

Tosin P. Adewumi*, Foteini Liwicki, and Marcus Liwicki

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

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 the 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 WordSim scores, corresponding Spearman correlation and better downstream performances (with significance tests) compared to the original model, trained on a 100 billion-word corpus.

Keywords: 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) [1]. 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 [2, 3, 4]. BERT generates contextual representations of words after been trained for extended periods on large corpora, unsuper-

vised, using the attention mechanisms [5]. Unsupervised learning provides feature representations using large unlabelled corpora [6].

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 [7, 8, 9]. Therefore, the authors seek to address the research question: what is the optimal combination of word2vec hyper-parameters for intrinsic and extrinsic NLP purposes, specifically NER and SA? There are astronomically high numbers of combinations of hyper-parameters possible for neural networks, even with just a few layers [10]. Hence, the scope of our extensive, empirical work over three English 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 a few extrinsic evaluation tasks [11, 12]. It is not our objective in this work to set new SotA results. Some main contributions 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 performance being task-specific for the downstream. The rest of this paper is organised 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

Breaking away from the non-distributed (high-dimensional, sparse) representations of words, typical of traditional bag-of-words or one-hot-encoding [13], [1] 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

*Corresponding author: Tosin P. Adewumi, SRT Department, EISLAB, Luleå University of Technology, 97187, Sweden, E-mail: tosin.adewumi@ltu.se

Foteini Liwicki, SRT Department, EISLAB, Luleå University of Technology, 97187, Sweden, E-mail: foteini.liwicki@ltu.se

Marcus Liwicki, SRT Department, EISLAB, Luleå University of Technology, 97187, Sweden, E-mail: marcus.liwicki@ltu.se

architectures have two training algorithms: negative sampling and hierarchical softmax [14]. 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 [15]. The Python implementations are slower to train, being an interpreted language [16, 17].

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 [1]. In further work, they extended the model to be able to do phrase representations and subsample frequent words [14]. Earlier models like latent dirichlet allocation (LDA) and latent semantic analysis (LSA) exist and effectively achieve low dimensional vectors by matrix factorization [18, 10].

It's been shown that word vectors are beneficial for NLP tasks [13], such as SA and NER. Besides, [1] 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: $vector("King") - vector("Man") + vector("Woman") \approx vector("Queen")$ 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 [19]. 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 a scaled score of 1 being the highest (very much similar or exact, in Google analogy case).

Like word embeddings, subword representations have proven to be helpful when dealing with out-of-vocabulary (OOV) words and [20] used such embeddings to guide the parsing of OOV words in their work on meaning representation for robots. Despite their success, word embeddings display biases (as one of their shortcomings) seen in the data they are trained on [21]. Intrinsic tests, in the form of word similarity or analogy tests, reveal meaningful relations among words in embeddings, given the relationship

among words in context [1, 22]. However, it is inappropriate to assume such intrinsic tests are sufficient in themselves, just as it is inappropriate to assume one particular downstream test is sufficient to generalise the performance of embeddings on all NLP tasks [23, 24, 25].

[1] 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 epochs of 5 and 10. Different observations were noticed 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 [26, 1].

3 Materials and methods

3.1 Datasets

The corpora used for word embeddings are the 2019 English Wiki News Abstract by [27] of about 15MB, 2019 English Simple Wiki (SW) Articles by [28] of about 711MB and the Billion Word (BW) of 3.9GB by [29]. The corpus used for sentiment analysis is the internet movie database (IMDb) of movie reviews by [30] while that for NER is the Groningen Meaning Bank (GMB) by [31], 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 [1] and WordSimilarity-353 (with Spearman correlation) by [19] were chosen for intrinsic evaluations.

3.2 Embeddings

The hyper-parameters tuned in a grid search for the embeddings are given in table 1. The models were generated in a shared cluster running Ubuntu 16 with 32 CPUs of 32x Intel Xeon 4110 at 2.1GHz. Gensim [15] Python library implementation of word2vec was used. This is because of

Tab. 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

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.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 BiLSTM network was trained on 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).

Tab. 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

4 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) [32] 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

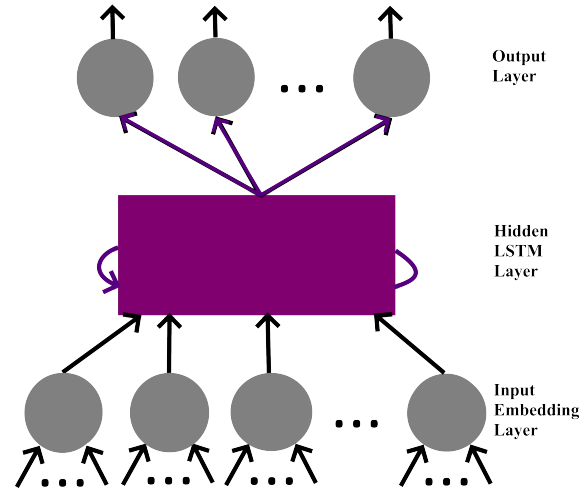


Fig. 1: Network architecture for NER

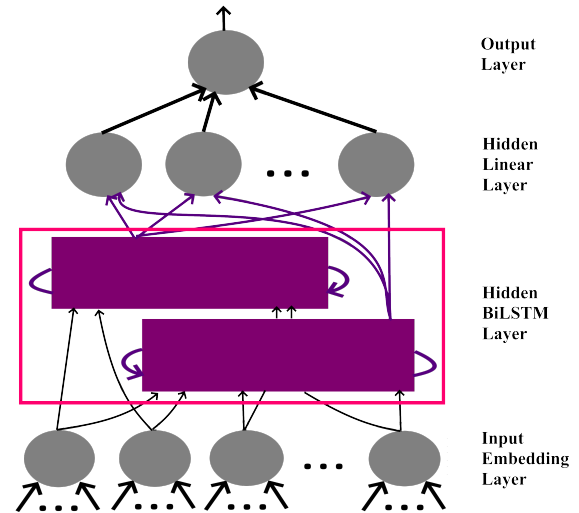


Fig. 2: Network architecture for SA

¹ <https://github.com/tosingithub/sdesk>

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 size 8 + skipgram + hierarchical 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) pre-trained 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.

5 Results and Discussion

The WordSim result output file from the Gensim Python program always has more than one value reported, including the Spearman correlation. The first value is reported as WordSim score1 in the relevant table. Table 3 summarizes key results from the intrinsic evaluations for 300 dimensions². Table 4 reveals the training time (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, the 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 [17].

Information on the length of training time for the original 100B model is not readily available. However, it's interesting to note that it is a skipgram-negative sampling (s1h0) model. Its analogy score, which we tested and report, is confirmed in the original paper [1]. It beats our best models in only analogy score (even for SW), performing worse in others, despite 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.

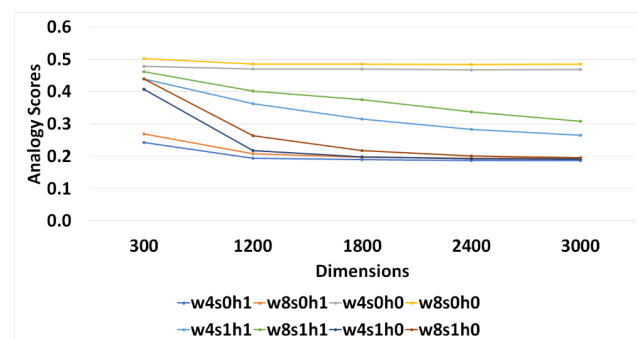


Fig. 3: Simple Wiki: Analogy Scores for 10 Epochs (color needed)

With regards to increasing dimension, though 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 with 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 [1]. This trend is true for all combinations for all tests. Polynomial interpolation may be used to determine the optimal dimension in both corpora.

With regards to NER, most pretrained embeddings outperformed the default Pytorch embedding, with our

² The results are to 3 decimal places

Tab. 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 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
Billion Word								
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
Google News - 100B (s1h0)								
Analogy: 0.740				WordSim: 0.624				
					Spearman: 0.659			

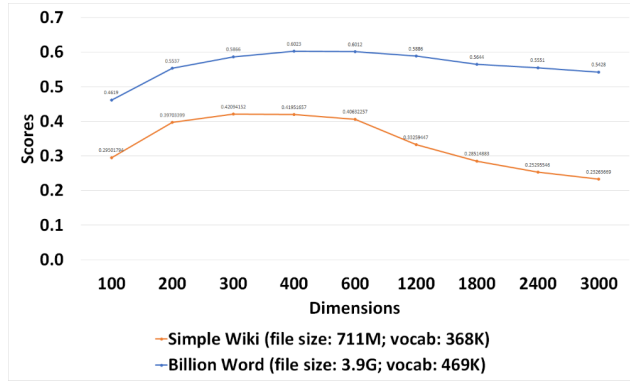


Fig. 4: Analogy Scores for w4s1h1 of SW for 5 Epochs & w8s1h1 of BW for 10 epochs (not drawn to scale from 400) (color needed)

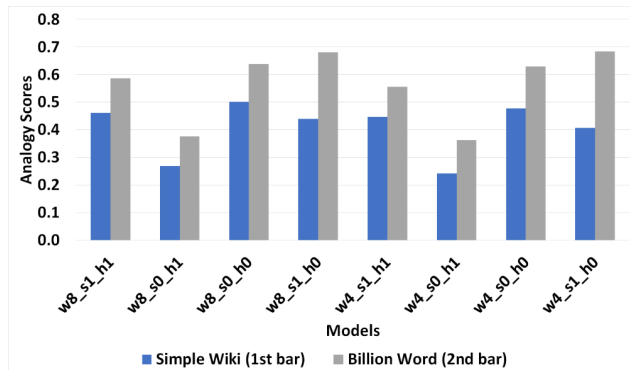


Fig. 5: Comparison of 300 dimension models for 10 epochs for SW & BW corpora

Tab. 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

Tab. 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

Tab. 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

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 [33], 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. Since one algorithm is involved in the comparisons in each case, unlike multiple algorithms [34], the applied bootstrap approach is adequate. The CI interval for

NER includes 0, thus we can conclude the difference was likely due to chance and fail to reject the null hypothesis but the CI for SA does not include 0, thus the difference is unlikely due to chance so we reject the null hypothesis.

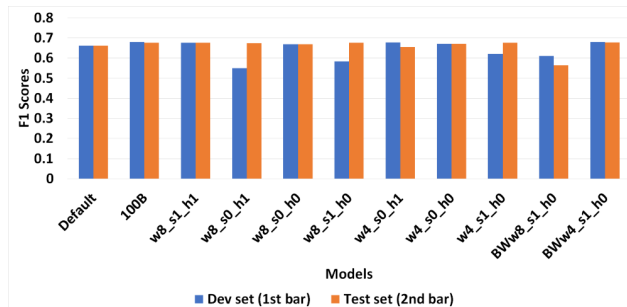


Fig. 6: Named Entity Recognition (NER) Mean F1 Scores on GMB Dataset

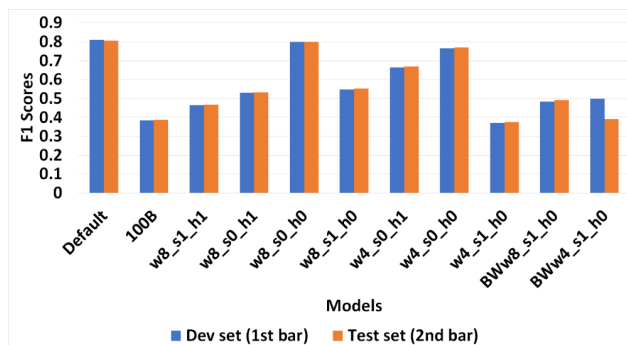


Fig. 7: Sentiment Analysis (SA) Mean F1 Scores on IMDB Dataset

6 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 [35] and [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

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 [10].

Future work that may be investigated are the 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.

Funding

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.

References

- [1] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [3] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.
- [4] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.
- [5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
- [6] Martin Längkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.

- [7] Bhuwan Dhingra, Hanxiao Liu, Ruslan Salakhutdinov, and William W Cohen. A comparative study of word embeddings for reading comprehension. *arXiv preprint arXiv:1703.00993*, 2017.
- [8] Marwa Naili, Anja Habacha Chaibi, and Henda Hajjaji Ben Ghezala. Comparative study of word embedding methods in topic segmentation. *Procedia computer science*, 112:340–349, 2017.
- [9] Yanshan Wang, Sijia Liu, Naveed Afzal, Majid Rastegar-Mojarad, Liwei Wang, Feichen Shen, Paul Kingsbury, and Hongfang Liu. A comparison of word embeddings for the biomedical natural language processing. *Journal of biomedical informatics*, 87:12–20, 2018.
- [10] Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.
- [11] Yijia Zhang, Qingyu Chen, Zhihao Yang, Hongfei Lin, and Zhiyong Lu. Biwordvec, improving biomedical word embeddings with subword information and mesh. *Scientific data*, 6(1):1–9, 2019.
- [12] Bin Wang, Angela Wang, Fenxiao Chen, Yuncheng Wang, and C-C Jay Kuo. Evaluating word embedding models: methods and experimental results. *APSIPA Transactions on Signal and Information Processing*, 8, 2019.
- [13] Joseph Turian, Lev Ratinov, and Yoshua 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*, pages 384–394. Association for Computational Linguistics, 2010.
- [14] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
- [15] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. <http://is.muni.cz/publication/884893/en>.
- [16] Tosin P Adewumi. Inner loop program construct: A faster way for program execution. *Open Computer Science*, 8(1):115–122, 2018.
- [17] Tosin P Adewumi and Marcus Liwicki. Inner for-loop for speeding up blockchain mining. *Open Computer Science*, 2019.
- [18] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.
- [19] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppín. Placing search in context: The concept revisited. *ACM Transactions on information systems*, 20(1):116–131, 2002.
- [20] Jesse Thomason, Aishwarya Padmakumar, Jivko Sinapov, Nick Walker, Yuqian Jiang, Harel Yedidion, Justin Hart, Peter Stone, and Raymond Mooney. Jointly improving parsing and perception for natural language commands through human-robot dialog. *Journal of Artificial Intelligence Research*, 67:327–374, 2020.
- [21] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in neural information processing systems*, pages 4349–4357, 2016.
- [22] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.
- [23] Albert Gatt and Emiel Krahmer. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170, 2018.
- [24] Tosin P Adewumi, Foteini Liwicki, and Marcus Liwicki. Corpora compared: The case of the swedish gigaword & wikipedia corpora. *arXiv preprint arXiv:2011.03281*, 2020.
- [25] Tosin P Adewumi, Foteini Liwicki, and Marcus Liwicki. The challenge of diacritics in yoruba embeddings. *arXiv preprint arXiv:2011.07605*, 2020.
- [26] Tosin P Adewumi, Foteini Liwicki, and Marcus Liwicki. Conversational systems in machine learning from the point of view of the philosophy of science—using alime chat and related studies. *Philosophies*, 4(3):41, 2019.
- [27] Wikipedia. Wiki news abstract. 2019.
- [28] Wikipedia. Simple wiki articles. 2019.
- [29] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. Technical report, Google, 2013.
- [30] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher 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*, pages 142–150. Association for Computational Linguistics, 2011.
- [31] Johan Bos, Valerio Basile, Kilian Evang, Noortje J Venhuizen, and Johannes Bjerva. The groningen meaning bank. In *Handbook of linguistic annotation*, pages 463–496. Springer, 2017.
- [32] Edward Loper and Steven Bird. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*, 2002.
- [33] Guillaume Calmettes, Gordon B Drummond, and Sarah L Vowler. Making do with what we have: use your bootstraps. *Advances in physiology education*, 36(3):177–180, 2012.
- [34] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.
- [35] Billy Chiu, Anna Korhonen, and Sampo Pyysalo. Intrinsic evaluation of word vectors fails to predict extrinsic performance. In *Proceedings of the 1st workshop on evaluating vector-space representations for NLP*, pages 1–6, 2016.