VecShare: A Framework for Sharing Word Representation Vectors

Jared Fernandez and **Zhaocheng Yu** and **Doug Downey**
Department of Electrical Engineering and Computer Science
Northwestern University
Evanston, IL 60208
{jared.fern|zhaochengyu2017}@u.northwestern.edu
ddowney@eecs.northwestern.edu

Abstract

Many Natural Language Processing (NLP) models rely on distributed vector representations of words. Because the process of training word vectors can require large amounts of data and computation, NLP researchers and practitioners often utilize pre-trained embeddings downloaded from the Web. However, finding the best embeddings for a given task is difficult, and can be computationally prohibitive. We present a framework, called VecShare, that makes it easy to share and retrieve word embeddings on the Web. The framework leverages a public data-sharing infrastructure to host embedding sets, and provides automated mechanisms for retrieving the embeddings most similar to a given corpus. We perform an experimental evaluation of VecShare's similarity strategies, and show that they are effective at efficiently retrieving embeddings that boost accuracy in a document classification task. Finally, we provide an open-source Python library for using the VecShare framework.[1]

1 Introduction

Word embeddings capture syntactic and semantic properties of words, and are a key component of many modern NLP models (Turian et al., 2010). However, high-quality embeddings can be expensive to train. As a result, rather than training their own embeddings, NLP researchers and practitioners often download pre-trained embeddings from the Web, e.g. (Limsopatham and Collier, 2016; Cheng et al., 2016).

However, existing methods for sharing embeddings on the Web are suboptimal. Current practice primarily consists of contributors posting embedding sets to their own Web sites. No central embedding repository exists, and it is difficult for users to know which embedding sets are available. Furthermore, determining the utility of an embedding set for a particular NLP task entails significant time and computational costs, as users must manually download and evaluate multiple complete embedding sets. Methods exist for automatically scoring an embedding set, but these are limited to specific tasks and lack integration with existing code bases (Faruqui and Dyer, 2014).

Our goal in this paper is to introduce a framework that makes sharing word embeddings easier for NLP researchers and practitioners. It should be simple and fast to post, browse, and retrieve embeddings from a public data store. Additionally, integration with existing NLP codebases should be more seamless: software libraries should automatically identify and download the particular embeddings that are likely to be relevant to a user's corpus.

This paper presents VecShare, a framework for sharing word embeddings. As its data store, it uses an existing public data-sharing platform, which provides searching and browsing capability. To solve the critical challenge of helping users quickly find relevant embeddings, we introduce embedding *indexers*. The indexers compute and share compact representations, called *signatures*, for each embedding set. Users employ a software library that downloads the signatures and compares them against the user's corpus. Using the signatures, the library efficiently evaluates the utility of each shared embedding set and determines which sets are most likely to be relevant. The library can then automatically download the relevant embeddings and make them available

[1] https://github.com/JaredFern/VecShare

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 316–320
Copenhagen, Denmark, September 7–11, 2017.

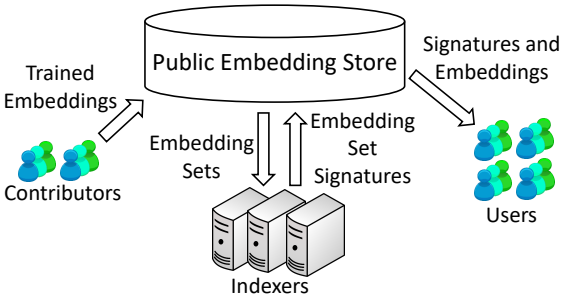

Figure 1: The VecShare Framework.

within the user's code. Finally, VecShare is an open source framework: new embeddings, indexers, and signature methods can be independently added at any time.

We perform experiments evaluating different signature methods in selecting embeddings for document classification tasks, and demonstrate that an ensemble signature based on simple features (e.g. vocabulary overlap with the user's corpus, or similarities between a sample of embedding pairs) can select helpful embeddings. We release a Python library for NLP researchers and practitioners that can query the embedding library, and automatically select and download relevant embeddings for a given corpus.

2 The VecShare Framework

The VecShare framework is illustrated in Figure 1. Contributors upload embedding sets to a *Public Embedding Store*, hosted on a public data-sharing platform. *Indexers* periodically poll the embedding store, detecting and indexing newly uploaded embedding sets. For each embedding set, indexers store a *signature* that compactly represents the content of the embedding set. The indexer uses these signatures to return relevant embedding sets to users. A *user* can then retrieve embeddings for their particular corpus from the data store, using a software library built for this purpose.

The current VecShare implementation uses the public data sharing website data.world as its embedding store.[2] We chose data.world for its ease of use and robustness, but any publicly-available data share that allows search by tags and programmatic access is sufficient to house VecShare. Below, we describe the details of the framework.

[2] http://data.world/

2.1 Contributors

A contributor who has computed a set of word embeddings adds the embeddings to VecShare by uploading the data to the share, following a simple standard format with $n + 1$ fields where the first field is a word, and the remaining fields give the n-dimensional embedding of the word. The contributor tags the data set with a designated tag so that indexers can automatically identify that the data set is an indexable word embedding set. For embedding sets to be applicable to certain kinds of signatures, contributors can also elect to upload additional metadata with their embeddings (in our initial implementation, metadata includes a frequency ranking of terms in the corpus, and the total number of tokens used to construct the embeddings).

2.2 Indexers and Signatures

Indexers periodically poll the embedding store, looking for new embedding sets uploaded by contributors. For each embedding set, the indexer computes and stores a signature designed to capture characteristics of the embeddings. To estimate the relevance of each embedding set for the user's task, these signatures are later compared to corresponding signatures created from the user's corpus. Thus, each signature method has an associated *similarity measure*, which takes in a pair of signatures (one from a VecShare embedding set, and the other from the user's corpus) and outputs a numeric similarity score for the pair.

We explore two primary signature methods in this paper. The first, *VocabRk*, consists of (up to) the T_v most frequent words in the embedding corpus, excluding a set of stop words. The similarity method is the negative average rank of the signature words within the user's frequency-ordered vocabulary. Words not in the user's corpus are assigned a rank of T. Thus, the most similar embedding set under the *VocabRk* signature is thus the one with the lowest average rank.

The *VocabRk* signature method relies only on vocabulary overlap, and entirely ignores the embeddings themselves. We also experiment with a second signature method that does utilize the embeddings. The *SimCorr* signature for a set of embeddings E consists of the embeddings for the T_s most frequent words in E's corpus (again excluding stop words). To estimate the similarity between the user's corpus C and the embeddings E,

the *SimCorr* method first computes a set of embeddings on the user's corpus. For all words found in both E's signature and in the user's corpus, *SimCorr* computes all pairwise cosine similarities between pairs of E embeddings, and pairs of C embeddings. The Pearson correlation coefficient between the E-based cosine similarities and the C-based ones is taken as the *SimCorr* similarity measure. Thus, two embedding sets that estimate similarity of terms in a similar manner will be deemed similar according the *SimCorr* measure, even if the vector values of the embeddings differ substantially between the two sets.

The indexers make their signatures available to users by simply sharing a signature data set on the public data store, which is read by the VecShare software library. Like the other components of VecShare, the indexers of the framework are extensible – new indexers, signature and similarity measures can be created at any time.

2.3 Libraries

Users access the VecShare framework using a code library. Embeddings can be requested by name if the user desires a particular embedding (e.g. "300 dimensional Google News word2vec embeddings"), or the user can query an indexer to find embedding sets most likely to be useful for the user's task. Importantly, the software library performs embedding selection locally on the user's machine, using signature similarity methods described previously. The user's corpus does not need to be uploaded or shared.

Once the target embedding set has been identified, the library downloads the target embeddings for the particular words in the user's vocabulary. Thus, if the user's vocabulary is much smaller than that of the embedding set, this download can be much more compact than the full embedding set.

A contribution of this paper is the release of a Python library for the framework, which implements the *VocabRk* similarity computation. With this library, leveraging the framework to select and retrieve the top-ranked embeddings for a given corpus requires just one line of code.

3 Experiments

We now evaluate the effectiveness of the signature methods described in the previous section at identifying high-quality embedding sets for a given corpus, for the task of text classification. We also quantify the improvement in efficiency when using VecShare rather than following the current practice of downloading and testing multiple embedding sets.

In our experiments, we set the parameters $T_v = 5,000$ and $T_s = 1,000$, and we discard the top 100 most frequent words as stopwords. When training embeddings on the user's corpus, we use word2vec.

3.1 Experimental Methodology

We perform experiments in two settings: first with *large-corpus embeddings*, where we use word2vec and GloVe embedding sets trained on billions of tokens. The large-corpus embeddings are representative of state-of-the-art models, but are trained over very broad-topic corpora (billions of tokens of newswire, Web or social media text). To better measure whether VecShare can harness more specific, targeted embedding sets, we also evaluate over *small-corpus* embeddings.

For the large-corpus embeddings, we utilize three sets of GloVe embeddings (Pennington et al., 2014): **wik+**, 100-dimensional embeddings trained on six billion tokens of Wikipedia and the Gigaword corpus; **web**, 300-dimensional embeddings trained on 42 billion tokens of the Common Crawl Web dataset; and **twtr**, 100-dimensional embeddings trained on 27 billion tokens of Twitter posts. We also utilize **gnws**, 300-dimensional word2vec embeddings trained on three billion tokens of Google News data.[3]

For the small corpus embeddings, we created a topically diverse collection of subsets of the New York Times corpus (Sandhaus, 2008), across seven categories (agriculture, arts, books, economics, government, movies, and weather). We then trained word2vec embeddings on each subset, to create seven distinct similarly-sized, small corpus embedding sets.

For our experiments, we utilize the embeddings as features for document classification within a convolutional neural network (Chollet, 2017). We evaluate on four document classification tasks: Reuters-21578 newswire topic classification (Lewis, 1997), subjectivity classification (Pang and Lee, 2004), IMDB movie review classification (Maas et al., 2011), and the 20news classification task.[4]

[3] https://github.com/mmihaltz/word2vec-GoogleNews-vectors.

[4] http://qwone.com/~jason/20Newsgroups/

	Reuters			Subjectivity			IMDB			20news			Average	
	ρ	Sel.	Acc.	ρ	Sel.	Acc.	ρ	Sel.	Acc.	ρ	Sel.	Acc.	ρ	Acc
Random	-	-	0.862	-	-	0.688	-	-	0.868	-	-	0.763	-	0.795
MaxTkn	**0.63**	web	**0.888**	0.19	web	0.728	0.38	web	0.881	**0.97**	web	**0.863**	**0.54**	0.840
VocabRk	0.46	gnws	0.882	0.02	gnws	**0.759**	0.40	gnws	**0.886**	0.20	gnws	0.719	0.27	0.812
SimCorr	-0.65	wik+	0.84	**0.81**	gnws	**0.759**	0.45	gnws	**0.886**	0.60	twtr	0.748	0.30	0.808
All	0.26	gnws	0.882	0.43	gnws	**0.759**	**0.49**	gnws	**0.886**	0.87	web	**0.863**	0.51	**0.848**
Oracle	-	web	0.888	-	gnws	0.759	-	gnws	0.886	-	web	0.863	-	0.85

Table 1: Experimental results using large-corpus embeddings. All of the signature methods outperform the random baseline, and the *All* method performs best in terms of both correlation ρ and text classification accuracy.

	Reuters			Subjectivity			IMDB			20news			Average	
	ρ	Sel.	Acc.	ρ	Sel.	Acc.	ρ	Sel.	Acc.	ρ	Sel.	Acc.	ρ	Acc
Random	-	-	0.844	-	-	0.667	-	-	0.829	-	-	0.610	-	0.738
MaxTkn	0.62	govt	0.856	-0.64	govt	0.568	-0.02	govt	0.763	**0.82**	govt	**0.647**	0.20	0.709
VocabRk	0.74	econ	**0.880**	0.51	mov	0.686	0.89	mov	0.835	0.64	book	0.629	**0.70**	0.758
SimCorr	0.51	econ	**0.880**	**0.62**	book	**0.706**	**0.93**	book	0.842	-0.25	agri	0.551	0.45	0.745
All	**0.82**	econ	**0.880**	0.16	book	**0.706**	0.87	book	**0.842**	0.67	book	0.629	0.63	**0.764**
Oracle	-	econ	0.880	-	book	0.706	-	book	0.842	-	govt	0.647	-	0.769

Table 2: Experimental results using small-corpus embeddings. The *VocabRk* and *SimCorr* methods outperform the baselines, and the *All* method performs best in terms of both correlation ρ and text classification accuracy.

In addition to the *VocabRk* and *SimCorr* methods described in the previous section, we also evaluate against two simple baselines: *Random*, which selects an embedding set at random, and *MaxTkn*, which adopts a "bigger is better" strategy, always selecting the embedding set trained over the largest text corpus.

Finally, as discussed below our experimental results reveal that the different signature methods have distinct strengths. Thus, we also evaluate *All*, a simple ensemble of the *VocabRk, SimCorr*, and *MaxTkn* methods. *All* simply takes an even average of the rankings output by its three constituent signature methods. The *All* method selects the single embedding set with the lowest average ranking, breaking ties in favor of the *VocabRk* method.

3.2 Results

Our results are shown in Tables 1 and 2. For each classification task and each method, "Sel." indicates the embedding set that the method selects (that is, the one the method ranks most similar to the text classification data set). We report two measures of the quality of a signature method: ρ, the Pearson correlation between the similarity scores assigned by the method and the set's accuracy on the classification task; and "Acc.," the accuracy of the embeddings selected by the method.

The results show that for the small-corpus embeddings, the *VocabRk* and *SimCorr* signature methods perform well, beating the random baseline overall. By contrast, for the large-corpus embeddings, the *MaxTkn* method performs the best of the individual methods (primarily due to its strong performance on the idiosyncratic 20news data set). The *All* ensemble method achieves performance nearly as high as the best possible embedding selection (represented as the *Oracle* method in the tables).

An alternative to using pre-trained embeddings is to train embeddings on the evaluation corpus itself. We found that this approach achieved an average accuracy of 0.746 across our four data sets, lower than our results using the *All* method. Utilizing the pre-trained embeddings, especially those computed over large corpora, provides a significant boost in text classification accuracy.

3.3 Efficiency Experiment

We also evaluated the relative gain in time and space efficiency that VecShare provides over the current practice of manually evaluating each embedding set and selecting the embedding that performs best. The efficiency experiment was performed on a single machine with a 2.3 GHz quad-core CPU and 8GB of main memory, using a test

framework containing 11 embedding sets.

Embedding selection was performed using both the *VocabRk* signature method on VecShare and the conventional method of selecting embeddings, which trains models for each embedding set and then evaluates those models on the test corpus. The conventional approach required an average of 177 minutes to train, evaluate, and select an embedding set for each test corpus. Whereas, the *VocabRk* signature method on the VecShare framework required an average of 38 seconds to select an embedding for each test corpus, an average speedup of 280x. Additionally, VecShare substantially reduces space cost: the total size of the signatures in the experiments is 4-5 orders of magnitude smaller than the full embedding sets.

4 Conclusions and Future Work

We presented VecShare, a framework for sharing word vector representations. The VecShare framework uses signatures to help researchers and practitioners quickly identify helpful embeddings for their task. We released a Python library that allows practitioners to access the framework. We also performed experiments quantifying the accuracy and efficiency of VecShare's embedding selection approach on text classification. Further experiments on additional data sets and NLP tasks are necessary.

In future work, we wish to explore embedding signatures that leverage richer knowledge of the practitioner's corpus and task. Finally, we hope to extend VecShare's embedding selection methods to consider syntheses of multiple distinct embedding sets, tailored to the practitioner's task and corpus.

Acknowledgments

This research was supported in part by NSF grant IIS-1351029 and the Allen Institute for Artificial Intelligence. We thank Mohammed Alam, Dave Demeter, Thanapon Noraset, and Zheng Yuan for helpful comments.

References

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. In *EMNLP*.

Franois Chollet. 2017. keras. https://github.com/fchollet/keras.

Manaal Faruqui and Chris Dyer. 2014. Community evaluation and exchange of word vectors at wordvectors.org. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Baltimore, USA.

David D Lewis. 1997. Reuters-21578 text categorization test collection, distribution 1.0. *http://www. research. att. com/˜ lewis/reuters21578. html* .

Nut Limsopatham and Nigel Collier. 2016. Modelling the combination of generic and target domain embeddings in a convolutional neural network for sentence classification. Association for Computational Linguistics.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 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, pages 142–150.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, page 271.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia* 6(12):e26752.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 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, pages 384–394.