Limits of Deep Learning

Simon Jenni
University of Bern
Email: simujenni@students.unibe.ch

Simon Brulhart
University of Fribourg
Email: simon.brulhart@unifr.ch

***Abstract*—The abstract goes here.**

I. Introduction

Recent state of the art natural language models represent words by embedding them as vectors in a continuous vector space [1] [2]. These embeddings are learned as weights of a recurrent neural network language model. It has been demonstrated that the distributed representation of words as vectors in a common vector-space captures not just syntactic similarities (e.g. cat is close to $cats$) but semantic similarities as well. Concretely, the model's ability to perform "word-arithmetic" has been demonstrated in [3]. To exemplify this, let f be the word-embedding function. Ideally, we would then have an embedding where distances in vector-space give a measure of semantic-similarity between the words. For two word pairs such as $(king : man)$ and $(queen : woman)$ which have the same relation (or similarity) we would obtain $f(king) - f(man) = f(queen) - f(woman)$ or equivalently $f(king) - f(man) + f(woman) = f(queen)$. This of course is equivalent to forming the analogy "$king$ is to man as $queen$ is to $woman$".

The goal of this work is to further evaluate the performance of the popular word2vec tool on these tasks. We evaluate the effect of different model and learning parameters on the performance and test the limits of the models semantic-analysis capabilities on new challenging test data. The ability to learn analogies has only really been demonstrated on rather general and hand-picked test-sets. We test the models ability to learn domain-specific knowledge by generating questions that test relationships such as $Brand \sim Country$ and $Movie \sim Director$. Previous work only considers the nearest resulting vector in the embedding space when evaluating the accuracy of this word-arithmetic. We generalise the notion of accuracy by considering the not only the nearest word-vector, but consider the set of the k nearest vectors and test wether the target word is present therein.

The content of this paper is structured as follows: Section II gives an overview of relevant previous work. In Section III we give a brief overview of the underlying model and in Section IV we outline the experimental setup and test-data used. The results of the experiments are then reported in Section IV-C before we conclude in V.

II. Related Work

The work by Tomas Mikolov [1], where the models were introduced, reports results on a test set they themselves created.

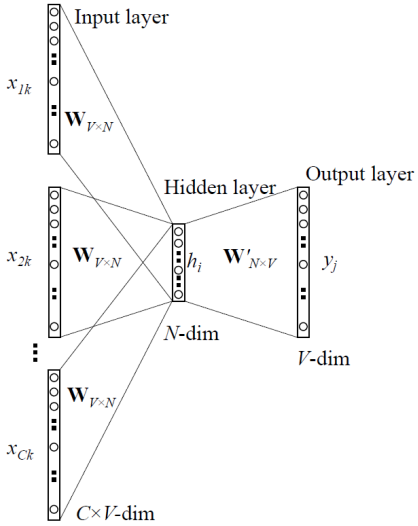

Fig. 1: Architecture of the CBOW model.

The set contained five types of semantic questions where relations such as $Country \sim Currency$ or $Country \sim Capitol$ were tested. In [3] the models performance on semantical similarity has been demonstrated on SemEval-2012 Task 2 [4]. The objective in this task is to decide how similar two target word-pairs such as $(glass : break)$ and $(soldier : fight)$ are with respect to a relation "an X typically Y". Though the model is not directly trained on this task, it achieved better results than the previous best methods.

In [5] Levy et. al. provide a comprehensive study of several word embedding techniques on the two tasks *Word Similarity* and *Analogy*. They conclude that much of the performance gains of new embedding methods stem from design-choices and hyper-parameter optimisation rather than the underlying algorithms. Performance estimation was based only on the test-data sets introduced in [1] and [3].

In a previous work [6] they show that generalising the Skip-Gram model's negative sampling to arbitrary contexts results in the model learning functional-similarities. They demonstrate this by manually inspecting the top k most similar words to a given query word. A query for *Hogwarts* for example returns a list of famous schools in their modified Skip-Gram implementation while CBOW returns a list of Harry Potter characters.

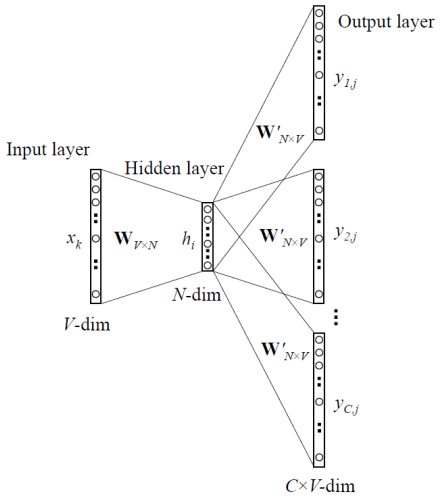

Fig. 2: Architecture of the Skip-Gram model.

III. The Models of word2vec

word2vec is an implementation of the Continuous Bag of Words (CBOW) and Skip-Gram models introduced in [1]. Both of these models are based on a two-layer recurrent neural networks with architectures depicted in Figure 1 and Figure 2. Both architectures take one-hot encoded words as inputs. Let x_i denote the input corresponding to word i and let the input vocabulary be V, then the inputs are given by the $|V|$-dimensional vectors:

$$w_a = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, w_{abandon} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \ldots, w_{zone} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \quad (1)$$

These one-hot vectors get mapped into a N-dimensional vector via the $|V| \times N$ weight matrix $\boldsymbol{W}$. The hidden unit then accumulates the sum of these vectors for all input words. The output of the model is generated by multiplying the hidden-units by weights W' and applying soft-max to the result in order to generate a valid probability-distribution over words in the vocabulary. Concretely, given a sequence of training words $w_1, w_2, \ldots, w_T$ the hidden-states are computed as

$$h(t) = \sigma(\boldsymbol{W} w(t) + \boldsymbol{H} h(t-1)), \quad (2)$$

where $\boldsymbol{H}$ is the $N \times N$ matrix of hidden-to-hidden weights and σ the sigmoid function:

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad (3)$$

Given the values of the hidden unites, the output can be computed as

$$y(t) = P_t(\boldsymbol{W'} h(t)) \quad (4)$$

with the softmax function P_t of a N-dimensional vector x given by:

$$P_t(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \quad (5)$$

The learnt word representation are then given by the columns of $\boldsymbol{W}$.

The models are learnt in an unsupervised fashion where either the centre-word of a series of words is predicted from its context (CBOW) or the context is predicted from the centre-word (Skip-Gram). Both models try to maximize the log-likelihood of their output. Training is done via gradient descent using back-propagation [7] to compute derivatives.

IV. Experiments

In this sections we first outline our experimental setup, introduce novel test-data and then report results obtained for different hyper-parameter choices and test scenarios.

A. Setup

We used the Python re-implementation of word2vec contained in the Gensim library [8]. This implementation is functionally identical to the original C-implementation provided by [1], but is more readable and has shown to perform just as well.

To compare the influence of hyper-parameters on the performance we train several models with different values for one of the parameters and keep all other parameters fixed. We identified the following set of hyper-parameters for our experiments:

1) Context-window size
2) Vector size (i.e dimension of hidden-layer)
3) Learning rate α
4) Training corpus
5) Hierarchical softmax vs. negative sampling for Skip-Gram

B. New Test Data

C. Results

V. Conclusion

The conclusion goes here.

Acknowledgment

The authors would like to thank...

References

[1] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[2] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation." in *EMNLP*, vol. 14, 2014, pp. 1532–1543.

[3] T. Mikolov, W.-t. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations." in *HLT-NAACL*, 2013, pp. 746–751.

[4] D. A. Jurgens, P. D. Turney, S. M. Mohammad, and K. J. Holyoak, "Semeval-2012 task 2: Measuring degrees of relational similarity," in *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*. Association for Computational Linguistics, 2012, pp. 356–364.

[5] O. Levy, Y. Goldberg, and I. Dagan, "Improving distributional similarity with lessons learned from word embeddings," *Transactions of the Association for Computational Linguistics*, vol. 3, pp. 211–225, 2015.
[6] O. Levy and Y. Goldberg, "Dependency-based word embeddings." in *ACL (2)*, 2014, pp. 302–308.
[7] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.
[8] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50, http://is.muni.cz/publication/884893/en.