Replicating Bayesian Distributional Word Embeddings

Elliot Pickens
pickense@carleton.edu

Kevin Ros
kevinros@vassar.edu

28 June 2019

Abstract

The Skip-gram model has been widely used to create word embeddings for over half a decade. One known implementation, *word2vec*, has reached a particularly high level of prominence in the natural language processing community. *word2vec* embeddings have been applied to nearly every problem in the realm of NLP, but it is not without limitations. The vector representations that it produces can sometimes be overly reductive in nature. One way to expand upon these embeddings is by dropping their reliance on vector representations and moving towards distributional representations of word embeddings in conjunction with Bayesian Statistics. The Bayesian Skip-Gram Model (BSG) produces distributional representations of words allowing for operations not normally possible with vector embeddings, such as evaluation within context and entailment. In this paper, we attempt to replicate the Bayesian Skip-gram model results described in *Embedding Words as Distributions with a Bayesian Skip-gram Model* by Arthur Bražinskas, Serhii Havrylov, and Ivan Titov.

1 Outline

In this paper we attempt to replicate the results presented in [2]. In section 2, we explain the background and significance of Bayesian word embeddings and provide some motivating examples of embedding applications. In Section 3, we describe our data collection process and discuss both the authors' and our own implementations of the model found in [2]. Following this, we discuss the results of the various implementations in Section 4. We finish with a discussion of the results and conclusion, which can be found in Sections 5 and 6, respectively.

2 Background and Significance of Word Embeddings

The phrase *word embedding* generally refers to a technique in the field of natural language processing, where individual words are represented as vectors of real numbers. Usually, these vectors encode various semantic relationships between the words. The dimensionality of these vectors can quickly become unmanageable, especially when techniques such as one-hot encoding are used on large corpora to create the vectors. To prevent this, researchers developed methods to maintain low dimensionality [3, 4, 6, 7]. Typically, the vectors remain uniformly fixed in 50-200 dimensional space, regardless of the size of the corpora of the number of words. One of the most popular methods of low-dimensional word embedding generation is word2vec [6], which leverages techniques such as the *Continuous Skip-gram Model* to generate the embeddings from unstructured text [5]. However, techniques like the ones mentioned above fix word embeddings across the entire data set, effectively ignoring context post-generation. In other words, only one vector is generated per word, regardless of if the word has multiple meanings within the corpora. One solution is to generate a probability distribution for each word. This was done in [8], where the authors encoded each word as a Gaussian distribution, referred to as word2gauss. More recently, Bražinskas et al. proposed a Bayesian version of the Skip-gram architecture [2]. The use of Bayesian methods allow for the creation of ”context-specific densities”, which ”encode uncertainty about the sense of a word given its context and correspond to the approximate posterior distributions within [the authors'] model” [2]. Put simply, the meaning of a word now varies based on its context.

One possible application of word embeddings is to find substitutions for words in sentences. For example, consider the following sentence: ”The rope was tightly **wound** around the post.” In a non-probabilistic setting that does not take into account context, possible substitutions may include ”injure, harm, hurt”. However, by using Bayesian approach, we can incorporate context to get more accurate substitutions, such as ”wrapped” or ”looped”. We formally explore this setting in Section 4.

3 Methods and Data Collection

The corpora used in [2], ukWaC and Wackypedia, were collected and distributed by [1]. ukWac was created by web-crawling .uk domains and consists of over one billion words. Wackypedia is an online encyclopedia that anyone can edit. In addition to the corpora, we used the authors' publically-available GitHub repository[1] as a starting point for our model. Due to technical limitations, some slight modifications of the original code were needed in order for the model to

[1]https://github.com/ixlan/BSG

run without error. We successfully generated five different models. We ran our first three models (WackFull, Wack10k, Wack20k) on the Wackypedia corpus, the fourth model (ukWac) on the ukWaC corpus, and the fifth model (Wack + ukWaC) on a concatination of the two corpora. WackFull ran on the entire Wackypedia corpus whereas Wack10k and Wack20k both ran on a smaller subset of the Wackypedia corpus. The Wack20k model looked at the most common 20,000 words, and every other model only looked at the most common 10,000 words. In all of our models, the learning rate was set to 0.00275, the vector dimensions were set to 100, and the word window was set to 5 on either side. Additionally, negative sampling was set to 10. Note that the original authors had a learning rate of 0.00055 and a vocabulary size of 280,000. In terms of pre-processing the corpora, for all of our models we removed only punctuation and sent each word to lower case. In the original authors' model, it was not clear how much pre-processing was performed on the corpora. Additionally, for a more thorough mathematical explanation of the Bayesian Skip-gram method, see Section 2 of [2].

4 Results of the Models

Our experiments produced quite mixed results. Although, throughout the course of our experimentation with the BSG model we ran well over 30 different instances of the model we faced serious limitations in terms of computing time and server access. Due to a cap on the number of cores we were allowed to employ during each simulation a single instance of the model could take up to five days to complete when running the full data-set used in the paper. Because of the computational limitations, we were unable to run the BSG model with the full set of hyperparameter settings used in the original paper. Even when using our notably more simplified version of the model, however, we were still able to score decently well on many metrics, and produce lexical substitutions that generally appear correct.

In Table 1, we compare the various models using the sum of scores of word similarity benchmarks (Score Sum) and the generalized average precision (GAP). More information about the word similarity benchmarks and GAP scores can be found in [2].

Numerical Results for Each Model		
Model	Score Sum	Mean Gap
ukWaC	4.3726	0.314831
WackFull	4.8977	0.285820
ukWaC + Wack.	4.9377	0.308338
Wack10k	5.8243	0.288952
Wack20k	5.9429	0.345963
Bražinskas et al.	**7.26**	**0.437**

Table 1: Summed similarity scores and mean GAP values for all models.

It is clear that the model presented in [2] by Bražinskas et al. outperforms all of our models, but our best results (coming from Wack20k) while markedly worse are within a reasonable distance of the best. Next we show some example substitution results from Wack20k:

Substitution Results for Wack20k	
Sentence	Top 3 Substitution Choices
press people loved the **film** as well	movie, production, documentary
finally this new rule will	also, ultimately, lastly
this meditation can **fix** many	repair, set, do
health information to provide and **manage** your--health care and related	control, handle, oversee

Table 2: Substitutions given by the BSG model trained on Wack20k.

Although it has comparable results with the original model the Wack20k model still failed in certain substitution tasks. One instance of this error can be seen in the following sentence: ”the summit did not **take** place until next year”, the top three substitutions proposed by Wack20k were ”be”, ”begin”, and ”include”. Of the three results ”begin” and ”be” make some sense in terms of sentiment, but are grammatically incorrect.

5 Discussion

Our results seem to indicate that the size of the vocabulary is one of the most important parameters when tasked with generating word embedding distributions. Although Wack20K was trained on a corpus three times smaller than WackFull and over five times smaller than ukWaC + Wack, it still achieved significantly better results. And when compared to the immense size of the vocabulary described in [2], it is understandable why their model outperforms ours.

Regardless of this, many questions still remain. To what effect does a difference in the pre-processing have on the model scores? Or how do different concatenations of the corpora effect the scores? Additionally, how do these changes compare to similar changes in traditional non-probabilistic models? All of these questions are potential areas of investigation.

In both our experiments and the Bražinskas paper we can see the potential of the BSG model on a number of similarity benchmarks, but moving forward it would be interesting to apply the distributional embeddings to the *GLUE* tasks [9]. In particular, it would be useful to see if these embeddings can be expanded to get distributional representations of sentences or documents. In paraphrase and duplicate question detection tasks, for example, distributional representations to sentence embeddings may be especially applicable given their ability to find show entailment. Common methods used for paraphrase and duplicate question detection use vector based sentence embeddings to measure similarity, so it would be fascinating to see how distributional embeddings work in this context.

6 Conclusion

During our series of experiments with the BSG model we had very mixed results. We were unable to fully recreate the results shown in the *Embedding Words as Distributions with a Bayesian Skip-gram Model*, but given our limited computational resources we still learned quite a bit about its properties especially in cases of limited data. It is clear that the BSG method requires significant data to function effectively, but when data is present and hyperparameters are chosen effectively the model can preform at a very high level while producing distributional outputs. Overall, we are optimistic that the future developments of probabilistic word embedding generation will continue to have an impact on the field of natural language processing.

7 Appendix

7.1 Acknowledgements

We would like to thank Dr. Jingchen (Monika) Hu for her guidance through the involved mathematics behind the Bayesian Skip-gram model. Additionally, we would like to thank Dr. Eros Zanchetta for providing us with the Wackypedia and ukWaC corpora.

References

[1] Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226, 2009.

[2] Arthur Bražinskas, Serhii Havrylov, and Ivan Titov. Embedding words as distributions with a bayesian skip-gram model. *CoRR*, abs/1711.11027, 2017.

[3] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(Aug):2493–2537, 2011.

[4] 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.

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[6] 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.

[7] Jeffrey Pennington, Richard Socher, and Christopher 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.

[8] Luke Vilnis and Andrew McCallum. Word representations via gaussian embedding. *arXiv preprint arXiv:1412.6623*, 2014.

[9] Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.