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*—Current natural language models try to capture semantic and syntactic relationships between words. This allows them to answer questions in the form of analogies: "X is to Y as U is to ?". We study the performance of the word2vec tool on this task. To this end, we analyse the sensitivity to different hyperparameter choices. We generate new questions and demonstrate the models (in)ability to learn and answer on domain-specific knowledge. Using new questions we also highlight the general limits of the model's capabilities. To offer a more detailed view of the model's performance, we introduce a broader definition of accuracy which is taking into account how far off the model is in a failure case. Finally, we experiment with different distance functions to compute the similarity between words and demonstrate when they can benefit the model's accuracy.**

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 shown 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 $f(king) - f(man) \approx f(queen) - f(woman)$ or equivalently $f(king) - f(man) + f(woman) \approx f(queen)$. This of course is equivalent to 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 choices for model-architectures and hyperparameters on the performance and test the limits of the models semantic-analysis capabilities on new challenging test data. The ability to learn analogies has 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 focuses on the nearest resulting vector in the embedding space when evaluating the accuracy of this word-arithmetic. We generalise the notion of accuracy by considering 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 present a brief overview of the underlying model and in Section IV we outline the experimental setup and testdata used. The results of the experiments are then reported in Section V before we conclude in VI.

II. Related Work

The work by Tomas Mikolov [1], where the models were introduced, reports results on a test set they themselves created. The set contained five types of semantic questions where relations such as $Country \sim Currency$ or $Country \sim Capital$ 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 wordpairs 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 parameter optimisation rather than the underlying algorithms. Performance estimation was based only on the testdata 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 *e.g.* for *Hogwarts* returns a list of famous schools in their modified Skip-Gram implementation while CBOW returns a list of *Harry Potter* characters.

III. The Models of word2vec

The word2vec tool is an implementation of the Continuous Bag-of-Words (CBOW) and Skip-Gram models introduced in [1]. Both of these models are based on two-layer recurrent neural networks with architectures depicted in Figure 1 and Figure 2. The two architectures reflect the objectives of both models: CBOW is trained to predict the centre word from its context in a sentence while Skip-Gram tries to predict the context from its centre. Both architectures take vectors of one-hot encoded words as inputs. Let x_i denote an input-vector

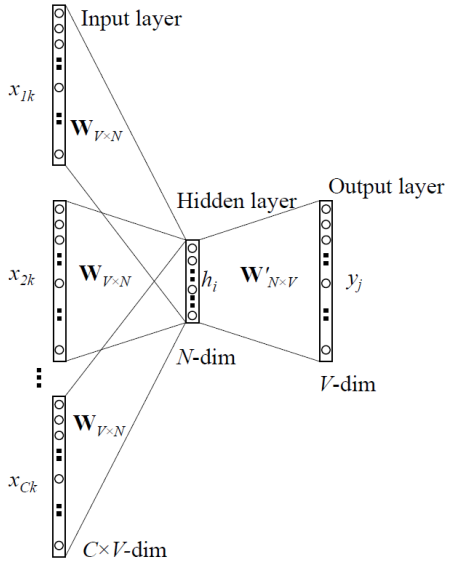

Fig. 1: Architecture of the CBOW model. The three one-hot input vectors represent the context of a centre-word in a sentence and the output vector is a vector with probabilities over possible centre-words.

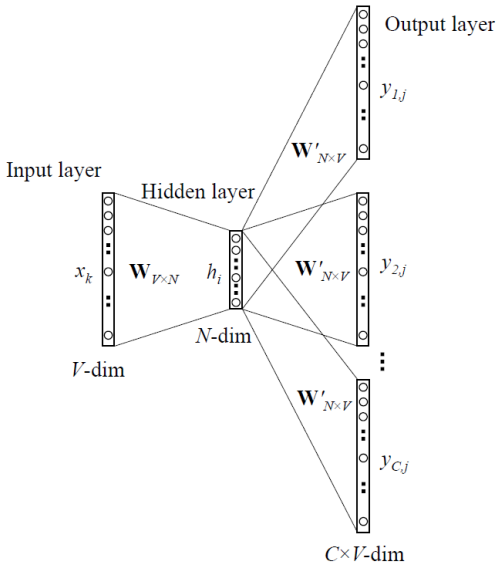

Fig. 2: Architecture of the Skip-Gram model. The output of this model are vectors with probabilities for the words occurring in the context of the input word.

corresponding to word i and let the input vocabulary be V, then the inputs to the model are given by the $|V|$-dimensional vectors:

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

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 in the case of CBOW and consists of only the vector associated with the single input in the Skip-Gram model. The output of the model is generated by multiplying the hidden-units by weights W' and applying softmax 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

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

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

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

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

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

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

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

for $i \in \{1, \ldots, |V|\}$. The learnt word representations 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 maximise the log-likelihood of their output. Training is done via gradient descent using backpropagation [7].

At test time when looking for the word most similar to a given word w we solve for

$$x = \text{argmin}_{v\in V} d(f(w), f(v)) \tag{6}$$

Where f is the embedding and d is a distance function. We therefore search for the word in the vocabulary which has the minimal distance in the embedding space. The model as proposed by [1] chose d to be the cosine similarity, but we investigate different choices such as L^p-norm or Dice similarity.

A. Mean-Shift for Better Discrimination

Solving for the closest word in Equation 6 based on the cosine similarity can be interpreted as projecting on an N-dimensional unit sphere and looking for the projected vector with smallest angle to the given query vector. This projection can lead to similar vectors being very close on the sphere and therefore hard to distinguish based on angular distance. This observation is also backed up by the plots shown Figure 7

where we observe that the correct word is often very close by. Being able to better distribute out the the points on the sphere would therefore likely improve the predictions.

We therefore propose the following mean-shift procedure:

1) Normalise all vectors (*i.e.* project on the unit sphere)
2) Compute the mean of the normalised vectors
3) Shift by mean
4) Normalise again (*i.e.* re-project on unit sphere)
5) Compute dot-products with query vector to get cosine-similarities

The intuition behind this procedure is depicted in Figure 3.

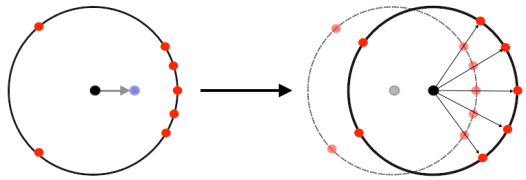

Fig. 3: Sketch of the proposed mean-shift procedure in 2D. *Left*: Unit sphere with projected vectors in red. The points on the right are relatively close together. The mean of the red points is indicated in blue. *Right*: The shifted and reprojected points. The points are now farther apart and better distinguishable.

IV. Experiments

In this section we first outline our experimental setup, introduce novel test data and then report results obtained for different 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 easier to extend and has shown to perform just as well.

To compare the influence of 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 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. Training Text Corpora

We trained models using multiple text corpora to check if hyperparameters influence performance differently depending on the corpus. These corpora include:

1) Wiki: The first 900 megabytes of the English Wikipedia dump on 03/03/2006. [1]

2) Text8: The first 100 megabytes of the English Wikipedia dump. Facilities for downloading and using this set are provided with word2vec. [2]

3) Brown: English corpus of one million words assembled from 500 sources of different genres, created in 1961 at Brown University. It is considerably smaller than our other corpora. We used this set through the NLTK python library. [3]

C. New Test Data

To test the models performance on domain-specific questions we constructed several questions sets. The questions are all of the form "X is to Y as U is to ?", *i.e.* they take the form of analogies. We constructed the following sets of questions:

1) Brand-Origin: A set of questions testing for the relationship between well-known brands and their countries of origin. To construct this questions set we gathered names and country of origin for the top 500 brands according to brandfinance.com. We excluded all the brands from countries with names composed of multiple words (*e.g.* United States of America, ...) due to the model working on single words. The result is a set of 38,220 questions.

2) Movie-Director: A set of analogy-questions for the relationship between famous movies and their respective directors. The set was constructed by gathering movie titles and last name of the director for some of the top rated movies according to IMDb. Similar to the Brand-Origin questions we excluded movies with titles composed of more than one word with the exception of titles starting with "The ..." or similar. The result is a set of 4,557 questions.

V. Results

A. Influence of Parameters on Model Performance

We first wanted to see the influence of parameters when using the generalised question set described in [1]. For this purpose we plotted the accuracy obtained for different values of a chosen parameter. Here, the accuracy is defined as the ratio of the number over right answers to the total number of questions.

In Figure 4 we see that the window size has only a limited influence on the accuracy. It is also difficult to pinpoint a common best value but window sizes between 5 and 8 seem to work well on all corpora. Generally the model is quite robust with respect to this parameter. Interestingly, a bigger window size seems beneficial when using CBOW on the wiki corpus, but detrimental when using Skip-Gram.

Conversely, we see in Figure 5 that a large enough vector size is essential for good performance. A larger value generally improves accuracy, although we get diminishing returns above a certain point. This is not surprising as the vector size defines the complexity of the model and therefore also matters more

[1] Instructions to download and build the test data on this website: http://mattmahoney.net/dc/textdata.html

[2] Ready-to-use corpus: http://mattmahoney.net/dc/text8.zip

[3] See http://www.nltk.org/book/ch02.html

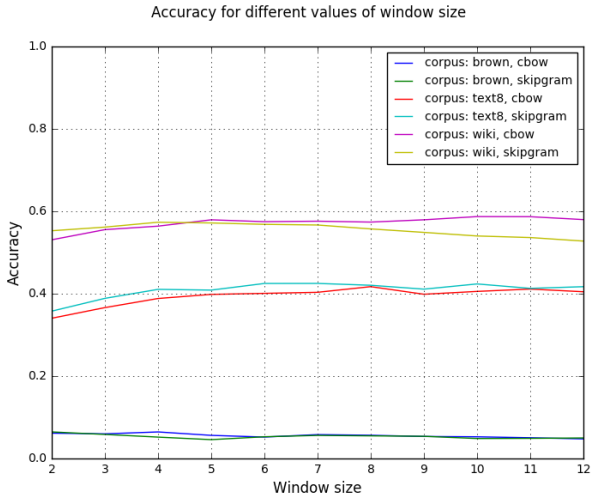

Fig. 4: Accuracy for different values of the context-window size.

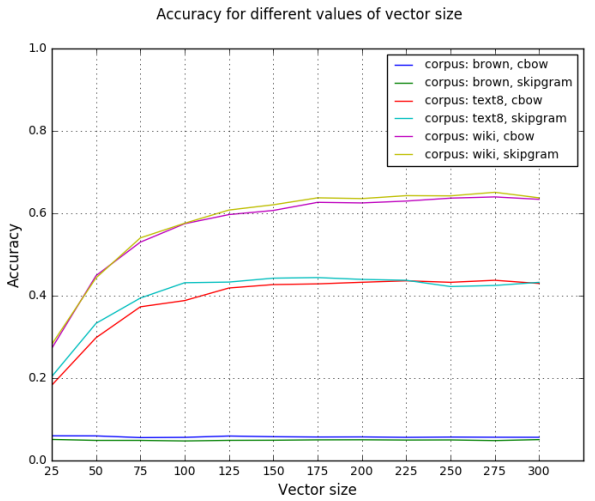

Fig. 5: Accuracy for different values of vector-size (i.e. dimension of embedding space).

for larger corpora. We noticed that the vector size also has some effect on training time, so it is wise to restrain the vector size.

The learning rate (α) greatly affects performance as can be seen in Figure 6. There appears to be a critical spot for α around which the accuracy drops dramatically (values around 0.03 - 0.15). However, a learning rate around 0.007 - 0.03 yields good results for every model we tried.

From a broader perspective, we observe that the corpus size is by far what influences performance the most. This is not surprising given that neural networks are dependent on large amounts of training data. We also notice that the Skip-Gram architecture slightly outperforms CBOW in most cases.

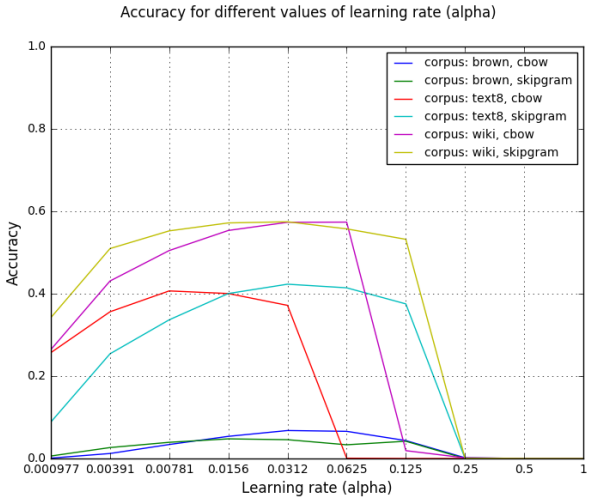

Fig. 6: Accuracy for different values of the learning rate α.

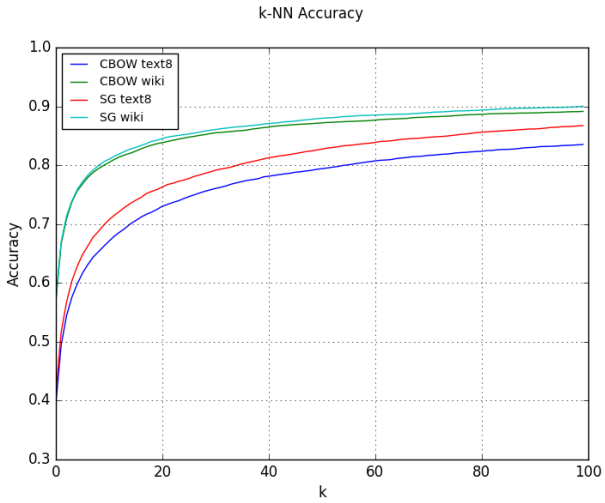

Fig. 7: k-NN accuracy according to k.

B. K-NN Accuracy

In cases when the model gives the wrong answer, the common definition of accuracy ignores how far off the model's prediction was from the correct answer. To get an idea of how wrong the model really is, we tried a different definition of accuracy where we consider the k nearest neighbours of the prediction. We count the answer as a success if this set of neighbours contains the correct answer.

We plotted this k-NN accuracy for different values of k in Figure 7. It is clearly observable that the accuracy increases dramatically when changing from $k = 1$ to 2 or 3 and only slowly starts saturating at around $k > 10$. This suggests that the prediction is in fact very not far off most of the time.

C. Distance Functions

Results for new distance-functions will go here...

Method	Capital-Country	Currency-Country	City-State	Family	Adjective-Adverb	Opposite	Superlative	Nationality-Adjective	Past-tense	Plural	Total
Cosine	45.2%	18.7%	21.3%	62.1%	**15.5%**	15.0%	36.8%	77.6%	**33.9%**	45.0%	42.0%
Dice	42.8%	20.5%	19.5%	**65.0%**	13.5%	**15.4%**	37.7%	**79.2%**	32.1%	44.8%	41.1%
L2	44.0%	14.9%	20.6%	60.1%	15.2%	11.8%	27.7%	74.9%	30.2%	42.8%	39.9%
Manhattan	39.5%	17.5%	17.4%	56.9%	12.8%	8.8%	25.7%	72.8%	24.8%	36.2%	36.1%
Cosine+RC	**47.6%**	**21.6%**	**23.0%**	63.4%	14.8%	14.4%	**42.9%**	77.9%	33.7%	**49.2%**	**43.5%**

TABLE I: Comparison of different similarity-measures. Results obtained on the generalised question set of [1] with a Skip-Gram model trained on the *Text8* corpus. We can observe that cosine-similarity overall performs better compared to the other measures. Although not a similarity-measure by its own we list the results of our proposed re-centering approach for comparison and note a significant increase in accuracy.

Model	Capital-Country	Currency-Country	City-State	Family	Adjective-Adverb	Opposite	Superlative	Nationality-Adjective	Past-tense	Plural	Total
CBOW	55.7%	15.7%	35.2%	77.1%	25.7%	30.4%	53.4%	79.8%	38.3%	67.0%	53.5%
CBOW+RC	63.9%	20.2%	**43.7%**	**80.0%**	29.1%	35.4%	**62.8%**	83.7%	45.2%	67.9%	59.0%
Skip-Gram	**70.7%**	20.8%	34.6%	71.3%	**29.2%**	37.1%	55.1%	88.6%	48.9%	68.3%	58.3%
Skip-Gram+RC	70.6%	**21.3%**	35.7%	70.8%	28.9%	**38.9%**	59.5%	**89.5%**	**49.8%**	**70.0%**	**59.6%**

TABLE II: Performance comparison of models with and without our proposed re-centering approach. Results obtained on the generalised question set of [1] with models trained on the *Wiki* corpus.

Model	Capital-Country	Currency-Country	City-State	Family	Adjective-Adverb	Opposite	Superlative	Nationality-Adjective	Past-tense	Plural	Total
CBOW	13.9%	7.1%	12.3%	72.2%	11.9%	19.0%	29.8%	59.5%	24.0%	43.0%	31.1%
CBOW+RC	34.4%	16.0%	15.6%	**79.7%**	15.3%	**21.2%**	42.3%	65.6%	31.2%	45.7%	40.0%
Skip-Gram	46.0%	18.7%	21.0%	62.1%	**15.5%**	15.0%	36.8%	77.6%	**33.9%**	44.9%	42.1%
Skip-Gram+RC	**47.6%**	**21.6%**	**23.0%**	63.4%	14.8%	14.4%	**42.9%**	**77.9%**	33.7%	**49.2%**	**43.5%**

TABLE III: Performance comparison of models with and without our proposed re-centering approach. Results obtained on the generalised question set of [1] with models trained on the *Text8* corpus.

VI. Conclusion

The conclusion goes here.

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.