KØBENHAVNS UNIVERSITET

# Bachelor thesis

Thor Steen Larsen (mvj665)

# Phase Transitions In Word Embeddings

Department of Computer Science

Supervisor: Daniel Hershcovich

8. juni 2020

# Abstract

Hershcovich et al. [2019] and Mckinney-Bock and Bedrick [2019] found anecdotally that word embedding of CBoW with context window size 1 judge similarity better than those trained with larger windows, but then performance improves gradually until it recovers. 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 windows sizes and evaluate on the WS353 dataset similarity. Furthermore, we look into the word embeddings using explorative data analysis and come up with a hypothesis...
Models are created and evaluated using BlazingText Algortihm from AWS SageMaker and Gensim.Word2Vec.

# Indhold

# 1 Theory

## 1.1 Context-predicting 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 (ref).

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 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 optimize the weights of these.

An alternative is a count-based models. using word n-gram, a sequence of words in a sentence $w_1, \ldots w_n$, we are interested in the prediction task:

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

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.

Predictive models have though 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].

## 1.2 CBoW

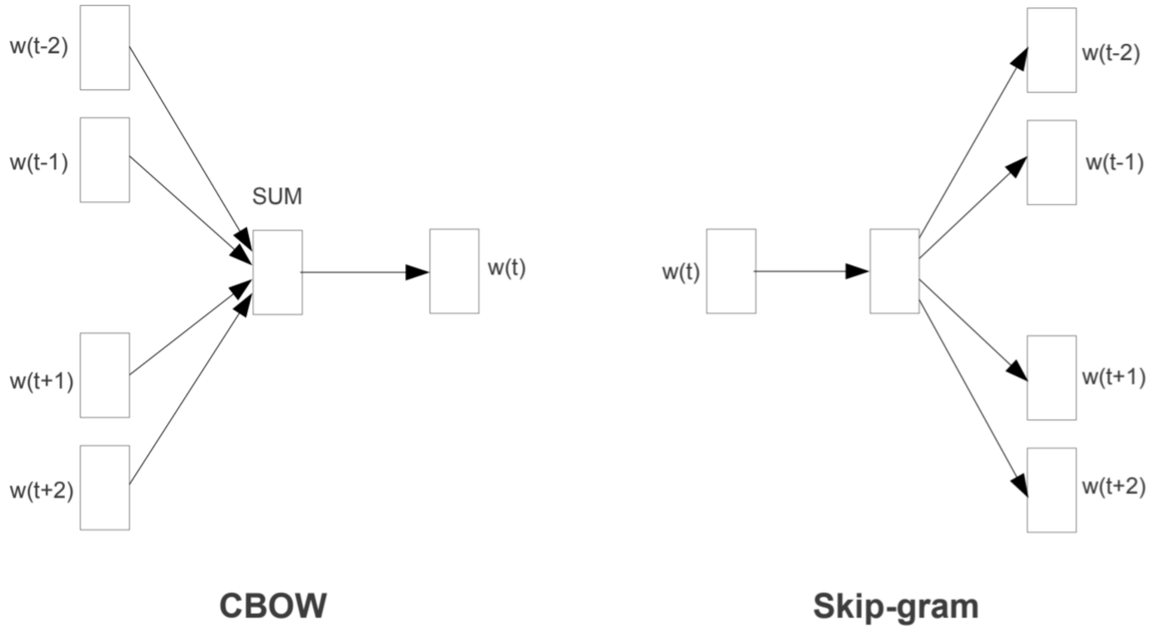



Fig 1: CBoW and Skip-gram architecture [Mikolov et al., 2013a]

In CBoW, we take some the ideas from the previous sub-section and use in 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 $\{\mathbf{x}_1, \dots \mathbf{x}_C\}$ 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

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

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 **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 optimize the weights. We can use the final output of a word vector $y \times \mathbf{W}$ through the softmax 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 softmax 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 stocastic gradient decent as in the architecture of CBoW.
In addition to the general softmax 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 input word Denmark, a sentence: "The capital of Denmark is Copenhagen ... is Aarhus"and a window size 1 then the network is going to learn the statistics of the word pairs (Denmark, of), (Denmark, in) and get many more training samples of (is, Copenhagen) than (is, Aarhus). Which mean that the model will learn there is a higher similarity between Denmark, Capital, Copenhagen than Denmark, Capital, Aarhus.
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.
Is there a difference between skip-gram and cbow?

## 1.5 Cosine similarity

To calculate the similarity score between words, 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. This make us able to avoid using the magnitude as by euclidean distance to compare the word vectors.

## 1.6 Spearman's rank correlation coefficient

To evaluate our models on a test data-set in AWS SageMaker and Gensim, we furthermore calculate Spearman's $\rho$ of the word rankings. In our case the words are ranked by cosine similarity from high to low where the highest cosine similarity will be of the most similar word in the n-gram.
Spearman $\rho$ can be used as a statistical test to determine if there exists a relation between two random variables X and Y. The test can be a bilateral test or a unilateral test [Dodge, 2008].
Spearman's $\rho$, where $\rho$ is Pearson correlation coefficient, and $rank_X, rank_Y$ are the ranking of the two random variables can be written as

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

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].

## 1.7 Similarity

In linguistics similarity both refers to what we humans associate with being similar and similar words in terms of attributes such as hypernyms. Following is short description of semantics more generally and how the data set, we will use for reference are put together.

### 1.7.1 Semantic relations

There are several different kinds of semantic relations where the most important ones for similarity are Synonyms, Antonyms, Hypernyms. whY?

1. Synonyms refer to words that have exactly the same meaning such as sick and ill.

2. Antonyms refer to words that have the opposite same meaning such as man and woman.

3. Hypernyms refer to words that the word refer to where root hypernym refers to the most basic hypernym such as entity or plant.

Words pairs with the same Hypernyms can be described as sister terms. These words will naturally also be very similar, but may not be assoiciated by humans. Examples of this is given in the following reference datasets.

### 1.7.2 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".' (ref).
As ... points out this results in many dissimilar word pairs receive a high rating and no associated but dissimilar concepts receive low ratings such as (coffee, cup) and (train, car) where the first pair is dissimilar but ? and the second pair is highly associated but ?. Furthermore, they point out there was a low annotator agreement (ibid.)
WS353 iis tough the most used gold standard data set for similarity measures (ref).

### 1.7.3 Simlex

Simlex was made to correct some of the shortcomings for WS353. The data set is made of word pairs which conceptually are highly similar or related with a clear distinction between relatedness and association in rating by the annotator.
To create a test of the ability of models to capture similarity as opposed to association the dataset is based on the USF dataset for high word pairs with high similarity and a random selection of word pairs with high similarity put together. Such as words pairs with same hypernyms and atonyms. In the Simlex dataset anntotators were ntroduced to similarity via the well-understood idea of synonymy, and 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])"(ref). 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 (ibid.).

## 1.8 Geometric distribution

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 kth 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 modeled is a sequence of independent trials, there are only two possible outcomes for each trial, often designated success or failure, the probability of success, p, is the same for every trial (ref).
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.
Why is this interesting in our case?
The expected value of the geometric distribution can be derived as follows.

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

Statistical inference?

# 2 Method

We take a data driven inductive approach to testing our hypothesis. 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 use the BlazingText algorithm from Amazon SageMaker and a similar implementation of Word2Vec from Gensim.
BlazingText is a version optimised version of the Word2Vec Algorithm and can be used for embedding as well as text classification [Amazon, 2020].
Gensim's Word2Vec is based on the orginal C implementation by Mikolov et al. [2013a] of Word2Vec [Rehurek, 2019]. The Word2vec algorithm maps words to high-quality distributed vectors. The resulting vector representation of a word is the word embedding. Using these embedding we can find which words that are semantically similar. The BlazingText and Gensim.Word2Vec are highly optimized for multi-core CPU architectures which makes it fast when given several million of words. Furthermore, they are both is able to provide both Skip-gram (SGNS) and continuous bag-of-words (CBoW) models (ibid.).

## 2.1 Dataset and pre-processing

We use fraction of a wiki dump called text8 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. Only regular article text was retained [Mahoney, 2011]. Image captions are retained, but tables and links to foreign language versions were removed. Citations, footnotes, and markup were removed. Hypertext links were converted to ordinary text, retaining only the (visible) anchor text. Numbers are spelled out ("20"becomes "two zero", a common practice in speech research). 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).

## 2.2 Size of Context Window

We define the windows size to go from 1 - 25 in both our CBoW and Skip-gram model. This give os a total of 50 different models.
The windows size determines what we feed into our model and model complexity due to the vectors direct relations ship with the size of the input word vectors. Mikolov et al. [2013a] found that increasing the windows size improves quality of the resulting word vectors, but it also increases the computational complexity. To be efficiently get the best results they chose a window size of 10 in their experiment (ibid.).

## 2.3 Word2Vec parameters

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].
Using BlazingText parameters were set similarly and in some instances lower.

## 2.4 Evaluation

Using Gensim we can calculate Spearman's $\alpha$ between similarity scores of our models and the human annotators. This is done using the WS353 dataset and SimLex dataset which were downloaded from there respective websites.

## 2.5 Qualitative analysis of semantics

Using WordNet, we can find all the semantics of the word pairs and look into there lingustic similarities. WordNet is a database of synsets developed by Stanford University which has public python API (ref). Furthermore, we can consider other attributes such hypotenyms and each word pair's similarity ranking in comparison with that of WS353 and SimLex.

# 3 Hypothesis

Using an inductive approach, we look at some at what data is our models produce. As shown in the earlier section, we were able to reproduce the fact that CBoW models with a context window size of 1 models better word similarity that a models with larger context windows.
When looking into the results, we furthermore found that the words rated with high similarity share many of the same hypernyms which indicate they could be sister terms. When running the same evaluations of the CBoW models on the SimLex dataset we get? (to do)

$Hyp_0$: Word embedding made with a Continuous-Bag-of-words method and a context window $= 1$ are significantly learns sister terms more accurately than similar models with context windows $> 1$.

$Hyp_1$: Word embedding made with a Continuous-Bag-of-words method and a context window $= 1$ are significantly does not learn sister terms more accurately than similar models with context windows $> 1$.

Other formulation:
$Hyp_0$: Word-embedding made with a Continuous-Bag-of-words method rates sister terms higher than other terms.
$Hyp_0$: Word-embedding made with a Continuous-Bag-of-words method does not rate sister terms higher than other terms.

Other formulation:
$Hyp_0$: Word-embedding made with a Continuous-Bag-of-words method with W2V are exposed to rating sister terms higher than other terms.
$Hyp_0$: Word-embedding made with a Continuous-Bag-of-words method with W2V find sister terms the most similar given narrow context windows.

# 4 Results

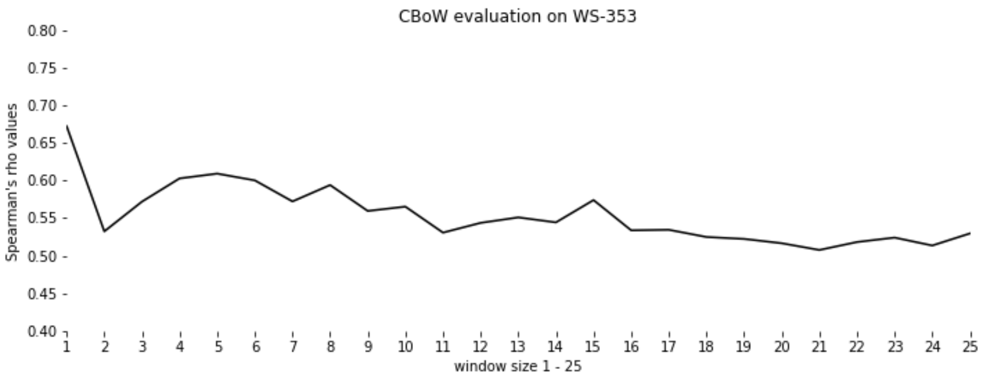
## 4.1 Execution

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-url.

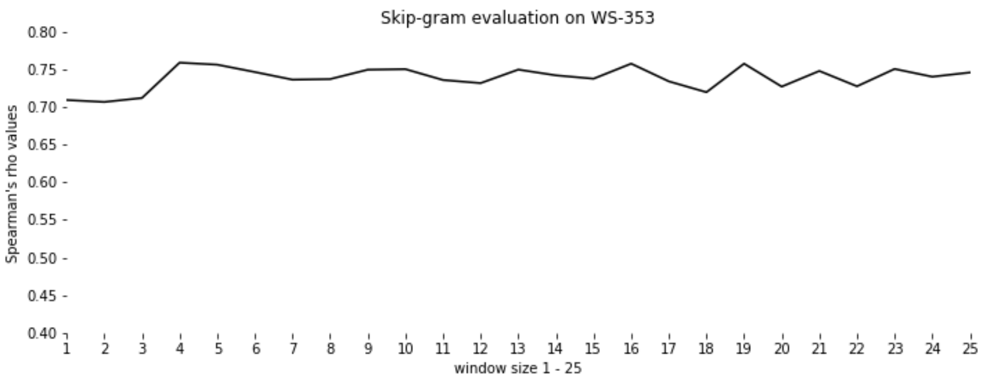
## 4.2 Evaluation

In our evaluation as well of testing of the models, we can only use input and output to find out how the model performs.
Each model is evaluated on WordSim353 for similarity and relatedness which is a dat aset 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.

Depending on the algorithm we use, we get different results when evaluating on WS-353. Using different parameters with BlazingText and a changing window size, we are not able to reproduce the finds 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 smaller when using the Skip-gram algorithm.




make display of word pair and their semantically attributes

# 5 Appendix

## 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.

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/`.

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