KØBENHAVNS UNIVERSITET

Bachelor thesis

Thor Steen Larsen (mvj665)

Phase Transitions In Word Embeddings

Department of Computer Science

Supervisor: Daniel Hershcovich

11. maj 2020

Abstract

McKinney-Bock and Bedrick (2019) and Hershcovich et al. [2019] found anecdotally that CBOW word embeddings 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 repreduce the results of Hershcovich et al. [2019] using the natural language processing model word-2-vec and look into the word embeddings using explorative data analysis.
Experiment is done using BlazingText Algortihm from AWS SageMaker.

Indhold

1 Introduction

2 Method

We take a data driven inductive approach to testing our hypthesis. We therefore want to recreate the setting where we find the phase transitions.
To reproduce the experiment by Hershcovich et al. [2019], we use the BlazingText algorithm from Amazon SageMaker.
BlazingText is a version optimised version of the Word2Vec Algorithm and can be used for embeddings as well as text classifcation (need ref).
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 implementation of the Word2vec algorithm is highly optimized for multi-core CPU architectures which makes it fast when given large datasets as input. BlazingText is able to provide both Skip-gram (SGNS) and continuous bag-of-words (CBoW) models.

2.1 Dataset and pre-processing

We use fraction of a wiki dump called text8. TO do: look into dataset attributes and source.

2.2 CBoW

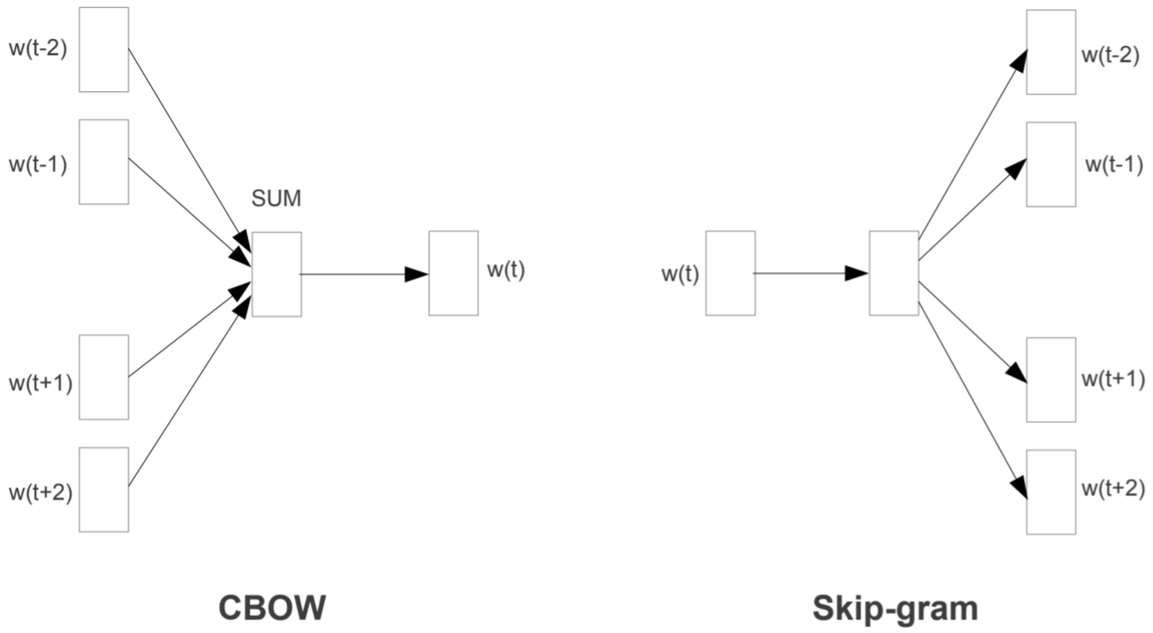

Fig 1: CBoW and Skip-gram architecture Mikolov et al. [2013]
In CBoW we use a similar model to a neural network language model to learn the embedding of each words context.
In the NNLM where the non-linear hidden layer is removed and the projection layer is shared for all the words. This is called a bag-of-words model as the order of words in the history does not influence the projection and all the word vectors are averaged [Mikolov et al., 2013]. This is the main difference compared to the Skip-gram.
The basic CBoW architecture is as following. The input layer consists of the one-hot encoded input context word for a word and vocabulary depending on the size of the window and vocabulary. The hidden layer is an N-dimensional vector. Finally, the out-

put layer is output word which is also one-hot encoded. The one-hot encoded input vectors are connected to the hidden layer via a weight matrix and the hidden layer is connected to the output layer via another weight matrix. Using forward propagation with a soft-max activation function to learn the weight matrices and backward propagation with stochastic gradient descent to optimize the weights, we obtain the word embedding as one-hot vector.

2.3 Skip-gram

In Skip-gram we use a simple neural network to obtain the embedding.
The basic Skip-gram architecture is as following. 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 in the form of a one-hot-vector which is our output, embedding. Or put as by the authors of Word2Vec, Mikolov et al. [2013], 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. Furthermore, negative sampling is used to decrease the computational cost need ref.

2.4 Window size

We define the windows size to go from 1 - 10 in both our CBoW and Skip-gram model. This give os a total of 20 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. [Mikolov et al., 2013] 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.).

2.5 Execution

The Models are executed and run in AWS SageMaker's cloud environment. Models can and code can be found on GitHub.

Evaluation

Each model is evaluated on WordSim353 for similarity and relatedness.
To dos:

1. Make script for unpacking eval.jsons
2. Make plot of rho-alpha values
3. Discuss results compared to Hershcovich et al. [2019]

3 Appendix

Litteratur

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`.

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