KØBENHAVNS UNIVERSITET

Bachelor thesis

Thor Steen Larsen <mvj665@alumni.ku.dk>

Phase Transitions In Word Embeddings

Department of Computer Science

Supervisor: Daniel Hershcovich <dh@di.ku.dk>

11. august 2020

Abstract

Word embedding models such as Word2Vec can model natural language by assigning vectors to words on the background of training data in the form of text (Mikolov et al., 2013a). In all algorithms to calculate such vectors, the size of the context of words is central. Given how similar the words are and how similar their context, real numerical values will be given to describe their relation to all words in the training data. Hershcovich et al. (2019) and Mckinney-Bock and Bedrick (2019) found that word embedding models anecdotally using a single word from the context to predict the middle word can achieve better results than models trained with more words on the data-set WS353 which evaluate model performance by human judgement of similarity. Using two words from the context yields in far worse results and results only improve gradually by using more words. To examine this phase transition, we look into relations between the most similar word pairs found by the word embedding. By examining the data generated by the evaluating on the WS353 data-set, we inductively come up with a hypothesis that the model's rating of sister-terms can explain the better performance on the WS353 data-set. We test our hypothesis on the SimLex999 data-set and find that we cannot confirm our hypothesis, but that the word embedding model is highly sensitive to sister-terms, synonyms and antonyms which in many cases share context. It indicates that the model is sensitive to these semantic relations due to their similar context, but in a way that does not always correspond with human judgement.

Indhold

1 Introduction

In this project, we will explore the handling of words in natural word processing. Word embedding model consists of words from a vocabulary mapped to vectors of real numbers depending on their context. This idea comes from the distributional hypothesis that linguistic items such as words with similar distributions have similar meanings. In this project, we will explore the handling of words by Word2Vec models. Hopefully, we will learn more about how word embedding models efficiently process natural language and capture word similarity.

Word2Vec is a network-based, predictive language model which can either predict the middle word of a sentence given the context around the word or predict the context around the word given the middle word depending on the used algorithm. Predictive models such as Word2Vec need a corpus and a context window to build word embedding. The training data provide the first while the other is a parameter which decides how many words the model uses at the time to predict the middle word.

By creating word embedding using different amounts of words to predict the middle word, Hershcovich et al. (2019) and Mckinney-Bock and Bedrick (2019) have shown there is a significant performance score between using one word and using several words when evaluating using human judgement. It means that it could be that the most simple word embedding has a strong prediction power. Potentially, if we understand in which context this works, we can in some cases greatly decrease model training time for Word2Vec and similar models and better understand how well word embedding model semantic relations.

This project is based on an inductive, data-driven approach, and the report is structured as follows. We start by describing language models, the Word2Vec architecture, the used evaluation data-sets and the statistical measures used to evaluate models and test results. We then go through the experimental set-up where the data, parameters and methods used will be explained. After this, we go through the preliminary results, which lead to a hypothesis that can explain why the simple word embedding of Word2Vec performs well in evaluation. We then test this on another evaluation data-set and end with a discussion of the results and their implications.

2 Theory

2.1 Natural Language Models

In natural language, we, in general, have a stream of words in sentences where a certain number of words fall into a certain bin. By calculating the probability that a word falls into a bin, we can from a frequentist point of view begin to understand word relations by their distribution 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 the distributional hypothesis, that is, words are used and occur in the same contexts tend to infer similar meanings (Harris, 1954).

Distributional semantic modelling in vector spaces makes language model work efficiently on computers and make us able to use architectures such as Word2Vec which is a predictive model, but not a language model mainly because it predicts the middle word instead of the next word[1].

A traditional language model could e.g. be a count-based model. This is a model which use word n-gram, a sequence of words in a sentence $w_1, \dots w_n$ where 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. Here we assume the Markov property that the probability of one word affects the probability of the next word.

The reason why most have turned to predictive models is that they have significantly better results in deciphering semantic context than count-based methods since merely counting the amount of word is not enough learning for the model (Baroni et al., 2014).

In predictive models such as Word2Vec, we can learn word vectors in a shallow, two-layer neural network which are trained to reconstruct a linguistic context of words in

[1]Note that in the literature and other sources Word2Vec is also called a method or a program. In this project, we simplify and call it a model and in some cases refer to the architecture of the model and its word embedding

the form of 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. It is thereby using the isomorphic properties of the learned vector space to predict the structure of language. Depending on the algorithm used, CBoW or Skip-gram, word vectors are created and positioned in the vector space. In our case, we want to experiment with context windows. In the following paragraph, we will go deeper into the Word2Vec algorithm and context windows.

2.2 Continuous Bag of Words algorithm (CBoW)

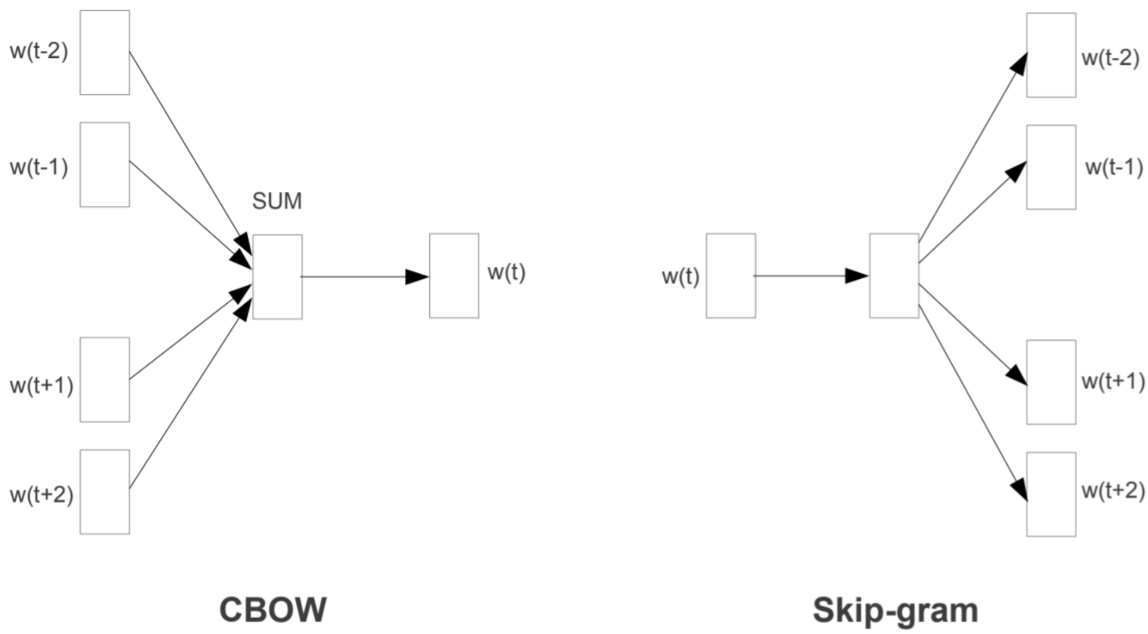


Fig 1: CBoW and Skip-gram architecture (Mikolov et al., 2013a)

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 $w \ldots w_{\text{window size}}$ want to predict the middle word. The basic W2V architecture using CBoW 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 **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 **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 **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 **W**'.The output y is calculated by passing the input $\mathbf{u}_j$ through the soft-max function

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

When we have learned the weight matrices **W** and **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 or next input word. The next step is to derive the update equation for the hidden-output layer weights **W**', then derive the weights for the input-hidden layer weights **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 gives us a bag-of-words type model where the order of words in the history do not influence the projection to the output layer and all the word vectors are averaged (Mikolov et al., 2013a). This computation which happens in the hidden layer together with the objective function is the main difference to the Skip-gram algorithm.

2.3 Skip-gram algorithm (SGNS)

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

The basic W2V architecture using Skip-gram 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.

Alternatively, 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 descent as in the architecture of CBoW.

In addition to the general soft-max function, Mikolov et al. (2013b) implements a hierarchical soft-max 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. Furthermore, they implement negative sampling to only modify a small part of the weights during training, instead of all weights for each training sample, which can be used to decrease the computational cost further (Mikolov et al., 2013b). We will not go into further into these two additions and instead focus on the matter at hand, the context window size.

2.4 Context window size

One of the main parameters of distributional semantic models is in terms of the context window. The window size parameter in Word2Vec or size of the context window defines the words we use to calculate the probability of each word. It thereby has a critical role in determining how the model predicts the middle word.

If we have a sentence: "The capital of Denmark is Copenhagen and its second largest city is Aarhus". A W2V network with a window size of one is going to learn that there is a 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 of the middle word and the context to learn this while a model with the CBoW algorithm would take the whole context to learn this.

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 a significant difference between similarity scores of CBoW models with a window size of 1 and 2.

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

2.5 Semantic relations and evaluation data-sets

In linguistics similarity both refer 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.

2.5.1 Semantic relations

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

1. Synonyms refer to words that have the same meaning, such as sick and ill.
2. Identity refers to words that have the same lemmas.
3. Antonyms refer to words that have the opposite meaning, such as man and woman.
4. Meronyms refers to a named part of the thing. E.g. Bark is a meronym of ”tree”.
5. Holonyms refers to a named part of the word that describes the thing. E.g. Tree is a holonym of bark. Thereby, the opposite 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 refers to a manner relation of two verbs.

8. Hypernyms refer to words that the word refer to in a type-of-relation. E.g. Bird is a hypernym of duck, crow and seagull.

9. Hypernym is the opposite relation of a hyponym 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 colour.

Words pairs with the same hypernyms or that are 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 with humans. Examples of this are given in the following section about the two used data-sets for evaluation.

2.5.2 WS353

WordSim3535 contains 353 word pairs, each associated with an average of 13 to 16 human judgements. In this case, both similarity and relatedness and similarity 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 the similarity of antonyms, consider them ”similar” i.e., belonging to the same domain or representing features of the same concept, not ”dissimilar”.' (Hill et al., 2015).

As Hill et al. (2015) points out, this results in many different word pairs receiving a high similarity rating such as (coffee, cup) which are associated but not similar and word pairs as (telephone, communication) receiving a low similarity rating. Furthermore, they point out that there was a low annotator agreement (ibid.).

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

2.5.3 Simlex999

Simlex999 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 similarity and relatedness or association in the rating by annotators.

To create a test of the ability of models to capture similarity as opposed to association and relatedness. The data-set is based on word pairs with high similarity and where 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 the 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])”(Hill et al., 2015). 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 authors, therefore, argue that is a context-free-dataset of word pairs (ibid.).

2.5.4 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

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

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. It means the magnitude does not influence the similarity between word vectors, but only how they are positioned in the vector space.

2.6 Statistical measures

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

2.6.1 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's ρ can be used on as a statistical test to determine if there exists a relation between these two variables.

Using ρ as Pearson's correlation coefficient on the two variables X and Y and their ranking $rank_X, rank_Y$, we get the ranking

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

which give us Spearman's rank correlation coefficient. 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).

2.6.2 The geometric and hyper-geometric distributions

Because we have occurrences as a part of the distribution compared to the rest of the distribution, here the top-most similar word pairs versus the rest, we can use the hyper-geometric distribution to test if the occurrences are significant.

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.

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

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 $\binom{n}{k} \equiv \frac{n!}{k!(n-k)!}$.

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:

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

The variables used is described in our significance test is described in the experimental setup section.

3 Experimental setup

3.1 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 compared 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 data-set. To find a hypothesis about the observed, using cosine similarity, we can then examine the word embedding of the model and the semantics of the most similar word pairs rated by the model. It will hopefully give us a hypothesis that can explain the phase transitions.

To test our hypothesis, we evaluate the same word embedding on the SimLex999 data-set to examine. We furthermore make sure our results are significant by using a hypergeometric test. In the following, we will go through the practical setup and method of the experiment.

3.2 Word2Vec Implementation

In our experiment, we use Gensim implementation of Word2Vec. Preliminary experiments also tried using BlazingText from Amazon SageMaker, but since we were unable to reproduce the results from before mentioned studies it was deselected[2].

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 map 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

[2]Preliminary results using BlazingText can be found in the appendix and on the GitHub (GitHub-url).

word pairs. Gensim.Word2Vec is optimised for multi-core CPU processes 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 or continuous bag-of-words algorithms (ibid.).

3.3 Dataset and pre-processing

Wikipedia is a fruitful web-page for natural language in digital text format. To decrease the training time of the W2V models, we use a 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 almost all pre-preprocessing techniques improve the model performance of word embedding.

In text8 the following pre-processing steps have been taken: The wiki dump which has been cleaned to words of the 27 characters English alphabet containing only the letters a-z and nonconsecutive spaces. The goal of the authors of the data-set was only to 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 spelt out, so "20"becomes "two zero". 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 not least, all words are tokenised for the W2V model.

3.4 Size of Context Window

We define the context windows size to go from 1 - 25 in our models using either CBoW or Skip-gram. It gives 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 relationship with the size of the input word vectors.

3.5 Word2Vec parameters and execution

Beyond window size, we 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 is set to 500 to get similar results to Mckinney-Bock and Bedrick (2019).

The models are trained on a personal multi-core computer. Word embedding from W2V models with context windows size from 1 - 25, data and the used code written in Python can be found on GitHub (GitHub-url).

3.6 Model evaluation

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

In our evaluation as well of testing of the models, we use a black-box approach. We can only control input and output to find out how the model performs.

Using Gensim's evaluate function, we can calculate Spearman's ρ between similarity scores of our models and the human annotators. We use the earlier described evaluation data-sets, the WS353 data-set and SimLex data-set, which were downloaded from their respective authors, Gabrilovich (2009) and Hill et al. (2015), and can be found on the projects GitHub.

3.7 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 which make us able to find all semantic relations between the word pairs in the data-set. We do the same for the word pairs in SimLex999.

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)[3].

3.8 Significance test

We use Sci Py's implementation of the hypergeometric test and its survival function to calculate p-values (SciPy Documentation link). In our hypergeometric test, we want to test the top 10% of distribution which is a standard quantity. To test if the occurrence of semantic relations is significant, we set X to be the occurrence of relations between top-30 word pairs, M the total number of pairs, n the total number of found semantic relations and N to be the 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 less than 10% of the original data-set. The same is done for the SimLex999 data-set where we also use a little less than 10% of the original data-sat.

We find the counts by comparing each data point with each relation and then perform the hypergeometric 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 the Bonferroni correction of the significance level, α/m, to compensate for the many relations tested for in the same data which should make us avoid randomness in our results (Dodge, 2008).

[3] The original script by Hershcovich et al. (2019) can be found on GitHub (GitHub link).

Tabel 1: Counts of semantic relations between word pairs included in model vocabulary

	synonyms	identity	hypernyms	hyponyms	member_holonyms	member_meronyms	part_holonyms	part_meronyms	substance_holonyms	substance_meronyms	antonyms	pertainyms	co_hyponyms	total word pairs
SimLex999	21	20	17	18	0	0	4	1	1	0	14	0	26	267
WS353	0	2	3	2	0	1	0	1	0	0	1	0	9	85

4 Preliminary results

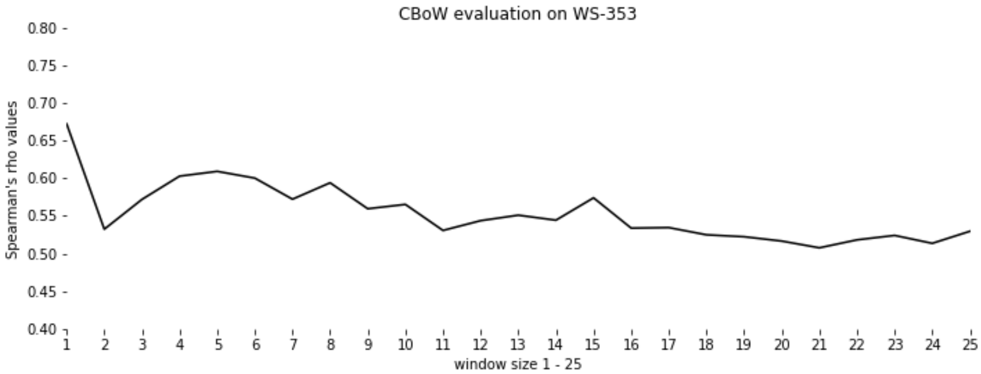
4.1 Descriptive analysis of evaluation data-sets

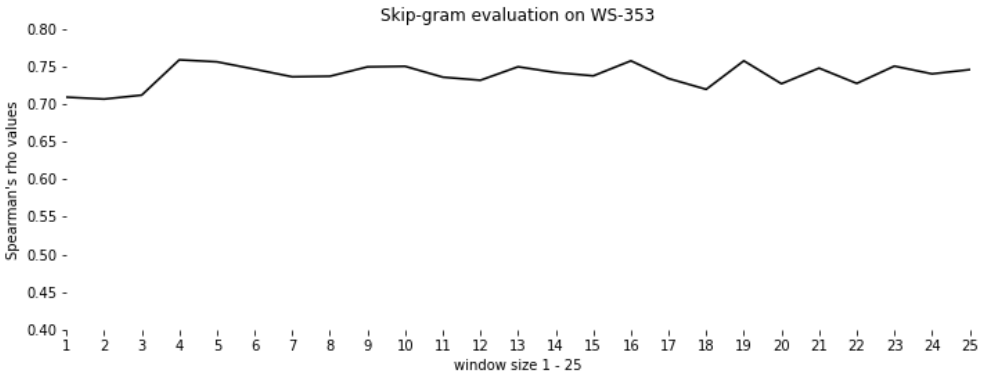
The word vectors of the word embedding produced by our W2V models do not cover all word pairs of the Ws353 data-set or SimLex999 data-set. This gives us two reduced dataset with 85 word pairs and 267 word paris. In these reduced data-sets, we find using the WordNet API the relation between the words in each word pair. We check for all possible semantic relations in the WordNet database, that is synonyms, hypernyms, hyponyms, holonyms, meronyms, holonyms, antonyms and pertainyms. This yields the statistics of the two data-sets seen in tabel 3. We see that there is a different distribution of word relations in the data-sets and that several of the word pairs do not have any relation. We will not test for these non-occurring relations in the following hyper-geometric tests.

4.2 Evaluation dataset WS353

We want to investigate the model findings of Hershcovich et al. (2019) and Mckinney-Bock and Bedrick (2019) and thereby need to re-create W2V models that yield similar results.

Creating W2V models with varying context windows sizes, we it is clear from the plots belpw that we are able to reproduce the results found in Hershcovich et al. (2019) and Mckinney-Bock and Bedrick (2019). As in Hershcovich et al. (2019), we see in the plots below that there is no significant difference between models with a context window size of 1 and 2 when using the Skip-gram algorithm.





4.3 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 2.

We see that the top-5 word pairs have a co-hyponym relation which mean that many of the most similar word pairs are sister-terms, and we see that the semantic relations of top-10 word pairs across models are composed of a similar distribution of co-hyponyms (tabel 3 and 4). This indicate that most models could rate sister terms higher.

4.4 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 co-hyponym relation seem to be significant for the word pairs of the WS353 data-set which indicate that the model rates this semantic relation as a very similar word pair.

Tabel 2: Top-10 word-pairs and their relation from 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

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

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 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)**

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 co-hyponym word pairs very similar across models with different window sizes. Due to the limited size of the reduced Ws353 dataset, results are not very clear. Testing on more word pairs will hopefully give clearer results.

5 Hypothesis

As shown in our preliminary results, we were able to reproduce evidence of W2V models using CBoW with a narrow context window size of 1 are better at finding similarity between word pairs than models with larger context windows.

In our qualitative analysis of the most similar word pairs and their semantic relations, we furthermore find that the words which were rated with a high similarity score across models have a co-hyponym relation (tabel 2). They thereby share the same hypernyms which indicate they are sister-terms. Furthermore, this relation is significant for models with a context window size of 1. This leads us to the hypothesis that sister-terms could explain the phase transition at model evaluation on the WS353 data-set. Using a larger data-set which a greater amount of semantic relations, we hope to find that the co-hypernym relation is only significant for models with narrow context windows.

Following the idea above, our hypothesis is formulated as

Hyp_1: Word2Vec models with narrow context windows find that the most similar word pairs are sister terms.

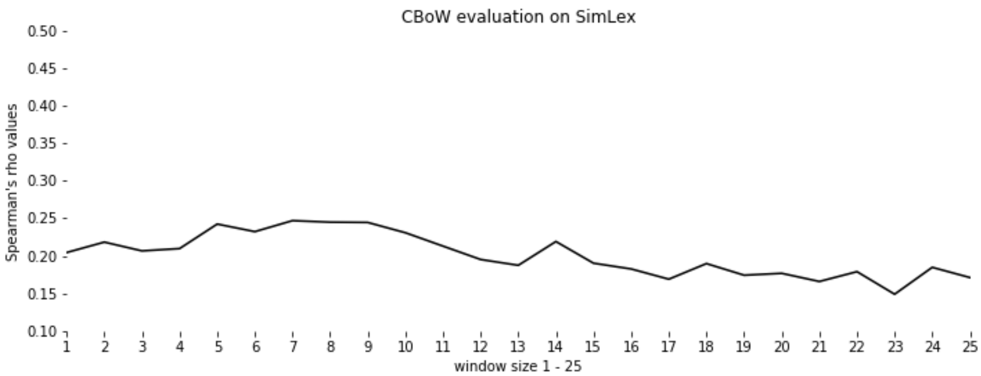
Hyp_0: Word2Vec models with narrow context windows find that the most similar word pairs are not sister terms.

In following section, we will test this hypothesis on our test data-set, SimLex999.

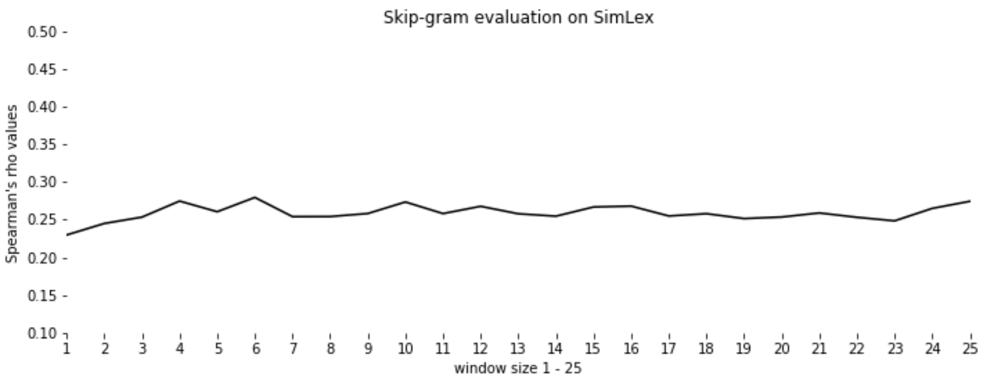
6 Test results

6.1 Evaluation dataset SimLex999

Running the same evaluation of our W2V model with the CBoW algorithm on SimLex999 as on WS353, we get a very different Spearmans ρ.



This could mean that, we do not see the same phase transition phenomena on the SimLex999 dataset as with Ws353. This was also found by Hershcovich et al. (2019). Sister-terms could though still be as the most similar in narrow context windows of 1 following our hypothesis.



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

6.2 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

Tabel 6: Top-10 word pairs Simlex999

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

3, 4). In the 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 (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 (tabel 8).

We will discuss what these results mean and the results adjusted with the Bonforenni correction in the next section.

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)

7 Discussion

7.1 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 have a similar amount of co-hyponyms across all context window sizes (table 5). Running the same experiment on the test data-set, SimLex999, we though find that only a model with a context window of 1 can 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 ($m = 9$, each tested relation) which lead to an $\alpha = 0.05/9 \approx 0.005$ we can more surely deny the significance of co-hyponyms in our test. It means that we cannot confirm our hypothesis using a conservative estimate. The results on SimLex999 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-90, but it is clear that the co-hyponyms stay in the top across all window sizes. Again this leads us to that we cannot confirm our hypothesis about the narrow context window's better performance is due to sister terms.

We furthermore see a high number of antonyms as part of the most similar word pairs in SimLex999 which mean that the model learns antonyms to be very similar. Probably this is because of the similar context which they appear in as with hypernyms and co-hyponyms. Similar studies have found that models cannot distinguish these semantic relations because of their similar context (Silke Scheible and Springorum, 2013). Several studies find that Word2Vec models are outperformed by vector space models using special Antonym-Synonym patterns such as AntSynNET and models using symmetric patterns (Schwartz et al., 2015), (Nguyen et al., 2017).

The high number of antonyms may explain the low evaluation score of the models on SimLex999. The computational understanding of the models does not correspond with a human understanding of word similarity. This is highlighted in table 8 where the antonym relation is very significant in the top word pairs of models across window sizes and the corresponding plot of ρ between the model and SimLex999.

7.2 Differences between WS353 and SimLex999

As shown in the preliminary results (table 1) there is a considerable difference in the distribution of word relations which results in two very different composition of top pairs (table 5 and 8). This shows that there is a big difference in how these data-sets have been made and annotated, which the authors of SimLex999 have also described as actual similarity in SimLex999 vs association and relatedness in WS353 (Hill et al., 2015).

Annotators in SimLex999 used context-free-rating, which 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 makes it easier for such a model. In WS353 annotators were not instructed to differentiate between words that are similar semantically and words which are associated or related. E.g. words used in the same context, such as "cup"and "coffee"will often be associated, but their meaning differs very much (Hill et al., 2015). It might explain why there are so few synonyms in the top of Ws353, but so many hypernyms and co-hyponyms in comparison. It might also explain why there are so many identity and synonyms relation in the top and other relations such as holonyms and antonyms in the SimLex999 data-set.

The considerably higher evaluation score on WS353 might mean that W2V models are better at predicting related and associated words. This makes sense because the model is trained on human-written text from Wikipedia and models using CBoW base their prediction on a similar context where the related and associated words occur. It could also mean that models with narrow context windows are especially good at predicting the similarity of sister-terms and the reason for the high evaluation score on WS353 and low on SimLex999 is because there are many more sister-terms in WS353 and less of other types of relations (table 1).

We can confirm that W2V models perform considerably better when rating similarity on the WS353 data-set than SimLex999, which was also found by Hershcovich et al. (2019) and Mckinney-Bock and Bedrick (2019).

7.3 Technical implications for Word2Vec models using CBoW

The CBoW architecture uses the context around the word to predict the next word. It 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 is very sensitive to word pairs with a co-hyponym relation since these words will often appear in the same context.

As the window size increases, we see that new word pairs in WS353 and SimLex999 with other relations becomes the most similar pairs. In SimLex999, 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.

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 because of less training time and more 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 further.

The results also confirm that language model such as W2V are only learn the form of language and not the meaning. For W2V and more advanced models such as Bidirectional Encoder Representations from Transformers (BERT) it is impossible to learn any meaning by just the form of language, such as text from Wikipedia (Bender and Koller, 2020). Meaning in our case is the semantics which we see the model is only able partly able to capture the similarity between different relations being only trained on the form. Supervised models trained on meaning, such as semantic relations, might perform differently, but unsupervised model such as our model seems to have clear limitation. Furthermore, the results are interesting in the discussion of how well prediction-based word embedding is compared to count-based word embedding. Levy et al. (2015) found in their studies that if hyper-parameters such as context window size is tuned optimally, then word embedding models are not always superior. It also shows that SGNS algorithm is superior to CBoW, which we also found evidence of in our results (Levy et al., 2015)[4].

[4]see plots in result sections

7.4 Further work

To further explore the effect of the context window size, it would be interesting to see how W2V models trained to perform tasks such as classifying semantic relations or leveraging co-hypernyms to perform other tasks. This project has only 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 context windows are good at classifying 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.

As shown in the preliminary results, the W2V models made with Skip-gram algorithm did not show the same kind of behaviour. It though scores 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 SimLex999 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 similar semantic relations such as SimLex999 in German compared to English may or may not yield similar results.

8 Conclusion

In this project, we have looked at Word2Vec, a word embedding model, and developed a method to investigate the phase transition between narrow context windows and larger. We investigated W2V models using the CBoW algorithm across different context windows sizes by evaluating these and found similar results to other studies. To go into depth why this is the case, we further looked into which semantic relations the models rate the highest and investigate how sensitive the models were to different types of relations in terms of similarity. Word2Vec models with a context window of 1 rate co-hyponyms / sister-terms very similar, but also less semantic similar relations such as antonyms.

From our preliminary results and our understanding of how Word2Vec models using CBOW functions, we came up with the hypothesis that the model is able to capture similarity better because of sister-terms, also called co-hyponyms, which models with narrow windows rate as very similar. Sister-terms are also rated as very similar by human judgement which yields good model performance on WS353. Testing this hypothesis on another evaluation data-set, we were not able to confirm our hypothesis. We find this is due to a different distribution of word pair relations in the test data-set and that the language model can not distinguish between with hypernyms, co-hypernyms / sister-terms, synonyms and antonyms which it all rates as very similar. Rating antonyms very high most likely results in a bad evaluation compared to the human annotation. This problem of antonyms is also found in studies of similar word embedding. We also found that the annotation and concept of similarity differ between evaluation data-sets which explain why the models are evaluated so differently on the two data-sets. Our models had far better prediction on the WS353, which is not annotated with actual similarity in mind which allows for associated and related word pairs to be annotated as similar. Furthermore, there are almost no antonyms in the set which makes it easier to achieve good model performance.

In this project, we have given no proof of word embedding ability to model certain semantic relations but a methodology to explain why models with narrow context windows predict similarity of words with different relations better. Interesting insights were found on the differences between models and data-sets, as expressed in semantic relations such as co-hyponyms and antonyms.

Further work should use a larger and more diverse amount of word pairs to test our hypothesis and evaluate models. It would also be interesting to investigate further how supervised models across windows sizes performs and potential efficiency gains.

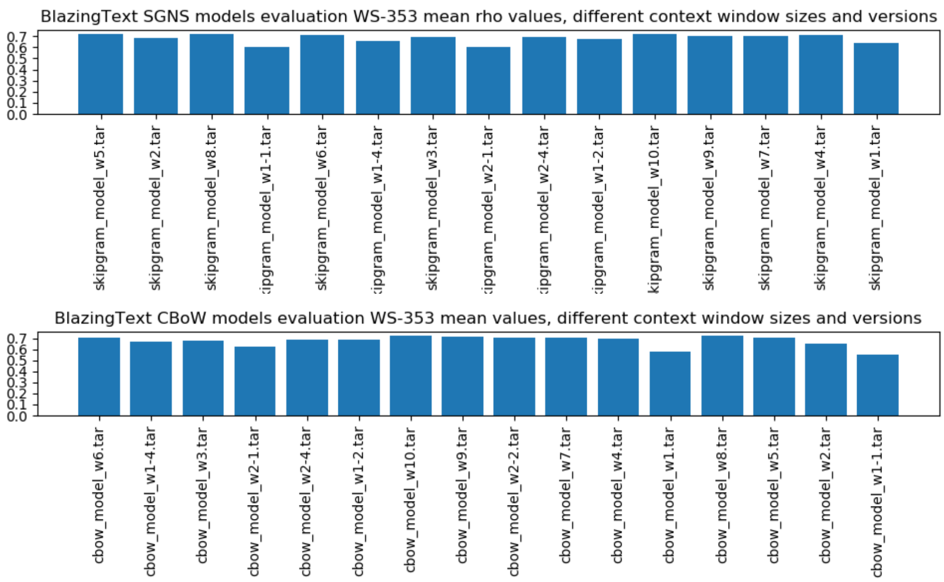
9 Appendix

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).

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

The models are trained and evaluated in AWS SageMaker's cloud environment (url)

Testing the BlazingText models before switching to the Gensim implementation 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). The following plot shows that similar results to these studies where not able to be found using diffrent parameters.



10 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`.

Emily M. Bender and Alexander Koller. Climbing towards nlu: On meaning, form, and understanding in the age of data. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020. doi: 10.18653/v1/2020.acl-main.463. URL `https://www.aclweb.org/anthology/2020.acl-main.463.pdf`.

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.

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`.

Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695, 2015. doi: 10.1162/coli_a_00237.

Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association* *for Computational Linguistics*, 3:211–225, 2015. doi: 10.1162/tacl_a_00134. URL https://www.mitpressjournals.org/doi/pdf/10.1162/tacl_a_00134.

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.

Kim Anh Nguyen, Sabine Schulte Im Walde, and Ngoc Thang Vu. Distinguishing antonyms and synonyms in a pattern-based neural network. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, page 76–85, 2017. doi: 10.18653/v1/e17-1008. URL https://www.aclweb.org/anthology/E17-1008.pdf.

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

Roy Schwartz, Roi Reichart, and Ari Rappoport. Symmetric pattern based word embeddings for improved word similarity prediction. *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, page 258–267, 2015. doi: 10.18653/v1/k15-1026. URL https://www.aclweb.org/anthology/K15-1026.pdf.

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.