Effective Dimensionality Reduction for Word Embeddings

Vikas Raunak
Carnegie Mellon University
vraunak@cs.cmu.edu

Vivek Gupta
University of Utah
vgupta@cs.utah.edu

Florian Metze
Carnegie Mellon University
fmetze@cs.cmu.edu

Abstract

Pre-trained word embeddings are used in several downstream applications as well as for constructing representations for sentences, paragraphs and documents. Recently, there has been an emphasis on improving the pre-trained word vectors through post-processing algorithms. One improvement area is reducing the dimensionality of word embeddings. Reducing the size of word embeddings can improve their utility in memory constrained devices, benefiting several real-world applications. In this work, we present a novel technique that efficiently combines PCA based dimensionality reduction with a recently proposed post-processing algorithm (Mu and Viswanath, 2018), to construct effective word embeddings of lower dimensions. Empirical evaluations on several benchmarks show that our algorithm efficiently reduces the embedding size while achieving similar or (more often) better performance than original embeddings. To foster reproducibility, we have released the source code along with paper [1].

1 Introduction

Word embeddings such as Glove (Pennington et al., 2014) and word2vec Skip-Gram (Mikolov et al., 2013) obtained from unlabeled text corpora can represent words in distributed dense real-valued low dimensional vectors which geometrically capture the semantic *'meaning'* of a word. These embeddings capture several linguistic regularities such as analogy relationships. Such embeddings are of a pivotal role in several natural language processing tasks.

Recently, there has been an emphasis on applying post-processing algorithms on the pre-trained word vectors to further improve their quality. For example, algorithm in (Mrkšic et al., 2016) tries to inject antonymy and synonymy constraints into vector representations, while (Faruqui et al., 2015) tries to refine word vectors by using relational information from semantic lexicons such as WordNet (Miller, 1995). (Bolukbasi et al., 2016) tries to remove the biases (e.g. gender biases) present in word embeddings and (Nguyen et al., 2016) tries to *'denoise'* word embeddings by strengthening salient information and weakening noise. In particular, the post-processing algorithm in (Mu and Viswanath, 2018) tries to improve word embeddings by projecting the embeddings away from the most dominant directions and considerably improves their performance by making them more discriminative. However, a major issue related with word embeddings is their size (Ling et al., 2016), e.g., loading a word embedding matrix of 2.5 M tokens takes up to 6 GB memory (for 300-dimensional vectors, on a 64-bit system). Such large memory requirements impose significant constraints on the practical use of word embeddings, especially on mobile devices where the available memory is often highly restricted. In this work we combine the simple dimensionality reduction technique, PCA with the post processing technique of (Mu and Viswanath, 2018), as discussed above.

In Section 2, we first explain the post processing algorithm (Mu and Viswanath, 2018) and then our novel algorithm and describe with an example the choices behind its design. The evaluation results are presented in section 3. In section 4, we discuss the related works, followed by the conclusion.

2 Proposed Algorithm

We first explain the post-processing algorithm from (Mu and Viswanath, 2018) in section 2.1. Our main algorithm, along with the motivations is explained next, in the section 2.2.

[1] https://github.com/vyraun/Half-Size

Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019), pages 235–243
Florence, Italy, August 2, 2019.

2.1 Post-Processing Algorithm

(Mu and Viswanath, 2018) presents a simple post-processing algorithm that renders off-the-shelf word embeddings even stronger, as measured on a number of lexical-level and sentence-level tasks. The algorithm is based on the geometrical observations that the word embeddings (across all representations such as Glove, word2vec etc.) have a large mean vector and most of their energy, after subtracting the mean vector is located in a subspace of about 8 dimensions. Since, all embeddings share a common mean vector and all embeddings have the same dominating directions, both of which strongly influence the representations, eliminating them makes the embeddings stronger. Detailed description of the post-processing algorithm is presented in Algorithm 1 (PPA).

Algorithm 1: Post Processing Algorithm PPA(X, D)

Data: Word Embedding Matrix X, Threshold Parameter D
Result: Post-Processed Word Embedding Matrix X

/* Subtract Mean Embedding */
X = X - $\overline{X}$;
/* Compute PCA Components */
u_i = PCA(X), where i = 1,2, . . . ,d. ;
/* Remove Top-D Components */
for ***all*** *v in X* **do**
    $v = v - \sum_{i=1}^{D}(u_i^T \cdot v)u_i$
end

Figure 1 demonstrates the impact of the post-processing algorithm (PPA, with D= 7) as observed on wiki pre-trained Glove embeddings (300-dimensions). It compares the fraction of variance explained by the top 20 principal components of the original and post-processed word vectors respectively [2]. In the post-processed word embeddings none of the top principal components are disproportionately dominant in terms of explaining the data, which implies that the post-processed word vectors are not as influenced by the common dominant directions as the original embeddings. This makes the individual word vectors more *'discriminative'*, hence, improving their quality, as validated on several benchmarks in (Mu and Viswanath, 2018).

2.2 Proposed Algorithm

This section explains our algorithm, 2 that effectively uses the the PPA algorithm, along with PCA for constructing lower dimensional embeddings.

[2] the total sum of explained variances over the 300 principal components is equal to 1.0

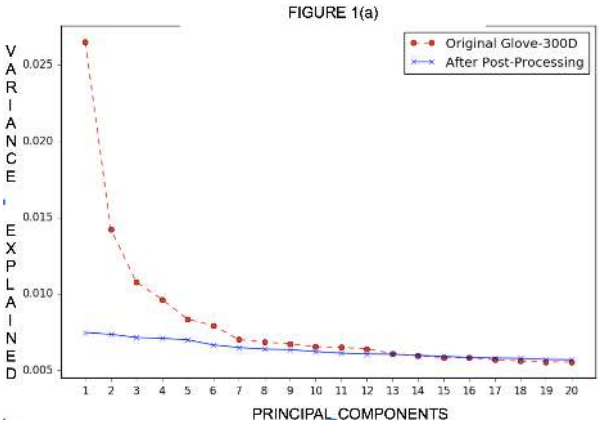

Figure 1: Comparison of the fraction of variance explained by top 20 principal components of the Original and Post-Processing (PPA) applied Glove embeddings (300D).

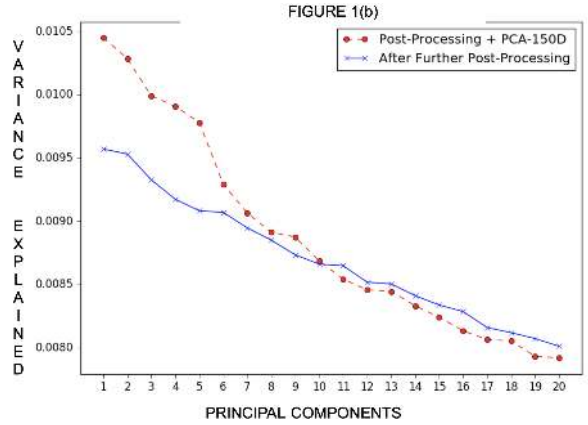

Figure 2: Comparison of the fraction of variance explained by top 20 principal components of the PPA + PCA-150D baseline and Further Post-Processed Glove embedding (150D).

We first apply the algorithm 1 (PPA) of (Mu and Viswanath, 2018) on the original word embedding, to make it more *'discriminative'*. We then construct a lower dimensional representation of the post processed *'purified'* word embedding using Principal Component Analysis (PCA (Shlens, 2014)) based dimensionality reduction technique. Lastly, we again *'purify'* the reduced word embedding by applying the algorithm 1 (PPA) of (Mu and Viswanath, 2018) on the reduced word embedding to get our final word embeddings.

To explain the last step of applying the algorithm 1 (PPA) on the reduced word embedding, consider the Figure 2. It compares the variance explained by the top 20 principal components for the embeddings constructed by first post-processing the Glove-300D embeddings according by Algorithm 1 (PPA) and then transforming the embed-

dings to 150 dimensions with PCA (labeled as Post-Processing + PCA); against a further post-processed version by again applying the Algorithm 1 (PPA) of the reduced word embeddings [3]. We observe that even though PCA has been applied on post-processed embeddings which had their dominant directions eliminated, the variance in the reduced embeddings is still explained disproportionately by a few top principal components. The re-emergence of this geometrical behavior implies that further post-processing (Algorithm 1 (PPA)) could improve the embeddings further. Thus, for constructing lower-dimensional word embeddings, we apply the post-processing algorithm on either side of a PCA dimensionality reduction of the word vectors in our algorithm.

Finally, from Figures 1 and 2, it is also evident that the extent to which the top principal components explain the data in the case of the reduced embeddings is not as great as in the case of the original 300 dimensional embeddings. Hence, multiple levels of post-processing at different levels of dimensionality will yield diminishing returns as the influence of common dominant directions decreases on the word embeddings. Details of reduction technique is given in Algorithm 2.

Algorithm 2: Dimensionality Reduction Algorithm

Data: Word Embedding Matrix X, New Dimension N, Threshold Parameter D
Result: Word Embedding Matrix of Reduced Dimension N: X

```
/* Apply Algorithm 1 (PPA)          */
X = PPA(X, D) ;
/* Transform X using PCA            */
X = PCA(X) ;
/* Apply Algorithm 1 (PPA)          */
X = PPA(X, D) ;
```

3 Experimental Results

In this section, we evaluate our proposed algorithm on standard word similarity benchmarks and across a range of downstream tasks. For all our experiments, we used pre-trained Glove embeddings of dimensions 300, 200 and 100, trained on Wikipedia 2014 and Gigaword 5 corpus (400K vocabulary) (Pennington et al., 2014) [4] and fastText embeddings of 300 dimensions trained on Wikipedia using the Skip-Gram model described in (Bojanowski et al., 2017) (with 2.5M vocabulary) [5]. The next subsection also presents results using word2vec embeddings trained on Google News dataset [6].

3.1 Word Similarity Benchmarks

We use the standard word similarity benchmarks summarized in (Faruqui and Dyer, 2014) for evaluating the word vectors.

Dataset: The datasets (Faruqui and Dyer, 2014) have word pairs (WP) that have been assigned similarity rating by humans. While evaluating word vectors, the similarity between the words is calculated by the cosine similarity of their vector representations. Then, Spearman's rank correlation coefficient (Rho $\times$ 100) between the ranks produced by using the word vectors and the human rankings is used for the evaluation. The reported metric in our experiments is Rho $\times$ 100. Hence, for better word similarity, the evaluation metric will be higher.

Baselines: To evaluate the performance of our algorithm, we compare it against different schemes of combining the post-processing algorithm with PCA [7] as following baselines:

- **PCA**: Transform word vectors using PCA.
- **P+PCA**: Apply PPA (Algorithm 1) and then transform word vectors using PCA.
- **PCA+P**: Transform word vectors using PCA and then apply PPA.

These baselines can also be regarded as ablations on our algorithm and can shed light on whether our intuitions in developing the algorithm were correct. In the comparisons ahead, we represent our algorithm as Algo-N ,where N is the reduced dimensionality of word embeddings. We use the scikit-learn (Pedregosa et al., 2011) PCA implementation.

Evaluation Results: First we evaluate our algorithm on the same embeddings against the 3 baselines, we then evaluate our algorithm across word embeddings of different dimensions and different types. In all the experiments, the threshold parameter D in the PPA algorithm was set to 7 and the new dimensionality after applying the dimensionality reduction algorithms, N was set to $\frac{d}{2}$, where

[3] the total sum of explained variances over the 150 principal components is equal to 1.0
[4] nlp.stanford.edu/projects/glove/
[5] github.com/facebookresearch/fastText/
[6] https://code.google.com/archive/p/word2vec/
[7] Generic non-linear dimensionality-reduction techniques performed worse than baselines, presumably because they fail to exploit the unique geometrical property of word embeddings discussed in section 2.

Table 1: Performance (Rho $\times$ 100) of Algo. 2 on different embedding and dimensions across multiple datasets. Bold represent the best value in each column.

Dataset	M Turk -771	WS 353 SIM	M Turk -287	VE RB -143	WS -353 -ALL	RW Stan ford	Men -TR -3K	RG -65	MC -30	Sim Lex -999	WS -353 -Rel	YP -130
Glove-300D	65.01	66.38	63.32	30.51	60.54	41.18	73.75	**76.62**	70.26	**37.05**	57.26	**56.13**
PCA-150D	52.47	52.69	56.56	28.52	46.52	27.46	63.35	71.71	70.03	27.21	41.82	36.72
P+PCA-150D	**65.59**	70.03	63.38	39.04	66.23	**43.17**	75.34	73.62	69.21	36.71	62.02	55.42
PCA-150D+P	63.86	70.87	**64.62**	40.14	66.85	40.79	75.37	74.27	72.35	33.81	60.5	50.2
Algo-150D	64.58	**71.61**	63.01	**42.24**	**67.41**	42.21	75.8	75.71	**74.8**	35.57	**62.09**	55.91
FastText-300D	66.89	**78.12**	**67.93**	**39.73**	**73.69**	**48.66**	**76.37**	79.74	81.23	**38.03**	68.21	**53.33**
Algo-150D	**67.29**	77.4	66.17	34.24	73.16	47.19	76.36	**80.95**	**86.41**	35.47	**69.96**	50.9
Glove-200D	**62.12**	62.91	61.99	28.45	57.42	38.95	71.01	71.26	66.56	34.03	54.48	52.21
Algo-100D	61.99	**68.43**	**63.55**	**36.82**	**65.41**	**39.8**	**74.44**	**71.53**	**69.83**	**34.19**	**61.56**	**49.94**
Glove-100D	58.05	60.35	61.93	30.23	52.9	**36.64**	68.09	69.07	62.71	**29.75**	49.55	**45.43**
Algo-50D	**58.85**	**66.27**	**64.09**	**33.04**	**62.05**	**36.64**	**70.93**	**64.56**	**68.79**	29.13	**59.55**	41.95

d is the original dimensionality. The value of parameter D was set to 7, because from Figure 1, we observe that the top 7 components are disproportionately contributing to the variance. Choosing a lower D will not eliminate the disproportionately dominant directions, while choosing a higher D will eliminate useful discriminative information from the word vectors. We choose $N = \frac{d}{2}$ as we observed that going below half the dimensions ($N < \frac{d}{2}$) significantly hurts performance of all embeddings.

Results Across Different Baselines: Table 1 shows the results of different baselines on the 12 datasets. As expected from discussions in Section 2, our algorithm achieves the best results on 6 out of 12 datasets when compared across all other baselines. In particular, the 150-dimension word embeddings constructed with our algorithm performs better than the 300-dimension embeddings in 7 out of 12 datasets with an average improvement of 2.74% across the 12 datasets, thus performing significantly better than PCA, PCA+P baselines and beating P+PCA baseline in 8 out of the 12 tasks.

Results Across Different Embeddings: Table 1 also shows the results of our algorithm on 300-dimension fastText embeddings, 100-dimension Glove embeddings and 200-dimension Glove embeddings. In fastText embeddings, the 150-dimension word vectors constructed using our algorithm gets better performance on 4 out of 12 datasets when compared to the 300-dimension embeddings. Overall, the 150-dimension word vectors have a cumulative score of 765.5 against the 771.93 of the 300-dimension vectors. The performance is similar to the 300-dimension embeddings with an average performance decline of 0.53% across the 12 datasets. With Glove embeddings of 100 and 200 dimensions, our algorithm leads to significant gains, with average performance improvements of 2.6% and 3% respectively over the original embeddings and achieves much better performance on 8 and 10 datasets respectively. Another observation is the embeddings generated by reducing Glove-200D to 100 dimensions using our algorithm outperform the original Glove-100D embeddings, with an average performance improvement of 6% across all 12 datasets. Hence, empirical results validate that our algorithm is effective in constructing lower dimension word embeddings, while maintaining similar or better performance than the original embeddings.

3.2 Downstream Tasks

Embeddings obtained using the proposed dimensionality reduction algorithm can be used as direct features for downstream supervised tasks. We experimented with textual similarity tasks (27 data-sets) and text classification tasks (9 data-sets) to show the effectiveness of our algorithm. We used the SentEval (Conneau and Kiela, 2018) toolkit for all our experiments. In all cases, sentences were represented as the mean of their words' embeddings and in the classification tasks, logistic regression was used as the classifier. Table 2 give an overview of the downstream tasks we evaluated our reduced representation.

Table 2: Downstream Task Overview

Task	# of Datasets
Textual Classification Task	9
Sentence Similarity Task	5 (27)

Sentence Similarity Task: We further evaluate our algorithm against the baselines on the SemEval dataset (2012-2016) which involved 27 semantic textual similarity (STS) tasks (2012 - 2016) (Agirre et al., 2012), (Agirre et al., 2013), (Agirre et al., 2014), (Agirre et al., 2015), and (Agirre et al., 2016). The objectives of these tasks are to predict the similarity between two sentences. We used the average Spearmans rank correlation coefficient (Rho $\times$ 100) between the predicted scores and the ground-truth scores as the evaluation metric. Table 4 show performance of all reduction methods with varying reduction dimensions. Figure 3 - 8 compare performance of all reduction methods with varying reduction dimensions (in all these figures, X-axis represents the number of dimensions of reduced embeddings while Y-axis represents the score: Rho $\times$ 100). Similar to previous observations[8] our reduction technique outperforms all other baselines.

Textual Classification Task: We also performed the experiments on several textual classification tasks using SentEval (Conneau and Kiela, 2018) toolkit, which includes binary classification (MR, CR, SUBJ, MPQA), multiclass classification (SST-FG, TREC), entailment (SICK-E), semantic relatedness (STS-B) and Paraphrase detection (MRPC) tasks, across a range of domains (such as sentiment, product reviews etc). We observe that our embedding is effective on downstream classification tasks and can effectively reduce the input size and the model parameters without distinctively reducing the performance ($<= 1.0\%$). Table 3 compares the accuracy of reduced embeddings for multiple dimensions to the original embeddings on classification for several datasets. It can be clearly seen that the reduced embeddings at 200-D perform comparable to the orignal embeddings. The results confirm that [9] one can effectively reduce the input size and the model parameters without distinctively reducing the performance ($<= 1.0\%$).

3.3 Analysis and Discussion

The performance of reduced embeddings matches that of unreduced embeddings in a range of word

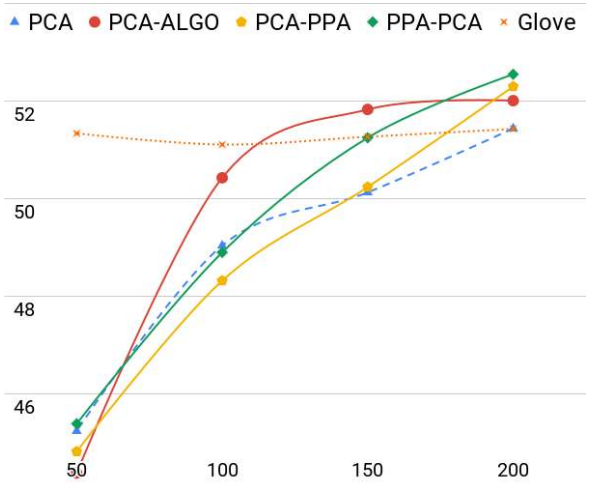

Figure 3: STS-12

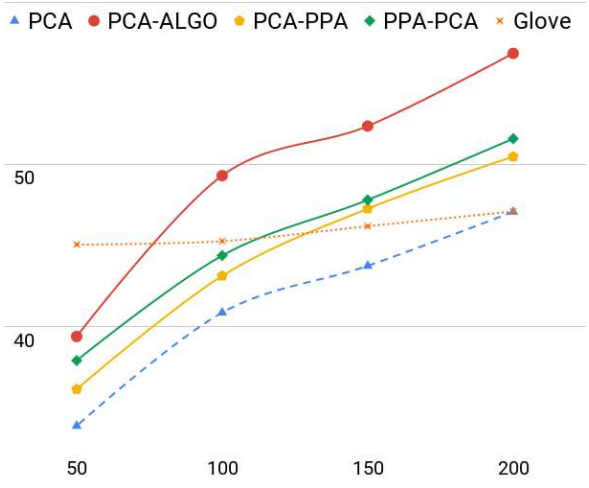

Figure 4: STS-13

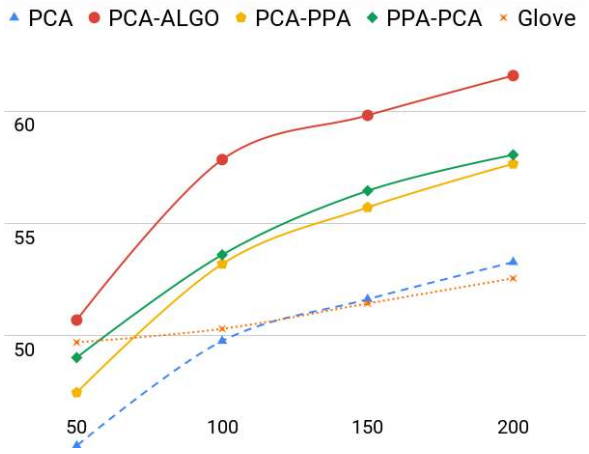

Figure 5: STS-14

[8] We only report the results with Glove embedding, however we obtain similar observations with other embeddings such as fasttText and word2vec. [9] We used the commonly used pre-trained 300 dimensional embeddings trained on wikipedia for our experiments. Lower-dimensional pre–trained embeddings trained on wikipedia were unavailable for most embedding types.

Table 3: Comparison of performance (in terms of test accuracy) on several classification datasets with original embeddings and the reduced embeddings obtained using the proposed algorithm. Bold represents the reduced embeddings performance with is within $<= 1\%$ of the original 300D dimensional embeddings.

Model	MR	CR	SUBJ	MPQA	STS-B	SST -FG	TREC	SICK-E	MRPC
Glove-300D	75.59	78.31	91.58	86.88	78.03	41	**68**	78.49	71.48
Algo-50D	66.52	70.49	85.6	77.5	68.92	35.48	50	73.25	71.01
Algo-100D	70.43	75.34	88.31	82.3	71.99	38.42	55	75.6	71.42
Algo-150D	73.45	77.43	89.86	85.59	76.33	40.18	59.6	76.76	71.54
Algo-200D	**75.23**	**78.17**	**90.61**	**86.51**	**78.09**	**41.36**	65.4	**77.35**	**73.1**
word2vec-300D	77.65	**79.26**	90.76	**88.3**	61.42	42.44	**83**	**78.24**	72.58
Algo-50D	71.84	72.79	88.1	83.53	54.89	39.37	64.6	73.03	71.07
Algo-100D	73.89	75.65	89.56	84.81	59.54	39.95	69	74.97	71.42
Algo-150D	75.88	77.06	**90.01**	86.13	**61.42**	41.27	73.4	76.42	71.19
Algo-200D	**76.77**	77.88	**90.15**	86.9	**61.5**	**41.45**	77.4	76.98	**71.77**
fastText-300D	78.23	80.4	92.52	87.67	**68.33**	45.02	**85.8**	**79.2**	73.04
Algo-50D	71.23	75.36	87.88	82.25	57.01	38.87	66.8	71.91	72.06
Algo-100D	73.94	77.64	89.88	84.34	62.67	40.86	72.8	74.79	73.16
Algo-150D	75.52	78.2	90.96	86.18	63.46	41.4	75.2	75.06	73.39
Algo-200D	**77.18**	**79.76**	**91.6**	**86.64**	64.32	**43.48**	77.4	76.76	**72.93**

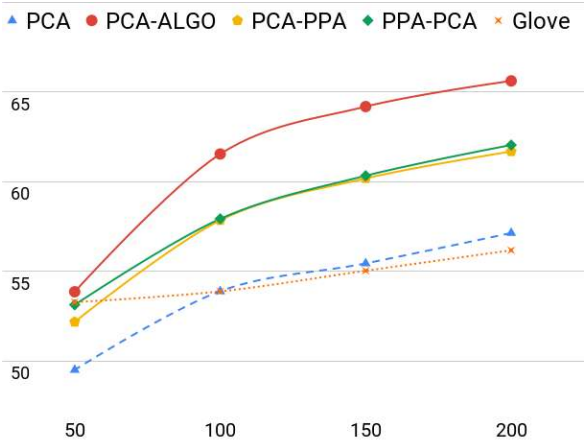

Figure 6: STS-15

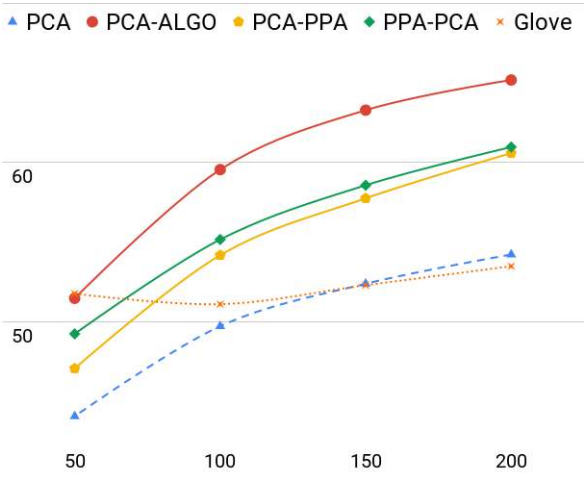

Figure 7: STS-16

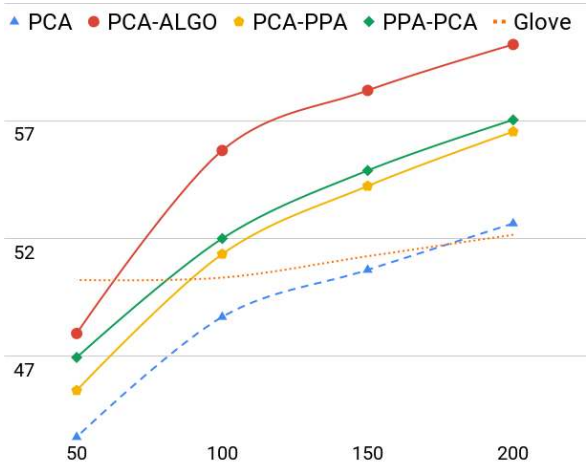

Figure 8: STS Average

and sentence similarity tasks. Considering the fact that semantic textual similarity (Majumder et al., 2016) is an important task across several fields, the resulting gains in efficiency, resulting from an efficient embeddings reduction, can prove useful to a range of applications. We obtain good performance on the similarity tasks since the proposed algorithm effectively exploits the geometry of the word embeddings (Mu and Viswanath, 2018) to reduce irrelevant noise in the word representations. In the sentence classification tasks, the reduced embeddings suffer from a slight performance loss in terms of test accuracy, which we suspect is due to the limitation of variance based techniques themselves in the context of word embeddings, i.e. owing to the disproprortionate distribution of linguistic features across the principal components themselves. Therefore, the loss of

STS-12	200	150	100	50
PCA	51.32	50.39	48.68	46.15
ALGO	**53.76**	**53.34**	**51.56**	**48.46**
PCA-PPA	51.76	51.26	49.12	42.45
PPA-PCA	51.91	50.91	49.27	46.73
Glove	51.42	51.26	51.1	51.33
STS-13	**200**	**150**	**100**	**50**
PCA	46.51	44.16	40.8	35.47
ALGO	**58.47**	**58.08**	**53.51**	**48.42**
PCA-PPA	56.71	54.26	50.11	43.46
PPA-PCA	56.98	55.12	52.43	45.94
Glove	47.1	46.19	45.27	45.06
STS-14	**200**	**150**	**100**	**50**
PCA	53.33	51.66	49.86	44.95
ALGO	**62.33**	**62.32**	**60.85**	**56.4**
PCA-PPA	61.62	61.04	58.59	54.24
PPA-PCA	61.66	61.13	58.84	54.67
Glove	52.56	51.44	50.31	49.71
STS-15	**200**	**150**	**100**	**50**
PCA	57.11	55.82	53.81	49.91
ALGO	**68.1**	**67.35**	**66.46**	**60.95**
PCA-PPA	65.49	64.78	61.17	56.23
PPA-PCA	65.48	64.92	62.49	57.55
Glove	56.18	55.04	53.89	53.29
STS-16	**200**	**150**	**100**	**50**
PCA	54.32	52.86	49.16	44.56
ALGO	**67.3**	**66.89**	**64.48**	**60.01**
PCA-PPA	65.06	63.86	60.07	55.27
PPA-PCA	65.36	64.12	61.16	54.89
Glove	53.52	52.33	51.14	51.8
STS Average	**200**	**150**	**100**	**50**
PCA	52.52	50.98	48.29	35.31
ALGO	**61.99**	**61.6**	**59.37**	**54.85**
PCA-PPA	60.13	59.04	55.81	50.33
PPA-PCA	60.28	59.24	56.84	51.96
Glove	52.16	51.25	50.34	50.24

Table 4: Performance in terms of (Rho $\times$ 100) between the predicted scores and the ground-truth scores for STS tasks.

information which is decorrelated with principal components (or the amount of variance explained) leads to decline in performance, since those properties of the embedding space are lost upon dimensionality reduction.

Another interesting analysis is to compare the performance of the reduced embeddings against state-of-the-art neural techniques on each of the datasets in Tables 3 and 4. In particular, the performance of the reduced embeddings of 200 dimensions (using Glove) obtained using the proposed algorithm suffers from an average drop of 4.1% in Spearman's Rank correlation scores (x 100) across the five sentence similarity datasets in Table 4, when compared against 4096 **(20X more)** dimensional sentence encoding obtained using InferSent [10] (Conneau et al., 2017). In the 9 sentence classification datasets, described in Table 3, the 200 dimensional reduced embeddings lead to an average drop of 7.3% in accuracy scores when compared against the 4096 dimensional Infersent encodings. If we exclude TREC (on which all pre-trained embeddings perform poorly), then the 200 dimensional embeddings lead to an average drop of 5.4% when compared against the 4096 dimensional InferSent encodings, across the remaining 8 sentence classification tasks in Table 3.

4 Comparison with Related Work

Most of the existing work on word embedding size reduction focuses on quantization (e.g. (Lam, 2018), which requires retraining with a different training objective), compression or limited precision training. In particular, (Ling et al., 2016) tries to reduce the embeddings' memory footprint by using limited precision representation during word embedding use and training while (Andrews, 2016) tries to compress word embeddings using different compression algorithms and (Shu and Nakayama, 2017) uses compositional coding approach for constructing embeddings with fewer parameters. There hasn't been much study on dimensionality reduction for word embeddings, with a general consensus on the use of publicly released pre-trained word embeddings of 300 dimensions, trained on large corpora (Yin and Shen, 2018). A recent work (Yin and Shen, 2018) has addressed the issue of exploring optimal dimensionality by changing the training objective, instead of dimensionality reduction. However, in this paper we mainly focus on the dimensionality reduction of the widely used pre-trained word embeddings (word2vec, Glove, fastText) and show that we can half the standard dimensionality by effectively exploiting the geometry of the embedding space. Therefore, our work is the first to extensively explore directly reducing the dimensionality of existing/pre-trained word embeddings, making it both different and complementary to the existing methods.

[10] https://github.com/facebookresearch/InferSent

5 Conclusions and Future Work

Empirical results show that our method is effective in constructing lower dimension word embeddings, having similar or (more often) better performance than the original embeddings. This could allow the use of word embeddings in memory-constrained environments. In future, an interesting area to explore would be the application of compressed and limited precision representations on top of dimensionality reduction to further reduce the size of the word embeddings. Deriving an algorithm to choose D, N and the levels of post-processing automatically, while optimizing for performance could also make the dimensionality reduction pipeline simpler for downstream applications. Further, owing to the growing popularity of contextualized embeddings such as ElMo (Peters et al., 2018) and BERT (Devlin et al., 2018), it would be interesting to explore whether the geometric intuitions used for developing the proposed algorithm for word embedding dimensionality reduction could be leveraged for contextualized embeddings as well.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, et al. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *(SemEval 2015)*, pages 252–263.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *(SemEval 2014)*, pages 81–91.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *(SemEval-2016)*, pages 497–511.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. * sem 2013 shared task: Semantic textual similarity. In *(SemEval 2013)*, volume 1, pages 32–43.

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *(SemEval 2012)*, pages 385–393. Association for Computational Linguistics.

Martin Andrews. 2016. Compressing word embeddings. In *International Conference on Neural Information Processing*, pages 413–422. Springer.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association of Computational Linguistics*, 5(1):135–146.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*, pages 4349–4357.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *NAACL-HLT 2015*, pages 1606–1615.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471.

Maximilian Lam. 2018. Word2bits-quantized word vectors. *arXiv preprint arXiv:1803.05651*.

Shaoshi Ling, Yangqiu Song, and Dan Roth. 2016. Word embeddings with limited memory. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 387–392.

Goutam Majumder, Partha Pakray, Alexander Gelbukh, and David Pinto. 2016. Semantic textual similarity methods, tools, and applications: A survey. *Computación y Sistemas*, 20(4):647–665.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Nikola Mrkšic, Diarmuid OSéaghdha, Blaise Thomson, Milica Gašic, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of NAACL-HLT*, pages 142–148.

Jiaqi Mu and Pramod Viswanath. 2018. All-but-the-top: Simple and effective postprocessing for word representations. In *International Conference on Learning Representations*.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2016. Neural-based noise filtering from word embeddings. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2699–2707.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Jonathon Shlens. 2014. A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*.

Raphael Shu and Hideki Nakayama. 2017. Compressing word embeddings via deep compositional code learning. In *International Conference on Neural Information Processing*, page arXiv:1711.01068. arXiv preprint.

Zi Yin and Yuanyuan Shen. 2018. On the dimensionality of word embedding. In *Advances in Neural Information Processing Systems*, pages 895–906.