KØBENHAVNS UNIVERSITET

Bachelor thesis

Thor Steen Larsen (mvj665)

Phase Transitions In Word Embeddings

Department of Computer Science

Supervisor: Daniel Hershcovich

16. juli 2020

Abstract

Hershcovich et al. (2019) and Mckinney-Bock and Bedrick (2019) found anecdotally that word embedding of Word2Vec models using the CBoW algorithm with a context window size 1 judge similarity better than those trained with larger windows and performance improves gradually. This bachelor thesis aims to research why this is happening, and what is changing in this phase transition. We reproduce Skip-gram and CBoW models with similar parameters on different context windows sizes and evaluate on the WS353 data-set. Furthermore, we look into relations between the most similar word pairs found by the word embedding and inductively come up with a hypothesis that the model judgement of sister-terms given the WS353 evaluation data-set can explain why a model with a narrow context windows performs better. We test our hypothesis on the SimLex data-set and find that we cannot confirm our hypothesis, but that Word2Vec model are highly sensitive to sister-terms, synonyms and antonyms which indicate word embedding alone does not and does capture similarity given the semantic relation.

Word2Vec Models are created using Gensim's implementation of the original Word2Vec model by Mikolov et al. (2013a), evaluated using Spearman's ρ and our hypothesis is tested using a hyper-geometric significance test.

Indhold

1 Theory

1.1 Natural Language Models

By looking at natural language, we in general have a stream of words in sentences where a certain number of words fall into a certain. By calculating the probability that a word falls into a bin, we can from a frequentist point of view begin to understand word relations and derive a good statistical estimator of word context. That is the context of word relations for every word in the stream. Contextual information provides a good approximation to word meaning, since semantically similar words tend to have similar contextual distributions Silke Scheible and Springorum (2013).

In linguistics, this is formulated as the distributional hypothesis, that is words are used and occur in the same contexts tend to infer similar meanings (Harris, 1954). In our coverage of the background of language model, we will not go further into this.

Distributional semantic modelling in vector spaces make language model work efficiently on computers and make us able to use architectures such as Word2Vec. In our case we want to experiment with the predictive models using Word2Vec which given semantic, word vectors assigns weights to differently distributed sentences and tries to optimise the weights of these.
There exist many alternatives to this such as count-based models which use word n-gram, a sequence of words in a sentence $w_1, \dots w_n$, we are interested in the prediction task:

$$P(w_n | w_1 \dots w_{n-1}) = \frac{P(w_1 \dots w_n)}{P(w_1 \dots w_{n-1})}$$

where P is the probability function of w conditioned on the word n-gram w_{n-1}. We then calculate P and find the Maximum Likelihood Estimator. The parameters of the MLE will be the words with the highest probability of being in the context of the word of interest. Here we assume the Markov property that the probability of one word affects the probability of the next word.

In turn, predictive models have shown to be highly superior in deciphering semantic context than count-based methods due to the fact that simply counting the amount of word does not tell you much about semantic relations between words (Baroni et al., 2014).

In predictive models, we use more advanced probability calculation to create word vectors such as shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words which is used the model of our interest, Word2Vec.

The Word2Vec model is maybe the most widely used predictive model to produce word embedding. Word2vec takes as input a large corpus of text and produces a vector space with each unique word in the corpus being assigned a corresponding vector in the space. Depending on the algorithm used, CBoW or Skip-gram, word vectors are positioned in the vector space. In the following paragraph, I will go deeper into this.

1.2 CBoW

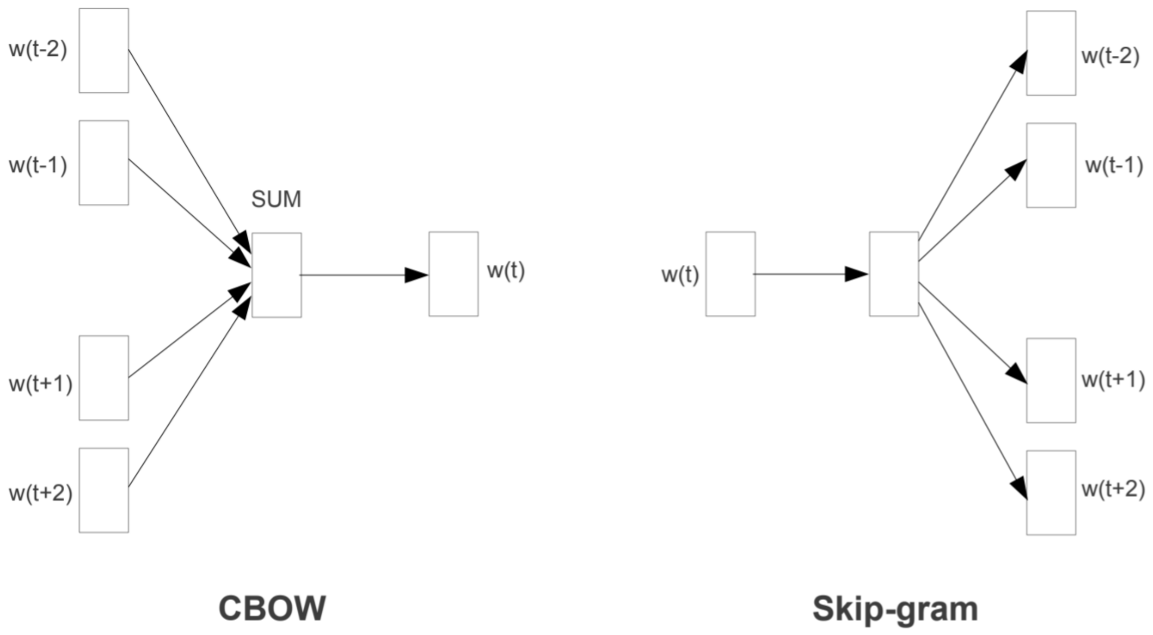


Fig 1: CBoW and Skip-gram architecture (Mikolov et al., 2013a)

In Word2Vec models using the CBoW algorithm, we use a similar model to a neural network language model to learn the embedding of each word given its context. More precisely, we from the word's context $ww_{\text{window size}}$ want to predict the middle word. The basic CBoW architecture is as following.

As the input layer we have an one-hot encoded input context words represented as the vectors $\{\mathbf{x}_1, \ldots \mathbf{x}_C\}$ for a word of size C and vocabulary V depending on the size of the window and vocabulary.

The hidden layer is an N-dimensional vector $\mathbf{h}$ connected to the hidden layer via a $V \times N$ weight matrix $\mathbf{W}$ and the hidden layer is connected to the output layer via a $N \times N$ weight matrix **W'**.

During forward propagation in the neural network $\mathbf{h}$ is computed by

$$\mathbf{h} = \frac{1}{C}\mathbf{W} \cdot (\sum_{i=1}^{C} \mathbf{x}_i)$$

which is the average of the input vectors weighted by the matrix $\mathbf{W}$. To compute the inputs to each node in the output layer

$$u_j = {v'_{w_j}}^T \cdot \mathbf{h}$$

where v'_{w_j} is the j'th of the output matrix $\mathbf{W}$'.The output y is calculated by passing the input $\mathbf{u}_j$ through the soft max function

$$y_j = p(w_{y_j} | w_1 \ldots, w_C) = \frac{\exp(u_j)}{\sum_{j'=1}^{V} \exp(u'_j)}$$

When we have learned the weight matrices $\mathbf{W}$ and $\mathbf{W}$', we back-propagate in the neural network by using negative logarithmic in an error function Mikolov et al. (2013a). Similar to the MLE, the objective is to maximise the conditional probability of a output word given a input context, therefore our loss function will be

$$\begin{aligned} E &= -\log p(w_O | w_I) \\ &= -u_{*j} - \log \sum_{j'=1}^{V} \exp(U_{j'}) \\ &= -\mathbf{v}_{w_O}^T \cdot \mathbf{h} - \log \sum_{j'=1}^{V} \exp(\mathbf{v}_{w_{j'}}^T \cdot \mathbf{h}) \end{aligned} \tag{1}$$

Where $*j$ is the index of the the actual output word, w_O is the middle output word and w_I is the previous/next input word. The next step is to derive the update equation for the hidden-output layer weights $\mathbf{W}$', then derive the weights for the input-hidden layer weights $\mathbf{W}$. We then update the weights for the input and output hidden layer by using stochastic gradient descent or a similar method to optimise the weights. We can use the final output of a word vector $y \times \mathbf{W}$ by running it through a soft-max function to learn the probability of randomly picking a word x nearby any word in our vocabulary V.

In the predictive model, the words are averaged and the projection layer is shared for all the words. This is the reason why CBoW is a bag-of-words model as the order of words in the history does not influence the projection to the output layer and all the word vectors are averaged (Mikolov et al., 2013a). This computation which happen in the hidden layer together with the objective is the main difference compared to the Skip-gram model.

1.3 Skip-gram

As in CBoW, we use a predictive model to obtain the word embedding. This time using a middle input word to infer the word's context.

The basic Skip-gram architecture is a mirror of the one in CBoW. Given a random word from the context of the window, we built up a data set of word pairs. We then calculate the probabilities of each word pair by feeding it to the hidden layer of a simple neural network in the form of a log-linear classifier with one layer per word in the vocabulary as features. From the hidden layer, we then obtain the weights which times our vector and through a soft-max give us the same result as CBoW.

Or as put by the authors of Word2Vec, Mikolov et al. (2013a), we use each current word as an input to a log-linear classifier with continuous projection layer, and predict words within a certain range before and after the current word. We then use an error function and stochastic gradient decent as in the architecture of CBoW.

In addition to the general soft-max function Mikolov et al. (2013b) implements a hierarchical softmax function to optimise the computation. This implementation uses a binary tree representation of the output layer with V words as its leaves and for each node the relative probabilities of its child nodes. This gives a random walk that assigns probabilities to the words. Furthermore, negative sampling can be used to decrease the computational cost (Mikolov et al., 2013b).

1.4 Windows size and context

One of the main ways distributional semantic models differ is in terms of the context window. The window size parameter in Word2Vec or size of the context window defines the word pairs we use to calculate the probability of each word's context. This means that it has a critical role in determining how the model predicts the next word.

If we have a sentence: ”The capital of Denmark is Copenhagen and the its second largest city is Aarhus”. A W2V network with a window size 1 is going to learn that there is high probability between (Capital, of, Denmark, is, Copenhagen) and less probability between (Capital, of, Denmark, is, Aarhus). A model using the Skip-gram algorithm would use word pairs to learn this while a model with the CBoW algorithm would take the whole sentence / context to learn this.

This means that the size of the context window has a great effect on how our model evaluates related and similar words. Hershcovich et al. (2019) and Mckinney-Bock and Bedrick (2019) have shown in their experiments that there is significant difference between similarity scores of CBoW models with a window size of 1 and 2.

Mikolov et al. (2013a) found that increasing the windows size improves quality of the resulting word vectors, but it also increases the computational complexity, and that the most efficient context window was of size 10 in their case.

1.5 Semantic relations and evaluation data-sets

In linguistics similarity both refers to what we humans associate with being similar and similar words in terms of attributes such as co-hyponyms. Following is a list with a short description of semantic relation and how the data set, we will use for evaluation of our model are annotated by humans.

1.5.1 Semantic relations

There are several different kinds of semantic relations. Following is short summery of these which will use in our analysis.

1. Synonyms refer to words that have exactly the same meaning such as sick and ill.
2. Identity refer to words that have the same lemmas such as king and queen.
3. Antonyms refer to words that have the opposite same meaning such as man and woman.
4. Meronyms refer to a named part of the thing. E.g. Bark is a meronym of tree.
5. Holonyms refer to a named part of the word that describes the thing. E.g. Tree is a holonym of bark. Therby, the oppisite of a meronym relation.
6. Pertainyms is a word, usually an adjective, which can be defined as "of or pertaining to"another word.
7. Troponyms refer to a manner relation of two verbs.

8. Hypernyms refer to words that the word refer to. E.g. Bird is a hypernym of duck, crow and seagull.
9. Hypernym is the opposite relation of a hyponyms which can be described as a type-of-relation. E.g. Duck, crow and seagull are all hyponyms of bird.
10. Co-hyponyms are words that share hypernyms such as red and blue which are co-hyponyms of color.

Words pairs with the same hypernyms or that is that they have a co-hyponym relation can be described as being sister terms. These words will naturally also be very similar, but may not be associated by humans. Examples of this is given in the following section about the two used data-sets for evalution.

Furthermore, there exist many other derivationally related forms which we will not use in our analysis?

1.5.2 WS353

WordSim3535 contains 353 word pairs, each associated with an average of 13 to 16 human judgements. In this case, both similarity and relatedness are annotated without any distinction on a scalar from 0 to 10. The Annotators were given the task to 'Assign a numerical similarity score between 0 and 10 (0 = words totally unrelated, 10 = words VERY closely related) ... when estimating similarity of antonyms, consider them "similar" i.e., belonging to the same domain or representing features of the same concept, not "dissimilar".' (Felix Hill, 2014).

As Felix Hill (2014) points out, this results in many dissimilar word pairs receiving a high similarity rating such as (coffee, cup) which are associated but not similar and word pairs as (telephone, communication) recieving a low similarity rating. Furthermore, they point out there was a low annotator agreement (ibid.).

The human annotators are though highly correlated in the data-set and it thereby give a real approximation of similarity in natural language, and therefore still widely used as a gold-standard data-set for evaluating similarity (Eneko Agirre, 2009).

1.5.3 Simlex

Simlex was made to correct some of the shortcomings for WS353. The data set is made of word pairs 999 which conceptually are similar or dissimilar with a clear distinction between relatedness and association in the rating by annotators.

To create a test of the ability of models to capture similarity as opposed to association. The dataset is based on the USF data-set for word pairs with high similarity and where clearly distinct word pairs have been randomly selected.

Furthermore, annotators were introduced to similarity by the well-understood idea of synonyms which was put in contrast to association: "In each case the participant was required to identify the most similar pair from a set of three options, all of which were associated, but only one of which was clearly similar (e.g. [bread, butter] [bread, toast] [stale, bread])"(Felix Hill, 2014). Furthermore, word pairs were not put in groups of context due to the fact that it introduces a high degree of subjectivity into the design process of the word pairs and the authours therefore argue that is a context-free-dataset of word pairs (ibid.).

1.5.4 Language model word similarity

To calculate the similarity score between the word vectors of the W2V model, we use cosine similarity.

The cosine similarity between two vectors **A** and **B** can be calculated by

$$\begin{aligned} \mathbf{A} \cdot \mathbf{B} &= \parallel \mathbf{A} \parallel \parallel \mathbf{B} \parallel \cos\theta \\ \cos(\theta) &= \frac{\mathbf{A} \cdot \mathbf{B}}{\parallel \mathbf{A} \parallel \cdot \parallel \mathbf{B} \parallel} \end{aligned} \tag{2}$$

Cosine similarity captures the angle of the word vectors which mean that a high similarity score of 1 is equivalent to a 0 degree difference between the two vectors and none-similarity score of 0 means a 90 degree difference between the vectors. This mean the magnitude does not influence the similarity between word vectors, but only how they are positioned in the vector space.

1.6 Statistical measures

To evaluate the language models and test our hypothesis we use different statistical measures.

1.6.1 Spearman's rank correlation coefficient

To compare the most similar words found by our model with the evaluation data-set, we use simple ranking in terms of the two numerical similarity scores. We furthermore calculate Spearman's ρ of the word rankings. In our case the words are ranked by cosine similarity and a 0 - 10 score which Spearman ρ can be used on as a statistical test to determine if there exists a relation between these two random variables.

Given ρ, Pearson correlation coefficient, X and Y, two random variables and $rank_X, rank_Y$, the ranking of the two random variables, we have the ranking

$$\rho_{rank_X,rank_Y} = \frac{cov(rank_X, rank_Y)}{\sigma_{rank_X}, \sigma_{rank_Y}}$$

Identical values are usually each assigned fractional ranks equal to the average of their positions in the ascending order of the values, which is equivalent to averaging over all possible permutations (Dodge, 2008).

1.6.2 The geometric and hyper-geometric distributions

The geometric distribution gives the probability that the first occurrence of success requires k independent trials, each with success probability p. If the probability of success on each trial is p, then the probability that the k'th trial (out of k trials) is the first success is

$$\Pr(X = k) = (1 - p)^{k-1}p$$

The geometric distribution is an appropriate model if the phenomenon being modelled is a sequence of independent trials with only two possible outcomes for each trial, designated success or failure, the probability of success, p, is the same for every trial (Dodge, 2008).

If these conditions are true, then the geometric random variable X is the count of the number of failures before the first success. The possible number of failures before the first success is 0, 1, 2, 3, and so on.

The expected value of the geometric distribution can be derived as follows.

$$\begin{aligned}
E(Y) &= \sum_{k=0}^{\infty}(1-p)^k p \cdot k \\
&= p\sum_{k=0}^{\infty}(1-p)^k k \\
&= p(1-p)\sum_{k=0}^{\infty}(1-p)^{k-1} \cdot k \\
&= p(1-p)\left[\frac{d}{dp}\left(-\sum_{k=0}^{\infty}(1-p)^k\right)\right] \\
&= p(1-p)\frac{d}{dp}\left(-\frac{1}{p}\right) = \frac{1-p}{p}
\end{aligned} \tag{3}$$

To test our hypothesis, we use a special formulation of the geometric distribution called the hyper-geometric distribution which is also closely related to the binomial distribution.

For the hyper-geometric distribution, the probability mass function can be formulated as

$$p(k, M, n, N) = \frac{\binom{n}{k}\binom{M-n}{N-k}}{\binom{M}{N}} k \in [\max(0, N - M + n), \min(n, N)]$$

where M is the population size, N is the sample size, n is the number of successes in the population, X is number of drawn successes and the binomial coefficients are defined as $\binom{n}{k} \equiv \frac{n!}{k!(n-k)!}$.

We find the variables in our Word2Vec model output which is specified in the method section.

Using a the survival function which is the inverse of the cumulative distribution function, we can then calculate the p value and determine if our findings are significant given a significance level α.

We assume that our samples are normal distributed and so that a standard normal distribution function Φ can describe our hyper-geometric distribution:

$$P(X \leq k) \approx \Phi\left(\frac{k - np}{\sqrt{np(1-p)}}\right)$$

2 Experimental setup

2.1 Description

We take a data driven inductive approach to finding a hypothesis that can explain the high performance of W2V models using the CBoW algorithm with a context window of size of 1 in relation to models with larger context windows, that is the phase transition from the smallest window size. We therefore want to recreate the setting where the phase transitions were found by Hershcovich et al. (2019) and Mckinney-Bock and Bedrick (2019). To reproduce the experiment by Hershcovich et al. (2019), we need to create a series of W2V models with increasingly large context windows and evaluate their performance on the Ws353 dataset. To find a hypothesis about the observed, we can then examine the word embedding of the model and the semantics of the most similar word pairs rated by the model. This will hopefully give us an hypothesis that can explain the phase transitions.

To test our hypothesis, we can use the SimLex dataset and as with Ws353 evaluate the model and look into the word embedding to see if the same phenomena occurs.

We furthermore make sure our results are significant by a hyper-geometric test.

In the following, I will go the practical setup and method of the experiment.

2.2 Word2Vec Implementation

In our experiment, we use two versions of the Word2Vec model to explore different Word2Vec methods, one from Amazon and one from Gensim.

BlazingText from Amazon SageMaker is based on FastText which is a more efficient version of the original Word2Vec model. BlazingText is furthere optimised by Amazon and effecitvely produce word embedding at Amazon Web Service's machine learning platform called Amazon Sagemaker (Amazon, 2020).

Gensim's Word2Vec implementation is based on the original C implementation by Mikolov et al. (2013a) of Word2Vec and is probably the closest we come to the original Word2Vec model when using Python libraries (Rehurek, 2019).

As described before, Word2vec models maps a corpus of words to high-quality distributed vectors. The resulting vector representation of a word is the word embedding. Using these embedding of a general corpus of words, we can find which words that are semantically similar in natural language put in sentences or other formats such as word pairs. BlazingText and Gensim.Word2Vec are highly optimized for multi-core CPU architectures which makes them even more efficient when given several million words. Furthermore, they are both able to provide word embedding from using either the Skip-gram (SGNS) or continuous bag-of-words (CBoW) algorithms (ibid.).

2.3 Dataset and pre-processing

Wikipedia is a excellent webpage for natural language in digital text format. Similarly to Hershcovich et al. (2019), we use fraction of all this text called text8 to create a corpus in our W2V models (Mahoney, 2011).

Pre-processing before Word2Vec-word embedding is very important when it comes to model performance. Nastaran Babanejad (2020) show in their analysis that model performance of word vectors models is improved by almost all pre-preprocessing techniques.

In text8 the following pre-processing steps have been taken: The wiki dump which has been cleaned to words of the 27 character English alphabet containing only the letters a-z and nonconsecutive spaces. The goal of the authors of the data-set was to only retain text that normally would be visible when displayed on a Wikipedia web page and read by a human. Therefore, only regular article text was retained, image captions are retained, but tables and links to foreign language versions were removed (Mahoney, 2011). Citations, footnotes, and markup were removed. Hypertext links were converted to ordinary text, retaining only the visible anchor text. Numbers are spelled out so ”20”becomes ”two zero”etc.. Upper case letters are converted to lower case. Finally, all sequences of characters not in the range a-z are converted to a single space. The Perl script which was used to clean the text can be found here (url). Last but no least all words a tokenized for the w2V model.

2.4 Size of Context Window

We define the windows size to go from 1 - 25 in our models using either CBoW or Skip-gram. This give us a total of 50 different models.

The windows size determines what we feed into our model and model complexity due to the vectors direct relationsship with the size of the input word vectors.

2.5 Word2Vec parameters and execution

Beyond window size, we all need to decide on the dimension of the feature vectors and the minimum threshold of the words in the model.

Feature Vector Dimension is set to 500 and minimum frequency for words are set to 500 to get a similar results to Mckinney-Bock and Bedrick (2019).

The models are trained and evaluated in AWS SageMaker's cloud environment (url) and on a personal multi-core computer. Models, data and the used code written in Python can be found on GitHub (GitHub-url).

Testing the BlazingText models before switching to the Gensim implemenatation some parameters were set similarly and in some instances lower to re-create the results of (Hershcovich et al., 2019) and Mckinney-Bock and Bedrick (2019).

2.6 Model evaluation

Each model is evaluated on WordSim353 for similarity and relatedness which is a data set of word which is human annotated (Gabrilovich, 2009). The same data set and approach is used in Hershcovich et al. (2019) which we wish to compare with our own results.

In our evaluation as well of testing of the models, we can only use input and output to find out how the model performs.

Using Gensim we can calculate Spearman's α between similarity scores of our models and the human annotators. This is done using the WS353 data-set and SimLex data-set which were downloaded from their respective websites.

2.7 Exploratory data analysis of semantic relations

Using WordNet, we can find all the semantic relations of the word pairs of WS353 and see which relations are most similar by using cosine similarity between the word vectors of the model. WordNet is a database of synsets developed by Princeton and Stanford University which has made public API which we make use of to retrieve all semantic relations (Wordnet Homepage link).

In our exploratory data analysis, all the word pair relation are investigated by retrieving all synsets from the WS353 word pairs evaluated by the model.This make us able to decide all the relations between the word pairs by making comparison of all lemmas, hypernyms, hyponyms, antonyms etc.

When using the WordNet API, we use a py script implementing a similar method to Hershcovich et al. (2019) with the word pairs (link to script). The original script by Hershcovich et al. (2019) can be found on GitHub (GitHub link).

2.8 Significance test

We use Sci Py's implementation of the hyper-geometric test and its survival function to calculate p-values (SciPy Documentation). In our hyper-geometric test, we want to test the top 10% of distribution which is a standard quantity. To test see if the occurrence of semantic relations are significant, we set X to be the occurrence of a relations between top-30 word pairs, M the total number of pairs, n the total number of relation occurrences and N to be count of word pairs in the top-30 for our first test. The original data-set of WS353 for similarity is of 203 word pairs which mean that we use a little more than 10 % of the original data set.

We find the counts by comparing each data point with each relation and then perform the hyper geometric test. The script can be found on Github (link to script).

When testing our hypothesis, we use a standard significance level $\alpha = 0.05$ 0.05, and investigate our hypothesis for each model by $p < \alpha$, that is we determine if our findings are significant. Because we are looking for different relations for each W2V model, we also need to be aware of Bonferroni correction of the significance level, α/m, to compensate for the many relations which should make us avoid randomness in our results.

Tabel 1: Descriptive stats of evaluation data-sets

	synonyms	identity	hypernyms	hyponyms	member_holonyms	member_meronyms	part_holonyms	part_meronyms	substance_holonyms	substance_meronyms	antonyms	pertainyms	derivationally_related_forms	co_hyponyms	total word pairs
SimLex	21	20	17	18	0	0	4	1	1	0	14	0	0	26	267
WS353	0	2	3	2	0	1	0	1	0	0	1	0	0	9	85

3 Preliminary results

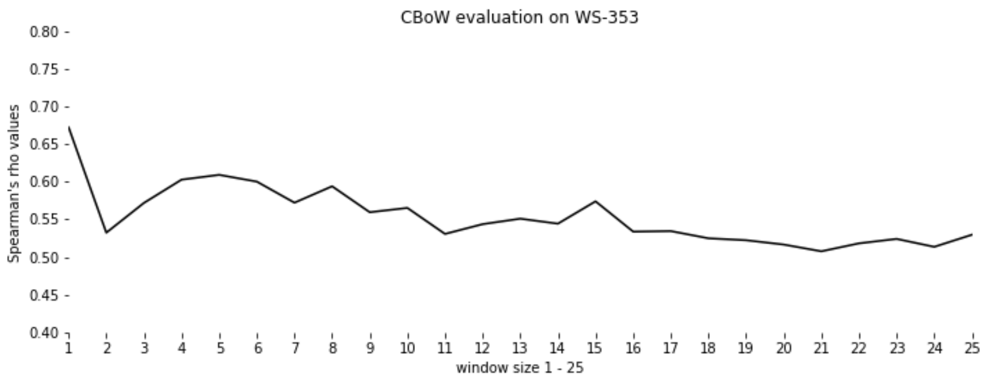
3.1 Descriptive analysis of evaluation data-sets

From the corpus from WikiPedia which we train our W2V models on, we get a reduced datasets due to some missing word vectors in our models. On these reduced datasets, according to WordNet we find the relation between word pairs. We check for alle posible sementic relations in the WordNet database: synonyms, hypernyms, hyponyms, member holonyms, member meronyms, part holonyms, part meronyms. substance holonyms, substance meronyms, antonyms, pertainyms and derivationally related forms. This yields the statistics of the two data-sets seen in tabel 1. We see that there is a different distribution of word relations in the data-sets and that several relations do not exist. We will not test for these non-ocurring relations in the following hypergeom tests.

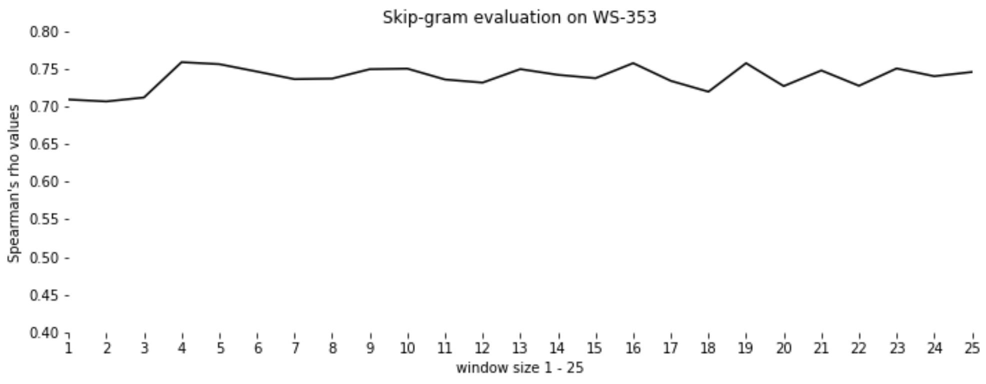
3.2 Evaluation dataset WS353

Depending on the algorithm we use, we get different results when evaluating on WS-353. Using different parameters with BlazingText and a increasing window size, we are not able to reproduce the findings of Hershcovich et al. (2019) and Mckinney-Bock and Bedrick (2019).

When using the Word2Vec implementation found in Gensim by Radimrehurek, which is based on the original implementation by Mikolov et al. (2013a) in C, we are though able to reproduce the results.



As found in Hershcovich et al. (2019) the difference between window size 1 and 2 is not evident when using the Skip-gram algorithm.



We from here on concentrate on the W2V models made with the CBoW algortihm in the Gensim implementation because we want to investigate the findings of Hershcovich et al. (2019) and Mckinney-Bock and Bedrick (2019).

3.3 Exploratory data analysis of words pair relations

The top-10 ranked word pairs by the CBoW model with a context window size 1 are shown in table 1.

We see that the top-5 word pairs have a hyponym relation and several pairs have hypernym relation. This indicate that the wordpairs are each others hyponyms and reversely each others hypernyms and thereby that they are sister terms.

When looking at the relation of top-10 word pairs across models, we see a similar distribution of hyponyms. This could indicate that all models rate sister terms higher.

Tabel 2: Top-10 word-pairs and their relation of W2V Model with a context window size of 1

football	basketball	co_hyponyms
physics	chemistry	co_hyponyms
television	radio	co_hyponyms
rock	jazz	co_hyponyms
liquid	water	hyponyms
man	woman	antonyms
skin	eye	None
glass	metal	None
type	kind	hypernyms
planet	moon	None
king	queen	identity
food	fruit	None
money	cash	co_hyponyms
planet	sun	co_member_meronyms
psychology	science	hypernyms

Tabel 3: Top-10 word pairs WS353

W1	**W2**	**W5**	**W10**	**W15**	**W25**	**WS353**
(football|basketball)	(football|basketball)	(physics|chemistry)	(physics|chemistry)	(physics|chemistry)	(physics|chemistry)	(money|cash)
(physics|chemistry)	(physics|chemistry)	(television|radio)	(television|radio)	(seven|series)	(football|basketball)	(money|currency)
(television|radio)	(television|radio)	(football|basketball)	(football|basketball)	(television|radio)	(television|radio)	(type|kind)
(rock|jazz)	(liquid|water)	(rock|jazz)	(rock|jazz)	(football|basketball)	(seven|series)	(king|queen)
(liquid|water)	(type|kind)	(type|kind)	(seven|series)	(liquid|water)	(liquid|water)	(planet|star)
(man|woman)	(rock|jazz)	(liquid|water)	(liquid|water)	(man|woman)	(five|month)	(money|dollar)
(skin|eye)	(glass|metal)	(seven|series)	(man|woman)	(rock|jazz)	(computer|news)	(man|woman)
(glass|metal)	(man|woman)	(king|queen)	(king|queen)	(psychology|science)	(rock|jazz)	(planet|moon)
(type|kind)	(planet|moon)	(psychology|science)	(type|kind)	(king|queen)	(man|woman)	(planet|sun)
(planet|moon)	(journal|association)	(man|woman)	(glass|metal)	(type|kind)	(king|queen)	(liquid|water)

Tabel 4: Top-10 word-pair relation

W1	**W2**	**W5**	**W10**	**W15**	**W25**	WS353
co_hyponyms	co_hyponyms	co_hyponyms	co_hyponyms	co_hyponyms	co_hyponyms	co_hyponyms
co_hyponyms	co_hyponyms	co_hyponyms	co_hyponyms	None	co_hyponyms	hypernyms
co_hyponyms	co_hyponyms	co_hyponyms	co_hyponyms	co_hyponyms	co_hyponyms	hypernyms
co_hyponyms	hyponyms	co_hyponyms	co_hyponyms	co_hyponyms	None	identity
hyponyms	hypernyms	hypernyms	None	hyponyms	hyponyms	co_hyponyms
antonyms	co_hyponyms	hyponyms	hyponyms	antonyms	None	None
None	None	None	antonyms	co_hyponyms	None	antonyms
None	antonyms	identity	identity	hypernyms	co_hyponyms	None
hypernyms	None	hypernyms	hypernyms	identity	antonyms	co_member_meronyms
None	None	antonyms	None	hypernyms	identity	hyponyms

Tabel 5: Count of relations in top (P-values) in Ws353. ** indicate p <0.05 and *** indicate p <0.01. Decimals are rounded without loss of mutual precision.

Model	**identity**	**hypernyms**	**hyponyms**	**member_meronyms**	**part_meronyms**	**antonyms**	**co_hyponyms**
W1	2 (0.122)	3 (0.041)**	1 (0.584)	1 (0.353)	0 (1.000)	1 (0.353)	6 (0.046)**
W2	2 (0.122)	2 (0.283)	1 (0.584)	1 (0.353)	0 (1.000)	1 (0.353)	6 (0.046)**
W3	1 (0.584)	2 (0.283)	1 (0.584)	1 (0.353)	0 (1.000)	1 (0.353)	6 (0.046)**
W4	1 (0.584)	3 (0.041)**	1 (0.584)	1 (0.353)	0 (1.000)	1 (0.353)	6 (0.046)**
W5	1 (0.584)	2 (0.283)	1 (0.584)	1 (0.353)	1 (0.353)	1 (0.353)	6 (0.046)**
W6	1 (0.584)	3 (0.041)**	1 (0.584)	1 (0.353)	0 (1.000)	1 (0.353)	5 (0.164)
W7	1 (0.584)	3 (0.041)**	1 (0.584)	1 (0.353)	1 (0.353)	1 (0.353)	6 (0.046)**
W8	1 (0.584)	2 (0.283)	1 (0.584)	0 (1.000)	1 (0.353)	1 (0.353)	7 (0.008)***
W9	1 (0.584)	2 (0.283)	1 (0.584)	0 (1.000)	0 (1.000)	1 (0.353)	6 (0.046)**
W10	1 (0.584)	3 (0.041)**	1 (0.584)	0 (1.000)	1 (0.353)	1 (0.353)	6 (0.046)**
W11	1 (0.584)	2 (0.283)	1 (0.584)	0 (1.000)	1 (0.353)	1 (0.353)	7 (0.008)***
W12	1 (0.584)	3 (0.041)**	1 (0.584)	0 (1.000)	0 (1.000)	1 (0.353)	7 (0.008)***
W13	1 (0.584)	2 (0.283)	1 (0.584)	1 (0.353)	1 (0.353)	1 (0.353)	7 (0.008)***
W14	1 (0.584)	2 (0.283)	1 (0.584)	0 (1.000)	0 (1.000)	1 (0.353)	7 (0.008)***
W15	1 (0.584)	3 (0.041)**	1 (0.584)	0 (1.000)	1 (0.353)	1 (0.353)	7 (0.008)***
W16	1 (0.584)	3 (0.041)**	1 (0.584)	1 (0.353)	1 (0.353)	1 (0.353)	6 (0.046)**
W17	1 (0.584)	2 (0.283)	1 (0.584)	0 (1.000)	1 (0.353)	1 (0.353)	5 (0.164)
W18	1 (0.584)	2 (0.283)	1 (0.584)	0 (1.000)	1 (0.353)	1 (0.353)	6 (0.046)**
W19	1 (0.584)	3 (0.041)**	1 (0.584)	0 (1.000)	1 (0.353)	1 (0.353)	7 (0.008)***
W20	1 (0.584)	2 (0.283)	1 (0.584)	0 (1.000)	1 (0.353)	1 (0.353)	7 (0.008)***
W21	1 (0.584)	2 (0.283)	1 (0.584)	1 (0.353)	0 (1.000)	1 (0.353)	7 (0.008)***
W22	1 (0.584)	2 (0.283)	1 (0.584)	0 (1.000)	1 (0.353)	1 (0.353)	6 (0.046)**
W23	1 (0.584)	2 (0.283)	1 (0.584)	0 (1.000)	1 (0.353)	1 (0.353)	7 (0.008)***
W24	1 (0.584)	2 (0.283)	1 (0.584)	0 (1.000)	1 (0.353)	1 (0.353)	6 (0.046)**
W25	1 (0.584)	2 (0.283)	1 (0.584)	1 (0.353)	1 (0.353)	1 (0.353)	0 (1.000)
W353	2 (0.122)	3 (0.041)**	1 (0.584)	1 (0.353)	1 (0.353)	1 (0.353)	6 (0.046)**

3.4 Hyper-geom test of word pair relations

To check if the word relations are significant, we calculate a p value for all relations in each model using the parameters described in the last section (table 5). We see that only the hyponym relation seem to be significant for the word pairs of the WS353 data-set which indicate that this relation is very important for the way the model measures similarity.

We see that this is true for models with a context windows of all sizes except 6, 17 and 25. which indicate that most models rate hyponym word pairs very similar across models with different window sizes. The most significant cases are for models with a context window size of 1 and 2.

3.5 Hypothesis

As shown in our preliminary results, we were able to reproduce the fact that CBoW models with a context window size of 1 models are better to find similarity between words than models with larger context windows.

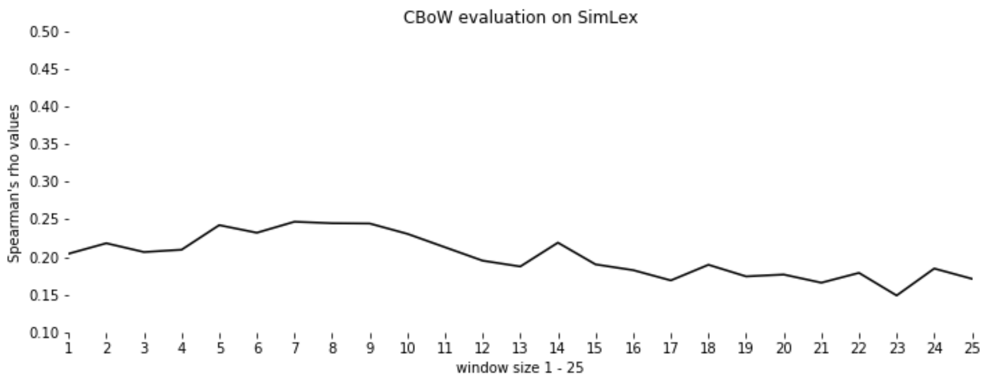
When looking into into the word embedding and the semantic relations of the top-word pairs, we furthermore found that the words which were rated with a high similarity across models have a hyponym relation and thereby share many of the same hypernyms which indicate they could be sister terms. This finding was especially significant for models with a window size of 1 and 2 which mean that this could explain the phase transition.

Hyp_0: Word2Vec models with narrow context windows find that the most similar word pairs are sister terms.
Hyp_1: Word2Vec models with narrow context windows find that the most similar word pairs are not sister terms.
In following section, we will test this hypothesis on our test dataset, SimLex.

4 Test results

4.1 Evaluation dataset SimLex

Running the same evaluation of our W2V model with the CBoW algorithm on SimLex as on WS353, we get a very different Spearmans ρ.



This could mean that, we will not see the same phase transition phenomena on the

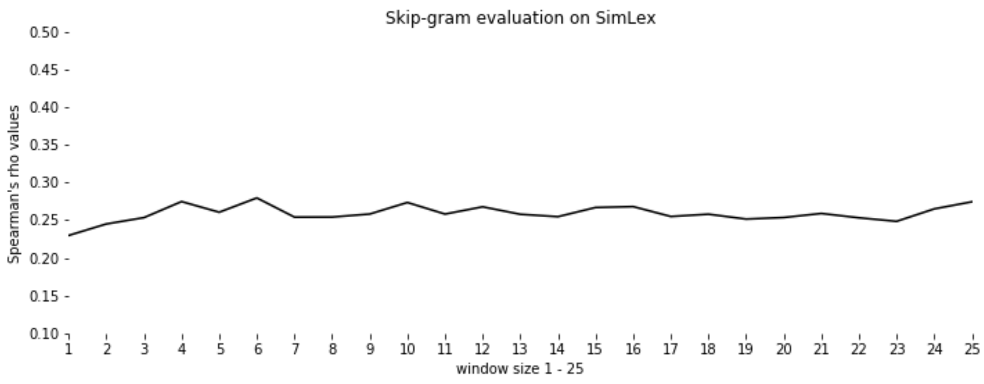
Tabel 6: Top-10 word pairs Simlex

W1	W2	W5	W10	W15	W25	SimLex
(actress|actor)	(actress|actor)	(actress|actor)	(actress|actor)	(actress|actor)	(actress|actor)	(large|big)
(molecule|atom)	(composer|writer)	(actor|singer)	(actor|singer)	(actor|singer)	(actor|singer)	(large|huge)
(winter|summer)	(actor|singer)	(composer|writer)	(son|father)	(south|north)	(south|north)	(area|region)
(south|north)	(south|north)	(south|north)	(composer|writer)	(son|father)	(son|father)	(simple|easy)
(actor|singer)	(basketball|baseball)	(son|father)	(south|north)	(composer|writer)	(composer|writer)	(boundary|border)
(composer|writer)	(winter|summer)	(winter|summer)	(movie|film)	(area|region)	(brother|son)	(business|company)
(dog|cat)	(son|father)	(movie|film)	(area|region)	(movie|film)	(father|brother)	(corporation|business)
(wife|husband)	(brother|son)	(mother|wife)	(mother|wife)	(brother|son)	(area|region)	(essential|necessary)
(basketball|baseball)	(mother|wife)	(brother|son)	(basketball|baseball)	(mother|wife)	(winter|summer)	(task|job)
(large|huge)	(understand|know)	(basketball|baseball)	(brother|son)	(north|west)	(movie|film)	(movie|film)

Tabel 7: Top-10 word pair relations SimLex

W1	W2	W5	W10	W15	W25	SimLex
hypernyms	hypernyms	hypernyms	hypernyms	hypernyms	hypernyms	identity
identity	None	None	None	None	None	similar_tos
co_hyponyms	None	None	None	antonyms	antonyms	identity
antonyms	antonyms	antonyms	None	None	None	also_sees
None	co_hyponyms	None	antonyms	None	None	synonyms
None	co_hyponyms	co_hyponyms	identity	identity	None	None
None	None	identity	identity	identity	None	indirect_hypernyms
antonyms	None	None	None		None	identity
identity						
co_hyponyms	None	None	co_hyponyms	None	co_hyponyms	identity
similar_tos	hyponyms	co_hyponyms	None	indirect_part_holonyms	identity	identity

SimLex dataset as with the Ws353.



Using Skip-gram, we as before do not see the phase transition.

4.2 Hyper-geom test of word pair relations

We see in the relations of the top 10 pairs of SimLex in table 6 and 7 that the most similar word pair relation are very different compared to those found in Ws353 (table 3, 4).

Tabel 8: Counts of word pair relations (P-values) in SimLex. * indicate $p < 0.1$. ** indicate $p < 0.05$ and *** indicate $p < 0.01$. Rounding is done without loss of precision.

Model	**synonyms**	**identity**	**hypernyms**	**hyponyms**	**part_holonyms**	**substance_holonyms**	**antonyms**	**co_hyponyms**
W1	9 (0.244)	9 (0.192)	8 (0.173)	7 (0.403)	0 (1.000)	0 (1.000)	11 (0.001)***	12 (0.117)
W2	11 (0.053)	9 (0.192)	7 (0.334)	6 (0.606)	1 (0.809)	0 (1.000)	10 (0.004)**	11 (0.222)
W3	12 (0.019)**	10 (0.090)*	9 (0.074)**	4 (0.912)	0 (1.000)	0 (1.000)	8 (0.056)*	13 (0.054)*
W4	7 (0.601)	9 (0.192)	10 (0.025)**	5 (0.788)	0 (1.000)	0 (1.000)	9 (0.016)**	13 (0.054)*
W5	8 (0.412)	9 (0.192)	7 (0.334)	5 (0.788)	1 (0.809)	1 (0.337)	9 (0.016)**	11 (0.222)
W6	8 (0.412)	10 (0.090)*	6 (0.538)	5 (0.788)	0 (1.000)	1 (0.337)	9 (0.016)**	10 (0.368)
W7	5 (0.895)	9 (0.192)	7 (0.334)	6 (0.606)	1 (0.809)	1 (0.337)	9 (0.016)**	12 (0.117)
W8	7 (0.601)	10 (0.090)*	6 (0.538)	7 (0.403)	1 (0.809)	1 (0.337)	9 (0.016)**	12 (0.117)
W9	6 (0.773)	9 (0.192)	6 (0.538)	8 (0.227)	1 (0.809)	1 (0.337)	8 (0.056)*	12 (0.117)
W10	5 (0.895)	10 (0.090)*	6 (0.538)	7 (0.403)	1 (0.809)	1 (0.337)	8 (0.056)*	10 (0.368)
W11	4 (0.963)	8 (0.348)	6 (0.538)	6 (0.606)	1 (0.809)	1 (0.337)	8 (0.056)*	11 (0.222)
W12	5 (0.895)	10 (0.090)*	6 (0.538)	8 (0.227)	1 (0.809)	1 (0.337)	8 (0.056)*	10 (0.368)
W13	6 (0.773)	9 (0.192)	7 (0.334)	7 (0.403)	0 (1.000)	1 (0.337)	8 (0.056)*	12 (0.117)
W14	4 (0.963)	9 (0.192)	7 (0.334)	7 (0.403)	0 (1.000)	1 (0.337)	8 (0.056)*	12 (0.117)
W15	5 (0.895)	9 (0.192)	7 (0.334)	7 (0.403)	0 (1.000)	1 (0.337)	8 (0.056)*	11 (0.222)
W16	5 (0.895)	7 (0.538)	5 (0.737)	7 (0.403)	1 (0.809)	1 (0.337)	8 (0.056)*	11 (0.222)
W17	5 (0.895)	10 (0.090)*	7 (0.334)	7 (0.403)	1 (0.809)	1 (0.337)	8 (0.056)*	11 (0.222)
W18	5 (0.895)	8 (0.348)	7 (0.334)	8 (0.227)	1 (0.809)	1 (0.337)	7 (0.151)	11 (0.222)
W19	4 (0.963)	9 (0.192)	6 (0.538)	8 (0.227)	0 (1.000)	1 (0.337)	8 (0.056)*	11 (0.222)
W20	5 (0.895)	9 (0.192)	6 (0.538)	5 (0.788)	1 (0.809)	1 (0.337)	9 (0.016)**	12 (0.117)
W21	5 (0.895)	9 (0.192)	5 (0.737)	6 (0.606)	1 (0.809)	1 (0.337)	8 (0.056)*	13 (0.054)*
W22	5 (0.895)	9 (0.192)	6 (0.538)	6 (0.606)	0 (1.000)	1 (0.337)	8 (0.056)*	11 (0.222)
W23	6 (0.773)	8 (0.348)	8 (0.173)	5 (0.788)	1 (0.809)	1 (0.337)	8 (0.056)*	12 (0.117)
W24	4 (0.963)	10 (0.090)*	7 (0.334)	7 (0.403)	1 (0.809)	1 (0.337)	8 (0.056)*	11 (0.222)
W25	4 (0.963)	8 (0.348)	6 (0.538)	6 (0.606)	0 (1.000)	1 (0.337)	8 (0.056)*	13 (0.054)*
SimLex	9 (0.244)	15 (0.000)***	9 (0.074)*	10 (0.041)**	0 (1.000)	1 (0.337)	0 (1.000)	5 (0.973)

When make a hyper-geom test, we also get a very different result. In this test, we use the top-90 pairs, X = 90, because SimLex has a size $\approx 3\times$ of WS353. Similar results are found using a X = 30 (table 8). As before non-occurring relations are not used in the hyper-geometric test. We see that the only significant relation for the W2V model with a context window 1 and 2 is antonyms. We though see a high count of co-hyponyms in models with a window context of 1 to 4.

5 Discussion

5.1 Results

The results obtained from our experiment gives different results. In our preliminary results using WS353, we see on one hand that w2V models with narrow context windows significantly rates word pairs with a co-hyponym relation as most similar, but that the models finds have a similar amount of co-hyponyms across all context window sizes (table 5). Running the same experiment on the test data-set, SimLex, we though find that only a model with a context window of 1 is able to capture an almost significant amount of co-hyponyms (table 8), but that the there is only significant occurrences in models with a context size of 3,4 and 25. Using Bonferroni correction (m = 7, each tested relation) which lead to an $\alpha = 0.05/9 \approx 0.01$ we can more surely deny the significance of co-hyponyms in our test. This means that we cannot confirm our null hypothesis using a conservative estimate. The results on SimLex though by the count of co-hyponyms in the top indicate that models with narrow context windows rate word pairs with a co-hyponym relation as very similar. As the context window increase, they may change around and new pairs come in the top-30, but it is clear that the co-hyponyms stay in the top across all window sizes.

We furthermore see a high number of antonyms as part of the most similar word pairs in SimLex. This mean that the model learn antonyms to be very similar which can be explained by the context which they appear as they as hypernyms and co-hyponyms appear in the same context. This is a classic problem for word space models which cannot distinct these relations because of the context (Silke Scheible and Springorum, 2013). This computational understanding does though not correspond with a human understanding of word similarity which is reflected in the low evaluation score of the model on SimLex, and the antonym relation's non-significance in SimLex's most similar word pairs (table 8.

5.2 Differences between WS353 and SimLex

As shown in the preliminary results (table 1) there is a considerable difference in the distribution of word relations and the same goes for which word relation's occurrence are the most similar (table 5 and 8). This shows that there is a big difference in how this data-sets have been made and annotated which the authors of SimLex have also described as actual similarity in SimLex vs. conceptual similarity in WS353 (Felix Hill,

2014).
Annotators in SimLex used context-free-rating which make might make it very difficult for a model such as a Word2Vec using the CBoW algorithm and explain the low evaluation score. Reversely, very context-dependent and subjective rating as in Ws353 make it easier for such a model. In WS353 annotators were not instructed to differ between words with similar association and meaning. This might explain why there are so few synonyms in the top of Ws353, but so many hypernyms and co-hyponyms in comparison. It also explain why there are so many identity and synonyms relation in the top and other relations such as holonyms and antonyms in the SimLex data-set.

The considerably higher evaluation score on WS353 might mean that W2V models are better at predicting conceptual similarity. It could also mean that models with narrow context windows are especially good at predicting the similarity of sister-terms as there are many sister-terms in WS353 and less of other types of relations (table 1).

It not for this study to determine what evaluation data-set is more correct, we can only say that W2V models perform considerably better when rating similarity on the WS353 data-set than SimLex which was also found by Hershcovich et al. (2019) and Mckinney-Bock and Bedrick (2019).

5.3 Technical implications for Word2Vec models using CBoW

The CBoW architecture uses the context of around the word to predict the next word. This means that when used in a W2V model with a narrow context window, it has a large influence on the word vectors. It most likely be very sensitive to word pairs with a co-hyponym relations due to the fact that these word will often appear in the same context.

As the window size increases, we see that new word pairs in WS353 and SimLex with other relations becomes the most similar pairs. In SimLex, the identity relations which are the highest rated by the annotators are better captured with bigger windows sizes. (table 8: W3, W6, W10, W17, W24). It is though still antonyms, hypernyms and co-hyponyms which are the highest rated relations by the model. This could mean that CBoW models because of their context based prediction best are better able to capture these relations in language.

As shown in the preliminary results, the W2V models made with Skip-gram algorithm did not show the same kind of behaviour. It though score higher than CBoW models when being evaluated on both data-sets. Further work might explore this.

Furthermore, it would be interesting to train the W2V models on a different data-set of natural language to see if the same relations occur. It would also have been interesting to test on all word pairs of Ws353 and SimLex and not just a subset. The latter could be done by expanding the vocabulary.

Using other evaluation data-sets or own annotators, one could also re-create the experiment on other Germanic languages with similar semantic relations such as Danish and German. Similar languages with same semantic relations should give the same results as words will be placed in a similar context in most cases.

5.4 Further work

Using the results obtained from this project, it would be interesting to see how W2V models trained to perform tasks such as classifying word relations or leveraging co-hypernyms to perform other tasks. This project has mainly been using unsupervised learning and using supervised might lead to similar or new results from the idea that W2V models using the CBoW algorithm and narrow are good at tasks involving certain types of word relations such as co-hyponyms. A further study might include training on tasks and testing in a similar way to Nastaran Babanejad (2020) which involved testing supervised W2V models ability in sentiment analysis, emotion classification and sarcasm detection with different pre-processing techniques.

W2V models are to this day still widely used in search engines and other applications using text prediction such as real-time type assist. Adjusting the context window to a smaller size when prediction in certain context might results in gains in efficiency and correctness. Furthermore, this project can help explain why certain word prediction models using W2V give some odd results which semantically are not correct or unlikely in natural language. E.g. why antonyms and sister-terms may be overrated in prediction. This would also be interesting for further studies to explore.

6 Appendix

7 Bibliography

Litteratur

Amazon. Blazing documentation, 2020. URL `https://docs.aws.amazon.com/sagemaker/latest/dg/blazingtext.html`.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/P14-1023. URL `https://www.aclweb.org/anthology/P14-1023`.

Yadolah. Dodge. *The Concise Encyclopedia of Statistics*. Springer reference. Springer New York, New York, NY, 1st. ed. edition, 2008. ISBN 0-387-32833-5.

Keith Hall Jana Kravalova Marius Pasca Aitor Soroa Eneko Agirre, Enrique Alfonseca. A study on similarity and relatedness using distributional and wordnet-based approaches. pages 19–27, 2009.

Anna Korhonen Felix Hill, Roi Reichart. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. 2014. doi: arXiv:1408.3456v1.

E. Gabrilovich. Wordsim353 - similarity and relatedness, 2009. URL `http://alfonseca.org/eng/research/wordsim353.html`.

Zellig S. Harris. Distributional structure. *<i>WORD</i>*, 10(2-3):146–162, 1954. doi: 10.1080/00437956.1954.11659520. URL `https://doi.org/10.1080/00437956.1954.11659520`.

Daniel Hershcovich, Assaf Toledo, Alon Halfon, and Noam Slonim. Syntactic interchangeability in word embedding models. pages 70–76, 2019. URL `https://www.aclweb.org/anthology/W19-2009`.

Matt Mahoney. About the test data, 9 2011. URL `http://mattmahoney.net/dc/textdata.html`. text8 can found on http://mattmahoney.net/dc/text8.zip.

Katy Mckinney-Bock and Steven Bedrick. Classification of semantic paraphasias: Optimization of a word embedding model. *Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for*, 2019. doi: 10.18653/v1/w19-2007.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv.org*, 2013a. URL `http://search.proquest.com/docview/2086087644/`.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *arXiv.org*, 2013b. URL `http://search.proquest.com/docview/2085905727/`.

Aijun An Manos Papagelis Nastaran Babanejad, Ameeta Agrawal. A comprehensive analysis of preprocessing for word representation learning in affective tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, page 5799–5810, 2020. URL `https://www.aclweb.org/anthology/2020.acl-main.514.pdf`.

Radim Rehurek. Gensim.word2vec documentation, 2019. URL `https://radimrehurek.com/gensim/models/word2vec.html`.

Sabine Schulte im Walde Silke Scheible and Sylvia Springorum. Uncovering distributional differences between synonyms and antonyms in a word space model. page 489–497, 2013. URL `https://www.aclweb.org/anthology/I13-1056.pdf`.