Saarland University

Department of Computational Linguistics

Master Thesis:

Synonym Extraction with Minimal Supervision

Author:
Artuur Leeuwenberg
Matriculation: 2555148

Supervisors:
Dr. Mihaela Vela
Dr. Jon Dehdari
Prof. Dr. Josef van Genabith

September 30, 2015

Master Thesis:

Synonym Extraction with Minimal Supervision

Author:
Artuur Leeuwenberg
Code étudiant: 31316906

Supervisors:
Dr. Mihaela Vela
Dr. Jon Dehdari
Prof. Dr. Josef van Genabith
Co-supervisor:
Dr. Miguel Couceiro

September 30, 2015

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Declaration

I hereby confirm that the thesis presented here is my own work, with all assistance acknowledged.

Saarbrücken, September 30, 2015 Signature:

Abstract

The goal of this thesis is to extract synonym pairs from a large collection of text, using minimal supervision. The initial motivation is to use the extracted synonyms to improve machine translation evaluation. The approach is based on the word embeddings popularized by Mikolov et al. (2013a). We analyze how distributional word vectors can be used to extract synonyms for English and German, and what are the frequent error categories. We propose the measure *relative cosine similarity*, to increase the precision of the extraction. Furthermore, we show that by combining differently trained word embeddings, or using a part-of-speech tagger, the performance of the extraction can be improved. The final system is evaluated manually, and in the task of machine translation evaluation for both languages. We show our system can extract synonyms from part-of-speech tagged text that can be used to improve machine translation evaluation.

Keywords: synonym extraction, minimal supervision, machine translation evaluation, word embeddings

Acknowledgments

I would like to thank my supervisors for their help and guidance.

I would like to thank Kristy and Max for their help with the annotation.

I would like to thank my friends from Nancy and Saarbrücken for a wonderful time.

I would like to thank my family and girlfriend for all their love and support.

Contents

1 Introduction

In this thesis we explore different methods to extract synonyms from text. We try to do this using as little supervision as possible, with the goal that the same method can be applied to multiple languages. With *supervision* we mean, the use of resources that involve some type of linguistic annotation (other than word boundaries), or methods that were trained using such resources.

1.1 Motivation

The initial motivation for this task comes from machine translation evaluation. In machine translation evaluation a hypothesis translation is compared to a reference translation. A *hypothesis translation* is a translation that is to be evaluated and possibly machine made. A *reference translation* is a translation that was made by a proficient human translator. To evaluate if a machine translation system works well, a lot of hypothesis translations are compared to reference translations. This comparison between hypothesis and reference is often done automatically. If a hypothesis sentence and reference sentence have the same meaning, the hypothesis translation is considered a good translation. A frequently used evaluation system that makes this comparison is Meteor (Denkowski and Lavie, 2014; Banerjee and Lavie, 2005). Meteor makes an alignment between the hypothesis and reference sentence to see to what extend they convey the same meaning. It searches for the best alignment from a set of possible alignments. The possible alignments are defined by what parts of the two sentences can match. Finding possible matches is done by means of four modules:

1. Exact matching
2. Stemmed matching
3. Synonym matching
4. Paraphrase matching

In the exact matching module, words that are exactly the same are matched. In the stemmed matching module, words are matched that are the same after stemming. In the paraphrase matching module, paraphrases of each other are matched. The synonym module, matches words that are synonyms.

(1) The practiced reviewer chose to go through it consistently .

(2) The expert reviewers chose to go through it in a coherent manner .

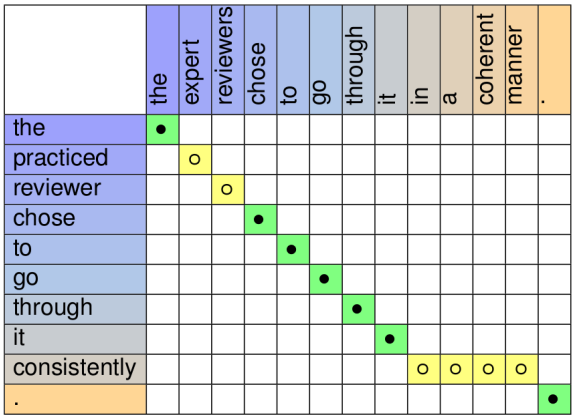

Figure 1: Meteor 1.5 alignment of hypothesis sentence (1), and reference sentence (2).

As an example, the best alignment for the hypothesis sentence (1) and reference sentence (2) is shown in Figure 1. In the example, the exact matches are indicated by the black filled dot. The stemming module matched "reviewer" with "reviewers". The paraphrase module matched "consistently" with "in a coherent manner", and the synonym module matched "practiced" with "expert".

Three of these matching modules use language dependent resources. Paraphrases and synonyms come from a pre-constructed lexical database, and stemming happens with a pre-trained stemmer. For this reason, not all modules are available for all languages. Currently, in Meteor 1.5, the synonym module is only available for English. It uses synonyms from the lexical database WordNet (Miller, 1995). Manual construction of lexical resources such as WordNet can be expensive, and time consuming. Also, the resource is to be created for each different language.

A resource that is available for many languages is raw text. Recently, research on word embeddings and distributional word vectors has contributed greatly to the value of unannotated text data. Many methods for training word vectors that reflect word similarity have been proposed that rely on the *distributional hypothesis*, i.e. words that are used and occur in the same contexts tend to purport similar meanings (Harris, 1954). Two out of these methods are the continuous bag-of-words model, and the skip-gram model from Mikolov et al. (2013a), which we will describe in the next section. It can be interesting to see if the lexical information that is captured in the word vector spaces can be extracted, and stored in a lexical database such as WordNet.

This thesis aims to extract synonyms from large collections of unannotated text. The motivations are the recent advances in distributional semantics and the large application domain of lexical databases, such as WordNet. Our major application of interest in the current work is machine translation evaluation.

1.2 Word and Synonym

In most recent work on synonym extraction the synonyms from WordNet are used for evaluation. In WordNet, synonyms are described as "words that denote the same concept and are interchangeable in many contexts". In the current work, our notion of words is merely a string of characters. Since there is *homography*, i.e. one word can have different lemmas, with different meanings and origins, we modifiy this notion of synonyms slightly. We think of *synonyms* as: words that denote the same concept and are interchangeable in many contexts, with regard to one of their senses.

1.3 Outline

In Chapter 2, we will proceed to describe the distributional word vectors we used in our experiments, and the related work in synonym extraction. In Chapter 3 we describe different experiments in which we explore synonym extraction using the continuous bag-of-words model, and the skip-gram model. Chapter 4 describes and evaluates few methods that introduce some supervision, such as using a synonym thesaurus from a different language together with a partial translation dictionary, or using a part-of-speech tagger. In Chapter 5 we do an evaluation of a system that combines different proposed findings, for English and German. We evaluate manually, and by using the extracted synonyms for the task of machine translation evaluation. Chapter 6 concludes the thesis by giving a summary of the findings and possibilities for future work.

2 Related Work

2.1 Distributional Word Vectors

Distributional word vectors, or *word embeddings*, are word representations that can be constructed from raw text, or a collection of documents, based on their context. The representation of each word will be a vector of numbers, usually real numbers. In some cases linguistic information, such as word dependency information, or morphological information, is also used during the construction process (Levy and Goldberg, 2014; Luong et al., 2013). These word vector representations can then be used to calculate, for example, word similarity and have a wide application domain.

In the last years, many new methods have been proposed to construct distributional word vectors based purely on raw text (Mikolov et al., 2013a; Pennington et al., 2014). Some methods also use the document structure that can be present in the data (Huang et al., 2012; Liu et al., 2015b,a).

In this work, we experiment mostly with word vectors trained using the *continuous bag-of-words model* (CBoW), and the *skip-gram model* (SG) developed by Mikolov et al. (2013a). It has been shown that these vectors, especially for the skip-gram model, can also encode relations between words in a consistent way (Mikolov et al., 2013b). This means that they not only encode word similarity, but also similarity between pairs of words. For example, the offset between the vectors for "queen" and "king" lies very close to the offset between "woman" and "man", i.e. $v(queen) - v(king) \approx v(woman) - v(man)$.

This property has been exploited to extract hypernyms from raw text by Fu et al. (2014) and Tan et al. (2015). In Fu et al. (2014), they automatically learned, in a supervised way, a piecewise linear projection that can map a word to its hypernym in the word vector space, for Chinese. To do this they clustered the vector offsets $(v_1 - v_2)$, and then found a projection for each cluster. Using this method they could succesfully find hypernym pairs. In Tan et al. (2015) they searched for hypernym pairs, but in English. They also used a projection to project a word to its hypernym in the word vector space. However, instead of automatically learning this projection by using a thesaurus, they concatenated the words "is", and "a" into an "is_a" token in the corpus, and used this as projection. So, $v(w) + v(is\_a)$ would lie very close to the vector for the hypernym of word w.

Both the CBoW and the SG model can be seen as a feed forward neural network, that is constructed from a word, and its context. The architecture of the neural network is shown in Figure 2. In the CBoW model a word representation is trained that is optimal for predicting the word, given its surrounding words, i.e. the *contextual window*. In the SG model, the word representation is trained such that the contextual window can be predicted from the word representation. In Figure 2, the word is represented as $w(t)$.

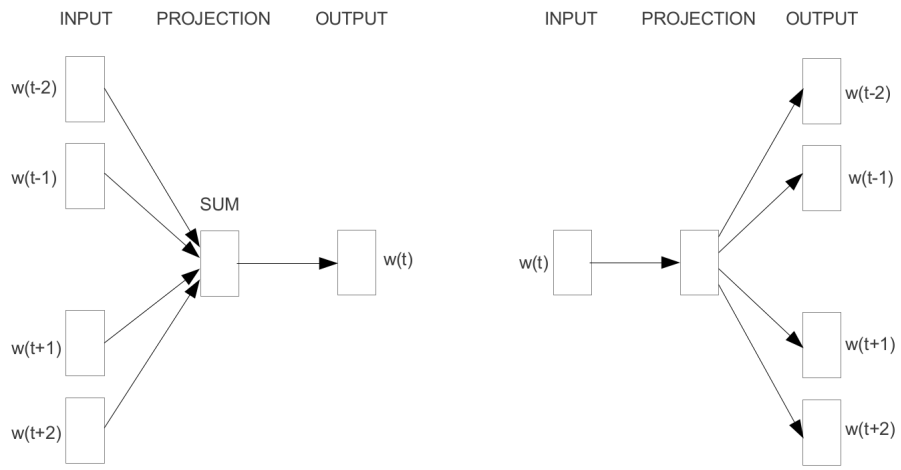

Figure 2: Continuous bag-of-words architecture on the left, and skip-gram on the right.

The contextual window, here of size 2 (two words to the left, and two to the right), is represented as $w(t-2), w(t-1), w(t+1), w(t+2)$. The final word vector is built from the weights from the projection layer. During training the window iterates over the text, and updates the weights of the neural network. Two training methods were proposed by Mikolov et al., *hierarchical softmax*, and *negative sampling*. In hierarchical softmax, the weights are updated based directly on the difference between the predicted and desired target word. In negative sampling, the weights are also updated based on a sample of randomly picked non-target words. It has been shown to be fast enough for training state-of-the-art word vectors, using the word2vec toolkit[1].

Depending on the application, it can be benificial to modify pre-trained word vectors towards specific properties. Faruqui et al. (2014) have shown a method to refine a vector space using relational information, such synonymy and hypernymy, from a lexical database. For the task of antonym detection, Ono et al. (2015) transformed a pre-trained vector space by maximizing the similarity between synonyms and maximizing the similarity between antonyms. Since we would like to use as little supervision as possible, we did not resort to these particular methods. Another method to transform vector spaces is *canonical correlation analysis* (CCA). This method was used by Faruqui and Dyer (2014) to project separately trained vectors from two different languages to a common vector space, using a word-to-word translation dictionary. We also experimented with this method so we will explain this method in more detail in the next paragraph. A different method that builds multilingual word vectors is used by Hermann and Blunsom (2014). This method uses a parallel corpus to train word vectors for both languages. The reason to choose the CCA method is because we can then make use of bigger monolingual corpora, with probably a less domain-restricted vocabulary.

In canonical correlation analysis, as used by Faruqui and Dyer (2014), the goal is to find

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

a projection for each vector space that maps words from each space to a common vector space. Words that have the same meaning (translations) should have a high correlation in the common vector space.

Let V_e and V_d be two vector vocabularies, say for English and German, and let ϕ_e be a projection vector that maps vectors from V_e to the common vector space V_c, and ϕ_d a projection that maps vectors of German words to the commmon space V_c. For the words "for" and "für", that are translations of each other, let $v(for)$ and $v(für)$ be their vectors in V_e and V_d respectively. The correlation ρ between $\phi_e v(for)$ and $\phi_d v(für)$ is calculated as given in Equation 1, where E stands for expectation.

$$\rho(x, y) = \frac{E[x \cdot y]}{\sqrt{E[x^2]E[y^2]}} \tag{1}$$

Given a set of word-to-word translations, CCA finds a ϕ_e and ϕ_d such that the correlation is maximized for the set of translations. Advantages of this method are that even though not for every word there is a translation, the full vocabulary of both languages can still be projected to the common space. Also, the number of dimensions of the two vector spaces doesn't have to be of the same size, which allows for optimizing the individual vector spaces before projecting them. How we exactly applied this method will be explained in Chapter 4. The implementation[2] we used for CCA is the one from Faruqui and Dyer (2014).

2.2 Synonym Extraction

Many methods that have been developed for synonym extraction use three main ideas. Firstly, the distributional hypothesis (Van der Plas and Tiedemann, 2006; Gupta et al., 2015; Saveski and Trajkovski, 2010; Pak et al., 2015; Plas and Bouma, 2005). Secondly, the assumption that words that translate to the same word have the same, or a very similar, meaning (Van der Plas and Tiedemann, 2006; Gupta et al., 2015; Saveski and Trajkovski, 2010; Lin et al., 2003) . And third, the use of linguistic patterns that are typical, or atypical for synonyms to occur in (Lin et al., 2003; Yu et al., 2002).

Van der Plas and Tiedemann (2006) used both distributional word similarity, and translational context for synonym extraction in Dutch. They used a big monolingual corpus to construct a measure for distributional similarity, which was based on grammatical relations. Furthermore, they used different parallel corpora, and automatic alignment, for the construction of a translational context. The authors remark that when only using the distributional similarity there were some word categories that show up frequently but

[2]https://github.com/mfaruqui/eacl14-cca

are not synonyms, which are antonyms, (co)hyponyms, and hypernyms. When using the translational context, these error categories were less frequent, and more synonyms were found.

These word categories seem a common problem when using purely distributional methods (Pak et al., 2015; Plas and Bouma, 2005; Lin et al., 2003). However, the advantage of using methods based on distributional properties is that the coverage is usually larger than that of manually constructed corpora, as also mentioned by Lin et al. (2003). They tackle the problem of discriminating synonyms from other strongly related words using linguistic patterns. For example, some English patterns in which synonyms hardly occur that they mention are "from X to Y", and "either X or Y".

In Yu et al. (2002), instead of filtering by means of linguistic patterns, they used particular patterns in which synonyms occur frequently. Their application domain was finding synonyms for gene and protein names. They found that in MEDLINE abstracts synonyms are often listed by a slash or comma symbol. This is probably a more domain dependent pattern. Some other patterns they found were "also called", or "known as", and "also known as". In this thesis, we do not resort to a pattern based approach, as they are language and domain dependent.

3 Synonyms in Word Vector Space

In this Chapter we explain different experiments we did to analyze how synonyms behave in different word vector spaces. First, we analyze the effect of contextual window size, the number of dimensions, and the type of word vectors on the precision of extraction, for English and German. Secondly, we take a closer look at the categories of words that are (cosine) similar in the vector space. Then, we take a look at cosine similarity, and the measure of relative cosine similarity. Last, we examine the overlap of the most similar words in different vector spaces. We begin by describing the data set, and the preprocessing.

3.1 Data and Preprocessing

For English and German, we use a 150 million word section of the NewsCrawl corpus from the Workshop on Machine Translation 2015[3]. As preprocessing, for both languages, we apply lowercasing, tokenization, and digits are conflated (3.45 → 5.55). In this thesis, we do not deal with multi word units. For example, for a separable verb in either German or English (e.g. abholen / to pick up) can only be found as one word in infinitive, or past perfect (abgeholt/picked up).

We only consider the vocabulary of words that occur at least 10 times in the corpus to ensure that the vectors have a minimum quality. We randomly split the vocabulary into a training, development, and testing sections with proportions 8:1:1 respectively. In Table 1, statistics about these sections are given.

Language	Corpus	V	$V_{\geq 10}$	$S_{V_{\geq 10}}$	V_{train}	S_{train}	V_{dev}	S_{dev}	V_{test}	S_{test}
English	150M	650.535	136.821	21.098	109.454	16.882	13.681	2.116	13.683	2.100
German	150M	2.421.840	279.325	16.304	223.458	13.056	27.933	1.599	27.933	1.649

Table 1: Dataset Statistics: V indicates the size of the full corpus vocabulary, $V_{\geq 10}$ indicates the vocabulary size for words with counts greater or equal to 10. S_x indicates the number of words for which at least one synonym is known, that also occurs in $V_{\geq 10}$.

For evaluation, we use the synonyms from WordNet 3.0 for English, and GermaNet 10.0 for German. In both WordNet and GermaNet words carry a corresponding part of speech. In WordNet these are nouns, verbs, adjectives, and adverbs. In GermaNet, synonyms are given for nouns, verbs, and adjectives. Because in the experiments in this chapter the part of speech of words is unknown, we consider the synonyms of each word to be those of all the parts of speech it can potentially have in WordNet or GermaNet.

[3]http://www.statmt.org/wmt15/translation-task.html

3.2 Evaluation

We evaluate several experiments in terms of precision, recall and f-measure. *Precision* is calculated as the proportion of correctly predicted synonym word pairs from all predictions. Because synonymy is symmetric, we consider the word pair (w_1, w_2) equivalent to (w_2, w_1) during evaluation. *Recall* is calculated as the proportion of synonym pairs that were correctly predicted from all synonym pairs present in WordNet, or GermaNet. In the experiments we sometimes only search for synonyms for words in a subset of the vocabulary (e.g. S_{train}). In this case, recall is calculated only with regard to the synonym pairs from WordNet or GermaNet that involve a word from the mentioned vocabulary. *F-measure* is the harmonic mean of precision and recall and is given by Equation 2.

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2}$$

3.3 Quantitative Analysis of Training Parameters

In this experiment, we trained CBoW, SG, and *Global Vectors* (GloVe) (Pennington et al., 2014) with different training parameters, and evaluated synonym precision for the $\{1^{st}, 2, 4\}$-most-similar word(s), for vocabulary S_{train}. With similarity we refer to cosine similarity. The parameters we varied are the contextual window size, and the number of dimensions of the vectors. The window size varied over {2, 4, 8, 16, 32}. The number of dimensions varied over {150, 300, 600, 1200}. The experiment is conducted for both English and German, and used 150M words per language. We fixed the number of training iterations, 5 for CBoW and SG, and 25 for GloVe. For the CBoW and SG training we used negative sampling with 5 negative samples. These are the default values given by the respective authors.

The results for the CBoW and SG vectors, for both English and German, are shown in Tables 2, 3, 4, and 5.

We excluded the results for the GloVe vectors, as they showed lower precision, and to limit the scope of the study. The general trends of the GloVe vectors were that they had higher precision for larger window sizes. The vectors with highest precision of 0.067 for English were of dimension 300, with a window size of 32. For German, the highest precision was 0.055, and the vectors were of dimension 1200, with a window size of 32 as well.

English CBoW																				
dim.	150					300					600					1200				
win.	2	4	8	16	32	2	4	8	16	32	2	4	8	16	32	2	4	8	16	32
P-1	0.077	0.076	0.072	0.066	0.058	0.084	0.083	0.079	0.072	0.068	0.086	**0.086***	0.081	0.074	0.068	0.083	0.083	0.082	0.073	0.067
P-2	0.058	0.056	0.055	0.051	0.046	0.062	0.061	0.059	0.055	0.052	0.063	0.063	0.060	0.056	0.052	0.061	0.061	0.060	0.055	0.050
P-4	0.039	0.039	0.038	0.036	0.032	0.042	0.042	0.041	0.039	0.036	0.043	0.043	0.042	0.039	0.036	0.042	0.042	0.041	0.039	0.036

Table 2: Precision for different window sizes and number of dimensions, using the CBoW model, for English.

English Skip-gram																				
dim.	150					300					600					1200				
win.	2	4	8	16	32	2	4	8	16	32	2	4	8	16	32	2	4	8	16	32
P-1	0.069	0.062	0.055	0.048	0.044	0.069	0.062	0.053	0.048	0.044	0.066	0.059	0.046	0.043	0.039	0.061	0.051	0.039	0.034	0.030
P-2	0.050	0.045	0.040	0.037	0.034	0.050	0.046	0.039	0.036	0.033	0.049	0.044	0.035	0.032	0.030	0.045	0.039	0.029	0.026	0.024
P-4	0.034	0.032	0.028	0.026	0.024	0.034	0.032	0.028	0.025	0.024	0.033	0.030	0.025	0.023	0.021	0.031	0.026	0.020	0.018	0.017

Table 3: Precision for different window sizes and number of dimensions, using the Skip-gram model, for English.

German CBoW																				
dim.	150					300					600					1200				
win.	2	4	8	16	32	2	4	8	16	32	2	4	8	16	32	2	4	8	16	32
P-1	0.073	0.082	0.082	0.083	0.080	0.076	0.084	0.086	0.086	0.082	0.076	0.087	**0.089***	0.088	0.080	0.076	0.083	0.086	0.085	0.081
P-2	0.052	0.057	0.057	0.058	0.056	0.054	0.060	0.062	0.061	0.059	0.054	0.060	0.062	0.062	0.059	0.053	0.059	0.062	0.060	0.058
P-4	0.034	0.036	0.038	0.038	0.037	0.036	0.039	0.041	0.040	0.039	0.035	0.039	0.041	0.041	0.040	0.035	0.039	0.041	0.040	0.039

Table 4: Precision for different window sizes and number of dimensions, using the CBoW model, for German.

German Skip-gram																				
dim.	150					300					600					1200				
win.	2	4	8	16	32	2	4	8	16	32	2	4	8	16	32	2	4	8	16	32
P-1	0.065	0.068	0.066	0.064	0.064	0.064	0.069	0.064	0.062	0.060	0.063	0.064	0.057	0.051	0.049	0.061	0.059	0.046	0.039	0.035
P-2	0.048	0.049	0.049	0.046	0.046	0.048	0.049	0.048	0.045	0.046	0.047	0.046	0.042	0.039	0.037	0.046	0.043	0.035	0.030	0.027
P-4	0.032	0.033	0.032	0.032	0.031	0.033	0.033	0.032	0.031	0.031	0.031	0.031	0.029	0.027	0.026	0.031	0.029	0.025	0.022	0.020

Table 5: Precision for different window sizes and number of dimensions, using the Skip-gram model, for German.

In general, it can be noticed from Tables 2, 3, 4, and 5 that the CBoW vectors give higher precision than SG for both German and English. A reason for this could be that CBoW vectors tend to be slightly more syntactical compared to SG vectors. It could be that the syntactical constraint on synonyms, as they are to appear in similar contexts, has enough influence for CBoW vectors to perform better.

It can also be noticed that for English, smaller contextual windows, 2 and 4, generally give better precision, for both CBoW and SG vectors. For German, the optimal window size lies between 8 and 16 for CBoW, and around 4 for SG vectors. A possible explanation for the difference in optimal window size between English and German can be the difference in types of synonyms that are available. As WordNet contains synonyms for nouns, verbs, adjectives and adverbs, whereas GermaNet does not include synonyms for adverbs. It could be that adverbs require only a small contextual window to be predicted, compared to nouns, verbs, and adjectives. Another observation that can be made is that for both English and German the optimal window size for SG tends to be slightly lower than for CBoW vectors. Again, this can be due to training difficulty. A larger window can make the training of the SG model more difficult, as a bigger context is to be predicted from one word.

To get an impression of the performance if we would use the most-similar words as synonyms we calculated precision, recall and f-measure on the test set S_{test}. For English, using choose the CBoW vectors of dimension 600 with window size 4, precision is 0.11, recall 0.03, and f-measure is 0.05. For German, using a CBoW model of dimension 600 with a window size of 8, precision is 0.08, recall is 0.05, and f-measure 0.06. For both languages, these scores are very low. In the next section, we look at some frequent error categories, with the goal to get more insight in the reason behind these low scores.

3.4 Distributionally Similar Words

Only looking at precision, calculated on WordNet or GermaNet, allows us to compare different vector spaces with regard to finding synonyms. However, it might not reflect actual precision, due to sparsity of WordNet and GermaNet. Also, it gives only few cues for possible improvements.

For this reason, we also looked more in depth to the most-similar words. For 150 randomly chosen English words from S_{train} we looked at the 1st-most-similar word, and 2nd-most-similar words and categorized them. This was done manually. Categories were made based on what was found during the analysis. The word vectors used to create the 1st-most similar, and 2nd-most-similar words were from the CBoW model of dimension 600, with window size 2, from the previous experiment. The results from this analysis are shown in Table 6. The categories we found are the following:

- *WordNet-Synonyms*: Synonyms as given in WordNet.
- *Human-Synonyms*: Synonyms judged by a fluent, but non-native, English speaker.
- *Spelling Variants*: Abbreviations, differences between American and British spelling, and differences in hyphenations.
- *Related:* The two words are clearly semantically related, but not consistently enough to make a separate category.
- *Unrelated / Unknown*: The relation between the two words is unknown.
- *Names*: Names of individuals, groups, institutions, cities, countries or other topographical areas.
- *Co-Hyponyms*: The two words share a close hypernym.
- *Inflections / Derivations*: Inflections or derivations other than plural.
- *Plural*: The found word is the plural version of the given word.
- *Frequent collocations*: The two words occur frequently next to each other.
- *Hyponyms*: The found word is conceptually more specific.
- *Contrastive*: There appears an opposition or big contrast between the meaning of the two words.
- *Hypernym*: The found word is conceptually more general.
- *Foreign*: A non-English word.

What can be noticed from Table 6 is that the number of human-synonyms is about twice as big as the number of synonyms given by WordNet, considering that WordNet considers spelling variants also to be synonyms. This suggests that the actual precision may lie a corresponding amount higher. As WordNet would give a precision of 0.12 for this set of words, where the human annotation gives 0.25. A reason for this big difference can be that resources such as WordNet are usually constructed by manually adding the synonyms for a given word. Which requires the annotator to think of all the word senses of a word, and their synonyms. This can be a difficult task. Here, the two words are presented and the question is whether they are synonyms. It is probably easier to find the corresponding word senses of both words in this case.

The two biggest error categories are the related words, and unknowns. Since both categories are rather vaguely defined, and consisting of many subcategories we will not go into much more detail on these. There appears some overlap with the error types that

Category	1st-most-similar	2nd-most-similar	Example
WordNet-Synonyms	18	7	laundry / washing
Human-Synonyms	29	20	masking / obscuring
Spelling variants	8	4	commander / cmdr
Related	27	33	head-on / three-vehicle
Unrelated / Unknown	13	20	gat / por
Names	15	15	consort / margherete
Co-hyponyms	15	13	sunday / saturday
Inflections / Derivations	12	10	figuring / figured
Plural	11	2	tension / tensions
Frequent Collocations	7	5	dragon / lantern
Hyponyms	5	12	swimsuit / bikini
Contrastive	3	7	rambunctious / well-behaved
Hypernym	2	4	laundry / chores
Foreign	2	4	inhumation / éventualité

Table 6: Counts per category for the most similar word and second most similar word, of 150 randomly chosen English words, in a CBoW model of dimension 600 with a window size of 2.

were also found by Lin et al. (2003), Plas and Bouma (2005) and, Pak et al. (2015), namely co-hyponyms, and hyponyms. However, contrastives and hypernyms do not seem as frequent in our experiment. Some other major error categories we found are different types of inflections and derivations, and in particular plurals. This category is for our application, machine translation evaluation, not a very big problem, as the inflections might already have been matched by the stem module of Meteor. Another category that is fairly frequent are names. The reason is probably that names might not have many single-word synonyms. The error category of frequent collocations can be explained by the fact that both words usually occur together, and are thus trained on a set of very similar contexts. This effect, that more frequently co-occurring words get similar vectors, we tried to tackle by adapting the training method. This is described in the next section.

3.5 Neighbors as Negative Samples

As observed in the previous section, frequently co-occurring words seem to get similar vectors. However, we do not expect words more likely to be synonyms when they co-occur more frequently. We tried to remove this effect when training the vectors. Normally, when training the vector for a word w, a number of randomly chosen words are picked from the vocabulary, i.e. *negative samples*, to estimate the learning gradient. Instead of only choosing arbitrary words from the corpus, we also added the word that directly precedes w, and the word directly following it. We expected that this way frequent collocating words would get more different representations in the vector space.

Random samples	Neighboring samples	P-1
5	0	0.11
5	2	0.10
4	2	0.10
3	2	0.10

Table 7: Precision for the most-cosine similar word when changing the number and type of negative samples.

We trained CBoW vectors of dimension 600, with a window size of 2, using 5 randomly chosen negative samples. We also trained CBoW vectors of the same dimension and window size, but 3 or 4 randomly chosen samples, and the two neighboring words as samples. For these vectors we evaluated the synonym precision of the most-cosine-similar word for the vocabulary S_{train}. The results are shown in Table 7.

What can clearly be seen from the table, is that adding the neighboring words as negative samples does not improve the precision. The neighboring words seem even worse samples than randomly chosen samples. This can be seen when we compare the vectors with 5 random samples and no neighboring samples, with the vectors using 3 random samples and the two neighboring samples. Since sometimes training vectors with the same training parameters can give slightly different vectors we repeated this experiment. The second time, the precision values were almost the same, and the same conclusions could be drawn. Since this approach did not seem to work, and frequent collocations are not the biggest error category we did not further research the error category of frequent collocations. We expect that it might also be possible to resolve the error category by concatenating the frequent collocations in the corpus.

3.6 Relative Cosine Similarity

One idea we tested with the goal to improve precision was to only consider word pairs that have very high cosine similarity. In practice this would mean, to set a threshold, and only consider those word pairs that have a cosine similarity higher than the threshold. Our expectation was that synonyms are most similar compared to the other word relations. We plotted precision, recall and f-measure on S_{train} against the cosine similarity threshold. This is shown in Figure 3.

What we found however, is that even when you increase the cosine similarity threshold, precision does not increase. It does not even reach the precision we got from our baseline, of taking the most-similar word. This indicates that cosine similarity on its own is not a good indicator for synonymy. Still, we get higher precision with choosing the most-similar word. We manually looked at the top-10 most similar words of the 150 words from the

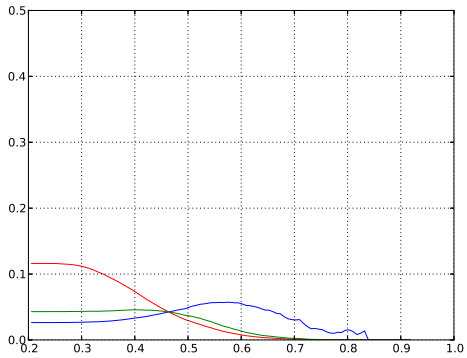

Figure 3: Precision (blue), recall (red), and f-measure (green) on S_{train} plotted against the cosine similarity threshold.

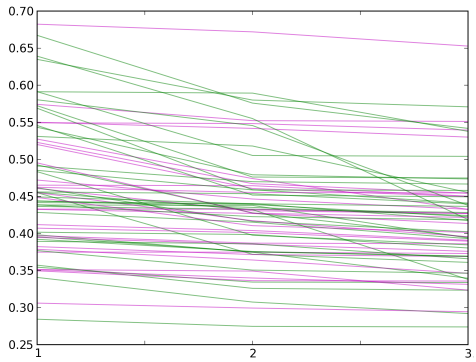

Figure 4: Cosine similarity against n-most similar, for the 3-most-similar words. If the most-similar word is synonym its line is green, and for related words purple.

previous section, and their cosine similarity. We noticed that when a synonym, inflection or contrastive occurs in the top-10, their cosine similarity is usually much higher than that of the other words in the top-10. That is, the difference in cosine-similarity between the most-similar word, and the second-most-similar word is very high for these categories. When we looked at this for other categories such as, co-hyponyms, unknowns, and just related words this was not the case. This can be seen when we plot the cosine similarity of the 3-most-similar words for synonyms, and related words taken from the previous experiment. This is shown in Figure 4.

In this figure two things can be noticed. Firstly, it is hardly possible to separate the start, at position 1, of the green synonyms from the purple related words by means of a horizontal cosine threshold. This corresponds to the observation we made earlier, that a cosine similarity threshold does not increase precision. Secondly, many green lines tend to decrease, and many purple lines stay relatively horizontal. This indicates that, in general, the difference in cosine similarity between synonyms and other similar words (from the top-10) is bigger compared to for example co-hyponyms. We also found this bigger difference for inflections, and contrastives. This observation could be used to increase precision, as we can possibly filter out some co-hyponyms, related words, and unknowns.

To test this hypothesis, we choose a difference measure to calculate similarity. We calculate similarity, relative to the top-n most similar words. We calculate *relative cosine similarity* between word w_i and w_j as in Equation 3.

$$rcs_n(w_i, w_j) = \frac{cosine\_similarity(w_i, w_j)}{\sum_{w_c \in TOP_n} cosine\_similarity(w_i, w_c)} \tag{3}$$

This will give words that have a high cosine similarity compared to other words in the top-10 most-similar words a high score. If all words in the top-10 most similar words

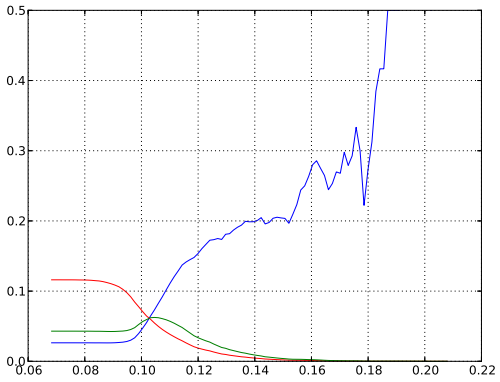

Figure 5: Precision (blue), recall (red), and f-measure (green) on S_{train} plotted against the relative cosine similarity threshold.

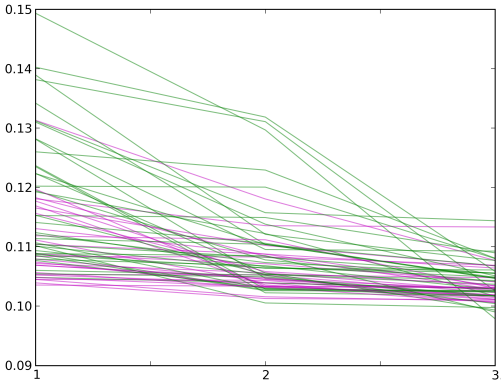

Figure 6: Relative cosine similarity against n-most similar position, for the 3-most-similar words. If the most-similar word is synonym its line is green, and for related words purple.

have almost an equal cosine similarity, they will get a lower score. When we do the same experiment again, changing the similarity threshold and plotting precision, recall and f-measure, but for relative cosine similarity instead, we can see that precision goes up when we increase the rcs-threshold. This is shown in Figure 5. In Figure 6, it can also be noticed that when we look at the relative cosine similarity for three most similar words of words where the most similar word is synonym (green), or just a related word (purple), part of the synonyms is now separable from the related words by a horizontal line, i.e. an rcs-threshold. This confirms our earlier hypothesis.

In this experiment, we used WordNet synonyms to calculate precision, recall and f-measure, and find the optimal rcs_{10}-threshold. However, what can be noticed is that the tilting point for the precision to go up lies at an rcs_{10}-threshold of 0.10. This is not a coincidence, as 0.10 is also the mean of the relative cosine similarities for 10 words. If a word has an rcs_{10} higher than 0.10, it is more similar than an arbitrary similar word. If for a language synonyms are more similar compared to other similar word relations, we can find this tilting point at $\frac{1}{n}$, where n is the number of most-similar words we consider for calculating rcs_n.

So, relative cosine similarity gives us the flexibility to increase precision, at the cost of recall. Also, we can identify the tilting point for precision to go up. Since it is a normalized measure we expect that the optimal thresholds for different languages will lie very close to each other, which makes it easy to generalize the method to other languages, without explicit synonym data in the respective language. This under the assumption that synonyms are one of the most similar words, among similar words in the respective language and vector space, as appears to be the case for English, and also German. That it also holds for German will be shown in the next section, particularly in Figure 7.

Constant	Varies	$P(both)$	$P(both\|synonym)$	$P(both\|synonym) - P(both)$
CBoW win. 2	dimension 300 & 600	0.38	0.67	0.29
CBoW dim. 600	window 2 & 4	0.31	0.60	0.30
CBoW dim. 600	window 4 & 8	0.32	0.60	0.28
CBoW dim. 600	window 2 & 8	0.24	0.52	0.28
dim. 300 win. 2	CBoW & SG	0.19	0.48	0.29

Table 8: Overlap between differently trained pairs of vector spaces, for arbitrary words, and synonyms.

3.7 Overlap of Similar Words in Different Vector Spaces

In this section, we explore if we could use a combination of different vector spaces, trained using different training parameters to improve the synonym extraction. For this we analyze the most-cosine-similar words of vocabulary S_{train} in different vector spaces. We considered pairs of vector spaces with different training parameters. Then, we calculated the probability that an arbitrary word is most-cosine-similar in both vector spaces, which we call $P(both)$. We also calculated the probability that a synonym is most-cosine-similar in both vector spaces, which we call $P(both|synonym)$. We altered the dimension, window size and training method (CBoW vs. SG). We mostly considered CBoW vectors, as they gave highest precision in previous experiments. The results of this experiment are shown in Table 8.

What can be seen in this table is that for all changes in parameters $P(both|synonym)$ is considerably higher than $P(both)$. This indicates that it can be a good cue for synonymy if a word is most-cosine-similar in differently trained vector spaces. We can also see that the general overlap seems highest when only changing the dimension, and lowest when changing the training paradigm, and fairly constant when doubling the window size. For all conditions, $P(both|synonym) - P(both)$ is fairly constant. This indicates that the cue for synonymy is almost equal for all pairs.

Because the numbers seem quite constant, we expected that it is maybe just the inflections that overlap between both vector spaces. For this reason we repeated the experiment, but the second time we only considered word-pairs that have a Levenshtein distance bigger than 3, to exclude the majority of the inflections. The results are shown in Table 9. In this table we can see that the observations made earlier still hold.

To use this observations in our earlier synonym extraction method we calculate rcs_{10}^m in each vector space m for the 10 most-cosine-similar words on S_{train} in each space, and simply sum the rcs_{10} of the different models. The *summed relative cosine similarity* between word w_i and w_j is calculated as in Equation 4, where $TOP_{10}^m(w_i)$ is the set

Constant	Varies	$P(both\|ld > 3)$	$P(both\|synonym, ld > 3)$	$P(both\|ld > 3) - P(both\|synonym, ld > 3)$
CBoW win. 2	dimension 300 & 600	0.31	0.61	0.30
CBoW dim. 600	window 2 & 4	0.23	0.55	0.32
CBoW dim. 600	window 4 & 8	0.24	0.56	0.32
CBoW dim. 600	window 2 & 8	0.17	0.48	0.31
dim. 300 win. 2	CBoW & SG	0.12	0.42	0.30

Table 9: Overlap between differently trained pairs of vector spaces, for arbitrary words, and synonyms, when only considering word-pairs with a Levenshtein distance larger than 3.

containing the 10 closest cosine-similar words of w_i in vector space m.

$$rcs_{10}^{M} = \sum_{m}^{M} \begin{cases} rcs_{10}^{m}(w_i, w_j) & \text{if } w_j \in TOP_{10}^{m}(w_i) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

As in the previous section, we again plot precision, recall, and f-measure against the threshold, but now using the summed rcs_{10} of a CBoW model, and a SG model. We did this for both German and English. For English, the CBoW model has dimension 600, and was trained with a window size of 4. The SG model is of dimension 150, and window size 2. For German, the CBoW model is of dimension 600 as well, and but has a window size of 8. The results are shown in Figure 7. If we compare it to the results from Figure 5, we

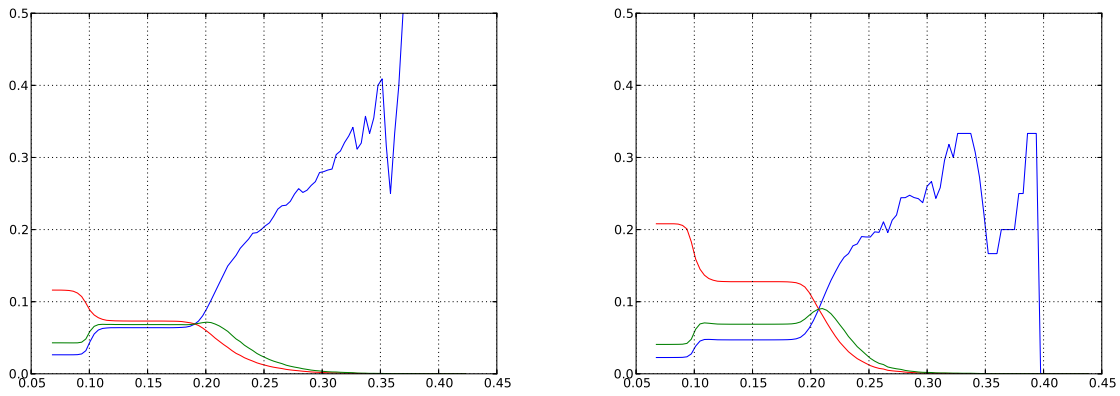

Figure 7: Precision (blue), recall (red), and f-measure (green), on S_{train} for English (left) and German (right), using the summed rcs_{10} score for a CBoW and SG model.

can see that for English, the general precision, recall, and f-measure lies slightly higher using two vector spaces. Also, we can see that the tilting point now lies at around 0.2 instead of 0.1. It lies twice as high, as we sum rcs_{10} of two spaces. Also, our expectation that for different languages this tilting point lies at the same threshold seems correct for German. The bump in both graphs around a threshold of 0.1 is because some words only occur in the top-10 most similar words in one of the two vector spaces.

When we choose the threshold that gives optimal f-measure on the S_{train}, and use it to extract synonyms for S_{test}, we find for English a WordNet precision of 0.12, a recall of

0.5, and an f-measure of 0.7. Compared to our baseline, of only taking the most similar word precision is 1% higher, recall is 2% higher, and f-measure 1%. For German, we find a precision of 0.12, recall of 0.07, and f-measure of 0.09. Compared to the baseline, it means precision went up with 4%, recall with 2%, and f-measure with 3%. From this, we conclude that combining differently trained models helps to extract synonyms, both in precision, and recall. Also, combining the scores from the different vector spaces does not prevent us from finding the tilting point, from where precision rises.

4 Adding Resources

We looked at using a part-of-speech tagger to improve the synonym extraction in different ways. Furthermore, we briefly explored the possibility to make clusters of word relations, and see how synonymy is distributed over these clusters. In the last experiment, we also look at the discriminative power of these relational clusters after projecting them to a common English–German vector space by means of CCA.

4.1 Homography

The initial motivation to resort to POS-tagging is *homography*, i.e. one word (here, string of non-space characters) has several word-senses. In Figure 8, an example of homography of the words <phone>, and <call> is given. The word-senses, and their respective part of speech are shown in the leaves of the tree. The dotted link represents the synonym relation between the word-senses of <phone>, and <call> for the action of making a telephone call.

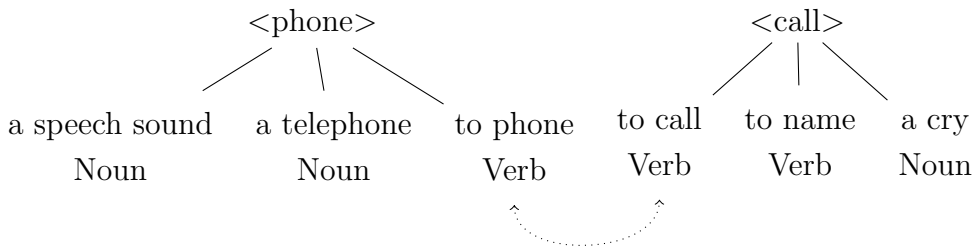

Figure 8: Schematic representation of the synonym relation between the corresponding word-senses of the words <phone>, and <call>.

Homography can be a problem for finding synonyms when using one vector for each word, as the vector for <phone> is trained on all the different word-senses that occur in the corpus. In the case of <phone>, it is probably used more frequently as the noun telephone, or as a verb for the action of calling, compared to the noun meaning of a speech sound, in our news corpus. This can make it difficult to find synonyms with regard to this less frequent meaning.

To train vector representations for each word-sense, ideally we would disambiguate each word in the corpus first, and then train the vectors on these disambiguated meanings. To our knowledge, there is not yet the possibility to do completely unsupervised word sense disambiguation. As can be seen in the example in Figure 8, some of the word-senses can be separated by their POS. Since POS-tagging is available for many languages, and there are also options for unsupervised POS-tagging (Christodoulopoulos et al., 2010) we experimented with this.

4.2 Simple Part-of-Speech Tagging

To very naively separate some word-senses we preprocessed both the English and German corpora from the previous chapter with the Stanford POS-tagger (version 3.5.2) (Toutanova et al., 2003), using the fastest tag-models. Afterwards, we rewrote the POS tags to the following simplified tags:

- Nouns
- Verbs
- Adjectives
- Adverbs
- Other (no tag)

A table with the simplification rules for the tags can be found in the Appendix. An example of what the text looks like after tagging and simplification is given in Sentence (1).

(1) Every day_N , I walk_V my daily_Adj walk_N .

In the example we can see that walk_V is separated from walk_N, which will give us two different vectors. We chose these four tags as they correspond to the POS-tags provided in WordNet and GermaNet. In this way, we can have a straightforward way to evaluate on the vocabulary (e.g. S_{train}). For each word, we now evaluate with regard to the synonyms that have the same POS in WordNet or GermaNet.

Another advantage of having these simple POS-tags is that we can filter bad synonyms from the 10-most cosine similar words. Synonyms are very similar also on a grammatical level, as they are interchangeable in many contexts, so they should be of the same part-of-speech.

Because the vocabulary has changed, and the average frequency of words is now lower, as some words are split, we again analyze what word vector training parameters work best. We train CBoW and Skip-gram vectors on the tagged corpus, varying the dimensions over {150, 300, 600}, and the contextual window size over {2, 4, 16, 32}. We calculate precision for the first-most-similar and second-most-similar word for all words in S_{train}. The results are shown in Tables 10, 11, 12, and 13.

If we look at table 10, we can see that the highest precision is obtained using a CBoW model with a window size of 4, and 600 dimensions. If we compare this to the best

CBoW (Tagged)															
dim.	150					300					600				
win.	2	4	8	16	32	2	4	8	16	32	2	4	8	16	32
P-1	0.079	0.080	0.073	0.067	0.060	0.084	**0.085***	0.080	0.074	0.066	0.084	0.084	0.081	0.073	0.069
P-2	0.058	0.056	0.053	0.049	0.045	0.061	0.061	0.059	0.055	0.050	0.061	0.062	0.059	0.055	0.053

Table 10: Precision for different window sizes and number of dimensions, using the CBoW model, for POS-tagged English.

Skip-gram (Tagged)															
dim.	150					300					600				
win.	2	4	8	16	32	2	4	8	16	32	2	4	8	16	32
P-1	0.068	0.065	0.057	0.049	0.045	0.069	0.066	0.057	0.052	0.046	0.067	0.062	0.052	0.046	0.041
P-2	0.050	0.047	0.041	0.038	0.036	0.050	0.047	0.042	0.038	0.035	0.050	0.045	0.038	0.034	0.031

Table 11: Precision for different window sizes and number of dimensions, using the Skip-gram model, for POS-tagged English.

CBoW (Tagged)															
dim.	150					300					600				
win.	2	4	8	16	32	2	4	8	16	32	2	4	8	16	32
P-1	0.086	0.092	0.094	0.092	0.090	0.092	0.100	0.100	0.099	0.094	0.090	0.102	**0.103***	0.101	0.101
P-2	0.060	0.065	0.066	0.065	0.063	0.065	0.069	0.072	0.070	0.069	0.064	0.070	0.072	0.071	0.071

Table 12: Precision for different window sizes and number of dimensions, using the CBoW model, for POS-tagged German.

Skip-gram (Tagged)															
dim.	150					300					600				
win.	2	4	8	16	32	2	4	8	16	32	2	4	8	16	32
P-1	0.084	0.085	0.086	0.082	0.080	0.085	0.085	0.083	0.077	0.077	0.082	0.079	0.072	0.066	0.065
P-2	0.059	0.061	0.061	0.059	0.058	0.061	0.063	0.059	0.057	0.056	0.058	0.059	0.053	0.049	0.047

Table 13: Precision for different window sizes and number of dimensions, using the Skip-gram model, for POS-tagged German.

results on the non-tagged corpus, from Table 2 in Chapter 3, the optimal window size has stayed the same. Also CBoW vectors still perform better than Skip-gram vectors, and also low windows work best for Skip-gram vectors. However, the best performing number of dimensions went from 600 to 300 when adding the POS-tag for English. A possible explanation can be that since the part of speech tags separate some of the word context, based on grammatical properties, the same information can be encoded with less dimensions.

For German, the precision went up when adding the POS-tags. This can be seen if we compare the precision from Tables 4 and 5 with Tables 12 and 13.The best vectors are still CBoW vectors of dimension 600, with a contextual window of 8. When we tried to find the reason why German has such a precision increase, compared to English, we found that it lies partially at the level of POS-tag simplification. As in the German

part-of-speech tagset, *the Stuttgart-Tübingen tagset* (STTS), names are not considered as nouns. For this reason we did not simplifify them to an N-tag, and they were excluded during evaluation. This was not the case for English. Names are one of the frequent error categories we found in Chapter 3.

This highlights another use of the POS-tagger, which is that we can simply exclude categories for which we don't want to find synonyms, and maybe even further filter bad synonym candidates from the 10-most-similar words. An example would be the frequent error category of plurals, but also other types of inflections, which can be filtered, as they are given a different POS-tag (before simplification). These insights will be used in the final system, presented in Chapter 5.

To compare using the simplified POS-tags with the previous approaches we also calculated precision, recall and f-measure on S_{test}. We found, compared to the baseline of looking only at the most-similar word, that for English precision did not change (11%). Recall increased from 3% to 4%, and f-measure as from 5% to 6%. For German, precision increased with 8% to 12%, recall from 5% to 7%, and f-measure from 6% to 9%.

From this experiments we conclude that POS-tags can help to increase synonym extraction in three ways. Firstly, they can separate some of the word-senses, however this effect is minor. Secondly, they can filter words that are not grammatically similar enough, such as plurals. And third, they can exclude synonyms for categories for which there no, or very few synonyms, such as names.

4.3 Clusters of Vector Offsets

In this section, we look at clusters of *vector offsets* $(v_1 - v_2)$. In previous research on especially Skip-gram vectors, semantic or lexical relations between words seem to be encoded in vector offsets (Mikolov et al., 2013b; Tan et al., 2015; Fu et al., 2014). We do not know if synonymy is also represented by the offset of vectors. In this thesis, we do not directly try to find a vector offset that represents synonymy. We expect that if we can make sensible relational clusters, this can already give information on if two words are synonyms or not. For example, if the vector offset of two words is in the plural-cluster, they are probably not synonyms. Not only the relation between the two words, but also the word class of the words might give information on synonymy. For example, as mentioned in the previous section, it can be beneficial to recognize names, as they have very few synonyms. For this reason, we also experiment with concatenating the vector of a word with its vector offset $(v_1||v_1 - v_2)$. In this way, We hope to find relations that go together with particular word categories.

We consider 1500 English word pairs, constructed by taking 1500 arbitrary words from

	CCA		Normal	
	CboW	SG	CboW	SG
$v_1 - v_2$	0.001	0.001	0.002	0.002
$v_1 \| \| v_1 - v_2$	0.002	0.003	0.005	0.003

Table 14: Variance in synonym probability of word-pair clusters for different word-pair vectors, in different vector spaces.

S_{train} and their most-cosine-similar word. We cluster the word pairs in 100 clusters, using k-means clustering. For this, we use the scikit-learn implementation of k-means in Python Pedregosa et al. (2011).

We calculate for each cluster the probability of a word to be a synonym, i.e. the proportion of synonyms in the cluster. We expect that a good clustering has a high variance in their synonym probability distribution over the clusters. If the variance is high, this means some clusters contain many synonyms, and some contain only few clusters. In this case, the cluster of a word-pair gives a strong cue on synonymy. When calculating the variance we only considered clusters that contain at least 10 word-pairs, since very small clusters have an unreliable estimate of their synonym probability.

We make clusters for both $v_1 - v_2$ and $v_1||v_1 - v_2$ vectors, for the CBoW and Skip-gram vectors. We also look at CboW and Skip-gram vectors that were projected using *canonical correlation analysis* (CCA), as explained in Chapter 2. This can be interesting, because if it is possible to get good clusters in such common vector space, it might be possible to use synonym resources in one language, to find improve finding synonyms in a second language.

To apply CCA, we require a table of word-to-word translations between English and German. We constructed this from the English-to-German, and German-to-English translation dictionaries from Freedict.org (Eyermann, 2015). We added the translations from either dictionary for all words in the respective vector vocabularies. This resulted in 32,387 word-to-word translations.

Because k-means clustering gives different clusters depending on the initialization of the centroids, we repeated the experiments three times, and report averaged results. The results are shown in Table 14.

What can be seen from the variance is that Skip-gram and CBoW clusters seem to perform equally well in separating synonyms when using the vector offset. We can also see that performance is slightly worse after projecting the vectors with CCA. This is in line with the findings of Faruqui and Dyer (2014), where they also found slightly worse performance in word-relation task when projecting Skip-gram vectors.

When we consider not only the offset, but the concatenation of the word-vector and the offset, we find better separating clusters. Again, after projecting, the variance is lower, especially for CBoW. We also looked manually at each of the clusterings. We noticed that when using only the offset vector, clusters generally represent relations, such as plurals, or present and past tense. Also, we noticed that the relations are generally clearer for Skip-gram clusters. However, when we looked at the concatenation of the vector and the offset the clusters seemed less clear. They seemed to be a mix between word categories and relations. So, the reason we think why the clusterings of concatenated vectors have bigger variance is because it is clustering the word categories maybe has higher impact on the synonym probability. In this thesis we did not look further into the properties of these clusters. But, it could be interesting to explore further in future work.

We also experimented with combining the synonym probability of a word pair's cluster, with the relative cosine score. We did this in a straightforward way, by taking the product of the relative cosine similarity of a word-pair, and the probability that the pair is a synonym based on its cluster. So the combined score for a word pair (w_i, w_j) is calculated as in Equation 5, where $g(w_i, w_j)$ is either the offset between the vectors of w_i and w_j, or the vector of w_i concatenated with the offset between the two vectors. And $cl(\cdot)$ assigns the cluster of the vector of the word-pair.

$$score(w_i, w_j) = rcs_n(w_i, w_j) \cdot P(synonym|cl(o(w_i, w_j)) \tag{5}$$

We made clusters in CBoW and Skip-gram vectors of all words in S_{train} and their most-cosine-similar word, and optimized the threshold of the combined score on S_{train}. Then we used the optimal threshold to extract synonym from the test set S_{test}. When using clusters of the offset of the two word vectors, precision is 0.08, recall 0.07, and f-measure 0.07. When using clusters of the concatenation of the vector for the first word, and the offset of the two word vectors, precision is also 0.08, recall is 0.6, and f-measure is 0.7. We observe that the recall is higher when only using the vector offsets, even though the variance for this condition in the experiment is lower, compared to using the concatenated condition. This might indicate that the clustering in the experiment might be too small to make proper estimations for their synonym probability.

From this observations we conclude that there is still room for improvement with regard to exploiting the vector offsets for synonym extraction. Mainly, because the results on the test set are lower than when not using the clusters.

5 Final System and Evaluation

In this Chapter we will describe the final systems, for English and German, that we constructed from the findings from the previous Chapters. We evaluate the system performance manually, and by using the extracted synonyms in machine translation evaluation.

5.1 The System

For the final systems we used bigger corpora than those used in the previous experiments. We used 500 million word sections from the same corpora as before, the English and German NewsCrawl 2014 corpora from the workshop on machine translation in 2015. We part-of-speech tagged the corpora using the same parser and models as in Chapter 4. However, we do not simplify the POS-tags, but only tag words with any of the different tags for nouns, verbs, adjectives or adverbs. We exclude the tags for names, as they have few to no synonyms. In Sentence (1), an example is given of a tagged sentence.

(1) I often_RB like_VBP to walk_VB my daily_JJ walk_NN .

The exact tags we consider for English and German are given in the Appendix. It should be noted that in the German tagset, there is only one tag for nouns, that covers both singular and plural nouns. This might results in more errors. For machine translation evaluation we do not expect this to have a big negative impact, as plurals would also have been matched by Meteor in the stemming module. However, it might result in a worse human evaluation.

For English, we train CBoW vectors of dimension 300, with a contextual window size of 4. We also train Skip-gram vectors of dimension 300, with a contextual window size of 2. For German, we train vectors with the same specifications, except for the German CBoW model we use a contextual window of size 8, and for Skip-gram a window of size 4.

We only consider words that occur at least 20 times in the corpus. The English vocabulary we consider, i.e. that is frequent and is tagged, contains 115,632 words, and the German vocabulary 311,664.

We then calculate the summed relative cosine similarity of both the CBoW and the Skip-gram vectors for the full vocabulary with regard to the top-10 most cosine-similar words. We select words with a summed rcs_{10} similarity higher than 0.22. We choose 0.22 as it lies slightly above the expected tilting point of 0.2. For English, we obtain 16,068 word-pairs. For German, we obtain 96,998 word-pairs. It should be noticed that the word-pairs are also tagged, which can be useful depending on the application.

5.2 Manual Evaluation

To evaluate the precision of the obtained synonyms, we took a random sample of 200 word pairs for both languages. The word pairs were then annotated for synonymy. The annotation categories are synonyms, non-synonyms, or unknown. In the description the unknown category is indicated for when an annotator does not know any of the two words. The annotators could also indicate hesitation, but still had to give a preference for any of the three categories.

For English, annotation is done by two annotators. One annotator is native English speaker, and one fluent non-native speaker. For German, annotation is also done by two annotators. One annotator is native German speaker, and one intermediate non-native speaker.

We discriminate several situations:

SS: Both annotators annotate synonymy

NN: Both annotators annotate non-synonymy

SU: One annotator annotates synonymy, and the other unknown

NU: One annotator annotates non-synonymy, and the other unknown

SN: One annotator annotates synonymy, and the other non-synonymy

UU: Both annotators annotate unknown

We assume that if both annotators do not know the words, there is no synonymy. We can calculate a *lower bound of precision* (P^{-}_{syn}), and an *upper bound of precision* (P^{+}_{syn}). For the lower bound, we only consider word-pairs of category SS as synonyms, and the rest as non-synonyms. For the upper bound, we consider word-pairs of category SS and SU as synonyms, and the rest as non-synonyms.

We also calculate a lower and upper bound for non-synonymy ($P^{-}_{\neg syn}$, and $P^{+}_{\neg syn}$), and the percentage of disagreement on the two categories synonym, and non-synonym ($P_{disagree}$). This way we can get a better idea of how many clear errors there are, and how many errors are discutable.

The results for both English and German are shown in Table 15. What can be noticed is that for German, the precision is quite a bit lower than for English. However, the number of found word-pairs is much higher. One reason can be that the threshold should be higher in order to get comparable precision. A second reason can be that for English the error categories, such as plurals, are separated by a POS-tag, resulting in higher precision. In

Manual Evaluation	P^{-}_{syn}	P^{+}_{syn}	$P^{-}_{\neg syn}$	$P^{+}_{\neg syn}$	$P_{disagree}$	P_{UU}
English	0.55	0.59	0.15	0.21	0.16	0.05
German	0.30	0.35	0.42	0.49	0.15	0.03

Table 15: Manual evaluation of the final systems.

the German tagset these are not separated. We found that 10% of the German word-pairs in this set are plurals. For English, there were no such cases. For our application, these errors should not be a big problem, as plurals would otherwise have been matched by the stemming module of Meteor.

The percentage of unknown words seems fairly small, and about the same for both languages. Also the disagreement on synonymy seems about the same for both languages (around 15%).

5.3 Application in Machine Translation Evaluation

To see if the extracted synonyms can be beneficial in an application we also used them in machine translation evaluation. We use them in the synonym module of the Meteor 1.5 evaluation metric. As mentioned in the introduction, Meteor already uses synonyms for English, but not yet for German. For English, we add our 16,068 word-pairs to the existing resource (WordNet) that Meteor uses. For German, the synonym resource will consist only of our 96,998 word-pairs. Meteor weighs the scores from each matching module. For English, we use the default weights as mentioned by Denkowski and Lavie (2014), as synonyms were already incorporated for English. For German, we use the default weights for all other modules, except we use the same weight for the synonym module as used for English (0.80).

We evaluate in two steps. First, we look at the impact of adding the synonyms on the Meteor score of a set of hypothesis and reference sentences. Secondly, we look if the updated Meteor score also better correlates with human judgments. For both steps, we use the data from the metrics task of the workshop on machine translation 2014[4] (WMT 2014) (Machácek and Bojar, 2014).

To evaluate the impact of adding the synonyms on the Meteor score, we use the news-test reference sentences from the language pair German-English, for English. This set consists of around 3000 segments, or sentences. For German, we use the reference sentences from the language pair English-German. This set consists of around 2700 segments, or sentences. We choose the news-test hypothesis sentences generated by the UEDIN-SYNTAX machine translation system for both English, and German (Williams et al.,

[4]http://www.statmt.org/wmt14/results.html

2014). As this systems has high scores for each language pair, based on the human-annotated system rankings (Bojar et al., 2014).

We calculate the Meteor score for the following two condition:

1. Using all four modules, with the default weights, and the existing synonym resource (no synonyms for German).
2. Using all four modules, using default weights, and adding our synonyms to the existing resource.

Meteor score	English	German
Condition 1	0.27658	0.36048
Condition 2	0.27674	0.36298

Table 16: Meteor scores, using the existing synonym resource (condition 1), and when adding our extracted synonyms (condition 2).

In Table 16, the Meteor scores for each condition and each language is shown. What can be noticed from the results is that in both cases the Meteor score is higher when adding the extracted synonyms. This means more synonym alignments could be made between the hypothesis, and reference segments of each language, after adding the new synonyms. Furthermore, it can be noticed that for German adding the synonyms has a bigger impact than for English. This can be caused by two things. Firstly, because there were no synonyms yet for German. Secondly, we extracted more synonyms for German than for English, as German has a bigger vocabulary size.

To evaluate if we do better machine translation, we evaluate the correlation to human judgments over the hypotheses sentences of different machine translation systems. We calculate *segment level Kendall's τ correlation* as calculated in the WMT 2014 for the default Meteor 1.5 metric, and after adding our synonyms to the synonym module. This correlation coefficient is expected to predict the result of the pairwise comparison of two translation systems. In the WMT 2014, this is calculated using human judgments on a ranking task of 5 systems per comparison. The correlation is then calculated as in Equation 6, where *Concordent* is the set of human comparisons for which the Meteor score suggests the same order, and *Discordant* is the set of all human comparisons for which a given metric disagrees. When the Meteor score gives the same rankings as the human judgments correlation will be high, and vise versa.

$$\tau = \frac{|Concordant| - |Discordant|}{|Concordant| + |Discordant|} \tag{6}$$

To calculate the correlation, we used the human judgments from the metric task of the WMT 2014, and calculated the Meteor scores for hypotheses from the 13 translation

German-**English**	segment-corr.	system-corr.
Condition 1	0.334*	0.927*
Condition 2	0.333*	0.927*

English-**German**	segment-corr.	system-corr.
Condition 1	0.238*	0.263*
Condition 2	0.243*	0.277*

Table 17: System level correlations, and segment level correlations for the default Meteor 1.5 score (condition 1), and when adding the extracted synonyms (condition 2)

systems for the language pair German-English, and the 18 translation systems for English-German.

We also calculated the system level correlation, which indicates to what degree the evaluation metric orders the translation systems in the same order as the human judgments do, based on the total system score that the evaluation metric gives to each system. This is calculated as the *Pearson correlation*, as described by Machácek and Bojar (2014), or in Equation 7, where H is the vector of human scores of all systems translating in the given direction, M is the vector of the corresponding scores as predicted by the given metric, here Meteor. $\bar{H}$ and $\bar{M}$ are their means respectively.

$$r = \frac{\sum_{i=1}^{n}(H_i - \bar{H})(M_i - \bar{M})}{\sqrt{\sum_{i=1}^{n}(H_i - \bar{H})^2}\sqrt{\sum_{i=1}^{n}(M_i - \bar{M})^2}} \tag{7}$$

Both the segment-based correlations and the system-level correlations are shown in Table 17 for the same conditions as mentioned before. What can be noticed is that for English it appears that adding the synonyms to the existing synonym resource has a small negative effect on the segment correlation. A reason for this can be that Meteor ignores the POS-tags that were given to the synonyms. If two words are synonymous with regard to their part of speech, but not synonymous if they are of different parts-of-speech, Meteor will align them in both situations. In the case when the words are of different POS, they will be falsely aligned by Meteor.

For German, adding the extracted synonyms improves the Meteor metric. Both the segment level correlation, and the system level correlation increase when adding the extracted synonyms. This might seem odd at first, since the German synonyms had a lower precision in manual evaluation compared to the English synonyms. But still, they perform better in machine translation evaluation. This can be explained by, what was already mentioned earlier, that a significant part of the German synonym errors are inflections, due to the difference in POS-tagset. A second explanation is that the synonyms extracted for German are less ambiguous with respect to their POS. German frequently uses compounding (e.g. schwierigkeitsgrade/ degree of difficulty), and grammatical case markers. This might result in less ambiguous words. The negative effect, that Meteor does not use the POS of the synonyms, could be smaller for German for this reason.

6 Conclusions & Future Work

In this thesis we explored different methods to extract synonyms from text. The initial motivation was to use the extracted synonyms to improve machine translation evaluation. We tried to extract the synonyms using as little supervision as possible, with the goal that the same method can be applied to multiple languages. We experimented with English and German.

We used word vectors trained using the continuous bag-of-words model (CBoW), and the skip-gram model (SG) proposed by Mikolov et al. (2013a). We evaluated different training parameters for training these vectors, with regard to synonym extraction. It appeared that for our experiments CBoW vectors gave higher precision and recall than SG vectors. The number of dimensions did not appear to play a very big role. For our experiments, dimensions of 300, and 600 seemed to give best results. The optimal contextual windows size seemed to be around 4 for English, and 8 for German. We hypothesized that the difference in window size can be because of the difference in the distributions of word categories of the synonyms in WordNet and GermaNet. If this is not the case, it could be interesting to see what would be the underlying reason for this difference, and what optimal window sizes are for other languages.

For English, we manually looked at frequent error categories when using these vectors for this task. The biggest well-defined error categories we found are inflections, co-hyponyms, and names. A more vague error category is that category of related words for which the exact type of relation is not clear.

We found that the cosine similarity on its own is a bad indicator to determine if two words are synonymous. We proposed *relative cosine similarity*, which calculates similarity relative to other cosine-similar words in the corpus. This seems to be a better indicator, and can help to improve precision, at the cost of recall. Also, the optimal thresholds for finding synonyms for English, and German using this measure, are almost the same. This gives hope for easy extension of this method to other languages, for which there is no synonym data. It would be very interesting to see to which other languages this method can generalize, and to which languages it cannot.

We also experimented with combining similarity scores from differently trained vectors, which seems to slightly increase both precision, and recall. Furthermore, we explored the advantages of using a part-of-speech tagger, as a way of introducing some light supervision. POS-tags can help performance in different ways. Firstly, it can disambiguate some of the meanings of homographs. Secondly, it can help filtering bad synonym candidates. And third, it can prevent extraction of synonyms for word categories that have no, or very few synonyms, such as names. For future research, it can be interesting to see what the effect is of using an unsupervised POS-tagger (Christodoulopoulos et al., 2010). Or, to

tackle the problem of homography in a different way. Future research could include using topical word embeddings (Liu et al., 2015b,a), or global context vectors (Huang et al., 2012). These types of vectors both make different vectors for each word, using topical information to disambiguate some of the different word-senses.

We evaluated our final approach for both English and German. We did a manual evaluation, with two annotators per language. Also we applied the extracted synonyms in machine translation evaluation. From the manual evaluation, for the parameters we chose, the English synonyms had higher precision than the German ones. The most probable reason for this is that the English POS-tagset better separates the frequent error categories mentioned in Chapter 3.

When applying the synonyms in the Meteor machine translation evaluation metric, for English there was a small negative effect. This is most probably due to the fact that Meteor does not use the POS-tags of the extracted synonyms, resulting in overgeneralization of synonymy on the extracted word-pairs. For German, using the extracted synonyms has a clear positive effect on the evaluation metric. The reason why the effect of ignoring the POS-tags seems smaller could be that German uses compounding, and more elaborate case marking. This could result in less ambiguous words, which makes the effect of ignoring POS-tags smaller. For future research on improving Meteor, it could be interesting to incorporate POS-tags, to prevent this overgeneralization of synonyms.

References

Banerjee, S. and Lavie, A. (2005). Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, volume 29, pages 65–72.

Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., et al. (2014). Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58. Association for Computational Linguistics Baltimore, MD, USA.

Christodoulopoulos, C., Goldwater, S., and Steedman, M. (2010). Two decades of unsupervised pos induction: How far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 575–584. Association for Computational Linguistics.

Denkowski, M. and Lavie, A. (2014). Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Eyermann, H. (2015). Freedict.org. http://freedict.org/en/. Accessed: August, 2015.

Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E., and Smith, N. A. (2014). Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*.

Faruqui, M. and Dyer, C. (2014). Improving vector space word representations using multilingual correlation. In *Proceedings of EACL*, volume 2014.

Fu, R., Guo, J., Qin, B., Che, W., Wang, H., and Liu, T. (2014). Learning semantic hierarchies via word embeddings. In *Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics: Long Papers*, volume 1.

Gupta, D., Carbonell, J., Gershman, A., Klein, S., and Miller, D. (2015). Unsupervised phrasal near-synonym generation from text corpora. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Harris, Z. S. (1954). Distributional structure. *Word*.

Hermann, K. M. and Blunsom, P. (2014). Multilingual models for compositional distributed semantics. *arXiv preprint arXiv:1404.4641*.

Huang, E. H., Socher, R., Manning, C. D., and Ng, A. Y. (2012). Improving word representations via global context and multiple word prototypes. In *Proceedings of the*

50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1, pages 873–882. Association for Computational Linguistics.

Levy, O. and Goldberg, Y. (2014). Dependencybased word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 302–308.

Lin, D., Zhao, S., Qin, L., and Zhou, M. (2003). Identifying synonyms among distributionally similar words. In *IJCAI*, volume 3, pages 1492–1493.

Liu, P., Qiu, X., and Huang, X. (2015a). Learning context-sensitive word embeddings with neural tensor skip-gram model. *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Liu, Y., Liu, Z., Chua, T.-S., and Sun, M. (2015b). Topical word embeddings. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Luong, M.-T., Socher, R., and Manning, C. D. (2013). Better word representations with recursive neural networks for morphology. *CoNLL-2013*, 104.

Machácek, M. and Bojar, O. (2014). Results of the wmt14 metrics shared task. *ACL 2014*, page 293.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Ono, M., Miwa, M., and Sasaki, Y. (2015). Word embedding-based antonym detection using thesauri and distributional information. In *Proc. of NAACL*.

Pak, A. A., Narynov, S. S., Zharmagambetov, A. S., Sagyndykova, S. N., Kenzhebayeva, Z. E., and Turemuratovich, I. (2015). The method of synonyms extraction from unannotated corpus. In *Digital Information, Networking, and Wireless Communications (DINWC), 2015 Third International Conference on*, pages 1–5. IEEE.

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

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12.

Plas, L. v. d. and Bouma, G. (2005). Syntactic contexts for finding semantically related words. *LOT Occasional Series*, 4:173–186.

Saveski, M. and Trajkovski, I. (2010). Automatic construction of wordnets by using machine translation and language modeling. In *13th Multiconference Information Society, Ljubljana, Slovenia*.

Tan, L., Gupta, R., and van Genabith, J. (2015). Usaar-wlv: Hypernym generation with deep neural nets. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Van der Plas, L. and Tiedemann, J. (2006). Finding synonyms using automatic word alignment and measures of distributional similarity. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 866–873. Association for Computational Linguistics.

Williams, P., Sennrich, R., Nadejde, M., Huck, M., Hasler, E., and Koehn, P. (2014). Edinburghs syntax-based systems at wmt 2014. In *Proc. of the Workshop on Statistical Machine Translation (WMT), Baltimore, MD, USA*, pages 207–214.

Yu, H., Hatzivassiloglou, V., Friedman, C., Rzhetsky, A., and Wilbur, W. J. (2002). Automatic extraction of gene and protein synonyms from medline and journal articles. In *Proceedings of the AMIA Symposium*, page 919. American Medical Informatics Association.

List of Figures

List of Tables

Appendix

Simplification of the English and German Tags

Simplification rules for rewriting tags from the Penn Treebank Tagset (PTTS), and for rewriting tags from the Stuttgart-Tübingen Tagset (STTS), as used in Chapter 4.

PTTS tag	Simplified tag
JJ	ADJ
JJR	ADJ
JJS	ADJ
NN	N
NNS	N
NNP	N
NNPS	N
RB	ADV
RBR	ADV
RBS	ADV
RP	ADV
MD	V
VB	V
VBZ	V
VBP	V
VBD	V
VBN	V
VBG	V

STTS tag	Simplified tag
ADJA	ADJ
ADJD	ADJ
ADV	ADV
NN	N
VVFIN	V
VVIMP	V
VVINF	V
VVIZU	V
VVPP	V
VAFIN	V
VAIMP	V
VAINF	V
VAPP	V
VMFIN	V
VMINF	V
VMPP	V

Table 18: Tagset simplifications.

Final Selection of English and German Tags

Selected tags for the final system as described in Chapter 5, for English from the Penn Treebank Tagset (PTTS), for German from the Stuttgart-Tübingen Tagset (STTS).

English

Nouns: NN, NNS
Verbs: MD, VB, VBZ, VBP, VBD, VBN, VBG
Adjectives: JJ, JJR, JJS
Adverbs: RB, RBR, RBS, RP

German

Nouns: NN
Verbs: VVFIN, VVIMP, VVINF, VVIZU, VVPP, VAFIN, VAIMP, VAINF, VAPP, VMFIN, VMINF, VMPP
Adjectives: ADJA, ADJD
Adverbs: ADV, PAV

Index