TECHNISCHE UNIVERSITÄT DRESDEN

Automated Thesaurus Generation Using Word2vec

Name	Sebastian Humenda
Mentor	Dr.-Ing. Maik Thiele
Supervisor	Prof. Dr.-Ing. Wolfgang Lehner
Published at	Technische Universität Dresden, Germany
Date	01/11/2017

Contents

1 Introduction

Thesauruses are applied in diverse use cases, ranging from general reformulation of text passages by authors to information retrieval applications for recall improvements of search queries. Thesauruses are usually hand-written dictionaries and are hence costly to produce. When creating domain-specific thesauruses, the costs are even higher, because the linguist needs additional domain knowledge as well.

In this work, the thesaurus generator ALT (**a**lways **l**earning **t**hesaurus) will be introduced. It uses a neural network called Word2vec, which is able to learn word relationships. Traditionally, thesauruses have been written by hand by dedicated linguists. While these usually have a high quality, it requires a lot of time to create and edit thesauruses manually, resulting in high costs for this kind of dictionaries.

Word2vec is a software to learn word relations using a shallow neural network. It does so by representing each word as a high-dimensional word vector and captures word relations by the similarity of these vectors, for instance through proximity. Terms can be both semantically or syntactically related. The models are independent of a particular language and hence training of these models can happen with any sufficiently large text corpus. The neural network learns unsupervised and can improve the quality of the resulting vectors by learning from larger text corpora.

To improve the quality of the generated thesauruses, word lists from FreeDict are used to filter the suggestions and to limit the number of words in the thesaurus. The FreeDict project strives to provide as many free and open (source) dictionaries as possible. It uses a standardized data format, which makes it suitable for human and machine use. It decouples the data from the actual use case in a format, which can encode human speech with all irregularities, called TEI.[1] The generated thesaurus dictionaries should be usable within the FreeDict project and should be therefore only freely licensed.[2] That implies in particular that existing corpora consisting of newspapers or other copyrighted material cannot be used.

[1] TEI is XML-based and standardized by the organization with the same name *Text Encoding Initiative*.
[2] **Free** is used here in the spirit of the Free Software Foundation.

2 Word2vec

> You shall know a word by the company it keeps
>
> John Rupert Firth, 1957

Word2vec has gained a lot of attention in the past years for its high accuracy in detecting semantic and syntactic relationships between words. The reduction in computational complexity is another reason for its high popularity [14, 17, 16]. It is based on the general observation that related words tend to appear in related contexts, as already stated by Firth.

The words from the corpus are modelled as word vectors, also called word embeddings. A vector contains real numbers and can have a dimensionality from 50 up to a few hundred [17, 14]. According to Schakel and Wilson, the dimensionality is relatively low compared to other training algorithms for word embeddings. Using distance measures such as the cosine distance, word relations can be extracted by measuring the distance between two word embeddings.

2.1 Overview

Word2vec learns word relations by the co-occurrance of words in a context, despite the input word order. This is based on the observation that related words occur in related contexts; for instance, the word *cow* occurs more frequently in sentences with *farm* than in sentences discussing *algebra* [4, 17]. The observation itself is not new [7], but the learning algorithm is more efficient.

To encode a term in a vector, a distributed word embedding technique is used. Instead of using a one-hot vector with the dimensionality of the vocabulary size, the real-value weights are distributed across the columns of the vector representation of the word. This allows for smaller vectors and also for a more fine-grained positioning of vectors

in the n-dimensional space, so that multiple relationships can be encoded for a term [18].

Word2vec consists of a log-linear neural network with two layers, a hidden layer and a projection layer (also called output layer). The output layer is non-linear, but is linear when applying a logarithm to the activation function of the neurons. It is a feed-forward neural network using backpropagation and learns unsupervised, so that no annotations are required for the input data.

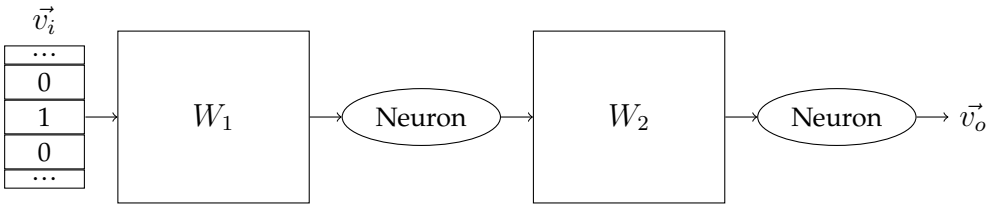

Figure 2.1: Generic overview over the two-layer neural network with input vector $\vec{v_i}$, output vector $\vec{v_o}$ and the two weight matrices W_1, W_2.

The simplicity of this network is one of the reasons for its high training efficiency. Figure 2.1 shows a simplified version of the neural network. The input vector $\vec{v_i}$ is multiplied by the first weight matrix of the first layer W_1 and sent through the first layer of neurons. This is repeated in the second layer. $\dim(\vec{v_i}) = V$ and $\dim(\vec{v_o}) = N$, where V is the vocabulary size of the used corpus and N the (hyperparameter) dimensionality of the word vectors to train. It should be noted that the input may consist of several vectors when using CBOW. Please see the subsequent sections for more details.

The input vector $\vec{v_i}$ is a one-hot vector and represents one word from the vocabulary. When multiplying it to the $V \times N$ weight matrix W_1, it effectively selects a row from this matrix, hence copying it. The activation function of the first neuron is linear (see next section) and therefore the computational complexity is linear [4].

In the projection layer, either hierarchical softmax or (negative) sampling are used, introducing the non-linear part [16]. The output is either one vector (for CBOW) or multiple vectors (Skip-Gram) with $\dim(\vec{v_o}) = N$. Using softmax, the vectors with weights can be transformed into a probability distribution. This can then be used to evaluate the guessed words from the network against the actual words from the training set.

The only input parameters for the neural network are its two weight matrices. Before the first iteration, the cells are initialised with pseudo-random numbers. Apart from these, there are hyperparameters which also affect the quality of the resulting word vectors. These parameters are the hidden layer size (of the neural network), (context) window size, minimum frequency, number of (negative) samples and number of iterations over

the whole corpus [13]. The window size is an important hyperparameter because it dictates how much context is attributed to a word. The focused word in a context is called the **centre word**. When evaluating the context in a window, a larger window means more computational complexity. Depending on the use case, increasing the window size might increase the word vector quality, but there is a point where the words are so distant that they are no longer related to the word (vector). In general, there is not a single optimal set of hyperparameters, but they need to be adjusted for a specific use case [14, 17, 1].

2.2 Training Models

Previous attempts to learn semantic and syntactic relations between words have been limited by their computational complexity. According to [17] the complexity of algorithms for the training of word embeddings is proportional to

$$O = E \times T \times Q$$

where E is the number of training epochs, T the number of words from the training set and Q a factor further specified by the used model. For instance, Schakel and Wilson specifies Q for a Feedforward Neural Net Language Model (**NNLM**) as

$$Q = N \times D + N \times D \times H + H \times V$$

where H is the hidden layer size, D the dimension, V the vocabulary size and N the number of previous words.
For Skip-Gram and CBOW, the computational complexity for the factor Q is much lower, which will be explained in the next sections.

2.2.1 Continuous Bag Of Words Model

The Continuous Bag of Words Model (**CBOW**) is an extension of the Bag of Words model, but utilises a (continuous) distributed representation to learn word embeddings. Bag of Words refers to the structure itself, where the word order in the context is irrelevant, but the number of word occurances is preserved. It is formally represented as a multiset [14].

The CBOW algorithm predicts a word, the *centre word*, by looking at the surrounding context [13]. When the window size is set to n, the one-hot vectors of the context words $\vec{c_1}, \ldots, \vec{c_{k-1}}, \ldots, \vec{c_{k+1}}, \ldots \vec{c_n}$ are passed to the network; $\vec{c_k}$ is the centre word to predict. The neural network now calculates the probability that a given word occurs in a given input context. In the hidden layer of the network, the activation function for the neurons are simply the average of all the weights obtained by the multiplication of the vectors $\vec{c_i}$ with the weight matrix W_1 from the first layer [4, 14].

The output vector $\vec{v_o}$ is the wanted centre word. By analysing the distance from $\vec{v_o}$ to the correct centre word, the error can be calculated and the weights of the network adjusted.

With the equation from 2.2 in mind, the complexity of the factor Q for this training model is

$$Q = N \times D + D \times \log_2(V)$$

Obviously, this is one addend less as for the Q-equation of the NNLM and makes the complexity independent of the hidden layer size. Furthermore, the vocabulary size is only a logarithmic factor.

2.2.2 Continuous Skip-Gram Model

The (continuous) Skip-Gram model uses the opposite method of CBOW to learn semantic information of words. It selects a word as the *centre word* and tries to guess the surrounding context given that word. It can be viewed as the opposite of the CBOW algorithm. More distant words are usually less related to the centre word and are hence weighted less. According to [14], the increasing computational complexity of larger windows can be reduced by sampling more distant words less. For small training corpora, the Skip-Gram model produces better results [13]. It furthermore captures semantic relations better for larger corpora [14].

Each centre word has exactly one probability distribution for the surrounding context. The probability distribution for a word is global for the model and must therefore fit for all possible contexts. To better catch semantic relations, words closer to the centre word are weighted higher. The word order in the context is ignored, the distance models the semantic relation between words best [14].

Looking at the complexity of the Continuous Skip-Gram model, we can see slightly different characteristics:

$$Q = C \times (D + D \times \log_2(V))$$

The new factor C is the maximum distance between two words and it becomes clear how the window size affects the runtime of the learning phase.

2.3 Optimisations

The general approach of Word2vec is not entirely new. Previous work already learned distributed word embeddings from unlabelled corpora using neural networks. One of the key differences is the speed with which the training can take place and the quality of the resulting word embeddings. Three of the optimisations will be discussed here, namely subsampling, negative sampling, and phrase detection.

When training the network using Skip-Gram, the output layer emits a vector with weights, then using softmax probabilities of individual context words for the given input centre word can be derived. This needs to be done for all words from the vocabulary. Comparing the guessed context with the correct context the calculation of the gradient descent is expensive, an update for both weight matrices has to be computed. For instance, for a training example with dimensionality 300 and vocabulary size of 5000000, we would need to compute and then update two weight matrices of size 300×5000000. This would need to be done for all words and contexts of the corpus. The aim of negative sampling is to avoid the expensive update by using a handful of negative examples for the neural network. For this, words not present in the context[1] are chosen using a unigram probability distribution. Instead of updating the whole matrices, only the columns/rows for the negative samples are updated, resulting in a dramatically reduced update cycle. The update for the correct words still need to happen. For smaller models, 5–20 negative examples are enough, for larger corpora, 2–5 words suffice. It has been shown that negative sampling increases the overall quality of the word embeddings [1, 4, 6].

Another technique to increase both word embedding quality and training speed is called Sub-Sampling. A word with a high frequency carries little meaning. Removing these words from the context of the centre words both minimises the number of contexts to consider slightly and enlarges the effective window size during training.
The quality increase comes from the enlarged window size, making the suggested words more topical. Usually, the sub-sampling rate should be between 1e$-$3 and 1e$-$5 [6, 2, 1].

[1] The procedure described here assumes Skip-Gram, but can be easily applied to CBOW as well.

2.4 Related Work

Due to the high costs and the massive amount of time required to edit a thesaurus manually, research has focussed on improving the productivity of the process. In the following section, different semi-automated and automated approaches will be presented.

2.4.1 Knowledge-Poor Techniques for automatic thesaurus generation

Already in 1993, Grefenstette [7] introduced a novel system to generate thesauruses from unlabelled plain text corpora. At this time, corpora of 6 MB in size were considered large. In comparison, the English text corpus from CRAFT[2] is around 430 times larger and is still regarded as a small corpus nowadays. Similar to the Word2vec approach, the input source is raw text data with no domain knowledge. The authors list two problems for the automated generation: the identification of words for the thesaurus and the constant update of the thesaurus when the underlying corpus changes. Both drive the costs of a thesaurus.

The generation process first tokenises the text using a regular grammar, queries the part of speech for each token and uses a stochastic model to decide on the most probable part of speech. When this information has been retrieved, the n related nouns for a noun are extracted using a weighted Jaccard measure. Two nouns are related if word w_2 is part of the n neighbours of w_1 and w_1 is part of the n neighbours of w_2. The problem of term-specific collocates is solved by checking the word relationships in both directions and only accepting words with an actual relation to the headword.[3] The identification of the words for the thesaurus is far easier for ALT because only base forms of words present in FreeDict dictionaries are used for the thesaurus. By using word lists the Word2vec/ALT approach is constantly evolving in quality due to progress made in the FreeDict project. The approach from this paper requires a lexicon-alike data source with part of speech information, whereas the ALT generator can work without additional data sources. In contrast to the system proposed by Grefenstette, ALT can generate general-purpose thesauruses, but requires much larger corpora for that. In contrast, the "Knowledge-Poor" approach is able to restrict the list of suggested entries to words with the same part of speech.

[2]The software is introduced in chapter 3.
[3]The headword is the indexed term in a dictionary.

2.4.2 The Sketch Engine

A different approach is taken by the Sketch Engine. The Sketch Engine is a corpora-based lexicographic software package, using different techniques such as word collocation, etc. to retrieve related words. Through its usage of tagged grammatical resources the most popular (or even all) senses, contexts, and usage scenarios of a word can be extracted. This is very rich information which aids a linguist in the semi-automated thesaurus editing process by presenting possible usage scenarios. Concordance is a linguistic approach to enrich terminology by common context or by frequent collocates and is a form of an extended index. The Sketch Engine is able to extract concordance information, providing the possibility to query examples for collocates and contexts. The Sketch Engine can generate thesauruses by extracting the common collocations of a word. A word w_1 from the corpus is compared to all other w_i from the corpus and those having the most common collocates are deemed to be similar [10].

ALT lacks most of the rich grammatical information, provided (and used) by the Sketch Engine. On the other hand, ALT is able to discover words with a similar meaning, where the Sketch Engine only finds words in the same or similar context. The intended audience for ALT is different, targeted mostly at end-users and as a start for the creation of a thesaurus, not for lexicographic research. The Sketch Engine does not generate a list of related words, but rather lists phrases, common prepositions, and nouns in similar contexts. This helps a human user to quickly find all word senses and related words, but does not result in a thesaurus on its own.

2.4.3 Alternate Equivalent Substitutes

Dao, Keller, and Bejnood conducted in [5] experiments on the optimal vector size for the Word2vec model to maximise the number of synonyms found by the neural network. The authors trained a model and used k-Means clustering afterwards, to partition the embeddings. After the training, a synonym list was used to determine a list of embeddings with a synonymity relation. With the help of the k-Means algorithm, the words were distributed among different classes using a random initialisation of word embeddings to classes. The list of known synonyms could then be used to rate how well the model recognised synonym relations by checking whether the headword and the synonym were both in the same class. The authors found that the vector size of the embeddings has a major impact on the relation quality and synonym discovery. While thesauruses should contain synonyms predominantly, they are not limited to this kind of semantic relations. This work is therefore a useful addition to thesaurus generation, but does not construct one itself. Furthermore, it requires a base synonym list to be present.

3 Corpus Training

As explained in chapter 1, one goal has been to only use free text sources to enable the inclusion of the resulting thesauruses in the FreeDict project. To acquire enough training data, free text sources had to be found and collected. In contrast to most Natural Language Processing algorithms (**NLP**), Word2vec only requires monolingual corpora. The data does not need to be tagged, which greatly eases the acquisition of sufficiently large text collections. Therefore, every free and open text source is a possible candidate for the inclusion.

In order to have an easily extensible and reproducible way of preparing texts to train the Word2vec model, a software called *CorpoRA-based Freedict Textextractor* (**CRAFT**) has been written. It parses texts from different free sources and extracts the words, discarding all formatting, punctuation, etc.

3.1 Source Acquisition

In a first step, importer scripts download the source material and save it to disk. At the moment, six importer scripts exist.

The two largest sources are from the Gutenberg project and from Wikipedia. Wikipedia is easy to handle, because it provides XML dumps of the latest versions of the articles. Downloading all books from the Gutenberg project is much harder. Neither the formatting of the book index, nor the format of the plain text formatted books is standardised, increasing the burden on the parser. Furthermore, some books have copyright restrictions, hence the parser needs to take care to not import these.
Another class of free sources are the multilingual corpora from the European Parliament and the "memory aids" from the Directorate-General for Translation (DGT) of the European commission. From these rich sources, only one language is used at a time.
A different scope have the texts from the Europeana project. It provides scans of historic newspapers, with decreasing quality the further back in time. While the data is very interesting, it is hard to distinguish between good and bad volumes and hence this importer is not used by default. For the future, it would be good to extend the importer

to only make use of the freely licensed and properly digitised material to broaden the knowledge domain of the model.

The importer for the code civil is the only one applicable for a single language, importing a collections of French laws. The French Wikipedia is considerably smaller than the German or English one and hence the code civil can fill this gap. It also adds formal and law-specific domain knowledge.

3.2 Text Preparation

Most of the data has markup information, not relevant for training a neural network. To remove the formatting, CRAFT relies on a multi-format converter called Pandoc. Pandoc reads the given input document with the requested format parser and transforms it to an internal Abstract Syntax Tree (AST). The AST is provided as a JSON representation and allows for a programmatic analysis of the document. With the help of Pandoc, input readers for the Markdown and Mediawiki formats have been implemented.

CRAFT extracts all text-only passages, leaving out formatting elements with no contextual value. Additionally, all article titles, images, table or mathematical formulas are removed, because they do not contribute real sentences and therefore no context for the neural network to learn from. This approach is very similar to that one described by Levy and Goldberg [11]. While CRAFT also lower-cases all tokens as well, it doesn't remove sentences with less than two words. Word2vec treats line breaks as a forced new context, a line with only one to four words will therefore not cause any problems.

In the original design, it was planned to delegate the parsing work entirely to Pandoc, making the processing independent from the input source format. It quickly turned out that the Pandoc parser is not feature-complete. For instance, unclosed syntax elements in the Mediawiki format can lead to a crash of the input reader, making the extraction of text impossible. Since it is better to omit only a fraction of the page in order to preserve the rest, the concept of a preprocessor was introduced. A preprocessor can parse the text before it is fed into Pandoc and can apply transformations to the text. For the Mediawiki format, the preprocessor removes certain syntax elements from the original source, to pass the simplified document to Pandoc, for instance broken tables. For the Gutenberg format, the preprocessor strips both preamble and epilogue from the books, containing English remarks about the project and the book license.

When CRAFT has received the AST from Pandoc, it extracts the text-only parts from it. The remaining plain text is freed from formatting, but not from punctuation and other non-letter characters. This is done in the last step. Punctuation and a few "enclosing"

characters are removed. Examples for enclosing characters are `(`, `)`, `[`, `]`. Only words which consist of letters are kept, except for tokens only consisting of numbers. For example:

> “Hi”, he said, “have you had a look at the 2nd letter, as I suggested?”

becomes

> Hi he said have you had a look at the letter as I suggested

The rules are as follows:

- If a word can be freed from enclosing characters, it is done and the word is kept.
- If a word consists only of digits, it is kept.
- If a word consists of numbers and letters, it is discarded.

The last rule is controversial. It removes words like *2nd* or *90s*, which are definitely part of the context. On the other hand, it reduces the noise by only keeping letter-based words. Finding a good trade-off is beyond the scope of this work.

Since Word2vec learns by predicting the centre word or its context, line breaks have a special meaning and mark the end of a context. Therefore, certain syntax elements such as headings, paragraphs and lists lead to hard line breaks in the resulting text output.

3.3 Stop Words

Words which occur very frequently in a text carry less semantic meaning [8]. Such words often have a grammatical function as conjunctions or prepositions, but do not encode a semantic meaning. These words can be regarded as noise for the learning, because they decrease the effective context of the training algorithm which operates on a fixed context window. This class of words are called **stop words**. For a thesaurus generator, it is best to remove these stop words, allowing the neural network to focus on the semantically important words.

The parameters for CRAFT can be set in a configuration file. An example configuration file for CRAFT might look like this:

```
craft:
  deu:
    wikipedia: data/dewiki-latest-pages-articles.xml.bz2
    gutenberg: data/gutenberg
    stopwords: >
      aber, alle, allem, allen, aller, alles, als, also, am, an, andere,
      anderem, anderen, anderer, anderes, anders, ansonsten, auch, auf, aus,
      bei, bis, da, daher, damit, dann, das, dasselbe, dazu, dem, demselben,
      welche, welchem, welchen, welcher, welches, wenn, weshalb, weswegen, wie,
      wieder, wo, wodurch, zu, zum, zur, zwar, zwischen
```

Listing 1: An excerpt of a YAML configuration for CRAFT. The stop word list has been shortened for this example.

4 Thesaurus Generation

When the word embeddings have been learned, a vector space with as many vectors as words in the model has been created. Vectors which are close to each other model a term relationship. According to Schakel and Wilson [17], the cosine distance is the most widely used measure for vector similarity. This is due to the nature of the neural network itself, which is the result of many vector and matrix multiplications, as explained in section 2.1. It best detects the position of the vectors to each other, because the dot product is close to the way the neural network was trained. Levy and Goldberg emphasizes in [11]that this measure ignores the spacial distance between two vectors completely, only yielding the distance of their angles.

Although the cosine distance of two vectors already indicates proximity and hence a kind of relationship, it does not allow conclusions about the type of relation. Word2vec captures both syntactic and semantic relations [14, 17, 4, 19].

4.1 Word Relation Types

Syntactic relations can be illustrated with an example. The word *apple* occurs in very similar contexts as *apples* and this is a singular/plural word relation. In an analogous way, different modes and tenses of a verb and comparative forms of adjectives are learned. Syntactically related words are not relevant for a thesaurus and the objective was to remove these.

There are a lot of different semantic relations and not all are relevant for a thesaurus. The table 4.1 gives a short overview over the different semantic relations. According to Handler [8], these are the most frequent relations discovered by Word2vec.

Linguistic Term	Short Explanation	Example
Synonym	Two terms are a synonym of each other, if they express the same fact.	pub → bar
Hypernym	A term is a hyponym of another term, if it is a generalization of the other.	wound → injury
Hyponym	A term is a hyponym of another term, if it specializes the meaning.	door → front door
Holonym	A term is a holonym of another, if it is semantically contained in the other.	face → nose
Meronym	A term is a meronym of another, if the other term is semantically contained in it.	chest → body

Table 4.1: Overview over word relationships most commonly discovered by Word2vec.

For a thesaurus, synonyms are the most helpful results and the ones present in most thesaurus dictionaries. Hypernyms can be helpful in some contexts, if they are labelled as such. For instance, *chocolate* might occur in very similar contexts as *sweets* and hence this hypernym is a good fit for a dictionary. The last group, holonyms, are less valuable for a thesaurus. This is because the relation between the two terms is too lose and often not relevant.

Word2vec favours some word relations over others, resulting in an unequal distribution of the semantic categories among the terms of the thesaurus [8].

4.2 Model Application

To make use of the trained word vector model, a software called ALT has been implemented. It is able to generate a thesaurus-like word list, listing related words for a given headword without any additional input from external sources and no human intervention.

ALT loads the binary representation of the learned word vectors and a list of valid words for the thesaurus from a separate file.[1] It then iterates over the words from the word list and computes the distance from their vectors to all other word vectors. As explained earlier, cosine distance is used to determine the closeness of the word embeddings and hence of the relationships of terms. The question arises whether other linear operations, such as plus and minus, might yield similar results. In a frequently cited example, Mikolov, Le, and Sutskever have shown that for their trained word vector

[1] The retrieval and reasoning of the word list is explained in the next section.

model $\overrightarrow{\text{King}} - \overrightarrow{\text{Man}} + \overrightarrow{\text{Woman}} \approx \overrightarrow{\text{Queen}}$. The pitfall with this example is that it uses domain-specific knowledge. In the process of building a thesaurus, this information is not available. Even if one would try to get a generic vector which could be subtracted or added to a word to get its synonyms, this cannot work. The particularity of Word2vec's word embeddings is that each vector encodes its relation to other vectors by its angle. A general word vector would hence need to encode all possible angles to each word. This is very similar to the averaging of frequent words (see Section 3.3 discussing stop words). Loosing the angle looses the trained word relation. Therefore, it is impossible to have a general method of determining synonyms.

Using the cosine distance as measure of proximity of related words means a high number of dot products. Each word needs to be compared to each other word. This is a compute-intense calculation:

$$\text{Number Of Dot Products} = \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} \vec{v_i} \bullet \vec{v_j}$$

$\vec{v}$ denotes a word vector and V is the full vocabulary set. Different techniques can be applied to reduce the computational complexity, which will be presented in the next section.

4.3 Word Filtering

Word2vec learns all forms of a word, not just the base forms commonly found in a dictionary. Syntactic relations, so for instance declined and inflected forms are of no use, because they do not bring any additional value to the base form and are not present in a thesaurus. The following sections will explain how ALT addresses these challenges.

4.3.1 FreeDict Word Lists

ALT restricts the words it considers as a valid word by restricting the vocabulary of the generator. While the cosine distance measure considers all word embeddings with a similar angle, word lists can help to filter words which are not in a base form, such as declined or inflected words. This avoids entries like "apple, apples" or "learn, learned, learning".

The filtering on the size of the vocabulary also brings a reduction in computation time, because less cosine products need to be calculated.

The white list of words to consider for the thesaurus is extracted from all FreeDict dictionaries available. For this, ALT parses all TEI-encoded dictionaries and extracts either headwords or translations, depending on the language pair of the dictionary. Dictionaries usually only list the infinitive or undeclined forms as a headword or a translation. Using the FreeDict vocabulary was an explicit design choice to restrict the number and kind of words, because terms from dictionaries are likely to be relevant for the inclusion in a thesaurus. Other possibilities would have been to use the list of Wikipedia articles, which also contain mostly base forms, but titles tend to contain many proper nouns, which are not relevant for a thesaurus. An additional benefit of the reuse of dictionary entries is the removal of rare coinages from the thesaurus.

As has been explained before, using only a fixed set of words for the cosine distance calculation of the vocabulary reduces computational complexity. Table 4.2 contrasts the vocabulary size of all vectors with the word count from the FreeDict-derived word list.

Language	**Distinct Words In Corpus**	**FreeDict-based Word List**
English	737 237	501 018
French	536 629	73 497
German	1 178 253	385 379
Spanish	419 223	35 069

Table 4.2: Word2vec Corpus Size In Contrast To FreeDict Monolingual Word List Size

For the Word2vec vocabulary size, German sticks out, because of the high number of compound nouns. German benefits a lot from the word list filtering, because compound nouns can be made up by a native speaker with fixed rules and even though these coinages are understood by another (German) speaker, these do not belong in a thesaurus. For instance, we could make up "Thesaurusgenerator" (thesaurus generator).

Spanish on the other hand suffers from the problem of insufficient Spanish FreeDict dictionaries, only 35 000 words are known. The filter is too aggressive, because too few words are known. The only solution is to extend the existing FreeDict dictionaries.

One might argue that using a stemming algorithm to derive base forms would be more effective, because this could help to use all words from the text corpus or the Word2vec model respectively. This approach would however make the thesaurus generation heavily language-dependent.

Another problem is that this method is unable to detect whether a headword is inflected or not. So while "Häuser" and "Hauses" might both be stemmed to "Haus", a stemming filter would not be able to detect that "Häuser" is a bad headword in the first place.

4.3.2 Word Length, Spelling And Pronunciation Similarity

Section 3.3 already discussed that high-frequency words such as conjunctions or articles carry less meaning than other words. In order for Word2vec to grasp the whole meaning of a word through its context, it is best to remove these from the corpus before training. In the learning phase, this is most important, because the less of these high-frequent words are in the context, the more words with semantic value fit into the sliding context window. However, not all meaningless words can be put on a stop word list, because such a list would be very long and is tedious to create. A rule of thumb is that more frequent words tend to be shorter[2]. Therefore, ALT offers an option to ignore words which are below a certain minimal word length. The threshold can be freely configured and is turned off by default. There is no fixed minimal word distance for ALT, because word length varies considerably among languages. For instance, there are languages with a general word length of one character, like for instance Chinese. Using a general minimal word length above one would lead to a vocabulary size of zero. The German thesaurus has been generated with a minimum word length of 4, the English and French versions with a minimum length of 3.

In some languages, grammatical functions of words are indicated through inflections of the words. If this principle is predominant in the language, it is called a synthetic language. Normally, languages tend to have either more or less synthetic elements. English is a less synthetic language, because there are only a few inflected forms of a word. In contrast, German has more inflected forms of a word and is hence a more synthetic language.
Word2vec has no understanding of inflection and treats each inflected form as a separate word embedding. Since inflected and unflected word variants often occur in the same context, Word2vec still relates them to each other (syntactic relation). Despite that inflected forms increase the vocabulary size of the trained model, they also lead to meaningless thesaurus suggestions. To prevent repetitions like "bird" and "birds" in the list of suggestions, the minimal edit distance between two words can be specified. Since this depends on the type of inflection, this value needs to be specified for each language individually. For the above example, a minimal edit distance of 2 would be appropriate, to only include words with at least two characters difference to the headword.

[2]See Zipf's law.

Depending on the term, the list of suggestions can be very long. The farther away two vectors are from each other, the more topical is the relation and hence inapplicable for the thesaurus. There is no general rule of thumb to determine when a word is too far away from its headword and hence from which (cosine) distance onwards words are irrelevant for the inclusion into the dictionary. For this reason, a relative threshold was used, utilising the first related word as a reference. Experiments have shown that most synonyms are not farther away than 80 % the distance of the first related word from the headword:

$$W_{\text{far}} \leq W_{\text{closest}} * 0,80$$

In the experiments, the list of synonyms from Wiktionary was used and compared with the list of suggestions from ALT. The maximum distance then served as a lower bound. This very simple measure shortens the list of suggestions considerably. Since the 80 percentile is relative to the first suggestion, the filtering mechanism is also able to deal with rare words. These sometimes have no closely related words, a fixed threshold would lead to no suggestions for this group of terms.

Experiments have shown that a relative threshold does not guard against terms without related words, such as interjections like “umm” or “emm”. These commonly do not have synonyms. This is reflected in the fact that the next closest word embedding has a very low cosine value (and a high distance) to the headword. When using the synonym lists from Wiktionary, the farthest synonym is around a value of 0.46 away. Since thesauruses consist mainly of synonyms, this is a useful lower bound.

Another way of identifying similar words would be their translation into a code encoding similarity and comparing the codes. The simplest algorithm for this is Soundex. It groups the letters of the alphabet to certain groups and labels those with a number. The Soundex code then consists of the first letter of the word and three more characters encoded as a number, ignoring vocals. This procedure has many disadvantages, for instance it discriminates against longer words and it is hard to apply for other languages. For German, there is the Kölner Phonetik (Colone Phonetics) with a similar approach, without the truncation after four characters and with some extension to improve similarity comparisons for German pronunciation. All these approaches have in common that they are language-dependent and hence are harder to integrate into a general-purpose solution like ALT.

5 Evaluation

Evaluating the quality of a generated thesaurus is challenging. The metrics depend both on the usage objective and the available comparison thesauruses. The goal is to use the auto-generated thesauruses both as a dictionary for end users and for information retrieval techniques.

When optimizing for human end users, it is best to compare the ALT dictionaries against existing solutions. The lack of standardisation of thesauruses is problematic. While some thesauruses might only include direct synonyms in the dictionary, others might also include hyper- and hyponyms. It is also unspecified how and whether the data is labelled with more information about the semantic relation to the headword. Comparison data could come from projects like WordNet, especially popular in machine-aided linguistics. It describes itself as a "large lexical database of English" `wordnetabout`. While the information is very rich and useful, it does not help evaluating the quality of the results from ALT, because the Word2vec model lacks part of speech information and other annotations. Therefore, we decided against the evaluation with WordNet or an annotated thesaurus because it would be an unequal evaluation. Even though Word2vec can discover relations between words, it is nearly impossible to determine the exact type of relation [8]. Completeness is another problem of direct comparisons, because there is no guarantee that a comparison resource contains all possible synonyms, hypernyms or antonyms detected by the trained model.

Before evaluating the generated thesauruses, it is best to get an overview of the general quality of the generated entries, as well as the distance between the headwords and their related terms. The examples have been randomly chosen. It also gives a feeling about the usefulness for a human user. The number of samples is too small to allow for a representative statement of the overall quality.

Headword	Suggestion 1	Suggestion 2	Suggestion 3
bric-a-brac	furniture 0.65	curiosities 0.58	marqueterie 0.55
consumptive	scrofulous 0.64	sickly 0.62	
convexity	convex 0.63	curve 0.61	curvature 0.60
deluge	flood 0.65	torrent 0.62	cataclysm 0.61
foiled	thwarted 0.78	baffled 0.66	
hunger	starvation 0.75	thirst 0.72	famine 0.65
impressing	impress 0.64	impresses 0.61	appreciating 0.61
self-regard	self-love 0.47	egoism 0.47	vanity 0.47
square-shouldered	broad-shouldered 0.69	thick-set 0.66	loose-jointed 0.65
subordination	subjection 0.64	subservience 0.63	obedience 0.62

Table 5.1: Overview over a few randomly selected words with their cosine distance.

Table 5.1 gives the first three suggestions of words for ten randomly selected headwords from the ALT-generated English thesaurus, leaving out proper nouns. The distance for the closest word lies for these words between 0.47 and 0.77 and shows that a fixed threshold for filtering too distant words is is hard to set correctly.

Self-regard is a brilliant example for Word2vec not finding appropriate suggestions. It is the headword with the most distant first hit and while the suggested words are all closely related, there are less related to the actual headword.
Table 5.1 also demonstrates the problem of insufficient FreeDict dictionaries, because inflected forms of verbs are present, polluting the hit list. The idea of the word list is to remove inflected forms, but this only works if the FreeDict dictionaries do not include inflected forms or mark them appropriately. Looking at the nouns from the same table, the filtering seems to have worked, because no genitive or plural forms are listed for the nouns.

In general, the related words from Table 5.1 are of varying quality, but we cannot draw conclusions from this due to the small sample size.

In the following sections, the generated thesauruses will be evaluated using traditional measures like Precision and Recall, but also with different evaluation methods. Precision and Recall are defined as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

The abbreviations stand for True Positives, False Positives and False Negatives, respectively. They are defined in comparison to a second, independent and hand-written thesaurus. A false positive is a term found in the generated dictionary, but not in the comparison dictionary. A false negative is a term found in the reference dictionary, but not in the ALT dictionary. A true positive is therefore found in both thesauruses.

Neither precision nor recall take the word relation into account, but do a direct word-by-word comparison. This results in several problems. First, the size of dictionaries should roughly match, both for the amount of words per entry as well as for the amount of distinct words. Second, the comparison dictionary should be complete, so that a *false negative* due to a insufficient entry from the comparison source can be avoided. Both is relatively hard to accomplish. If not noted otherwise, all evaluations of two thesauruses have been restricted to the common words they share, omitting words not present in both dictionaries and hence only evaluating a subset.
The third problem is that is completely ignores the semantic relation of the terms to the headword, treating them all equally.

False positives create another problem, which is hard to solve. Either they are an unrelated word discovered by Word2vec or they are a good semantically related word, not present in the comparison dictionary. It is impossible to distinguish between the two cases, to the resulting figures should be viewed with a grain of salt.

5.1 Comparison Using Precision And Recall

Subsequent sections will focus on the German English versions of the generated synonym dictionaries. Even though dictionaries have been created for Spanish and French as well, these languages lack enough suitable dictionaries in the FreeDict project, as has been already shown in Table 4.2. As a consequence, The French and Spanish thesauruses have to few entries and possibly lack important synonyms, even though they might have been discovered by Word2vec. This problem can be mitigated by importing more dictionaries for those languages and should disappear in the future.

Table 5.2 shows the amount of raw text data imported by CRAFT, with all punctuation, parenthesis and stop words removed. As usual, English has the most articles and books available, so the corpus is the biggest. German and French are roughly of the same size and Spanish is considerably smaller. This illustrates another problem of the generation of a free thesaurus, the problem of insufficiently large data sources.

Language	Tokens In Corpus	Size in MB
English	15 697 990	2.63 GB
French	14 913 256	2.03 GB
German	14 735 293	1.92 GB
Spanish	1 536 731 360	0.97 GB

Table 5.2: Corpora

In comparison to other training sets, the number of tokens is relatively small and do not qualify as big data. For instance, [11] use a corpus with around 1.5 billion words and the pre-trained Google News (paper) model was trained with around 100 billion tokens [1]. The Google News corpus is however only trained on one type of text, *news articles*. The CRAFT data is much more heterogeneous, because it contains books, Wikipedia articles, law texts and translation aids for the European Parliament and Commission.

5.1.1 Precision And Recall

Precision and recall are widely used metrics in information retrieval. A second thesaurus has been used as the "ground truth". Ground truth is difficult in this context, because a comparison thesaurus might be incomplete or classify word relations differently (e.g. classify hyponyms not to be part of a thesaurus). Evaluating the usefulness of a thesaurus-to-thesaurus comparison is beyond this work and is only used as a rough estimate for the quality of the generated word lists. The figures in the subsequent sections should be viewed with this perspective. For a more elaborate quality measure, see 5.2 for more details.

As was already stated in the introduction, the focus was on freely licensed thesauruses, which is why OpenThesaurus and Wiktionary were used.

OpenThesaurus is a free and open thesaurus for German,[1] developed by a community over the web. It is different to the projects of the Wikimedia Foundation, because it is supervised by an administrator. While the community contributes to the amount and the diversity of the entries, the administrator reviews and ultimately controls which definitions enter the thesaurus. Due to its adoption as the official German thesaurus for Libreoffice, it is in wide-spread use [12, 15].

Wiktionary is a Wikimedia project to create a dictionary (both monolingual and multilingual), based on the wiki principle. It is organized in articles, where each article is a term and can consist of multiple lexemes and even languages. Further structuring can include part of speech or etymology. Additionally, word relations (and even

[1] See `http://openthesaurus.de`

translations) can be listed in separate sections and are links to other articles of the same wiki [12]. Since Wiktionary is a wiki-based format, parsing the text data is a very challenging task, which can be only partly solved by Wiktionary's efforts to use templates and style guides. With the help of the WikDict project[2], this parsing step can be eased. The author of this project kindly provided word lists with synonyms, hyper- and hyponyms.

Both Wiktionary and OpenThesaurus are treated as simple word lists, where a term is mapped to multiple related words, discarding all information about the actual relationship. While this ignores valuable semantic information, it is enough for the comparison with the results from ALT. This has two reasons: first, OpenThesaurus doesn't come with annotations of the word relationship and second ALT lacks this information as well.

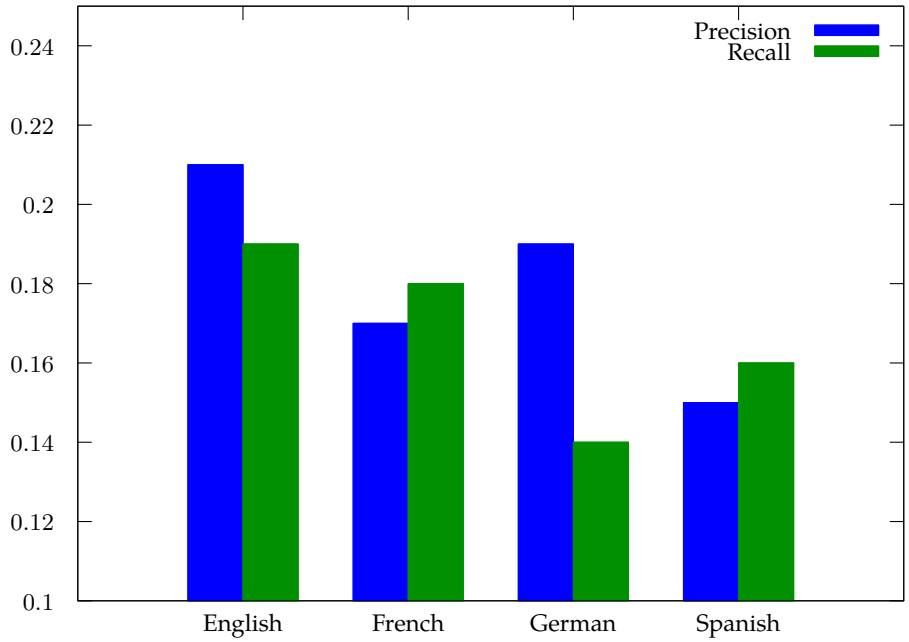

Figure 5.1: Overview over precision and recall for all evaluated dictionaries in comparison with the WikDict/Wiktionary synonym data, using only common words.

Figure 5.1 shows precision and recall for all processed languages in comparison to the Wiktionary word lists. To make the comparison more fair, only common words found in both thesauruses are kept. Testing for the existence of a word in the synonym list, which does not exist in the target vocabulary, reduces the recall and leads to an unfair evaluation. (see Section 5). Even though the same training and word sources

[2] `https://www.wikdict.com`

have been used English and German show the best precision. We cannot correlate this easily with the corpora size, as can be seen in Table 5.2.
By only keeping words from a thesaurus if they are present in the comparison thesaurus as well the recall rate should be independent of the vocabulary size. Comparing the different recall rates from Figure 5.1 with the number of words in the vocabulary (see 4.2), there is indeed no correlation.

According to [13], the quality of the word vector model can be significantly increased by providing more training data. Since there are not enough free data sources available, the pre-trained Google News corpus has been as used a reference. Using the trained word vector model and the same English word list, the precision increased to 0.24 and the recall to 0.23. This is only a very slight increase. When comparing the thesauruses directly, without stripping words which they do not share, the precision and recall rates even stay the same at 0.14 and 0.11 respectively. Even though it seems as if more data could help to increase the overall quality, there are examples from both dictionaries, where one outperforms the other. For the word **replaceable**, the suggestions look like this

Data Source	**Common Words**	Differing Words
CRAFT corpus	replaceable, interchangeable	removable, detachable, plastic, adjustable, machined, portable
Google News Corpus	replaceable, interchangeable	swappable

For this example, the precision is significantly better: three out of three words are synonyms of the requested words. The recall of the CRAFT-based model is higher, but it also lists more partly-related words. For this particular word, the WikDict/Wiktionary thesaurus has no entry, so a manual validation is impossible.

5.1.2 Skip-Gram Vs. CBOW

Word2vec offers two training algorithms, (continuous) Skip-Gram and CBOW. Since both models approach the learning of word contexts differently, a potential difference in the results could be expected. For the languages French, German, English and Spanish, Figure 5.2 compares the precision changes between the languages.

In general, Skip-Gram performs slightly worse than CBOW. A surprise is French, where Skip-Gram outperforms CBOW by 3 %. At this point, the reason for this remains unclear. According to Mikolov, Le, and Sutskever, Skip-Gram yields better results for infrequent words. Since the precision values are an average over all words, we cannot conclude anything from Figure 5.2. According to Mikolov et al., Skip-Gram

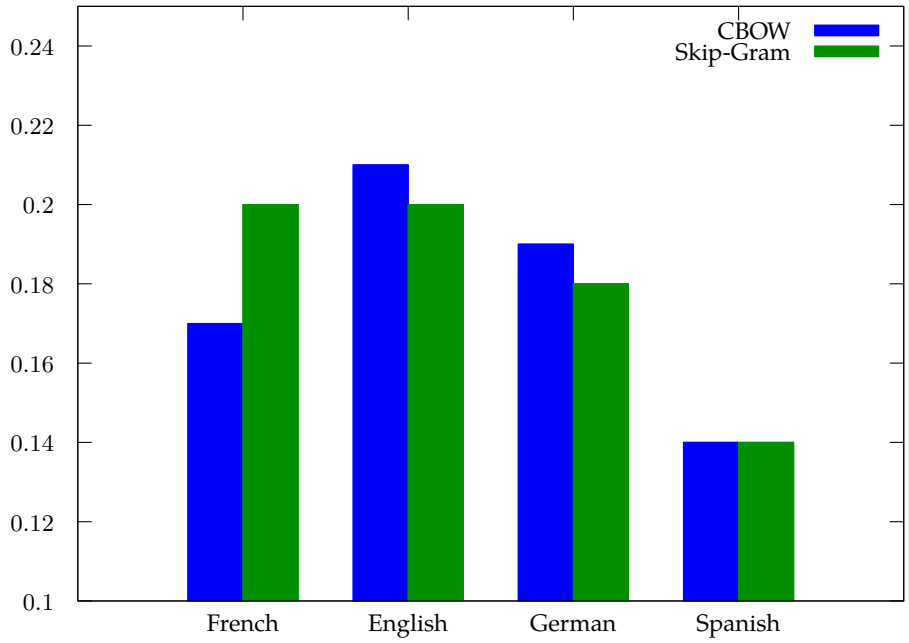

Figure 5.2: Comparison of precision for the Skip-Gram and CBOW algorithms.

performs better for semantic tasks, something we cannot conclude from the available measurements.

German consists of many compound words and native speakers can make up new words with general intelligibility for other (native) speakers. The Duden language corpus lists over 9 million base word forms and this number is even larger in normal texts, because of the inflected word forms [3]. This is a challenge for Word2vec, because each compound word and each inflected word is treated as a completely independent entity. In contrast, languages as English have only minimal inflection and use word groups instead of compound words, as for instance "lemon tree" (compare German "Zitronenbaum"). Word2vec will see two independent words and possibly relate "lemon" to "tree".[3] The high number of compound words and the high number of word forms due to inflection result in lower precision and recall for German. This problem can be potentially mitigated by a larger text corpus and by a larger vector size, hence allowing Word2vec to learn more about the high number of compound words. Please see 5.1.3 for more details.

The recall between the Wiktionary data and the generated thesaurus look very similar to those of Figure 5.2 and are for the sake of completeness shown in Table 5.3.

[3] Word2vec brings phrase detection, but this needs additional processing.

Language	CBOW	Skip-Gram
English	0.19	0.18
French	0.22	018
German	0.14	0.13

Table 5.3: Overview of the recall of the Skip-Gram and CBOW algorithms when using Wiktionary as a reference.

5.1.3 Hyperparameters

The quality of the training results can be influenced by the training parameters, also called hyperparameters. In the following section the influence on precision and recall are evaluated, when adjusting the model parameters window size, vector size and iteration count. Other hyperparameters have not been analysed, these include the negative sampling factor, the sample rate and whether or not hierarchical softmax is activated. These parameters were left to their defaults, as used by the reference implementation of Word2vec from `https://code.google.com/archive/p/word2vec`. The negative sampling rate has therefore been set to 25 negative samples, the sampling rate was specified as 1e−4 and hierarchical softmax was left turned off.

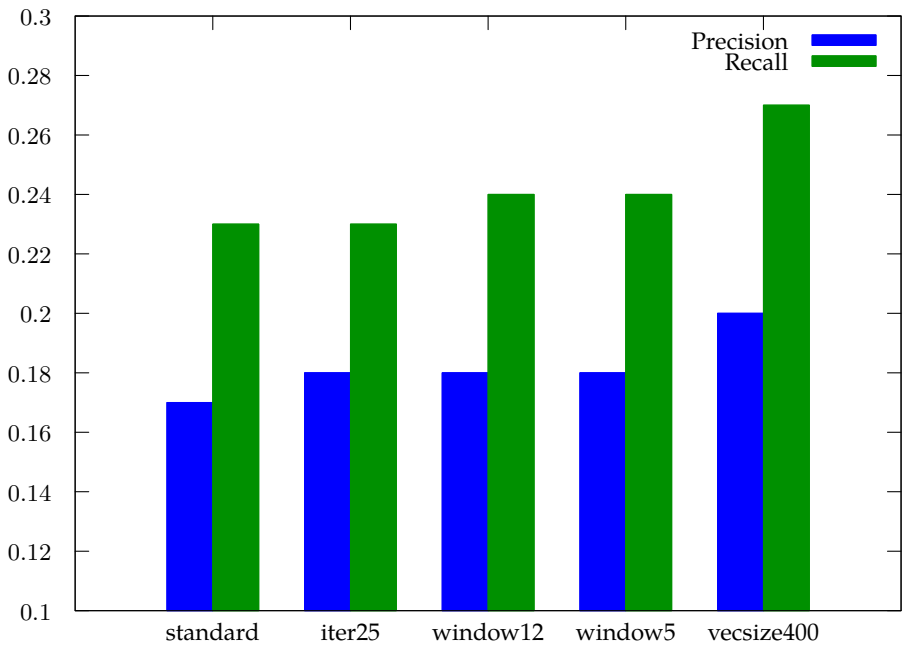

Figure 5.3: Influence of the different hyper parameters on precision and recall in comparison with OpenThesaurus.

Figure 5.3 visualizes how the different hyperparameters influence precision and recall, using OpenThesaurus as the comparison thesaurus. For the evaluation, only one hyperparameter at a time was changed, while the others have been set to their defaults. The influence on precision for the hyperparameters is clearly visible, where recall benefits more than precision. The difference between the smallest measured value with the standard hyperparameters and those from the highest with a vector size of 400 are small. The recall increases about 5 % and precision around 3 %. Although it is apparent that the hyperparameters can have an influence on the resulting dictionary quality, the effect is relatively small.

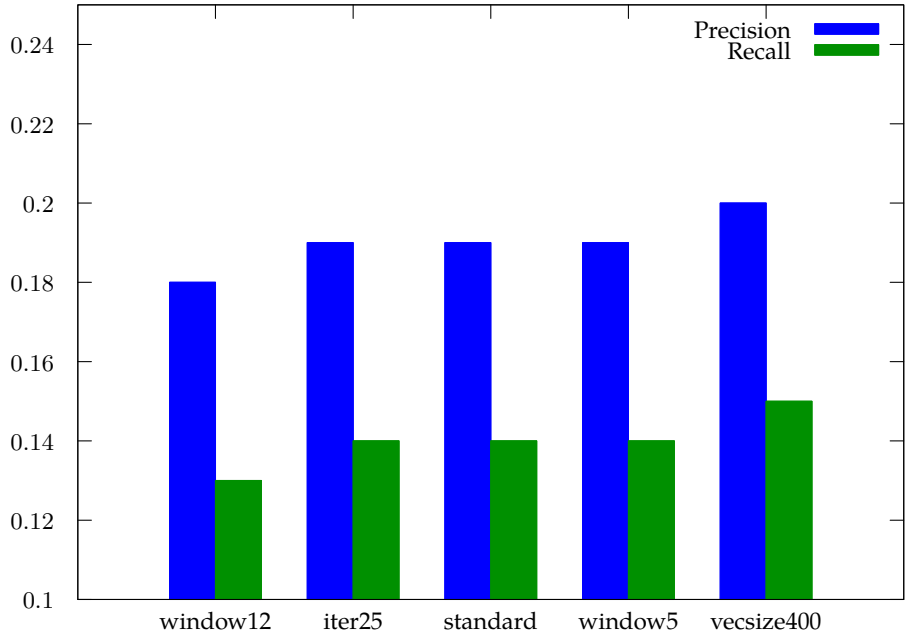

Figure 5.4: Precision and recall changes while adjusting hyperparameters, compared to Wiktionary word lists.

When comparing the generated thesaurus against the Wiktionary/WikDict data, the ranking of the different hyperparameters looks different, as can be seen in Figure 5.4. The worst performing run is the one with a window size of 12. Increasing the number of iterations has no effect, the same is true for decreasing the window size. That the increase of learning iterations does not improve the overall quality of the model lies within the construction of the neural network itself. The authors have shown that samples between 5 and 25 words are enough to learn sensible word relations. Increasing the iteration count does therefore not alter the weights of the neural network much, because the weights already were already very close to its steady state.

Figure 5.4 shows that the difference between the lowest and the highest measured value for both recall and precision is only 2 %, so even smaller than compared to OpenThesaurus.

In Figure 5.4, both a smaller and a higher window size (5 and 12) lead to worse results, but Figure 5.3 shows a general improvement when varying the window size. As explained in Section2.2, the window size influences the context which gets attributed to a word. Therefore, it could be that OpenThesaurus lists more Hyper- and Hyponyms, while Wiktionary/WikDict contains more direct synonyms. Smaller windows capture more syntactic relations [9], which is undesirable for a thesaurus. recommends to use smaller window sizes (around 5) for CBOW and larger window sizes (around 10) for (continuous) Skip-Gram. Therefore, the window size of 8 is a compromise for both algorithms, though should be adjusted when deciding on an algorithm.

There are only two certain conclusions which can be drawn from Figure 5.4 in comparison to Figure 5.3. The increase of the vector dimensionality improves the precision and recall rates, when training with the CBOW algorithm. As pointed out by Mikolov et al., increasing solely dimensionality of the word embeddings brings only diminishing effects on the quality of the word vector model. Instead, a larger data set requires a larger vector size, which in turn results in a increased computational complexity. Doubling the amount of raw training data has roughly the same effect as doubling the dimension of the word vectors in the model [14]. Larger vectors are able to store more syntactic and semantic information about a word, but according to [5] are less able to detect synonyms.

According to [1], the parameters to choose heavily depend on the kind of task to solve. While CBOW is faster to train, Skip-Gram performs better on rare words. The same is true for hierarchical softmax for infrequent and negative sampling for frequent words, the latter best with vectors with a lower dimensionality.[4]

As a sample, a German thesaurus with vector dimensionality of 640 was trained using Skip-Gram. While usual executions of Skip-Gram took around 10 hours on a quadcore machine, training with the high dimensionality took around 92 hours. Precision and recall were around 0.22 and hence comparable to the figures of the previous sections. This supports [1] that Skip-Gram performs better with lower dimensions and shows at the same time that more isn't necessarily better.

[4]The authors do not clearly explain "low" and "high", but given that the different papers train vectors with dimensions around $200 - -640$, the "lower" dimension must be within this range.

5.1.4 Word Frequency

The word frequency has a major impact on the kind of word relations Word2vec can learn. The more frequent a word occurs within a context, the better it can detect related words. Extremely frequent words however tend to be used in so many contexts, that clear relations cannot be extracted anymore. The following evaluation has been done with words from the generated thesaurus only and therefore excludes many of the most common stop words.

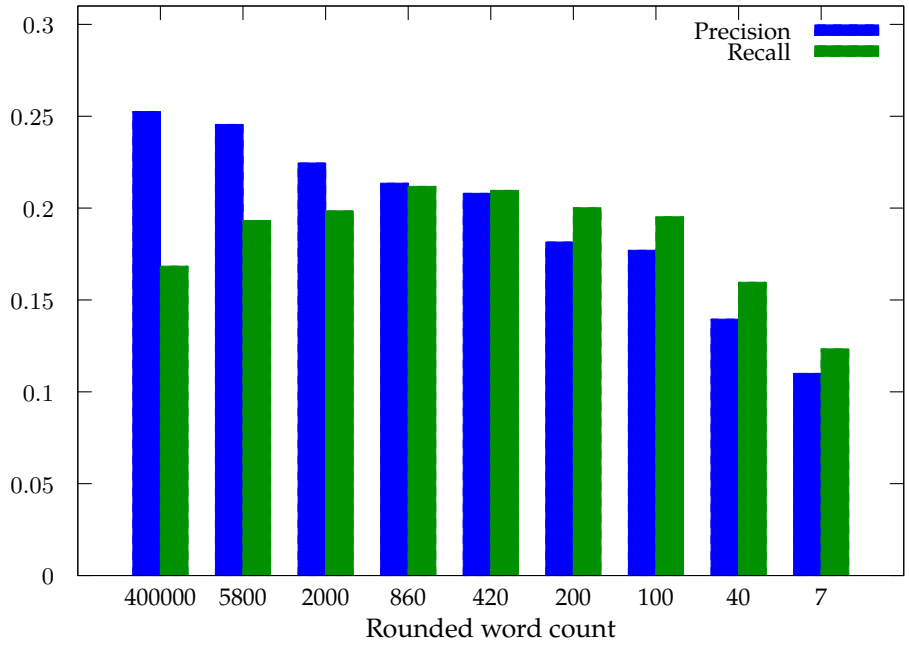

Figure 5.5: Precision and recall dependent on the word count, measured over partitions of 1000 words using the CBOW training model.

Figure 5.5 gives an overview about the course of recall and precision sorted by word frequency in descending order. For this, the thesaurus vocabulary was partitioned into slices of one thousand and the average of recall and precision calculated. [11] has very similar word frequency counts from around 474 000 for a corpus roughly seven times as large.

The above graph shows that the precision decreases with the frequency of terms, decreasing with a relatively constant rate. From this, we can draw the conclusion that CBOW can decrease the number of false positives for more frequent words. This also leads to the conclusion that larger text corpora would lead to a decreased false positive rate.

The recall curve is more hill-shaped, having its peak at a total word count of 860 and a recall rate of around 21 %. We see that words neither common nor infrequent have a lower number of false negatives. Since false negatives are words found exclusively in the comparison dictionary, we can conclude that a lower number translates to a higher thesaurus quality for neither frequent nor infrequent words.

5.2 Thesaurus Relevance

In the previous section, it was shown that an evaluation solely on recall and precision does not address the problem of the actual relevance of a related term for a headword. Different thesaurus have been build for different purposes and hence the authors might have a different view on related words. Therefore, a additional measure, called Thesaurus Relevance (**TR**, is proposed. It should increase the comparability of two thesauruses:

$$\text{TR} = \frac{\sum_{i=0}^{|S|} 3 + \sum_{j=0}^{|H_e|} 1.5 + \sum_{k=0}^{|H_o|} 1 + \sum_{l=0}^{|H_l|} 0.5}{|S| + |H_e| + |H_o| + |H_l|}$$

S is the set of synonyms, H_e the set of hypernyms, H_o the set of hyponyms and H_l the set of holonyms. In the numerator, each category gets a specific weight, which gets summed up for the members of each set. The denominator is the total amount of words of an entry. Weighting the synonyms considerably higher than the other word relations makes sure that this category is considered more useful, without discarding the other word forms as useless. From a linguist point of view, all word relations are equally important. From the perspective of a thesaurus editor, synonyms are the most useful, because they are the terms most useful when trying to substitute a term. This measure ignores the problem of homonyms, words with a different meaning, but the same spelling. See Section 5.3 for a more in-depth discussion of homonyms. For each semantic definition of a homonym, synonyms are counted with the same weight as for words with a single meaning.

When broadening the scope of a formulation within a text, hypernyms can be helpful for an author. Hypernyms are also more rare than synonyms, but in their function more important than other word relationships. The weights for hypo- and holonyms have been chosen arbitrarily, but considerably lower than synonym and hypernym, due to their low value for a thesaurus. Last but not least, words not present in this formula are weighted with 0 and are hence treated as not useful.

Because the sum of all weights gets divided by the number of entries of all sets, the resulting number is independent from the number of words per entry. There is a slight tendency to discriminate entries with a large amount of related terms against entries with only one synonym. However, entries with only one related form are rare.

Language	**Precision**	**Recall**	TR (ALT)	**TR (Wiktionary)**
German	0.15	0.097	1.527	2.008
English	0.2	0.1	1.66	2,18

Table 5.4: Overview over precision, Recall and Thesaurus Relevance for the most frequent 20 nouns for both the ALT and the Wiktionary thesaurus.

Table 5.4 shows measurements for the 20 most frequent words, extracted from the raw text corpus as generated by CRAFT. We decided to count nouns exclusively, excluding proper nouns and other part of speech. This is due to the most applicability of the relations to nouns. Proper nouns are even more problematic: what would be a good synonym for "Berlin"?

After the extraction of the most frequent nouns, the precision and recall has been calculated. Compared to Figure 5.1, the precision is roughly the same for English and lower for German. Recall is dramatically lower. We cannot conclude anything from these figures for a sample set as small as 20 words, but this is another confirmation of the fact that Word2vec has difficulties with too common words. This lies within the nature of Word2vec; according to Schakel and Wilson more frequent words have shorter word vectors. This happens during the learning process, more frequent words occur in a wider range of contexts and their vectors shrink to fit for the different contexts they are found in.

We can conclude from Table 5.4 that for both English and German, the quality of the found words is lower, but only about 17 % and 16 % respectively. That both ALT thesauruses have a TR around 1.5–1.6 shows that the trained Word2vec model discovers less synonyms than are present in the reference Wiktionary word list, but discovers enough related hyponyms and synonyms to not go below the category weight of a single hyponym. Word2vec favours synonyms, hyper- and hyponyms over other word forms [8] and although the above figures are only for a small subset of the corpus, we can at least conclude that synonyms do not form the majority of the discovered semantic relations.

Head Word	ALT Thesaurus Entries
fair	fairest, lovely, beauteous, good, goodly, fine, comely, dame, free, gracious
firm	company, consultancy, brokerage, business
general	particular, subordinate, universal, formal
hail	thunder, sleet, tempest, storm, roar, shower, shout

Table 5.5: Excerpt of homonymous definitions from the English ALT thesaurus; the list of homonymoms has been taken from `https://en.wikipedia.org/wiki/List_of_true_homonyms`.

5.3 Model Deficiencies

As was explained before, Word2vec discovers semantic relations by the context of the terms. It is hence solely spelling-based and cannot distinguish between different senses of a word, as is the case for homonyms. Human readers can easily related the correct sense to one of the homonymous spellings of a term, but machines fail when using raw word lists, as produced using ALT:

Table 5.5 gives an overview over a few homonyms found in the English ALT thesaurus. For both "fair" and "hail", ALT extracted one sense of the homonym with a fairly good success rate, while the other is underrepresented with one or two terms, respectively. For the term "firm", the thesaurus lacks synonyms for the adjective form and instead only lists synonyms and hyponyms of the noun. The term "general" is especially hard, because it is a relatively common word (about 132 000 occurrences), compared to "firm" (roughly 21 000). Even though a higher term frequency results in a shorter vector, the Word2vec model recognised both senses.

Word2vec can discover certain semantic and syntactic relationships using subtraction and addition, as explained in 4.2. Subtracting the vector representation of a synonym from the representation of its headword yields a vector close to most synonyms of the headword. This observation itself is not very useful, because this approach needs knowledge about synonyms to be present in the first place. Therefore, the list of synonyms from Wiktionary was used as a starting point. Each synonymity vector, resulting from a subtraction of the word embeddings from head word and synonym, was computed and the average of the vector (component-wise) calculated.

When subtracting this average synonymity vector from a word from the model vocabulary, ALT fails to detect synonyms properly; the cosine distance is considerably lower and due to the filtering mechanisms, this results in thesauruses between 800–1000 entries (for the different hyperparameters). The reason for this lies withing the averaging process, which has a similar effect as the training of very frequent words

(stop words), where the angle is lost and with it the encoded information. For the thesauruses with around 10 000 entries, precision and recall were lower than for the measurements from the previous sections. It can hence be concluded that subtraction (or addition) itself is not helpful as a measure to more reliably detect synonyms.

6 Conclusion And Future Work

This work has introduced CRAFT and ALT, a language-independent system to fully automatically generate thesauruses containing word relationships as synonyms, hypernyms, hyponyms and more. This is achieved using an unsupervised learning neural network called Word2vec. All resulting thesauruses are free and openly licensed. The resulting thesauruses have both precision and recall rates around 17 %–22 % in comparison to hand-written community-edited dictionaries.

The relatively low concordance values have a variety of reasons. First of all, the reference thesauruses are both far from complete and hence are likely to miss important definitions, found by ALT. Furthermore, Word2vec does not distinguish between synonyms and other relationships. Some of the related terms are therefore useless for a thesaurus. The English thesaurus for instance lists "beefsteak" as the fourth suggestion for "bacon", which is despite the topical relation , absolutely unrelated.

The evaluation has also shown some deficiencies of the FreeDict dictionaries, because their quality varies considerably. This ranges from simplistic word lists without any grammatical information to fully tagged entry structures with grammatical information, usage hints and/or inflected forms. Other dictionaries listed inflected forms as headwords, without including information about the inflection. This led to entries like

> **change:** changing, shift, alter
> **changing:** shifting, change, evolving, changed, altered

This example illustrates that the present participle was included in the ALT vocabulary, even though it should have been omitted. It is common practise to only list base forms of a word as headword in a dictionary. It is hence problematic, if FreeDict dictionaries also list inflected forms. We can therefore conclude that without improvements in the quality of the FreeDict dictionaries, the quality cannot be improved. This also affects the extractor, which should be able to remove questionable entries.

[11] suggests that there are several types of word relations, for instance comparative relationships of adjectives as "good" to "best" or singular to plural relations. Since this information is available for a subset of the existing relations from Wiktionary, vectors

for each type of relation could be calculated and used to exclude those from the list of thesaurus suggestions. It is advisable to couple this with a part-of-speech tagging using a probabilistic model as in [7].

It could be demonstrated that the quality of the generated thesaurus scales with the vector size and the size of the text corpus. This was done using the Google News corpus (see Table 5.1.1). Finding and including more free and open text sources is therefore an important task for the future and at the same time one of the current weaknesses of the system.

With the performed experiments, it has been shown that Word2vec is unable to distinguish the senses of homonyms. Homonyms are rare in comparison to synonyms, but this drawback has to be kept in mind when using the thesaurus data in information retrieval applications.

Alt can help with the semi-automated creation of thesauruses by providing a starting point for manual post-processing by lexicographers. A specific strength is its independence from any domain-specific knowledge, so that both domain-specific or generic thesauruses can be generated, solely depending on the input texts fed to CRAFT and learned by Word2vec. Using all the present words in the text corpus, a specialized word list can be derived from the general FreeDict dictionary list. When this list is provided to ALT, it will generate a word list with mainly domain-specific term relationships. Those can be further manually edited and significantly speed up the thesaurus creation process.

The experiments have shown that ALT thesauruses favour mostly the same part of speech for the list of related words. For FreeDict dictionaries lacking this piece of information, the ALT word list could be used to do a majority vote on the potential part of speech, for instance using cross-queries on other dictionaries. By comparing the different grammatical information of each word from a thesaurus entry, a majority vote could decide on the part of speech for the dictionary entry in question.

It has been shown that the continuous version of the Skip-Gram algorithm yields better results for less frequent words than CBOW at the cost of a slower training phase. This is similar for hierarchical softmax [1]. It seems therefore advisable to split the training and generation phase into two steps. First, the whole text corpus is fed into Word2vec using the CBOW algorithm. IN the second step, the words with a low frequency are determined and larger context chunks are extracted from the new corpus. The subset corpus is then passed to Word2vec using the Continuous Skip-Gram model with hierarchical softmax and a potentially larger vector size. ALT then needs to pick the correct trained model depending on the word frequency. This could lead to a higher over-all quality of the thesaurus entries.

A different model, Sense2vec, builds on the principle of Word2vec, while extending its capabilities. Instead of treating words as equal units, Sense2vec disambiguates the part of speech for the words in a sentence (using already existing models) and trains the model using this information. With the help of the automatically labelled data, Sense2vec can learn multiple word embeddings for homonyms. Furthermore, the authors introduce a clustering of the trained model, hence allowing for sense disambiguation for a term [18]. It would be interesting to apply a similar strategy to ALT.

To improve the results for certain languages, it would be great to enhance ALT to apply optimisations dependent on a given input language. This could help to further reduce the number of syntactic relations. For instance, German would benefit from the Kölner Phonetik, because it would allow for a simple inflection-based filtering.

FreeDict dictionaries have varying quality. Some list only base forms of a word, others also list inflected forms. While some might label inflected forms as such, others might not. This results in unwanted word forms in the word list. The solution to this is two-fold:

1. FreeDict dictionaries have to be improved and extended. Inflected forms have to be removed or have to be labelled accurately. The word list generator, used for the filtering mechanism, should be extended to only consider base forms. This will help to reduce the number of syntacticly related, but irrelevant suggestions, for instance “house” and “houses”.

2. Improve ALT to consider word forms in the filtering process. For cases, where the part of speech can be safely determined, this would help filter terms with an unrelated part of speech. In the current English thesaurus, the term “impregnability” lists “escalade”, which is unrelated, due to its part of speech.

The threshold for filtering unrelated or too distant words from the list of suggestions could be enhanced by using the word frequency of that term. Depending on the popularity of a term, a different threshold could be used to filter unrelated words. Given that more frequent words tend to have shorter vectors, it is likely that the angles/inclinations of related words are different to rare words. Therefore, it should be verified that a correlation exists and how it affects the overall quality of the dictionary.

The ALT thesaurus is based on the words found in FreeDict dictionaries and hence has a knowledge about the available vocabulary. This can be of help when searching for the translation of a word not present in the current bilingual dictionary. Current attempts to mitigate a lookup failure include finding words with a similar spelling, e.g. through the usage of the Levenshtein distance or by ranking the subsequent searches of other users, when this missing headword was entered. ALT could be extended to

find related words for every word of the text corpus, but restrict the synonyms, hypo- and hypernyms to a list of words present in the FreeDict dictionaries. For a user in a supermarket, searching for the translation of “poultry”, the dictionary interface could then suggest “chicken” and this would be indeed very helpful.

Although not subject to research, the size and quality of the FreeDict dictionaries needs to grow. It is unacceptable, if a word is removed from the list of suggestions solely because it was not present in one of FreeDict’s dictionaries, even though it would have been the best synonym. Similarly, the size of the text corpora has to grow, by finding more freely licensed texts to parse.

Bibliography

[1] Sept. 2, 2017.
Web: https://code.google.com/archive/p/word2vec/.

[2] Oct. 15, 2017.
Web: http://mccormickml.com/2017/01/11/word2vec-tutorial-part-2-negative-sampling/.

[3] Aug. 25, 2017.
Web: http://www.duden.de/sprachwissen/sprachratgeber/zum-umfang-des-deutschen-wortschatzes.

[4] Adrian Colyer: *The amazing power of word vectors*. Apr. 2016.
Web: https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/.

[5] Tri Dao, Sam Keller, and Alborz Bejnood: "Alternate Equivalent Substitutes. Recognition of Synonyms Using Word Vectors". In: (Dec. 12, 2013).

[6] Yoav Goldberg and Omer Levy: "word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method". In: *CoRR* abs/1402.3722 (2014).

[7] Gregory Grefenstette: "Automatic Thesaurus Generation from Raw Text using Knowledge-Poor Techniques". In: *MAKING SENSE OF WORDS. NINTH ANNUAL CONFERENCE OF THE UW CENTRE FOR THE NEW OED AND TEXT RESEARCH*. Oxford University Press. Oxford, United Kingdom, Sept. 1993.

[8] Abram Handler: "An empirical study of semantic similarity in WordNet and Word2Vec". MA thesis. University of New Orleans, Dec. 2014.

[9] Daniel Jurafsky and James H. Martin: *Speech and Language Processing*. Preprint from https://web.stanford.edu/~jurafsky/slp3/16.pdf. Aug. 2017.

[10] Adam Kilgarriff et al.: "The Sketch Engine: ten years on". In: *Lexicography* (2014), pp. 7–36.

[11] Omer Levy and Yoav Goldberg: "Linguistic Regularities in Sparse and Explicit Word Representations." In: *CoNLL*. Ed. by Roser Morante and Wen tau Yih. ACL, 2014, pp. 171–180. ISBN: 978-1-941643-02-0.

[12] Christian M. Meyer and Iryna Gurevych: “Worth Its Weight in Gold or Yet Another Resource — a Comparative Study of Wiktionary, Openthesaurus and Germanet”. In: *Proceedings of the 11th International Conference on Computational Linguistics and Intelligent Text Processing*. CICLing'10. Iaşi, Romania: Springer-Verlag, 2010, pp. 38–49.

[13] Tomas Mikolov, Quoc V. Le, and Ilya Sutskever: “Exploiting Similarities among Languages for Machine Translation”. In: *CoRR* abs/1309.4168 (2013).

[14] Tomas Mikolov et al.: “Efficient Estimation of Word Representations in Vector Space”. In: *CoRR* abs/1301.3781 (2013).

[15] Daniel Naber: “OpenThesaurus. Ein offenes deutsches Wortnetz”. In: *Sprachtechnologie, mobile Kommunikation und linguistische Ressourcen: Beiträge zur GLDV-Tagung 2005 in Bonn*. Frankfurt: Peter-Lang-Verlag, Mar. 2005.

[16] Xin Rong: “Word2vec Parameter Learning Explained”. In: *arXiv preprint arXiv:1411.2738* (2014).

[17] Adriaan M. J. Schakel and Benjamin J. Wilson: “Measuring Word Significance using Distributed Representations of Words”. In: *CoRR* abs/1508.02297 (2015). Web: `http://arxiv.org/abs/1508.02297`.

[18] Andrew Trask, Phil Michalak, and John Liu: “sense2vec - A Fast and Accurate Method for Word Sense Disambiguation In Neural Word Embeddings”. In: *CoRR* abs/1511.06388 (2015).

[19] Benjamin J. Wilson and Adriaan M. J. Schakel: “Controlled Experiments for Word Embeddings”. In: *CoRR* abs/1510.02675 (2015).
Web: `http://arxiv.org/abs/1510.02675`.

Selbstständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich reiche sie erstmals als Prüfungsleistung ein. Mir ist bekannt, dass ein Betrugsversuch mit der Note ”nicht ausreichend” (5,0) geahndet wird und im Wiederholungsfall zum Ausschluss von der Erbringung weiterer Prüfungsleistungen führen kann.”