Pattern Recognition Letters
journal homepage: www.elsevier.com

Word2Vec: optimal hyper-parameters and their impact on NLP downstream tasks

Tosin P. Adewumi[a,**], Foteini Liwicki[b], Marcus Liwicki[b]

[a] *SRT Department, EISLAB, Lule University of Technology, 97187, Sweden*
[b] *firstname.lastname@ltu.se — SRT Department, EISLAB, Lule University of Technology, 97187, Sweden*

ABSTRACT

Word2Vec is a prominent model for natural language processing (NLP) tasks. Similar inspiration is found in distributed embeddings for new state-of-the-art (SotA) deep neural networks. However, wrong combination of hyper-parameters can produce poor quality vectors. The objective of this work is to empirically show optimal combination of hyper-parameters exists and evaluate various combinations. We compare them with the released, pre-trained original word2vec model. Both intrinsic and extrinsic (downstream) evaluations, including named entity recognition (NER) and sentiment analysis (SA) were carried out. The downstream tasks reveal that the best model is usually task-specific, high analogy scores don't necessarily correlate positively with F1 scores and the same applies to focus on data alone. Increasing vector dimension size after a point leads to poor quality or performance. If ethical considerations to save time, energy and the environment are made, then reasonably smaller corpora may do just as well or even better in some cases. Besides, using a small corpus, we obtain better human-assigned WordSim scores, corresponding Spearman correlation and better downstream performances (with significance tests) compared to the original model, trained on 100 billion-word corpus.

Keywords: Word2Vec, NLP, Named Entity Recognition, Sentiment Analysis, Hyperparameters

1. Introduction

There have been many implementations of the word2vec model in either of the two architectures it provides: continuous skipgram and continuous bag-of-words (CBoW) (Mikolov et al., 2013a). Similar distributed models of word or subword embeddings (or vector representations) find usage in SotA, deep neural networks like bidirectional encoder representations from transformers (BERT) and its successors (Devlin et al., 2018; Liu et al., 2019; Raffel et al., 2019). BERT generates contextual representations of words after been trained for extended periods on large corpora, unsupervised, using the attention mechanisms (Vaswani et al., 2017). Unsupervised learning provide feature representations using large unlabelled corpora (Längkvist et al., 2014).[1]

It has been observed that various hyper-parameter combinations have been used in different research involving word2vec, after its release, with the possibility of many of them being sub-optimal (Dhingra et al., 2017; Naili et al., 2017; Wang et al., 2018). Therefore, the authors seek to address the research question: what is the optimal combination of word2vec hyper-parameters for intrinsic and extrinsic NLP purposes? There are astronomically high numbers of combinations of hyper-parameters possible for neural networks, even with just a few layers (Levy et al., 2015). Hence, the scope of our extensive, technical work over three corpora is on dimension size, training epochs, window size and vocabulary size for the training algorithms (hierarchical softmax and negative sampling) of both skipgram and CBoW.

The objective of this work is to determine the optimal combinations of word2vec hyper-parameters for intrinsic evaluation (semantic and syntactic analogies) and extrinsic evaluation tasks (Zhang et al., 2019; Wang et al., 2019). It is not our objective in this work to set new SotA results. Some main con-

[**] Corresponding author
e-mail: `tosin.adewumi@ltu.se` (Tosin P. Adewumi)

[1] This paper is under consideration at Pattern Recognition Letters

tributions of this research are the empirical establishment of optimal combinations of word2vec hyper-parameters for NLP tasks, discovering the behaviour of quality of vectors vis-a-vis increasing dimensions and the confirmation of embeddings being task-specific for the downstream. The rest of this paper is organised as follows: materials and methods used, experimental that describes experiments performed, results and discussion that present final results, and conclusion.

1.1. Related Work

Breaking away from the non-distributed (high-dimensional, sparse) representations of words, typical of traditional bag-of-words or one-hot-encoding (Turian et al., 2010), Mikolov et al. (2013a) created word2vec. Word2Vec consists of two shallow neural network architectures: continuous skipgram and CBoW. It uses distributed (low-dimensional, dense) representations of words that group similar words. This new model traded the complexity of deep neural network architectures, by other researchers, for more efficient training over large corpora. Its architectures have two training algorithms: negative sampling and hierarchical softmax (Mikolov et al., 2013b). The released model was trained on Google news dataset of 100 billion words. Implementations of the model have been undertaken by researchers in the programming languages Python and C++, though the original was done in C (Řehůřek and Sojka, 2010).

Continuous skipgram predicts (by maximizing classification of) words before and after the center word, for a given range. Since distant words are less connected to a center 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, 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 bag-of-words. A log-linear classifier is used in both architectures (Mikolov et al., 2013a). In further work, they extended the model to be able to do phrase representations and subsample frequent words (Mikolov et al., 2013b). Earlier models like latent dirichlet allocation (LDA) and latent semantic analysis (LSA) exist and effectively achieve low dimensional vectors by matrix factorization (Deerwester et al., 1990; Levy et al., 2015).

It's been shown that word vectors are beneficial for NLP tasks (Turian et al., 2010), such as SA and NER. Besides, Mikolov et al. (2013a) showed with vector space algebra that relationships among words can be evaluated, expressing the quality of vectors produced from the model. The famous, semantic example: *vector("King") - vector("Man") + vector("Woman") ≈ vector("Queen")* can be verified using cosine distance. Syntactic relationship examples include plural verbs and past tense, among others. WordSimilarity-353 (WordSim) test set is another analysis tool for word vectors (Finkelstein et al., 2002). Unlike Google analogy score, 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 score of 1 being the highest (very much similar or exact, in Google analogy case).

Mikolov et al. (2013a) tried various hyper-parameters with both architectures of their model, ranging from 50 to 1,000 dimensions, 30,000 to 3,000,000 vocabulary sizes, 1 to 3 epochs, among others. In our work, we extended research to 3,000 dimensions and 5 and 10 epochs. Different observations were noted from the many trials. 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 choice of optimal hyper-parameter 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 (Adewumi et al., 2019; Mikolov et al., 2013a) .

2. Materials and methods

2.1. Datasets

The corpora used for word embeddings are the 2019 English Wiki News Abstract by Wikipedia (2019b) of about 15MB, 2019 English Simple Wiki (SW) Articles by Wikipedia (2019a) of about 711MB and the Billion Word (BW) of 3.9GB by Chelba et al. (2013). The corpus used for sentiment analysis is the internet movie database (IMDb) of movie reviews by Maas et al. (2011) while that for NER is Groningen Meaning Bank (GMB) by Bos et al. (2017), containing 47,959 sentence samples. The IMDb dataset used has a total of 25,000 sentences with half being positive sentiments and the other half being negative sentiments. The GMB dataset has 17 labels, with 9 main labels and 2 context tags. Google (semantic and syntactic) analogy test set by Mikolov et al. (2013a) and WordSimilarity-353 (with Spearman correlation) by Finkelstein et al. (2002) were chosen for intrinsic evaluations.

2.2. Embeddings

The models were generated in a shared cluster running Ubuntu 16 with 32 CPUs of 32x Intel Xeon 4110 at 2.1GHz. Gensim (Řehůřek and Sojka, 2010) 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 for confirmation and source codes are available on github.[2]

2.3. 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 BiLSTM network was trained on

[2] https://github.com/tosingithub/sdesk

Table 1. Upstream hyper-parameter choices

Hyper-parameter	Values
Dimension size	300, 1200, 1800, 2400, 3000
Window size (w)	4, 8
Architecture	Skipgram (s1), CBoW (s0)
Algorithm	H. Softmax (h1), N. Sampling (h0)
Epochs	5, 10

the IMDb dataset for SA. The BiLSTM includes an additional hidden linear layer before the output layer. Hyper-parameter details of the two networks for the downstream tasks are given in table 2. The metrics for extrinsic evaluation include F1, precision, recall and accuracy scores (in the case of SA).

Table 2. Downstream network hyper-parameters

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

3. Experimental

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 (NLTK) (Loper and Bird, 2002) and additional data pre-processing carried out. Table 1 describes most hyper-parameters explored for each dataset and notations used. In all, 80 runs (of about 160 minutes) were conducted for the 15MB Wiki Abstract dataset with 80 serialized models totaling 15.136GB while 80 runs (for over 320 hours) were conducted for the 711MB SW dataset, with 80 serialized models totaling over 145GB. Experiments for all combinations for 300 dimensions were conducted on the 3.9GB training set of the BW corpus and additional runs for other dimensions for the window 8 + skipgram + heirarchical softmax 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 a model 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 were negligible.

For both downstream tasks, the default Pytorch embedding was tested before being replaced by the original (100B) pretrained embedding 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 as optimizer. For each task, experiments for each embedding was conducted four times and an average value calculated.

4. Results and Discussion

Table 3 summarizes key results from the intrinsic evaluations for 300 dimensions[3]. Table 4 reveals the training time

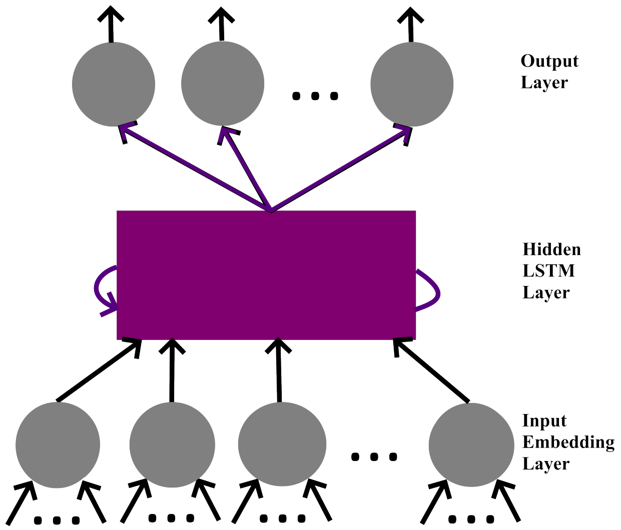

Fig. 1. Network architecture for NER

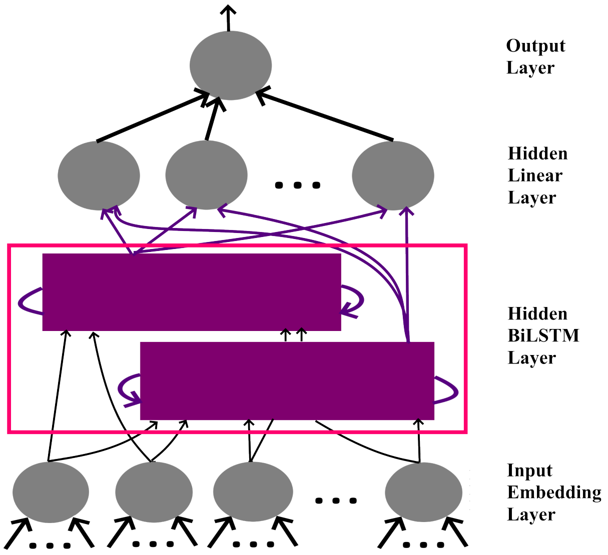

Fig. 2. Network architecture for SA

[3] The results are to 3 decimal places

(in hours) 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 3, 4, 5, 6 and 7 present line graph of the eight combinations for different dimension sizes for SW, trend of SW and BW corpora over several dimension sizes, analogy score comparison for models across datasets, NER mean F1 scores on the GMB dataset and SA mean F1 scores on the IMDb dataset, respectively. Results for the smallest dataset (Wiki Abstract) are so poor, because of the tiny file size (15MB), there's no reason reporting them here. Hence, we have focused on results from the SW and BW corpora.

Best combination in terms of analogy sometimes changes when corpus size increases, as will be noticed from table 3. In terms of analogy score, for 10 epochs, w8s0h0 performs best while w8s1h0 performs best in terms of WordSim and corresponding Spearman correlation for SW. Meanwhile, increasing the corpus size to BW, w4s1h0 performs best in terms of analogy score while w8s1h0 maintains its position as the best in terms of WordSim and Spearman correlation. Besides, considering quality metrics, it can be observed from table 4 that comparative ratio of values between the models is not commensurate with the results in intrinsic or extrinsic values, especially when we consider the amount of time and energy spent, since more training time results in more energy consumption (Adewumi and Liwicki, 2019).

Information on the length of training time for the original 100B model is not readily available. However, it's interesting to note that it's skipgram-negative sampling (s1h0). Its analogy score, which we tested and report, is confirmed in the original paper (Mikolov et al., 2013a). It beats our best models in only analogy score (even for SW), performing worse in others. This is inspite of using a much bigger corpus of 3,000,000 vocabulary size and 100 billion words while SW had vocabulary size of 367,811 and is 711MB. It is very likely our analogy scores will improve when we use a much larger corpus, as can be observed from table 3, which involves just one billion words.

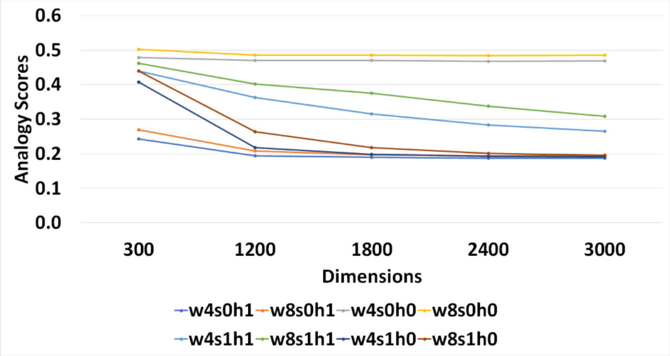

Fig. 3. Simple Wiki: Analogy Scores for 10 Epochs (color needed)

With regards to increasing dimension, although the two best combinations in analogy (w8s0h0 & w4s0h0) for SW, as shown in fig. 3, decreased only slightly compared to others, the increased training time and much larger serialized model size render any possible minimal score advantage over higher dimensions undesirable. As can be observed in fig. 4, from 100 dimensions, scores improve but start to drop after over 300 dimensions for SW and after over 400 dimensions for BW, confirming the observation by Mikolov et al. (2013a). This trend is true for all combinations for all tests. Polynomial interpolation may be used to determine the optimal dimension in both corpora.

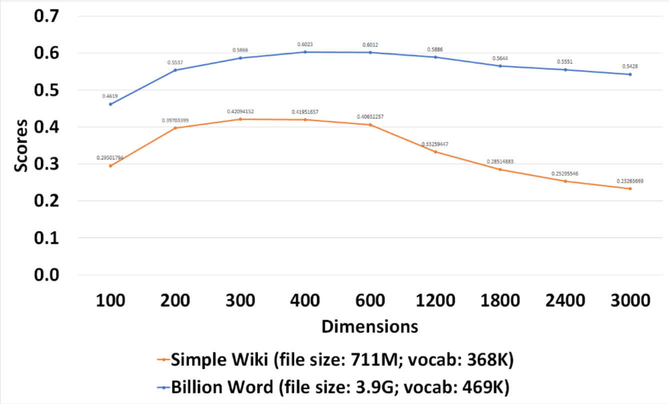

Fig. 4. Analogy Scores for w4s1h1 of SW for 5 Epochs & w8s1h1 of BW for 10 epochs (not drawn to scale from 400) (color needed)

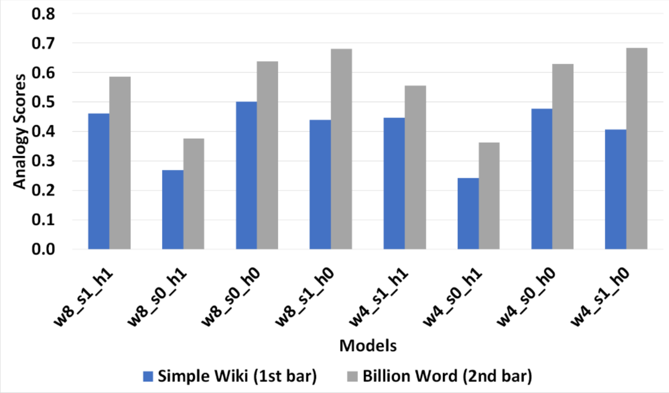

Fig. 5. Comparison of 300 dimension models for 10 epochs for SW & BW corpora

With regards to NER, most pretrained embeddings outperformed the default Pytorch embedding, with our BW w4s1h0 model (which is best in BW analogy score) performing best in F1 score and closely followed by the 100B model. On the other hand, with regards to SA, Pytorch embedding outperformed the pretrained embeddings but was closely followed by our SW w8s0h0 model (which also had the best SW analogy score). 100B performed second worst of all, despite originating from a very huge corpus. The combinations w8s0h0 & w4s0h0 of SW performed reasonably well in both extrinsic tasks, just as the default Pytorch embedding did.

Significance tests using bootstrap, based on Calmettes et al. (2012), on the results of the differences in means of the 100B & BW w4s1h0 models for NER shows a 95% confidence interval (CI) of [-0.008, 0.01] but [0.274, 0.504] for 100B & SW w8s0h0 for SA. The CI interval for NER includes 0, thus we can conclude the difference was likely due to chance and accept the null hypothesis but the CI for SA does not include 0,

Table 3. Scores for 300 dimensions for 10 epochs for SW, BW & 100B corpora.

	w8s1h1	w8s0h1	w8s0h0	w8s1h0	w4s1h1	w4s0h1	w4s0h0	w4s1h0
Simple Wiki								
Analogy	0.461	0.269	**0.502**	0.439	0.446	0.243	0.478	0.407
WordSim	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
Billion Word								
Analogy	0.587	0.376	0.638	0.681	0.556	0.363	0.629	**0.684**
WordSim	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
Google News - 100B (s1h0)								
Analogy: 0.740			WordSim: 0.624			Spearman: 0.659		

Table 4. Training & embedding loading time for w8s1h0, w8s1h1 & 100B

Model	Training (hours)	Loading Time (s)
SW w8s1h0	5.44	1.93
BW w8s1h1	27.22	4.89
GoogleNews (100B)	-	97.73

Table 5. NER Dev and Test sets Mean Results

Metric	Default	100B	w8 s0 h0	w8 s1 h0	BW w4 s1 h0
	Dev, Test	Dev, Test	Dev, Test	Dev, Test	Dev, Test
F1	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

Table 6. Sentiment Analysis Dev and Test sets Mean Results

Metric	Default	100B	w8 s0 h0	w8 s1 h0	BW w4 s1 h0
	Dev, Test	Dev, Test	Dev, Test	Dev, Test	Dev, Test
F1	**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

thus the difference is unlikely due to chance so we reject the null hypothesis.

5. Conclusions

This work analyses, empirically, optimal combinations of hyper-parameters for embeddings, specifically for word2vec. It further shows that for downstream tasks, like NER and SA, there's no silver bullet! However, some combinations show strong performance across tasks. Performance of embeddings is 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. (2016) and Wang et al. (2019). 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 reasonably smaller corpora may suffice or even be better in some cases. The on-going drive by many researchers to use ever-growing data to train deep neural networks can benefit from the findings of this work. Indeed, hyper-parameter choices are very important in neural network systems (Levy et al., 2015).

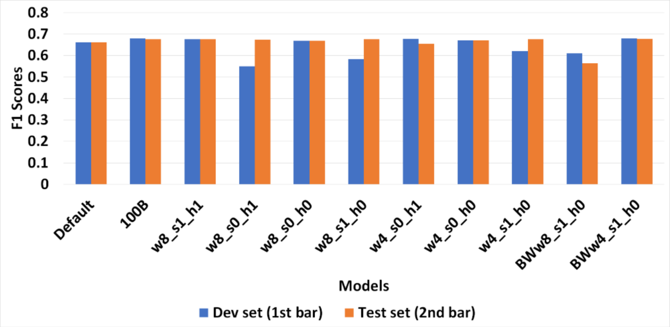

Fig. 6. Named Entity Recognition (NER) Mean F1 Scores on GMB Dataset

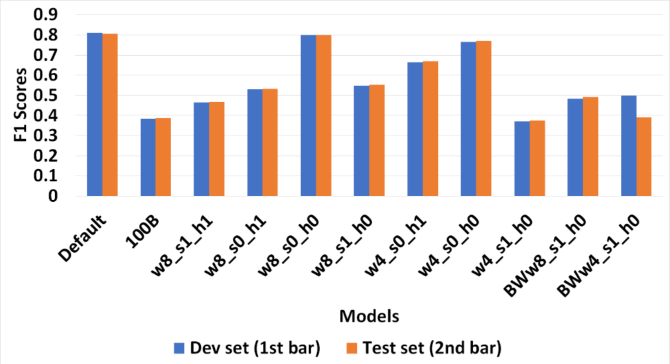

Fig. 7. Sentiment Analysis (SA) Mean F1 Scores on IMDb Dataset

Future work that may be investigated are performance of other architectures of word or sub-word 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. In addition, the actual reason for the changes, sometimes noticed, in intrinsic best model as corpus size increases is another task worth investigating.

Funding

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

References

Adewumi, T.P., Liwicki, F., Liwicki, M., 2019. Conversational systems in machine learning from the point of view of the philosophy of scienceusing alime chat and related studies. Philosophies 4, 41.

Adewumi, T.P., Liwicki, M., 2019. Inner for-loop for speeding up blockchain mining. Open Computer Science .

Bos, J., Basile, V., Evang, K., Venhuizen, N.J., Bjerva, J., 2017. The groningen meaning bank, in: Handbook of linguistic annotation. Springer, pp. 463–496.

Calmettes, G., Drummond, G.B., Vowler, S.L., 2012. Making do with what we have: use your bootstraps. Advances in physiology education 36, 177–180.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., Robinson, T., 2013. One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. Technical Report. Google. URL: `http://arxiv.org/abs/1312.3005`.

Chiu, B., Korhonen, A., Pyysalo, S., 2016. Intrinsic evaluation of word vectors fails to predict extrinsic performance, in: Proceedings of the 1st workshop on evaluating vector-space representations for NLP, pp. 1–6.

Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R., 1990. Indexing by latent semantic analysis. Journal of the American society for information science 41, 391–407.

Devlin, J., Chang, M.W., Lee, K., Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 .

Dhingra, B., Liu, H., Salakhutdinov, R., Cohen, W.W., 2017. A comparative study of word embeddings for reading comprehension. arXiv preprint arXiv:1703.00993 .

Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., Ruppin, E., 2002. Placing search in context: The concept revisited. ACM Transactions on information systems 20, 116–131.

Längkvist, M., Karlsson, L., Loutfi, A., 2014. A review of unsupervised feature learning and deep learning for time-series modeling. Pattern Recognition Letters 42, 11–24.

Levy, O., Goldberg, Y., Dagan, I., 2015. Improving distributional similarity with lessons learned from word embeddings. Transactions of the Association for Computational Linguistics 3, 211–225.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V., 2019. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 .

Loper, E., Bird, S., 2002. Nltk: the natural language toolkit. arXiv preprint cs/0205028 .

Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C., 2011. 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. pp. 142–150.

Mikolov, T., Chen, K., Corrado, G., Dean, J., 2013a. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 .

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J., 2013b. Distributed representations of words and phrases and their compositionality, in: Advances in neural information processing systems, pp. 3111–3119.

Naili, M., Chaibi, A.H., Ghezala, H.H.B., 2017. Comparative study of word embedding methods in topic segmentation. Procedia computer science 112, 340–349.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J., 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint arXiv:1910.10683 .

Řehůřek, R., Sojka, P., 2010. Software Framework for Topic Modelling with Large Corpora, in: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, ELRA, Valletta, Malta. pp. 45–50. `http://is.muni.cz/publication/884893/en`.

Turian, J., Ratinov, L., Bengio, Y., 2010. 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. pp. 384–394.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need, in: Advances in neural information processing systems, pp. 5998–6008.

Wang, B., Wang, A., Chen, F., Wang, Y., Kuo, C.C.J., 2019. Evaluating word embedding models: methods and experimental results. APSIPA Transactions on Signal and Information Processing 8.

Wang, Y., Liu, S., Afzal, N., Rastegar-Mojarad, M., Wang, L., Shen, F., Kingsbury, P., Liu, H., 2018. A comparison of word embeddings for the biomedical natural language processing. Journal of biomedical informatics 87, 12–20.

Wikipedia, 2019a. Simple wiki articles URL: `https://dumps.wikimedia.org/backup-index.html`.

Wikipedia, 2019b. Wiki news abstract URL: `https://dumps.wikimedia.org/backup-index.html`.

Zhang, Y., Chen, Q., Yang, Z., Lin, H., Lu, Z., 2019. Biowordvec, improving biomedical word embeddings with subword information and mesh. Scientific data 6, 1–9.