Fingerring**	Boxring	**Autobahn**	**Mathematik**	-
Rock	Kleidungsstück	**Musikrichtung**	-	-	-
Rute	**Gerte**	**Hundeschwanz**	-	-	-
Scheibe	Fensterglas	Brotstück	**Sonnenscheibe**	-	-
Schein	Zettel	**Geld**	**Sonnenschein**	**Anschein**	-
Schimmel	**Belag auf Lebensmitteln**	**das Pferd**	-	-	-
Schirm	Pilz	Lampen	**Regenschirm**	-	-

Table 7: German homonym table 3



1. NIBC Bank Deutschland
2. Bank CenterCredit
3. Bank Internasional Indonesia
4. DZ Bank
5. Darmstädter und Nationalbank
6. Euram Bank
7. Piraeus Bank
8. Laiki Bank
9. VKB-Bank
10. Bank Hapoalim
11. Eurasian Bank
12. Banken
13. Chinesische Zentralbank
14. Chase Manhattan Bank
15. Merkur Bank
16. Bank Medici
17. Oddo BHF
18. MBank
19. HSBC Guyerzeller Bank
20. Banco do Brasil

Figure 2: Plot for query "bank"


dame

Wikipedia (DE)

Limit 50

1. Damé
2. Dame (Schach)
3. Dame
4. Dame (Spielkarte)
5. Dame (Ritterwürde)
6. Abzug (Schach)
7. Spieß (Schach)
8. Dame-Leichtfiguren-Endspiel
9. Umwandlung (Schach)
10. Dame (Spiel)
11. Between Showers
12. Dame Edna Everage
13. Fesselung (Schach)
14. Cicisbeo
15. Schwerfigurenendspiel
16. Lady (Begriffsklärung)
17. La dame blanche
18. Peruanische Unterbliche
19. Gardez (Schach)
20. Schatrandsch

Figure 3: Plot for query "dame"