Extending Thesauri Using Word Embeddings and the Intersection Method

Jörg Landthaler, Bernhard Waltl, Dominik Huth, Daniel Braun and Florian Matthes
Software Engineering for Business Information Systems
Department for Informatics
Technical University of Munich
Boltzmannstr. 3, 85748 Garching bei München, Germany
joerg.landthaler@tum.de, b.waltl@tum.de, dominik.huth@tum.de, daniel.braun@tum.de, matthes@in.tum.de

Christoph Stocker and Thomas Geiger
Department for Portals and Collaboration (EM45)
DATEV eG
Fürther Straße 111, 90329 Nürnberg, Germany
christoph.stocker@datev.de, thomas.geiger@datev.de

ABSTRACT

In many legal domains, the amount of available and relevant literature is continuously growing. Legal content providers face the challenge to provide their customers relevant and comprehensive content for search queries on large corpora. However, documents written in natural language contain many synonyms and semantically related concepts. Legal content providers usually maintain thesauri to discover more relevant documents in their search engines. Maintaining a high-quality thesaurus is an expensive, difficult and manual task. The word embeddings technology recently gained a lot of attention for building thesauri from large corpora. We report our experiences on the feasibility to extend thesauri based on a large corpus of German tax law with a focus on synonym relations. Using a simple yet powerful new approach, called intersection method, we can significantly improve and facilitate the extension of thesauri.

Keywords

thesaurus, synsets, word embeddings, word2vec, parameter study, intersection method, tax law

1. INTRODUCTION

Legal content providers offer their customers access to large amounts of different text documents. Clients expect a search engine that serves all relevant search results in an ordered manner with most relevant results at the top. The expectation of users encompasses that all relevant documents are returned be a major task in information retrieval and receives much attention, also in the legal domain, cf. Qiang and Conrad [14] or Grabmair et al. [7]. One possibility is the employment of thesauri to capture the ambiguous nature of natural language. Thesauri capture binary relations such as synonyms or antonyms and some thesauri additionally cover hierarchical relationships. Large general purpose thesauri have been built, for example WordNet (Fellbaum, 1998). Thesauri can be used for search query expansion to increase the recall of information retrieval systems and particularly to include documents that use synonymous words.

Maintaining a thesaurus is expensive and error prone, especially for large thesauri, see for example Dirschl [3]. In the area of computational linguistics, the automated creation of thesauri has been investigated since the 1950s, cf. Section 2. The distributional hypothesis claims that words that share contexts likely have a more similar meaning (perceived by humans) than others. Since 2013 there has been an increasing interest in a technology called word embeddings that combines machine learning (ML) technologies with the distributional hypothesis. In contrast to distributed thesauri calculated based on statistical evaluations, the relatedness of words is calculated in a softer/iterative fashion and is easy to access using the cosine similarity measure.

In this paper we investigate the applicability of the word embeddings technology, in particular the word2vec implementation [16], to support humans to extend an existing thesaurus. While the overall goal is to find new relevant synonym relations that can be suggested to humans to consider for inclusion in the existing thesaurus, one focus of this paper is how word embeddings can be trained such that the quality of the word embeddings is good. The basic assumption is that high-quality word embeddings will lead to better suggestions for synonym relations that are not present in the current thesaurus. Related use cases are the creation of thesauri from scratch or the automated merging with 3rd party thesauri.

Moreover, an unsolved problem is to determine only relevant synonyms given a word, i.e. to build sensible synonym sets (synsets). Most approaches need to resort to

In: Proceedings of the Second Workshop on Automated Semantic Analysis of Information in Legal Text (ASAIL 2017), June 16, 2017, London, UK.

Published at http://ceur-ws.org

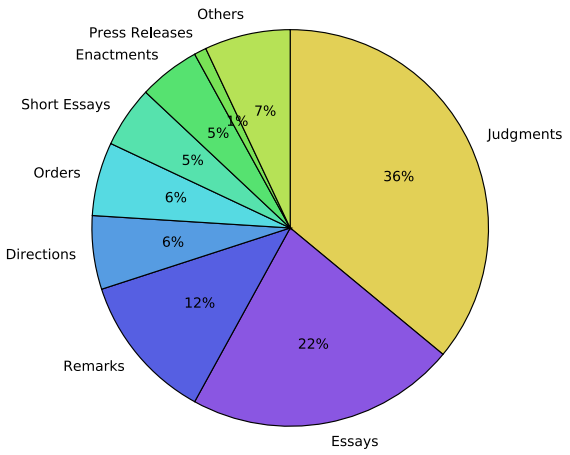

Figure 1: The used text corpus comprises different document types on the topic of German tax law with a total amount of approximately 130.000 documents. The corpus comprises roughly 150 million tokens yielding a vocabulary size of circa half a million words (not pre-processed).

a fixed amount of relevant words or to rely on the identification of a suitable threshold of relatedness. We investigate a straight-forward approach to identify semantically closed synsets without resorting to unreliable thresholds for a large corpus of German tax law and report our experiences. Using a given thesaurus that is manually maintained specifically for this corpus, we conduct parameter studies for the different parameters of word2vec. We propose and evaluate a novel approach to intersect result lists of a relatedness ranking of all words in the vocabulary of the corpus. Multiple word2vec word embeddings models are calculated with different parameters. For a given word (target word), we calculate the relatedness ranking of all words in the corpus and intersect the lists of the first top results among the word embeddings models calculated with different word2vec parameters. We can report promising results of our evaluation of the intersection method with the given corpus and the corresponding manually maintained thesaurus.

The remainder of this work is organized as follows: In Section 2 we give a brief summary of automatic thesauri generation and related work. In Section 3 we give an overview of the corpus and the corresponding manually maintained thesaurus used for all our experiments. The word embeddings technology is introduced in Section 4. We study the different word2vec parameters in Section 5 and present our intersection list method in Section 6. We evaluate the novel intersection method in Section 7 and discuss limitations in Section 8. Finally, Section 9 summarizes our findings and discusses future work.

2. RELATED WORK

The manual creation of thesauri is a very labor intensive process. There have been various attempts to automate the process. A popular approach emerged from the *distributional hypothesis* formulated by Harris in 1954 [9]. The distributional hypothesis claims that words with

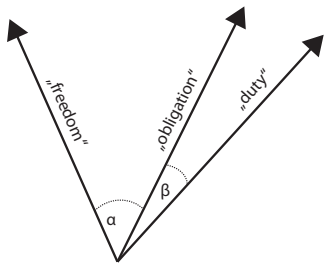

Figure 2: Illustration of the characteristics of word embeddings in two-dimensional space [12]. The angle α between the two vectors for the words *freedom* and *obligation* is larger than the angle β, which reflects that the two words *duty* and *obligation* are semantically more related than *duty* and *freedom* or *obligation* and *freedom*.

Table 1: The occurrence frequency of a token in the corpus has a strong impact on the quality of the resulting embeddings for individual words. Therefore, we choose different evaluation selections of synonym sets from the given thesaurus - that is specifically maintained for this corpus - based on the minimal individual term frequency of the tokens; N is defined as the minimum occurrence frequency of each individual token in the corpus to be included in a synset in a evaluation thesaurus.

N	Synsets	Terms	Terms/Group	Relations
250	260	587	2.26	874
625	125	289	2.31	464
1000	84	195	2.32	312

similar or related meanings tend to occur in similar contexts. The hypothesis is supported by many studies [25, 15, 10, 22].

In 1964, Sparck Jones [27] used the distributional hypothesis to automatically create thesauri using count-based methods. Many followed this approach, for example Grefenstette [8]. Thesauri are useful for several natural language processing problems and much effort has been put into improving distributional thesauri. Rychly and Kilgarriff [26] developed a system called SketchEngine that efficiently generates distributional thesauri from large datasets. They pre-process the dataset and remove word pairs that have nothing in common before the actual calculation is performed. Hence, their algorithm can process a dataset with 2 billion words in less than 2 hours (compared to 300 days without the removal).

In their project JoBimText, Riedl and Biemann [23] use a parallelized approach based on MapReduce and a Pointwise Mutual Information (PMI) measure to improve calculation speed as well as the quality of the generated thesaurus.

Word embeddings can be seen as an evolution of distributional statistics enhanced with machine learning approaches. Traditional distributed thesauri are calculated based on co-occurrence counts, while word embeddings leverage sub-sampling methods that are heavily used in the machine learning domain. Word embeddings provide an easy access to word relatedness via the cosine similarity measure.

Kiela et al. proposed that during the training phase,

word embeddings can be pushed in a particular direction [11] and optimized for detecting relatedness. In the TOEFL synonym task Freitag et al. [4] report considerably better result than for non-specialized embeddings. Thesauri often not only contain synonyms, but also antonyms. Ono et al. [19] presented an approach to detect antonyms, using word embeddings and distributional information. Nguyen et al. [18] improve the discrimination of antonyms and synonyms by integrating lexical contrast into their vector representation. In 2015, Rothe and Schütze [24] presented AutoExtend, an extension of word embeddings, optimized to train embedding vectors for synonym sets (one vector per synset) and their composing lexemes. To the best of our knowledge and in contrast to our intersection method, all approaches use fixed-length result sets or fixed thresholds to build synsets.

Recently, word embeddings have been used for query-expansion for information retrieval directly, i.e. without the creation of knowledge-bases. Examples for such query expansion using word embeddings are Ganguly et al., Zamani et al. and Amer et al. [5, 28, 20]. Query expansion using word embeddings specialized for the legal domain has recently been proposed by Adebayo et al. [1].

3. DATASET & PRE-PROCESSING

We conduct our parameter studies and the evaluation of our intersection method for extending thesauri on a legal texts corpus provided by our industry partner DATEV eG. The corpus comprises different document types on the topic of German tax law, cf. Figure 3. The corpus of 150 million pre-processed tokens yields a vocabulary of circa 180.000 entries. Our industry partner manually maintains a high-quality thesaurus specifically for this corpus including approximately 12.000 synonym sets with around 36.000 terms.

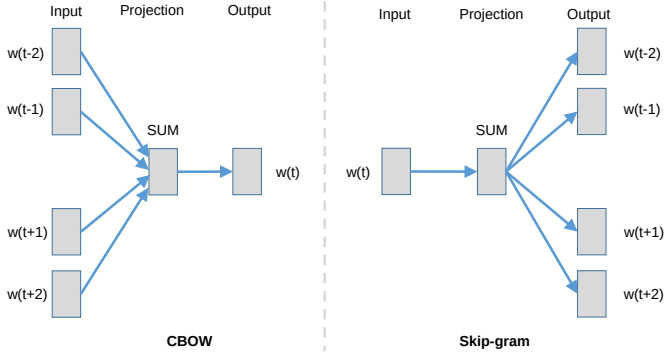

Figure 3: Continuous Bag-of-words (CBOW, CB) and Skip-gram (SG) training model illustrations adapted from [16]. The word2vec algorithm implements both training models. The basic idea behind the two training models is that either a word is used to predict the context of it or the other way around - to use the context to predict a current word. This task is iterated word by word over the corpus. The prediction can be thought of as a classifier from machine learning. The vectors are collected from the weights of the artificial neural network that serves as a classifier. Hence, large amounts of text can be used in an unsupervised fashion to train word embeddings.

We pre-process both the corpus and the thesaurus. The main corpus is pre-processed such that we include only tokens with more than four characters, remove all special characters and punctuation and lowercase all tokens. A single line with all cleaned and white-space-separated tokens is entered into the word2vec algorithm. For the thesaurus, we additionally restrict ourselves to terms that consist of a single token. It is well known that the occurrence frequency of words is crucial to the quality of resulting word embeddings. We extracted three different evaluation sets where all words in the evaluation thesaurus occur at least N={250,650,1000} times in the corpus, see Table 1.

All experiments have been carried out on an Intel Core i5-2500 (4x2333MHz) machine with 8 GB DDR3-1333 RAM running Ubuntu 14.04, Python 2.7.6, Numpy 1.13.1, scipy 0.16.1 and word2vec 0.1c (only versions $\geq$ 0.1c support the iterations parameter).

4. WORD EMBEDDINGS

Word embeddings are a family of algorithms producing dense vectors that represent words in the vocabulary of a corpus. The word embeddings can be trained using the Continuous Bag-of-words (CBOW) or Skip-gram training models depicted in Figure 3.

Word embeddings combine the distributional hypothesis with artificial neural networks [13]. Due to new efficient methods of calculating word embeddings, Mikolov et al. [16], word embeddings for several gigabytes of text data can be calculated within hours. While word embeddings are still considered a Bag-of-words approach, word embeddings do encode the general context of words in dense vectors. Mathematical operations, for example vector addition, can be carried out on the vectors while preserving their inherent semantic characteristics. Mikolov et al [17] show that word embeddings trained on fictional English literature capture semantic relationships among words. We illustrate such semantic relationships encoded in word embeddings in Figure 2. We noticed that relevant characteristics are recognizable even for word embeddings trained on comparably very small training corpora [12], at least regarding text similarity tasks. Hence, we assume that our corpus with 150 million tokens is large enough to produce word embeddings with sufficient quality.

Next, we give a short summary of a selection of the most important implementations to calculate word embeddings:

- **word2vec:** The original C++ implementation of word2vec by Mikolov et al. [16], [17] is very fast. It provides a multi-threaded implementation, but it does not support check-pointing (i.e. resuming computations after stopping).[1]
- **gensim:** The gensim implementation of word2vec provides a Python interface to calculate word embeddings and supports check-pointing.[2]
- **Apache Spark:** Apache Spark includes a Java/Scala implementation of word2vec that can be run in a

[1]https://github.com/kzhai/word2vec, word2vec version 0.1c for MAC OS X, accessed on 22/January/2017
[2]https://github.com/nicholas-leonard/word2vec, accessed on 22/January/2017

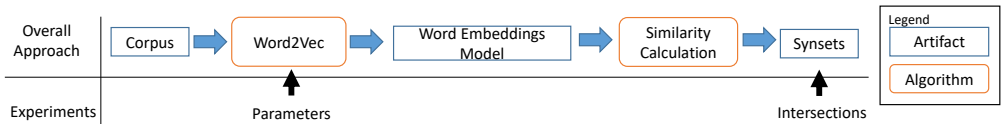

Figure 4: Illustration of our overall approach: word2vec parameters need to be chosen. Our intersection method can be applied after the (repeated) calculation of (fixed length) synsets.

Hadoop cluster.[3]

- **deeplearning4j:** This Java implementation is embedded in the general deeplearning4j framework for machine learning and is similar to gensim but implemented for usage with Java and Scala.[4]
- **GloVe:** GloVe vectors, presented by Pennington et al., [21] are a count-based algorithm implemented in C. GloVe vectors are similar to the word2vec embeddings, but optimize a different objective function.[5]
- **Torch:** Torch is, similar to the well-known Tensorflow framework, a deep learning and scientific computing framework that provides a Lua interface to a word2vec implementation.[6]
- **Tensorflow:** Tensorflow is a deep learning framework provided by Google and offers (among others) a Python interface to its own word2vec implementation.[7]

5. WORD2VEC PARAMETERS

The word2vec implementation has a large number of parameters. For the most relevant parameters, we conducted a parameter study using a manually maintained thesaurus as the ground truth for an evaluation of the quality of the resulting word embeddings. While a thesaurus by nature cannot be perfectly sharp, we assume that relations identified by humans have sufficient truth and by using a large number of relations identified by humans our assumption is that this is sufficient to identify general trends. The following list gives an overview of the most important parameters:

- **Size (Dimensionality):** The size of the resulting vectors is chosen manually. From an information entropy point of view this value needs to be large enough so that all relevant information can be encoded in the vectors. However, the larger the vector size is chosen, the more computationally expensive training and subsequent calculations become.
- **Window Size:** The window size is a training parameter that defines the size of the context window around each word during the training phase. Arguing from a statistical linguistics point of view, large (word-)distances between two words (for example in two consecutive sentences) usually lead to less influence of the words on each other [6].
- **Iterations (I)):** The iterations parameter defines the number of iterations over the full corpus and can be thought of as an artificial enlargement of the corpus. The choice for this parameter heavily depends on the corpus size. A larger number of iterations is particularly useful for small corpora.
- **Minimum Count:** The occurrence frequency of individual tokens has a strong impact on the quality of the resulting word embeddings. Using the minimum count parameter, one can control that words occur sufficiently often in the corpus. The downside of this parameter is that words in the vocabulary that do not occur often enough in the corpus will not have a vector.
- **Alpha:** The initial learning rate is a parameter that is derived from artificial neural network training and not investigated, because we assume that it is very specific to a concrete dataset.
- **Negative:** The number of negative examples presented during the training. Consult [6] for a comprehensive explanation.
- **CBOW or Skip-gram:** The two possible training models that can be used to train word embeddings with word2vec. Either the current word is used to predict the context of the current word or vice versa the context is used to predict the current word, cf. Figure 3. In our experiments the CBOW model results faster in high quality word embeddings in less training time.

We assume that the vector size, window size, negative and iterations parameters are the most important parameters for the creation or extension of a thesaurus given a corpus. We set the minimum count parameter to zero, because we want a vector for each word present in the corpus.

The manually maintained thesaurus for our corpus contains groups of synonymously used words. A simple example for such a group of synonyms is *(lawsuit, case, dispute)*. We are interested in how the vector size, window size, iterations and negative parameters affect similarity score lists calculated based on word embeddings trained by word2vec. We introduce the *RP-Score* measure that measures the average positions of synonyms in ranking lists of the terms. The RP-Score provides a measure to compare the relateness of two words obtained from humans and obtained by our word2vec approach. In contrast to the mean reciprocal rank (MRR), the RP-Score is not

[3]https://spark.apache.org/, accessed on 22/January/2017
[4]https://deeplearning4j.org/word2vec.html, accessed on 22/January/2017
[5]http://nlp.stanford.edu/projects/glove/, accessed on 22/January/2017
[6]https://github.com/yoonkim/word2vec_torch, accessed on 22/January/2017
[7]https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/tutorials/word2vec, accessed on 22/January/2017

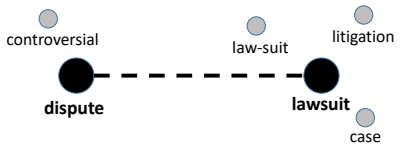

Figure 5: The cosine similarity distance measure is symmetric between (the word embeddings vectors of) two words. The ranking positions of two words can be asymmetric, because other words can be closer to one of the two words, but more distant to the other word, cf. Table 2.

Table 2: For our parameter study we use a measure that we call RP-Score. Using the cosine similarity measure, we compute an ordered list of the relatedness of all words in the vocabulary to one selected word from the vocabulary (target word). For three target words, the five most related words are listed. The word *dispute* has the ranking position (RP) 4 for the target word *lawsuit*. The RP-Score for this example synset is $\frac{1+4+2+5+3+4}{6} \approx 3.16$.

RP	lawsuit	case	dispute
1	**case**	trial	controversial
2	law-suit	**lawsuit**	trial
3	litigation	litigation	**lawsuit**
4	**dispute**	law-suit	**case**
5	trial	**dispute**	litigation

normalized to 1 and provides a more intuitive understanding for the quality of a word embeddings model.

The RP-Score measure is calculated as follows: For all target words in a synset we calculate a sorted list of all words in the vocabulary using the cosine similarity measure. The ranking list is ordered and *most related* words are at the top of the list. We determine the position for each combination of two words in a synset and accumulate all ranking positions. We perform this calculation on all synsets and finally divide the value by the total number of relations among all words in a synset and aggregate among all synsets. $RP_{w_1}(w_2)$ is defined as the position of w_2 in the result list of w_1.

$$\text{RP\_Score} := \sum_{s \in synsets} \frac{\sum_{w_1, w_2 \in s, w_1 \neq w_2} RP_{w_1}(w_2)}{|s|(|s|-1)}$$

Note that the RP-Scores are not a sharp measure in our evaluation. On the one hand, a thesaurus maintained by humans can lack (and contain) similarity relations that are included (or ignored) by an unsupervised word embeddings calculation. For example, spelling mistakes are often not contained in a thesaurus but detected by our overall approach. Wrongly typed words can be good candidates for an inclusion in a thesaurus, because documents like judgments cannot be corrected. Nevertheless, we are able to observe reasonable results in our parameter studies, due to the averaging of the RP-Score over a larger number of evaluation synsets.

For all parameters investigated, we applied both CBOW and Skip-gram model. For all experiments (unless otherwise stated) we used a vector size of 300 and ran 20 iterations.

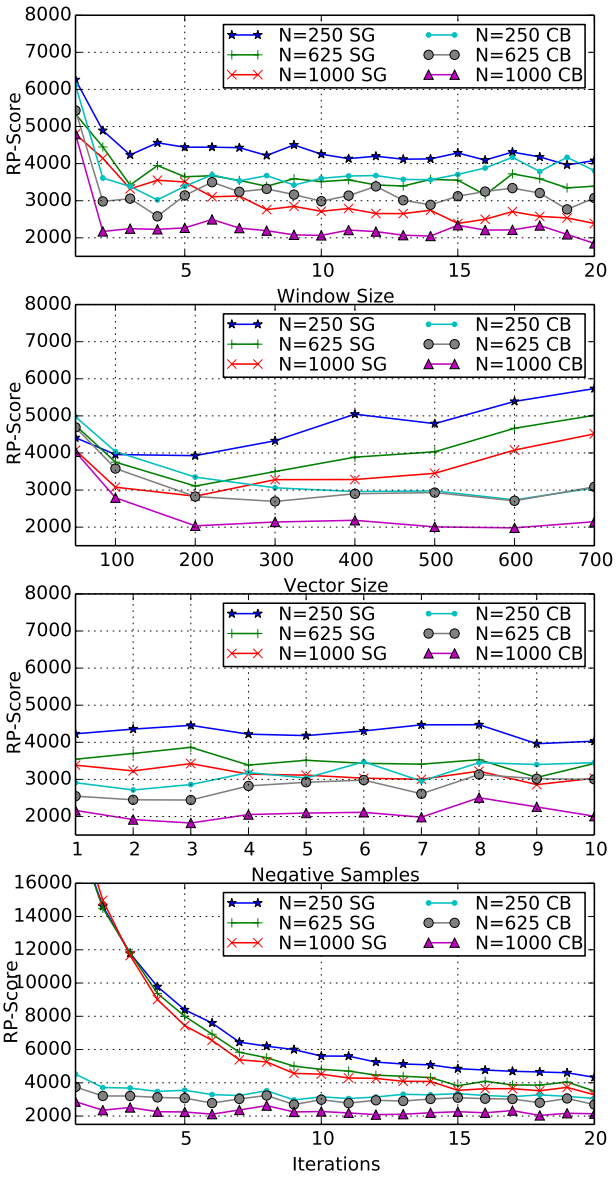

Figure 6: Parameter studies of different relevant word2vec parameters: *window size*, *vector size*, *negative samples* and number of *iterations*. For all parameters the computing costs increase with larger parameter values (cf. Figure 8) and a trade-off between computing costs and quality is inevitable. We measure the impact of the different word2vec parameters on the quality of the resulting word embeddings using the RP-Score with a thesaurus that is manually and specifically maintained for the given text corpus. Good parameter choices have a reasonably small RP-Score. For example, the default window size of 5 from the word2vec implementation is a good choice already. Note that the CBOW training model results in better RP-Scores while having smaller runtime values for equal parameter choices. Best viewed in color.

For the window size parameter study, the window parameter

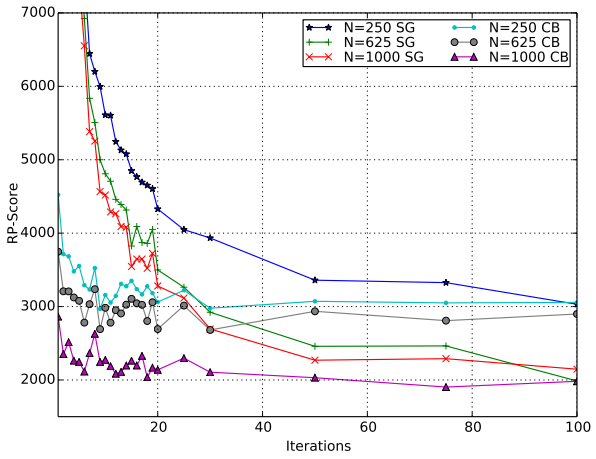

Figure 7: We evaluated the word2vec iterations parameter for larger values: Reasonable quality of the word embeddings can - for this corpus - be obtained with I=20, but slight improvements, especially for individual evaluation thesaurus and training parameter choices, can be achieved with larger iteration parameter choices. Note that for a larger number of iterations we conducted fewer runs, hence the perceived relative smoothness for I>20 is not representative. Best viewed in color.

runs from 1 to 20. For the vector size parameter study, we increase the vector size from 100 to 700 by 100, plus one run with vector size 50. For the number of iterations, we run 1 to 20 iterations incrementally and only selected samples up to 100 iterations, cf. Figures 6 and 7. For the negative (sampling size) we consecutively vary the negative parameter between 1 and 10. For each parameter study, we calculate the RP-Scores using the three different evaluation thesauri as described in Section 3 for each computed word embeddings model.

6. INTERSECTION METHOD

Finding good parameters for the word2vec algorithm is an important step towards the creation or extension of thesauri. However, several challenges remain. A ranking with respect to the similarity score of all words in the vocabulary is not enough. One major challenge is to decide which words should be included in a synset and which should not. One possibility is to include a fixed number of related words according to the ranking list, for example the first ten. Another possibility is to define certain thresholds for the similarity score between two words. However, the similarity scores are very different among result lists for different target words. For example, the first related result in a ranking list for the word *law* might have a similarity score of 0.7 while the first result for the word *doctor* could have a similarity score of 0.4 while both are considered as true synonyms by humans.

During our experiments with the parameters of word2vec, we recognized that the result lists differ substantially from one word2vec parameter selection to another. This led us to the idea of calculating intersections of the result lists for target words. This approach has two advantages at once.

Table 3: Artificial example to illustrate our intersection method. We calculate the k=5 most related words from the word embeddings models for the word *car*. The left column shows the most related words to the target word *car* for the word embeddings model calculated with 20 iterations. The center column shows the most related words to the word *car* for the word embeddings model calculated with 19 iterations. The right column displays the intersection of the left and center columns. Irrelevant words (for example *animal* or *chair*) are dropped from the final result list by the intersection, because they tend to change their positions heavily among result lists from word embeddings models calculated with different parameters.

Iter.=20	Iter.=19	Intersection
car	**car**	**car**
auto	**automobile**	**auto**
automobile	chair	**automobile**
animal	sheep	
doctor	egg	
tree	**auto**	

First, we do not need to define a threshold or fixed number of words to form a synset for a given word. Moreover, we experience that bad results are filtered out because their positions in ranking lists vary a lot. Table 3 illustrates the intersection approach. From a different point of view, for a specific word, words that are always at the top of different ranking lists form a fix-point set that is stable and as we will show in Section 7 returns higher quality results. We also found that the resulting intersection lists have different sizes that reflect that the approach identifies sensible synset sizes. For example, specific words like *scrapping bonus* result in few words in the intersection lists, while words like *doctor* yield a large number of synonyms in the intersection lists. This behavior reflects the human perception of good synsets, too.

The intersection of result lists of the size of the vocabulary results in vectors of the size of the vocabulary. Hence, we use a fixed number of the first k entries of a result list serving as an intermediate result. This additional parameter serves as an upper bound for the size of returned synsets. Due to the strong variation among result lists obtained from different word embeddings models, this upper limit is not assumed most of the time.

7. EVALUATION

Our evaluation is an attempt to quantify the perception of humans that synsets obtained by the intersection method have higher quality than synsets obtained using thresholds. We compare precision/recall values calculated per synset and accumulate the individual results. We use precision (P), recall (R) and F1-Score (F1) defined as follows:

$$P := \frac{TP}{TP+FP} \quad R := \frac{TP}{TP+FN} \quad F1 := 2 * \frac{P*R}{P+R}$$

True positives (TP) are present in the evaluation thesaurus and in the result of our method. False positives (FP) are present in the result of our method, but not in

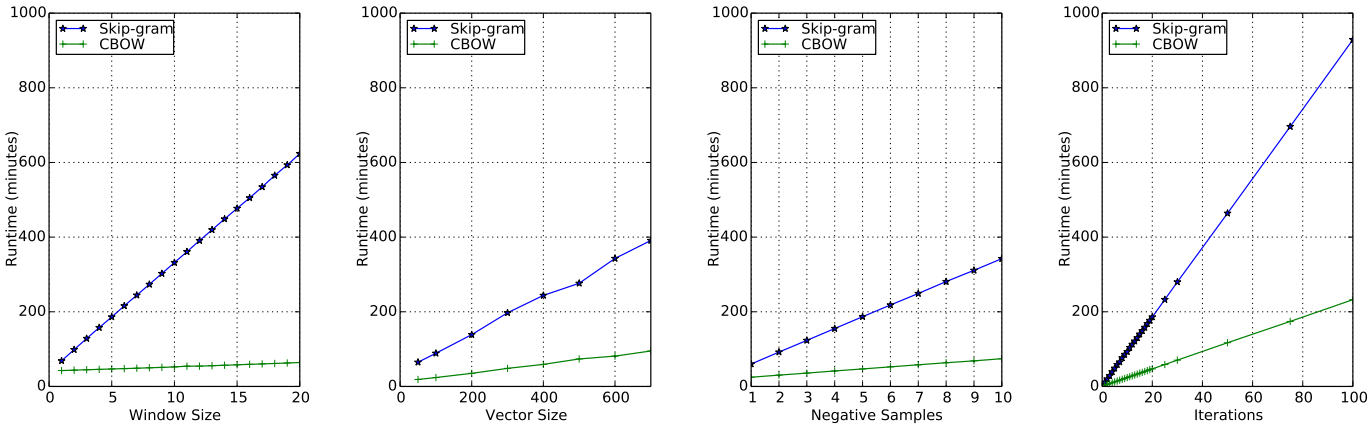

Figure 8: The computational costs in terms of runtime for the different parameter selections shown in Figure 6 increase linearly for all investigated parameter values (for the linearly increased parameter values from the parameter study). The calculation of reasonably good word embeddings models takes around one hour for the given dataset on the machine described in Section 3. Note that not all word embeddings models have been calculated in a sequential order of time. Thus, we can exclude cache effects.

the synset of the evaluation thesaurus. False negatives (FN) are present in the synset of the evaluation thesaurus, but not in the result lists. First, we use k=30 as the intermediate result list length.

Table 4: The synsets returned by our intersection method for the target words *Abwrackprämie (scrapping bonus)* and *Arzt (doctor)*. Here, the intersection of the result lists calculated from five different word embeddings models on our training corpus has been carried out (word embeddings models with iterations parameter I=20 to I=15 from the parameter study).

Tax law-specific example: Abwrackprämie (scrapping bonus)
Abwrackhilfe (scrapping help)
Umweltprämie (environmental bonus)
Prämie (bonus)

General example (unspecific to legal domain): Arzt (doctor)
Zahnarzt (dentist)
Chefarzt (chief physician)
Facharzt (consulting physician)
Rechtsanwalt (lawyer)
Assistenzarzt (assistant doctor)
Oberarzt (senior physician)
Tierarzt (veterinarian)
Kassenarzt (preferred provider)
Laborarzt (camp physician)
Krankenhausarzt (hospital physician)

In Figure 9, we present the precision/recall curves for successively more intersection steps. The points labeled with *20* represent the precision recall values comparing the evaluation thesauri with a fixed number of results for a target word. This is equivalent to an approach using a fixed synset size and serves as a baseline. The other precision/recall data points are calculated by intersecting the results from the models with *20*, *19*, *18*, etc. iterations. For $I = 19$, two lists are intersected obtained from the models with *20* and *19* iterations. Subsequently, result lists from one additional word embeddings model are included per data point. All used models were calculated during the parameter study described in Section 5.

The more that result lists are intersected, the better the precision. The increasing precision reflects the opinion of experts that intersected result lists are much better than fixed length synsets calculated by word2vec and cosine similarity. The recall drops slightly, which can be expected, because the more lists are intersected the fewer entries remain in the final result lists, and a suitable trade-off needs to be chosen. The overall low values of the precision stem from the large number of false positives, cf. Table 5, i.e. results obtained by the word2vec approach, but not present in the thesaurus. Remember that a manually maintained thesaurus is not complete. For example, many spelling errors in the corpus are not reflected in the thesaurus. Since the creation of a high-quality thesaurus by humans is difficult and expensive, it cannot be excepted that all sensible synonym relations for a huge corpus are present in the thesaurus. We therefore show real results obtained using the intersection method from the corpus for two exemplary target words from our training corpus, in Table 4 (one tax-law specific example and one general, not law-specific example).

The precision values are very low, but an optimization of the precision values is not the goal here. The relevant measure is the change in precision (and recall). Also, the parameter k (the size of the intermediate result list lengths) needs to be chosen manually. We conducted a parameter study on the parameter k, see Figure 10. The recall drops quite a lot in the beginning, but then stabalizes for k>30 while the precision continues to increase. However, the larger the individual result lists become, the more computing resources are necessary to calculate the intersections. Again, a good trade-off needs to be chosen.

Table 5: The detailed F1-Score (F1), precision (P) and recall (R) values for the results depicted in Figure 9 for N=250. For I=20 no intersections have been calculated (baseline), for I=15 intersections of the result lists from five different word embeddings models have been computed. True positives (TP), false positives (FP) and false negatives (FN) values are listed. The precision values are very small, because of the high number of false positives, which reflects that word2vec returns much more words than a human made thesaurus contains. Note that high precision and recall values are not the goal nor necessary for this evaluation, but the relative improvement. In fact, it is desired to find additional relevant words for inclusion in the thesaurus.

I	F1	P	R	TP	FP	FN
CBOW						
20	0.051	0.027	0.543	475	17135	399
19	0.060	0.032	0.538	470	14376	404
18	0.064	0.034	0.535	468	13246	406
17	0.067	0.036	0.532	465	12515	409
16	0.069	0.037	0.524	458	12002	416
15	0.070	0.038	0.519	454	11599	420
14	0.072	0.039	0.517	452	11267	422
13	0.073	0.039	0.514	449	10963	425
12	0.075	0.040	0.513	448	10687	426
Skip-gram						
20	0.052	0.027	0.553	483	17127	391
19	0.060	0.032	0.538	470	14212	404
18	0.065	0.034	0.530	463	12974	411
17	0.067	0.036	0.522	456	12198	418
16	0.069	0.037	0.515	450	11653	424
15	0.072	0.038	0.514	449	11232	425
14	0.073	0.039	0.508	444	10852	430
13	0.075	0.040	0.505	441	10506	433
12	0.076	0.041	0.497	434	10167	440

8. DISCUSSION & LIMITATIONS

Building synsets by intersecting result lists from word embeddings models calculated with different parameters is a good step towards an automated creation or extension of thesauri. However, there are several limitations to the overall process. An open question remaining is the selection of the different word embeddings models that are used to calculate intersection lists. An efficient approach for creating different word embeddings models would be to dump word embeddings models at checkpoints with the iterations parameter. After each iteration, a word embeddings model could be saved to disk. However, the word2vec algorithm does not support check-pointing (i.e. resuming training after stopping the algorithm) out of the box. Other implementations, such as gensim, do support check-pointing. In our experience, the intersections from word embeddings models calculated with varying more than one parameter give better results. This might be due to larger variation among word embeddings models that differ in multiple parameters (for example, iterations and vector size). However, we did not evaluate this in a structured fashion. Also, we did not evaluate the results with experts in a structured way. This is an important next step for the future. Another issue for the automated creation of thesauri from scratch is that with the intersection method we can calculate synsets for given

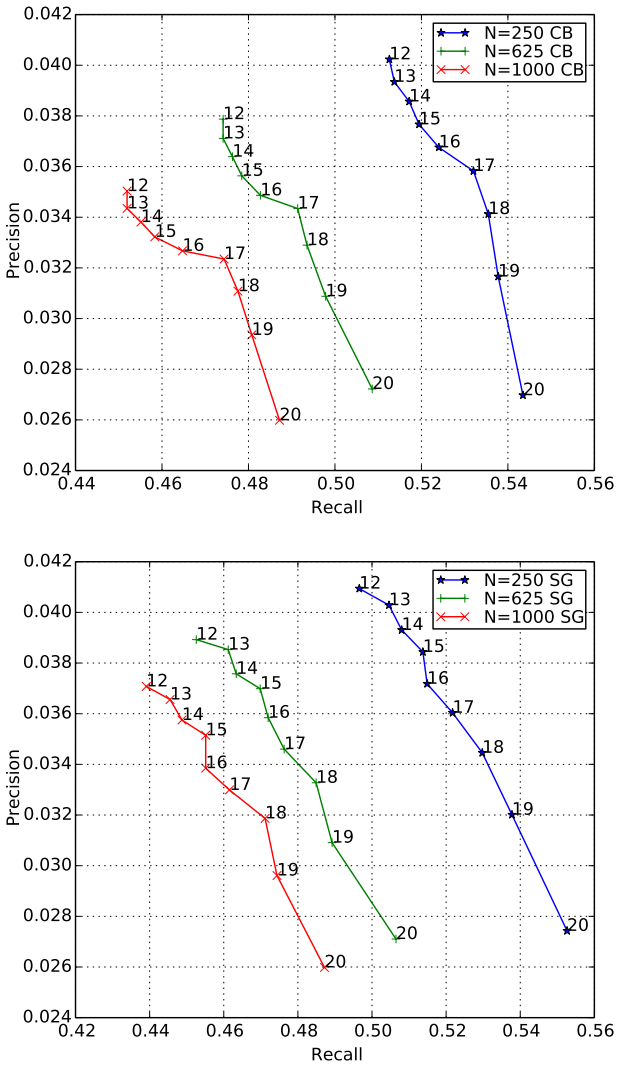

Figure 9: The precision recall curves show promising results for our intersection method. The upper graph shows the results for the CBOW model, the lower graph for the Skip-gram model. In both graphs, each data point is labeled with the iterations parameter (I) that the corresponding - additionally included - word embeddings model has been calculated with. For I=20 no intersection between result lists for the words of the synset has been calculated yet. For M=19 the intersection of the result lists from I=20 and I=19 have been computed. Subsequently, more and more result lists are intersected from the word embeddings models with I=18 to I=12 iterations. More intersections lead to significant increased precision while the recall only drops slightly. Please note the description for Table 5 regarding the low precision values. Best viewed in color.

words, but we did not tackle the problem of selecting the target words for which synsets are to be calculated. The word2vec implementation comes with a clustering algorithm that can be used to identify different clusters of synonymous words. However, the question of which

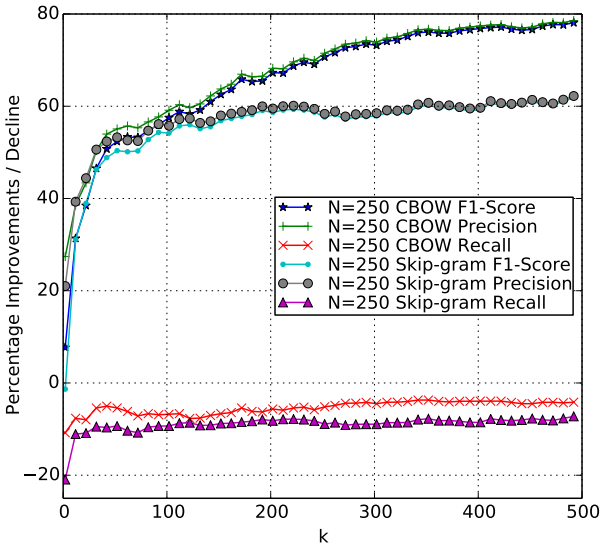

Figure 10: We evaluate the size of the result lists that are the input to our intersection method. Up to k=40 the recall drops and stabalizes for k>40 while precision and F1-Score constantly increases. A decent size of final synonyms that are presented to humans to consider an inclusion in the thesaurus is desirable. Therefore, we suggest to use a value of k around 30 to 40 for this particular corpus and thesaurus and can report a 50 percent improvement in F1-Score. Larger values of k lead to even larger percental improvements. Best viewed in color.

clusters can be considered relevant remains. For a practical application in a search context, one starting point could be to find synsets for the words in the most frequent search queries from the users. Search query expansion methods based on word embeddings could overtake the manual creation of thesauri in many cases. However, the manual creation of a thesaurus allows for more manual control over search results.

Note that the word2vec implementation is deterministic in the sense that the exact same corpus and parameter selection results in identical word embeddings models.

In our experience, abbreviations tend to give very bad results using our approach. We believe that using existing abbreviation lists is the better option in this case. Besides abbreviations, open compound words (terms consisting of multiple words) are problematic, too. Calculating all combinations of multiple words is very time- and resource-intensive for large corpora. One solution is to convert known open compound words as single tokens before entering them into the word2vec algorithm. Open compound words can be detected, for example, using the phrase2vec algorithm and implementation that ships with the word2vec implementation.

So far, we did not use any sufficiently reliable word sense disambiguation algorithm in our approach (such as POS-tagging). Hence, each token can map only to one vector for all meanings. Moreover, our approach sometimes has interesting effects on the meaning of individual words. For example, the result list to the word *doctor* yields a list of mostly different types of doctors (dentist, chief of medicine, ENT physician and the alike, where the corresponding German compound words are closed compound words). In contrast to this, the result list for the word physician returns all sorts of different job types, for example, lawyer, teacher and the alike. We also did not include antonyms or a discrimination of synonyms and other types of relations in thesauri.

9. CONCLUSION & FUTURE WORK

We investigated an approach to extending thesauri for a large German tax law text corpus based on word embeddings, in particular with word2vec. These newly detected relations can be presented to experts for possible inclusion in the thesaurus. We use a large existing, manually and specifically for this corpus maintained, thesaurus to identify good parameters for word2vec. We introduced a novel intersection method that intersects result lists of related words calculated by word2vec and cosine similarity for given target words. We showed that the intersection method returns synonym sets with higher F1-score and precision. The intersection approach provides an elegant solution to mitigate the problems associated with fixed length or threshold based approaches to decide on synset sizes.

For the future, an interesting question is to understand and quantify the impact of extending the corpus before entering the word2vec algorithm. It remains unclear why certain resulting synonym sets encompass certain specific meanings (doctors are related to different types of doctors and physicians are related to other professions). Related to that, a word sense disambiguation could significantly improve the quality of resulting synonym sets. We plan to include label propagation approaches based on word embeddings and to evaluate results with experts. Finally, a feasible solution to deal with open compound words (n-grams) and the automated selection of target words (words that synsets will be calculated for) are important. Different similarity measures, investigated for distributional thesauri methods, for example by Bullinaria and Levy 2007 [2], could be investigated. Finally, it will be necessary to evaluate the suitability of additionally suggested synonym relations by our approach with humans.

10. ACKNOWLEDGMENTS

The authors thank DATEV eG for providing the dataset and all people involved for their support.

11. REFERENCES

[1] G. B. Adebayo Kolawole John, Luigi Di Caro and C. Bartolini. - an approach to information retrieval and question answering in the legal domain. In *Proceedings of the 10th International Workshop on Juris-informatics (JURISIN 2016)*, 2016.

[2] J. A. Bullinaria and J. P. Levy. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39(3):510–526, Aug 2007.

[3] C. Dirschl. Thesaurus generation and usage at wolters kluwer germany. *IRIS: Internationales Rechtsinformatik Symposium, Salzburg, Austria*, 2016.

[4] D. Freitag, M. Blume, J. Byrnes, E. Chow, S. Kapadia, R. Rohwer, and Z. Wang. New experiments in distributional representations of synonymy. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 25–32. Association for Computational Linguistics, 2005.

[5] D. Ganguly, D. Roy, M. Mitra, and G. J. Jones. Word embedding based generalized language model for information retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 795–798, New York, NY, USA, 2015. ACM.

[6] Y. Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420, 2016.

[7] M. Grabmair, K. D. Ashley, R. Chen, P. Sureshkumar, C. Wang, E. Nyberg, and V. R. Walker. Introducing luima: an experiment in legal conceptual retrieval of vaccine injury decisions using a uima type system and tools. In *Proceedings of the 15th International Conference on Artificial Intelligence and Law*, pages 69–78. ACM, 2015.

[8] G. Grefenstette. *Explorations in automatic thesaurus discovery*, volume 278. Springer Science & Business Media, 1994.

[9] Z. S. Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[10] J. Karlgren and M. Sahlgren. From words to understanding. *Foundations of Real-World Intelligence*, pages 294–308, 2001.

[11] D. Kiela, F. Hill, and S. Clark. Specializing word embeddings for similarity or relatedness. In L. Màrquez, C. Callison-Burch, J. Su, D. Pighin, and Y. Marton, editors, *EMNLP*, pages 2044–2048. The Association for Computational Linguistics, 2015.

[12] J. Landthaler, B. Waltl, P. Holl, and F. Matthes. Extending full text search for legal document collections using word embeddings. In F. Bex and S. Villata, editors, *Legal Knowledge and Information Systems - JURIX 2016: The Twenty-Ninth Annual Conference*, volume 294 of *Frontiers in Artificial Intelligence and Applications*, pages 73–82. IOS Press, 2016.

[13] O. Levy and Y. Goldberg. Neural word embedding as implicit matrix factorization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc., 2014.

[14] Q. Lu and J. G. Conrad. Next generation legal search-it's already here. *Vox Populii blog, Legal Information Institute (LII), Cornell University*, 2013.

[15] S. McDonald and M. Ramscar. Testing the distributional hypothesis: The influence of context on judgements of semantic similarity. In *Proceedings of the 23rd annual conference of the cognitive science society*, pages 611–616, 2001.

[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[17] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[18] K. A. Nguyen, S. Schulte im Walde, and N. T. Vu. Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 454–459, Berlin, Germany, 2016. Association for Computational Linguistics.

[19] M. Ono, M. Miwa, and Y. Sasaki. Word embedding-based antonym detection using thesauri and distributional information. In R. Mihalcea, J. Y. Chai, and A. Sarkar, editors, *HLT-NAACL*, pages 984–989. The Association for Computational Linguistics, 2015.

[20] N. Ould Amer, P. Mulhem, and M. Géry. Toward Word Embedding for Personalized Information Retrieval. In *Neu-IR: The SIGIR 2016 Workshop on Neural Information Retrieval*, volume abs/1606.06991, Pisa, Italy, July 2016.

[21] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[22] P. Resnik. Wordnet and distributional analysis: A class-based approach to lexical discovery. In *AAAI workshop on statistically-based natural language processing techniques*, pages 56–64, 1992.

[23] M. Riedl and C. Biemann. Scaling to large3 data: An efficient and effective method to compute distributional thesauri. In *Conference on Empirical Methods on Natural Language Processing*, pages 884–890, 2013.

[24] S. Rothe and H. Schütze. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1793–1803, Beijing, China, July 2015. Association for Computational Linguistics.

[25] H. Rubenstein and J. B. Goodenough. Contextual correlates of synonymy. *Communications of the ACM*, 8(10):627–633, Oct. 1965.

[26] P. Rychlý and A. Kilgarriff. An efficient algorithm for building a distributional thesaurus (and other sketch engine developments). In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 41–44, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[27] K. Sparck Jones. *Synonymy and semantic classification*. PhD thesis, University of Edinburgh, 1964.

[28] H. Zamani and W. B. Croft. Embedding-based query language models. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, ICTIR '16, pages 147–156, New York, NY, USA, 2016. ACM.