Research Article

Tosin Adewumi*, Foteini Liwicki, and Marcus Liwicki

Word2Vec: Optimal hyperparameters and their impact on natural language processing downstream tasks

https://doi.org/10.1515/comp-2022-0236
received December 18, 2020; accepted March 01, 2022

Abstract: Word2Vec is a prominent model for natural language processing tasks. Similar inspiration is found in distributed embeddings (word-vectors) in recent state-of-the-art deep neural networks. However, wrong combination of hyperparameters can produce embeddings with poor quality. The objective of this work is to empirically show that Word2Vec optimal combination of hyper-parameters exists and evaluate various combinations. We compare them with the publicly released, original Word2Vec embedding. Both intrinsic and extrinsic (downstream) evaluations are carried out, including named entity recognition and sentiment analysis. Our main contributions include showing that the best model is usually task-specific, high analogy scores do not necessarily correlate positively with *F*1 scores, and performance is not dependent on data size alone. If ethical considerations to save time, energy, and the environment are made, then relatively smaller corpora may do just as well or even better in some cases. Increasing the dimension size of embeddings after a point leads to poor quality or performance. In addition, using a relatively small corpus, we obtain better WordSim scores, corresponding Spearman correlation, and better downstream performances (with significance tests) compared to the original model, which is trained on a 100 billion-word corpus.

Keywords: Word2Vec, hyperparameters, embeddings, named entity recognition, sentiment analysis

*** Corresponding author: Tosin Adewumi,** ML Group, EISLAB, Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, 97187, Luleå, Sweden, e-mail: tosin.adewumi@ltu.se
Foteini Liwicki: ML Group, EISLAB, Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, 97187, Luleå, Sweden, e-mail: foteini.liwicki@ltu.se
Marcus Liwicki: ML Group, EISLAB, Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, 97187, Luleå, Sweden, e-mail: marcus.liwicki@ltu.se

Acronyms

BERT	Bidirectional Encoder Representations from Transformers
Bi-LSTM	bi-directional long short-term memory network
BW	billion word
BoW	bag-of-words
CBoW	continuous bag-of-words
CNN	convolutional neural network
GDC	Gothenburg dialog corpus
GMB	Groningen meaning bank
IMDB	internet movie dataset
LSTM	long short-term memory network
NER	named entity recognition
NLG	natural language generation
NLP	natural language processing
NN	neural network
SA	sentiment analysis
SoTA	state-of-the-art
SRT	department of computer science, electrical and space engineering
SW	simple wiki

1 Introduction

There are two architectures in the implementation of the Word2Vec model: continuous skipgram and continuous bag-of-words (CBoW) [1]. Similar distributed models of word or subword embeddings find usage in recent state-of-the-art (SoTA), deep neural networks like Bidirectional Encoder Representations from Transformers (BERT) and its successors [2–4]. BERT generates contextual representations of words after it has been trained for extended periods on large corpora, unsupervised, using the attention mechanisms [5]. Unsupervised learning provides feature representations using large unlabelled corpora [6].

It has been observed that various hyperparameter combinations have been used in different works involving

Word2Vec, with the possibility of some of them being sub-optimal [7–9]. Therefore, the authors seek to address the research question: what is the optimal combination of Word2Vec hyperparameters for intrinsic and extrinsic natural language processing (NLP) purposes, specifically for named entity recognition (NER) and sentiment analysis (SA)? The number of combinations of hyperparameters that is possible for a neural network (NN) is very high, even for an NN with only a few layers [10]. Hence, the scope of our extensive, empirical work over three English corpora is on the dimension sizes, training epochs, window sizes, architectures, and the training algorithms (hierarchical softmax and negative sampling) of both skipgram and CBoW, as given in Section 3.

The objective of this work is to determine the relative optimal combinations of Word2Vec hyperparameters for intrinsic evaluation (semantic and syntactic analogies) and a couple of extrinsic evaluation tasks [11,12]. It is not our objective in this work to set new SoTA results. Some of the main contributions of this work are the discovery of the behaviour of quality of word-vectors *vis-a-vis* increasing dimensions for a given data size, the empirical establishment of more optimal combinations of Word2Vec hyperparameters for the identified NLP tasks, and the confirmation of the performance of embeddings being task-specific. The rest of this article is organized as follows: related work, materials and methods used, experimental that describes experiments performed, results and discussion that present final results, and conclusion.

2 Related work

2.1 Background

Breaking away from the non-distributed (high-dimensional, sparse) representations of words, typical of traditional bag-of-words (BoW) or one-hot-encoding [1,13] created Word2Vec. Word2Vec consists of two shallow NN architectures: continuous skipgram and CBoW, as shown in Figure 1. It uses distributed (low-dimensional, dense) representations of words that group similar words. This model traded the complexity of deep NN architectures for more efficient training over large corpora. It has two training algorithms: negative sampling and hierarchical softmax [14]. The publicly released model was trained on Google news dataset of 100 billion words (BWs). Implementations of the model have been undertaken in the programming languages: Python and C++, though the original was done in C [15]. The Python implementation is slower to train, being an interpreted language [16,17].

Continuous skipgram predicts (by maximizing classification of) words before and after the centre word, for a given range. Since distant words are less connected to a centre word in a sentence, less weight is assigned to such distant words in training. CBoW, on the other hand, uses words from the history and future in a sequence, with the objective of correctly classifying the target word in the middle. It works by projecting all history or future words within a chosen window into the same position,

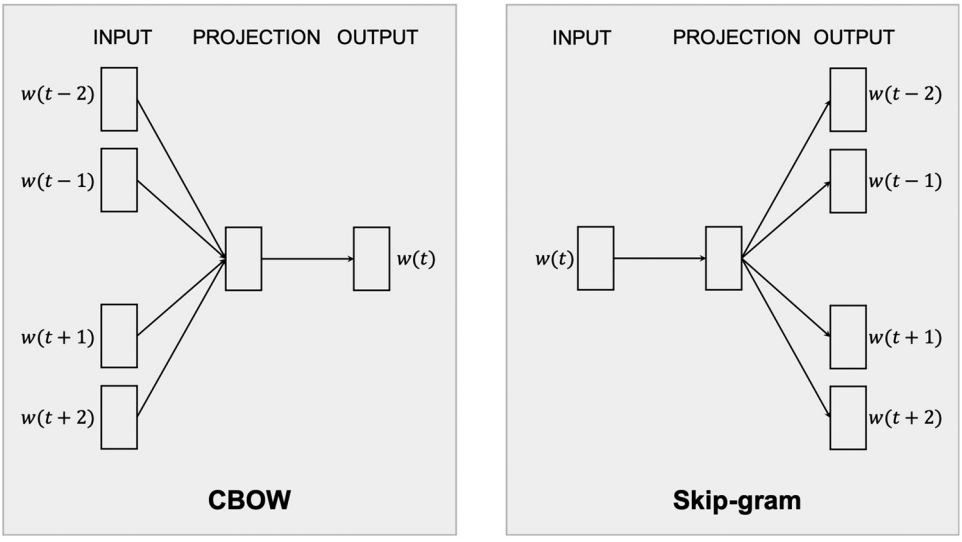

Figure 1: The CBoW and continuous skipgram model architectures [1].

averaging their vectors. Hence, the order of words in the history or future does not influence the averaged vector. This is similar to the traditional BoW. A log-linear classifier is used in both architectures [1]. In further work, Mikolov et al. [14] extended the model to be able to do phrase representations and subsample frequent words. Earlier models like latent Dirichlet allocation and latent semantic analysis effectively achieve low-dimensional vectors by matrix factorization [10,18].

2.2 Previous work

Recent SoTA architectures like BERT and Robustly optimized BERT approach [2,3], which use embedding layer, are used for sequence tagging, such as NER. Huang et al. [19] used the bi-directional long short-term memory network (Bi-LSTM) and other variants, which is what we use in this work. Embeddings have been shown to be beneficial for NLP tasks [13]. Mikolov et al. [1] showed with vector space algebra that relationships among words can be evaluated, expressing the quality of vectors produced from the model. The popular semantic example: $vector(\text{“}King\text{”}) - vector(\text{“}Man\text{”}) + vector(\text{“}Woman\text{”}) \approx vector(\text{“}Queen\text{”})$ can be verified using cosine distance. Dimensionality reduction algorithms (such as principal component analysis and t-distributed Stochastic Neighbor Embedding may be used to visualize the relationship among a set of words [20]. Such algorithms work by reducing the dimensionality of the embeddings further. WordSimilarity-353 (WordSim) test set is another analysis tool for word vectors [21]. Unlike the analogy metric, which is based on vector space algebra, WordSim is based on human expert-assigned semantic similarity on two sets of English word pairs. Both tools measure embedding quality, with a scaled score of 1 being the highest (i.e. very similar or exact).

Similar to word embeddings, subword representations have proven to be helpful when dealing with out-of-vocabulary (OOV) words and [22] used such embeddings to guide the parsing of OOV words in their work on meaning representation for robots. Intrinsic tests, in the form of word similarity or analogy tests, reveal meaningful relations among words in embeddings [1,23]. However, it is inappropriate to assume that such intrinsic tests are sufficient in themselves, just as it is inappropriate to assume that one particular downstream test is sufficient to generalize the performance of embeddings on all NLP tasks [24–26]. Furthermore, word embeddings display biases (as one of their shortcomings) seen in the data they are trained on [27].

Mikolov et al. [1] tried various hyperparameters with both architectures of the Word2Vec model, ranging from 50 to 1,000 dimensions, 30,000 to 3,000,000 vocabulary sizes and 1 to 3 epochs, among others. They observed diminishing returns after a certain point, despite additional dimensions or larger, unstructured training data. However, quality increased when both dimensions and data size were increased together. Although they pointed out that the choice of optimal hyperparameter configurations depends on the NLP problem at hand, they identified the most important factors as architecture, dimension size, subsampling rate, and the window size. In addition, it has been observed that larger datasets improve the quality of word vectors and, potentially, performance on downstream tasks [1,28]

3 Materials and methods

3.1 Datasets

The corpora used for word embeddings are the 2019 English Wiki News Abstract of about 15M by Wikipedia [29], 2019 English Simple Wiki (SW) articles of about 711M by Wikipedia [30], and the BW of 3.9G by Chelba et al. [31]. The corpus used for SA is the training set of the internet movie dataset (IMDB) of movie reviews by Maas et al. [32]. It has a total of 25,000 sentences with half being positive sentiments and the other half being negative sentiments. The training set is what was available with the ground truth. The dataset for NER is the Groningen meaning bank (GMB) by Bos et al. [33], containing 47,959 samples. It has 17 labels, with 9 main labels and 2 context tags. Google (semantic and syntactic) analogy test set by Mikolov et al. [1] and WordSimilarity-353 (with Spearman correlation) by Finkelstein et al. [21] were chosen for intrinsic evaluations.

3.2 Embeddings

The hyperparameters tuned in a grid search for the embeddings are given in Table 1. The models were trained on a shared cluster running Ubuntu 16 with 32 CPUs of 32x Intel Xeon 4110 at 2.1 GHz. Gensim [15] Python library implementation of Word2Vec was used. This is because of its relative stability, popular support and to minimize the time required in writing and testing a new implementation in Python from scratch. Our models are available

Table 1: Upstream hyperparameter choices (notations based on Gensim library convention)

Hyper-parameter	Values
Dimension size	300, 1,200, 1,800, 2,400, 3,000
Window size (w)	4, 8
Architecture	Skipgram (s1), CBoW (s0)
Algorithm	H. Softmax (h1), N. Sampling (h0)
Epochs	5, 10

for confirmation and the source codes are available on Github.[1]

3.3 Downstream architectures

The downstream experiments were run on a Tesla GPU on a shared DGX cluster running Ubuntu 18. Pytorch deep learning framework was used. A long short-term memory network (LSTM) was trained on the GMB dataset for NER. A Bi-LSTM network was trained on the IMDB dataset for SA. The Bi-LSTM includes an additional hidden linear layer before the output layer (Figures 2 and 3). Hyperparameter details of the two networks for the downstream tasks are given in Table 2. The metrics for extrinsic evaluation include *F*1, precision, recall and accuracy (for SA).

4 Experimental

In our work, we extended research to 3,000 dimensions and epochs of 5 and 10. To form the vocabulary for the embeddings, words occurring less than 5 times in the corpora were dropped, stop words removed using the natural language toolkit [34] and additional data pre-processing carried out. Table 1 describes the hyperparameters explored for each dataset and notations used. In all, 80 runs per data were conducted for both the Wiki News Abstract and the SW. Experiments for all combinations for 300 dimensions were conducted on the BW corpus and additional runs for the window size 8 + skipgram (s1) + hierarchical softmax (h1) combination to verify the trend of quality of word vectors as dimensions are increased.

Preferably, more than one training instance would have been run per combination for each embedding and an average taken; however, the long hours involved made this prohibitive. Despite this, we randomly ran a few combinations more than once and confirmed the difference in intrinsic scores was negligible. For both

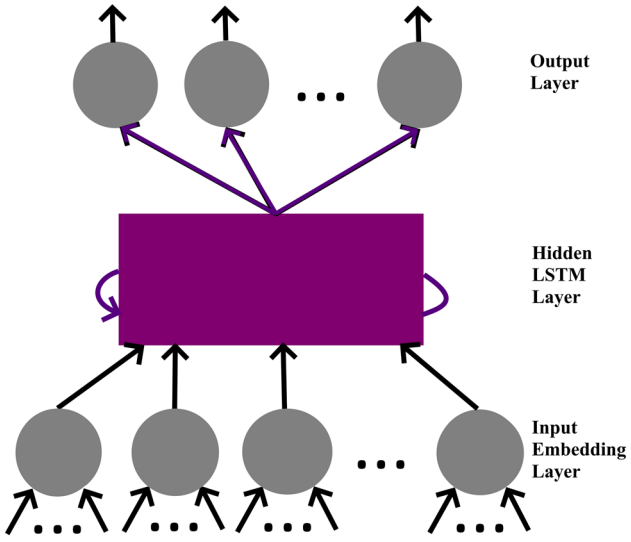

Figure 2: Network architecture for NER.

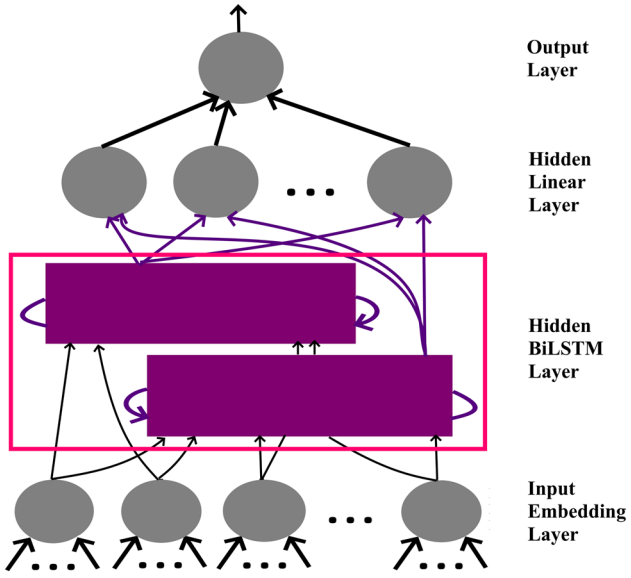

Figure 3: Network architecture for SA.

Table 2: Downstream network hyperparameters

Archi	Epochs	Hidden Dim	LR	Loss
LSTM	40	128	0.01	Cross entropy
BiLSTM	20	128* 2	0.0001	BCELoss

1 github.com/tosingithub/sdesk.

Table 3: Scores for 300 dimensions for ten epochs for SW, BW, and GoogleNews corpora

Data/Metrics	w8s1h1	w8s0h1	w8s0h0	w8s1h0	w4s1h1	w4s0h1	w4s0h0	w4s1h0
SW								
Analogy	0.461	0.269	**0.502**	0.439	0.446	0.243	0.478	0.407
WordSim score1	0.636	0.611	0.654	**0.655** *	0.635	0.608	0.620	0.635
Spearman	0.670	0.648	0.667	**0.695** *	0.668	0.648	0.629	0.682
BW								
Analogy	0.587	0.376	0.638	0.681	0.556	0.363	0.629	**0.684**
WordSim score1	0.614	0.511	0.599	**0.644**	0.593	0.508	0.597	0.635
Spearman	0.653	0.535	0.618	**0.681**	0.629	0.527	0.615	0.677
GoogleNews - 100B (s1h0)								
Analogy: 0.740*			WordSim score1: 0.624			Spearman: 0.659		

w: window size, s1: skipgram, s0: CBoW, h1: hierarchical softmax, h0: negative sampling | notations are based on Gensim convention |* shows best values. $p < 0.0001$.

downstream tasks, we evaluated the default Pytorch embedding, the pre-trained embedding by Mikolov et al. [1] (based on GoogleNews of 100B tokens) and ours. In each case, the dataset was shuffled before training and split in the ratio 70:15:15 for training, dev, and test sets. Batch size of 64 was used, and Adam was used as an optimizer. For each task, each experiment was conducted four times and an average value calculated.

Table 4: Embedding training and loading time

Model	Training (h)	Loading time (s)
SW w8s1h0	5.44	1.93
BW w8s1h1	27.22	4.89
GoogleNews [1]	—	97.73

w: window size, s1: skipgram, h1: hierarchical softmax, h0: negative sampling.

5 Results and discussion

We obtain comparable accuracy in SA to ref. [32], though we used less than half of the dataset for training and also evaluated on a smaller different set. The WordSim output file from the Gensim library always has more than one value reported, including the Spearman correlation. The first value is reported as WordSim score1 in the relevant table. It is a cosine similarity variant. Table 3 summarizes key results from the intrinsic evaluations for 300 dimensions.[2] Table 4 reveals the training time and average embedding loading time (in seconds), representative of the various models used. Tables 5 and 6 summarize key results for the extrinsic evaluations. Figures 4 and 5 present the line graph of SW for the eight hyperparameter combinations for different dimension sizes and the score trend for SW and BW corpora over several dimension sizes, respectively. Results for the smallest dataset (Wiki Abstract) are so poor, because of the tiny file size (15M), and there is no reason for reporting them here. Hence, we have focused on results from the SW and BW corpora.

From Table 3, we observe that there is a general better performance of the hyperparameter combination: skipgram-negative sampling (s1h0). Indeed, the embedding with the highest analogy score is the one by Mikolov et al. [1]; however, the highest WordSim score1 and Spearman correlation values are obtained by the skipgram-negative sampling embedding of window size 8 of the SW. The analogy score of the GoogleNews embedding, which we tested, is confirmed in the original article [1]. It is based on a vocabulary size of 3,000,000 words while SW has a vocabulary size of 367,811. Furthermore, it can be observed from Table 4 that the training and the loading times for the embedding of the SW are the lowest. Information on the training time for the released GoogleNews model [1] is not readily available. The other two embeddings take multiple times longer to train or load. The marginal gain of the GoogleNews or BW embedding in analogy score over that of the SW is not directly proportional to the length of time required to train or load them. This may put the embeddings based on the larger corpora at a disadvantage in cases where factors like the training time and the energy spent on training are critical. Energy-saving and time-saving methods are crucial for the environment and humanity as a whole.

2 The results are to three decimal places.

Table 5: NER Dev and test set mean results

Metric	Default Dev, Test	100B Dev, Test	w8 s0 h0 Dev, Test	w8 s1 h0 Dev, Test	BW w4 s1 h0 Dev, Test
*F*1	0.661, 0.661	**0.679**, 0.676	0.668, 0.669	0.583, 0.676	**0.679**, **0.677** *
Precision	0.609, 0.608	**0.646**, **0.642**	0.636, 0.637	0.553, 0.642	0.644, **0.642**
Recall	0.723, **0.724**	0.716, 0.714	0.704, 0.706	0.618, 0.715	0.717, 0.717

w: window size, s1: skipgram, s0: CBoW, h0: negative sampling. $p < 0.0001$.

Table 6: SA Dev and test set mean results

Metric	Default Dev, Test	100B Dev, Test	w8 s0 h0 Dev, Test	w8 s1 h0 Dev, Test	BW w4 s1 h0 Dev, Test
*F*1	**0.810**, **0.805** *	0.384, 0.386	0.798, 0.799	0.548, 0.553	0.498, 0.390
Precision	0.805, 0.795	0.6, 0.603	**0.814**, **0.811**	0.510, 0.524	0.535, 0.533
Recall	**0.818**, **0.816**	0.303, 0.303	0.788, 0.792	0.717, 0.723	0.592, 0.386
Accuracy	**0.807**, **0.804** *	0.549, 0.55	0.801, 0.802	0.519, 0.522	0.519, 0.517

w: window size, s1: skipgram, s0: CBoW, h0: negative sampling. $p < 0.0001$.

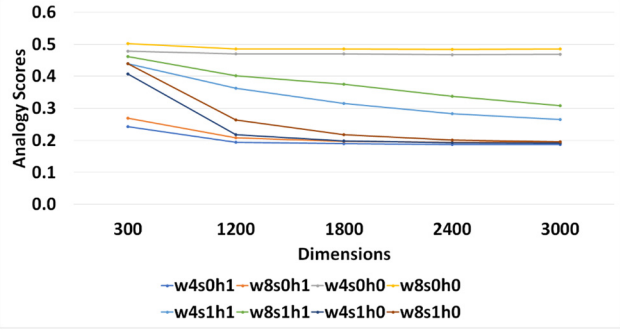

Figure 4: SW: Analogy scores for ten Epochs (w: window size, s1: skipgram, s0: CBoW, h1: hierarchical softmax, h0: negative sampling).

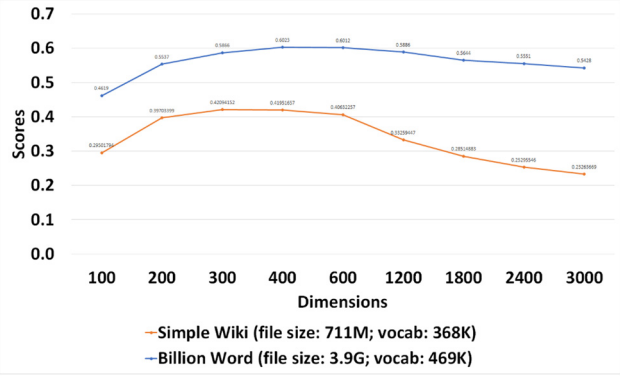

Figure 5: Analogy scores for w4s1h1 of SW for five Epochs and w8s1h1 of BW for ten epochs (not drawn to scale from 400).

We observe from Figure 4 that analogy scores for all the embeddings start to drop after 300 dimensions for the SW dataset. The rate of decrease, however, differs for each embedding. Also, from Figure 5, we note that from 100 dimensions, scores improve but start to drop after over 300 dimensions for SW and after over 400 dimensions for BW, confirming a similar observation by Mikolov et al. [1]. This trend is true for all the combinations. Polynomial interpolation may be used to determine the optimal dimension in both corpora.

For the NER downstream task, our BW skipgram-negative sampling of window size 4 (w4s1h0) embedding performs best in *F*1 score. This embedding is also the best in analogy score among our models. Most pretrained embeddings outperform the default Pytorch embedding in *F*1 scores. With regards to SA, the default Pytorch embedding outperforms the pretrained embeddings in accuracy and *F*1 scores but is closely followed by our CBoW-negative sampling embedding (w8s0h0) of the SW. The combination of CBoW-negative sampling for both window sizes 4 and 8 (w8s0h0 and w4s0h0) of the SW performs relatively well in both the extrinsic tasks.

For alpha of 0.05 and power of 1, significance tests of the differences of two means for the *F*1 scores give *p*-values of 0 in both cases: for the 100B and the skipgram-negative sampling (w4s1h0) of the BW embedding for NER, and the CBoW-negative sampling (w8s0h0) for the SW for SA. Since both *p*-values are 0, the differences

are unlikely due to chance, so we reject the null hypothesis in both cases.

6 Conclusion

This work analyses, empirically, relatively optimal combinations of hyperparameters for embeddings, specifically for Word2Vec. It further shows that for downstream tasks, such as NER and SA, there is no silver bullet! However, some combinations show strong performance across tasks. Performance of embeddings appears to be task-specific and high analogy scores do not necessarily correlate positively with performance on downstream tasks. This point on correlation is somewhat similar to results by Chiu et al. [35] and Wang et al. [12]. It was discovered that increasing embedding dimension size depreciates performance after a point. If strong considerations of saving time, energy, and the environment are made, then relatively smaller corpora may suffice or even be better in some cases. Indeed, hyperparameter choices are very important in NNs [10].

Future work that may be investigated are the performance of other architectures of word or subword embeddings in SoTA networks like BERT (based on a matrix of hyper-parameters), the performance and comparison of embeddings applied to other less-explored languages, and how these embeddings perform in other downstream tasks.

Acknowledgements: The authors wish to thank the anonymous reviewers for their valuable contributions.

Funding information: This work was supported partially by Vinnova under the project number 2019-02996 "Språkmodeller för svenska myndigheter." They, however, had no involvement in any stage of this work, including study design, interpretation of data, and report writing.

Conflict of interest: Authors state no conflict of interest.

Data availability statement: The data used in this study are available publicly for free, as provided in the reference section.

References

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

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2018, arXiv: http://arXiv.org/abs/arXiv:1810.04805.

[3] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, et al., *Roberta: a robustly optimized bert pretraining approach*, 2019, arXiv: http://arXiv.org/abs/arXiv:1907.11692.

[4] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, et al., *Exploring the limits of transfer learning with a unified text-to-text transformer*, 2019, arXiv: http://arXiv.org/abs/arXiv:1910.10683.

[5] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," In: *International Conference on Learning Representations*, ICLR, 2015.

[6] M. Längkvist, L. Karlsson, and A. Loutfi, "A review of unsupervised feature learning and deep learning for time-series modeling," *Pattern Recogn. Lett.*, vol. 42, pp. 11–24, 2014.

[7] B. Dhingra, H. Liu, R. Salakhutdinov, and W. W. Cohen, *A comparative study of word embeddings for reading comprehension*, 2017, arXiv: http://arXiv.org/abs/arXiv:1703.00993.

[8] M. Naili, A. H. Chaibi, and H. H. BenGhezala, "Comparative study of word embedding methods in topic segmentation," *Proc. Comput. Sci.*, vol. 112, pp. 340–349, 2017.

[9] Y. Wang, S. Liu, N. Afzal, M. Rastegar-Mojarad, L. Wang, F. Shen, et al., "A comparison of word embeddings for the biomedical natural language processing," *J. Biomed. Inform.*, vol. 87, pp. 12–20, 2018.

[10] O. Levy, Y. Goldberg, and I. Dagan, "Improving distributional similarity with lessons learned from word embeddings," *Trans. Assoc. Comput. Linguist.*, vol. 3, pp. 211–225, 2015.

[11] Y. Zhang, Q. Chen, Z. Yang, H. Lin, and Z. Lu, "Biowordvec, improving biomedical word embeddings with subword information and mesh," *Scientif. Data*, vol. 6, no. 1, pp. 1–9, 2019.

[12] B. Wang, A. Wang, F. Chen, Y. Wang, and C. C. Jay Kuo, "Evaluating word embedding models: methods and experimental results," *APSIPA Trans. Signal Inform. Process*, vol. 8, 2019. doi: 10.1017/ATSIP.2019.12.

[13] J. Turian, L. Ratinov, and Y. Bengio, "Word representations: a simple and general method for semi-supervised learning," In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2010, pp. 384–394.

[14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," In: *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2013, pp. 3111–3119.

[15] 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*, ELRA, Valletta, Malta, May 2010, pp. 45–50. http://is.muni.cz/publication/884893/en.

[16] T. P. Adewumi, "Inner loop program construct: A faster way for program execution," *Open Comput. Sci.*, vol. 8, no. 1, pp. 115–122, 2018.

[17] T. P. Adewumi and M. Liwicki, "Inner for-loop for speeding up blockchain mining," *Open Comput. Sci.*, vol. 10, no. 1, pp. 42–47, 2019. doi: 10.1515/comp-2020-0004.

[18] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *J. Am. Soc. Inform. Sci.*, vol. 41, no. 6, pp. 391–407, 1990.
[19] Z. Huang, W. Xu, and K. Yu, *Bidirectional lstm-crf models for sequence tagging*, 2015, arXiv: http://arXiv.org/abs/arXiv:1508.01991.
[20] G. E. Hinton and S. Roweis, "Stochastic neighbor embedding," *Adv. Neural Inform. Process. Sys.*, MIT Press, vol. 15, 2002.
[21] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, et al., "Placing search in context: The concept revisited," *ACM Trans. Inform. Sys.*, vol. 20, no. 1, pp. 116–131, 2002.
[22] J. Thomason, A. Padmakumar, J. Sinapov, N. Walker, Y. Jiang, H. Yedidsion, et al., "Jointly improving parsing and perception for natural language commands through human-robot dialog," *J. Artif. Intell. Res.*, vol. 67, pp. 327–374, 2020.
[23] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1532–1543. doi: 10.3115/v1/D14-1162.
[24] A. Gatt and E. Krahmer, "Survey of the state of the art in natural language generation: Core tasks, applications and evaluation," *J. Artif. Intell. Res.*, vol. 61, pp. 65–170, 2018.
[25] T. P. Adewumi, F. Liwicki, and M. Liwicki, *Corpora compared: The case of the Swedish Gigaword & Wikipedia Corpora*, 2020, arXiv: http://arXiv.org/abs/arXiv:2011.03281.
[26] T. P. Adewumi, F. Liwicki, and M. Liwicki, *The challenge of diacritics in Yoruba embeddings*, 2020, arXiv: http://arXiv.org/abs/arXiv:2011.07605.
[27] T. Bolukbasi, K.-W. Chang, J. Y. Zou, V. Saligrama, and A. T. Kalai, "Man is to computer programmer as woman is to homemaker? debiasing word embeddings," In: *Advances in Neural Information Processing Systems*, Barcelona, Spain, 2016, pp. 4349–4357.
[28] T. P. Adewumi, F. Liwicki, and M. Liwicki, "Conversational systems in machine learning from the point of view of the philosophy of science using alime chat and related studies," *Philosophies*, vol. 4, no. 3, p. 41.
[29] Wikipedia, Wiki news abstract, 2019.
[30] Wikipedia, Simple wiki articles, 2019.
[31] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn, et al., *"One billion word benchmark for measuring progress in statistical language modeling,"* Technical report, Google, 2013.
[32] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-volume 1*, Association for Computational Linguistics, 2011, pp. 142–150.
[33] J. Bos, V. Basile, K. Evang, N. J. Venhuizen, and J. Bjerva, "The Groningen meaning bank," In: *Handbook of Linguistic Annotation*, Springer, 2017, pp. 463–496
[34] E. Loper and S. Bird, *Nltk: the natural language toolkit*, 2002, arXiv: http://arXiv.org/abs/cs/0205028.
[35] B. Chiu, A. Korhonen, and S. Pyysalo, "Intrinsic evaluation of word vectors fails to predict extrinsic performance," In: *Proceedings of the 1st Workshop on Evaluating Vector-space Representations for NLP*, Association for Computational Linguistics, Berlin, Germany, 2016, pp. 1–6.