Effective Dimensionality Reduction for Word Embeddings

Vikas Raunak [†] Vivek Gupta [‡] Florian Metze [†]

[†]Carnegie Mellon University [‡]University of Utah

Overview

- We present a dimensionality reduction algorithm to construct lower dimensional word embeddings by exploiting a geometric property of the embedding space.
- Empirical evaluations on several benchmarks (Word Similarity, Sentence Classification, Semantic Similarity) show that our algorithm efficiently reduces the embedding size upto 50% while obtaining similar or better performance than original embeddings.

Introduction

- Pre-trained word embeddings are used in several downstream applications as well as for constructing representations for sentences, paragraphs and documents.
- A prohibitive issue related with word embeddings is their size. For example, loading a word embedding matrix of 2.5M tokens takes up to 6 GB memory (for 300-dimensional vectors, on a 64-bit system).
- Reducing the size of word embeddings through dimensionality reduction can improve their utility in memory constrained devices, benefiting several real-world applications.
- 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 [1].
- The Post-Processing algorithm in [1] improves the embeddings by subtracting the (large) mean vector and by projecting the embeddings away from the common dominant directions.
- In this work, we present a novel algorithm that effectively combines PCA based dimensionality reduction with the post-processing algorithm in [1], to construct word embeddings of lower dimensions.

Analysis

A simple analysis of word embeddings (in terms of fraction of variance explained by the top principal components) is shown below:

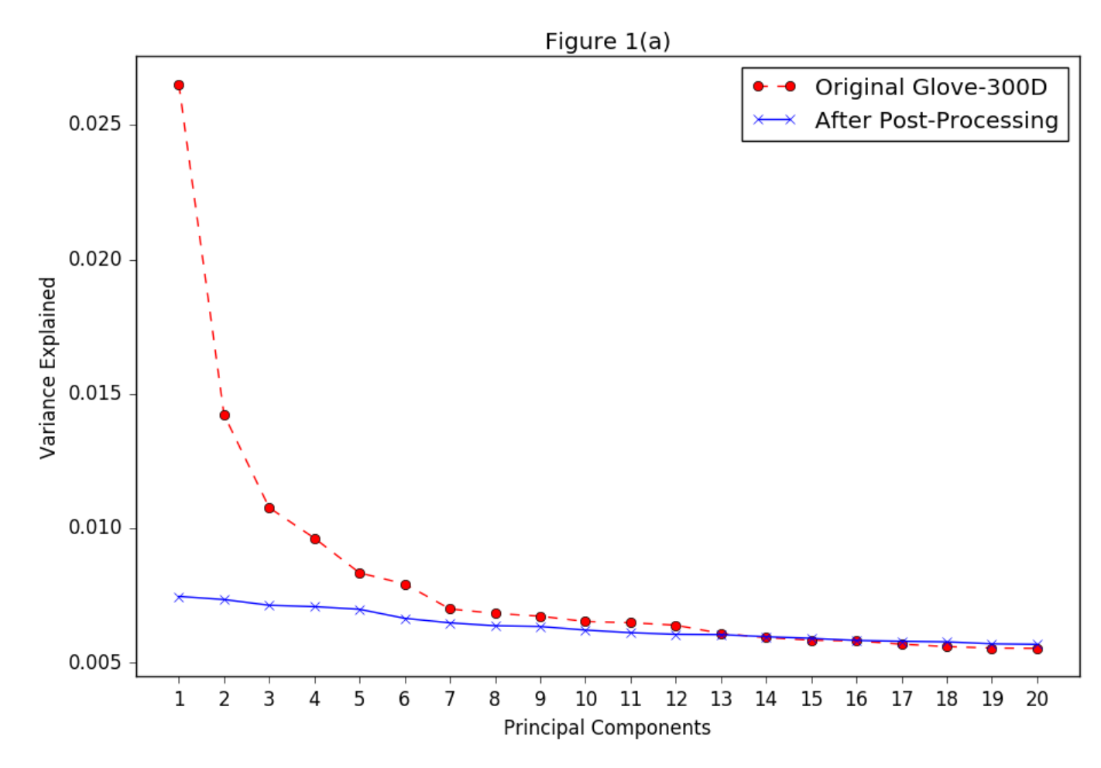

Figure 1: Comparison of the Original and Post-Processed Glove Embeddings (300-Dimensional) in terms of fraction of variance explained by top 20 Principal Components.

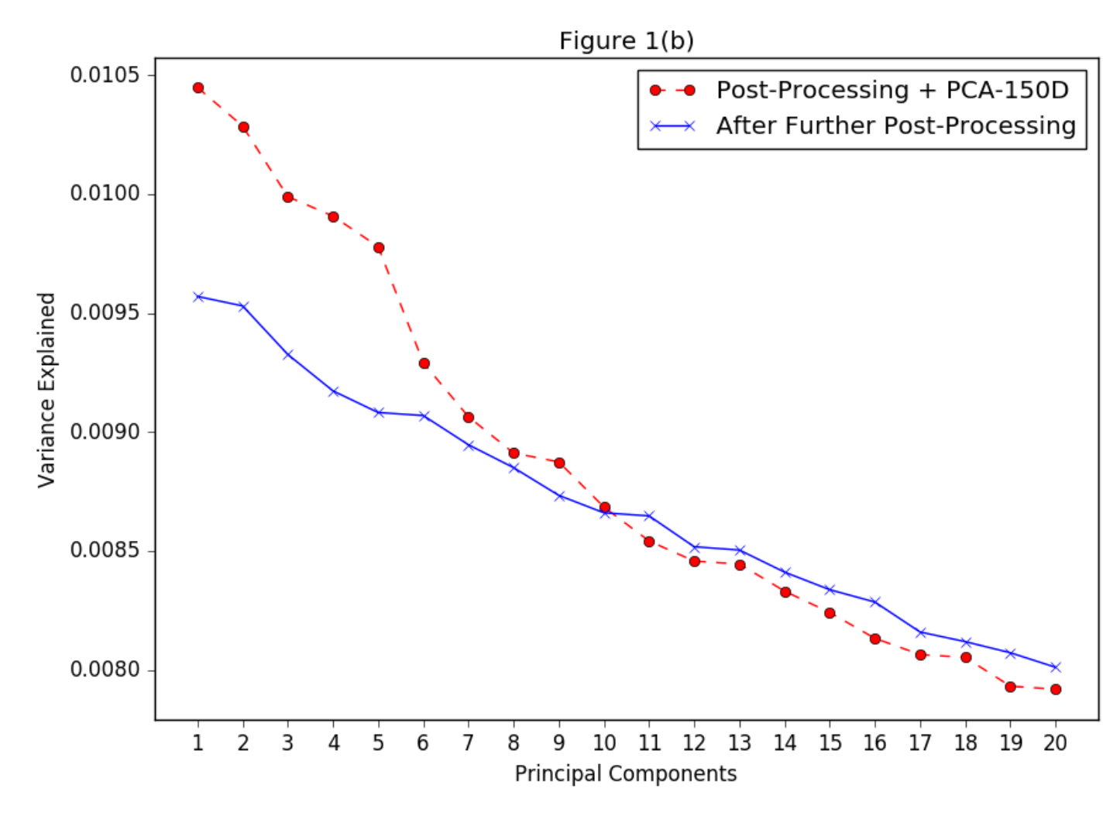

Figure 2: Comparison of the Post-Processing + PCA-150D Baseline and Further Post-Processed Glove Embeddings (150-Dimensional) in terms of fraction of variance explained by top 20 Principal Components.

Algorithm

The Post-Processing Algorithm (PPA)

```
Input: Word Embedding Matrix X, Threshold Parameter D.

1. Subtract the Mean:
     $X = X - mean(X)$.
2. Compute the PCA Components:
     $u_i = PCA(X)$, where $i = 1,2,...,d$.
3. Eliminate the Top D Components: $\forall$ v in X:
     $v = v - \sum_{i=1}^{D}(u_i^T \cdot v)u_i$

Output: Post-Processed Word Embedding Matrix X.

end
```

Algorithm

The Dimensionality Reduction Algorithm

```
Input: Word Embedding Matrix X, New Dimension N, Threshold Parameter D.

1. Apply the Post-Processing Algorithm:
     $X = PPA(X, D)$.
2. Transform X Using PCA:
     $X = PCA(X)$.
2. Apply the Post-Processing Algorithm:
     $X = PPA(X, D)$.

Output: Word Embedding Matrix of Reduced Dimension N: X.

end
```

Hence, our algorithm is simply to apply post-processing on either side of a PCA based dimensionality reduction procedure. The first post-processing is required since PCA should be applied on higher quality (base) embeddings. The second post-processing step is motivated by observation 1 from the analysis.

Observations from the Analysis

1. The property of common dominant directions re-emerges in lower dimensional embeddings even when PCA was applied on **already post-processed embeddings**.
2. 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 high dimensional embeddings. Hence, multiple levels of post-processing at different levels of dimensionality will yield diminishing returns as the influence of common dominant directions decrease on the word embeddings.

Baselines: To evaluate the performance of our algorithm, we compare it against different schemes of combining the post-processing algorithm with PCA. We consider the following baselines: PCA: Transform word vectors using PCA, PPA+PCA: Apply the algorithm 1 (PPA) and then transform word vectors using PCA, PCA+PPA: Transform word vectors using PCA and then apply the algorithm 1 (PPA).

Evaluation

Evaluations were done with Glove (100, 200, 300), word2vec (300) and fastText (300) embeddings on 12 word similarity datasets (metric is Spearman's rank correlation coefficient), 9 Sentence classification datasets and 5 Semantic Similarity (STS) Datasets.

1. **Word Similarity Benchmarks** 150D embeddings obtain 2-3% better performance than the original embeddings. 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 the 12 datasets.

2. **Sentence Classification Tasks** 200D embeddings obtain performance within 1% of the original embeddings (except TREC).

3. **Semantic Similarity Tasks**

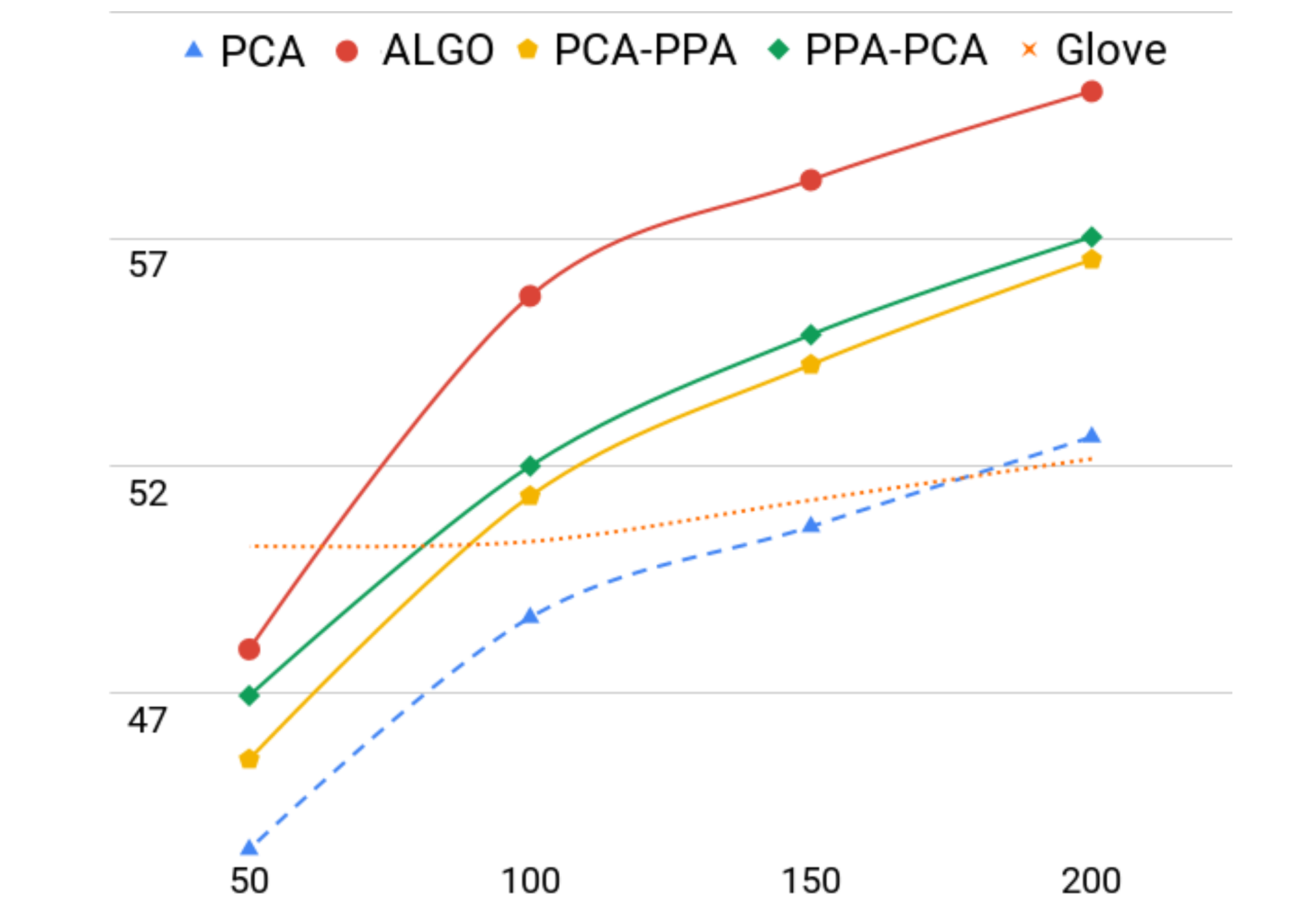

Figure 3: Baseline Comparison on STS 12-16 Tasks.

References

[1] Jiaqi Mu, Suma Bhat, and Pramod Viswanath. 2017. All-but-the-top: Simple and effective postprocessing for word representations. arXiv preprint arXiv:1702.01417.

https://github.com/vyraun/Half-Size