Low-Dimensional Embedding for Improved Training Performance

Submitted as final project report for the NLP course, IDC

Linoy Elimeleh (ID. 319122619), Roee Barak (ID. 206446221)

Semester B, August 2023

1 Introduction

When dealing with complex datasets, especially in the context of deep learning models, the input data can be high-dimensional, meaning it has a large number of features or dimensions. As the dimension increases, the computational resources required to process and train models grow significantly. Training deep learning models on high-dimensional data can be computationally expensive and time-consuming. In high-dimensional spaces, data points become sparse, which can lead to over-fitting or poor generalization performance. The model might struggle to find meaningful patterns in such data, and this adversely affects its ability to make accurate predictions on unseen samples.

To address these challenges, we use dimensional reduction techniques to map the data to a lower-dimensional space while preserving its essential information.

The key idea behind low-dimensional embedding is to learn a compressed representation of the data that captures its intrinsic structure, reducing redundancy and noise. By compressing the data into a lower-dimensional space, the model can efficiently process and learn from the data, leading to faster training times and improved performance.

In the context of natural language processing, word embedding is a well-known application of low-dimensional embedding. Word embedding represents words as dense, low-dimensional vectors in such a way that similar words have similar vector representations. We will show that our research of reducing word embeddings dimension can improve the performance of various NLP tasks, such as machine translation.

1.1 Related Works

Dimensionality reduction techniques have been used in many applications such as representing and capturing data intrinsic structure for feature selection (e.g. t-SNE, PCA, and other SVD-based techniques), storing essential data properties where storage or compute resources are limited (e.g. sketches such as

AMS sketch which was proposed in [1]) and other applications. Apart from SVD-based techniques and sketching, many techniques rely on random projections of the data into lower dimensions, one of the most famous is the Johnson–Lindenstrauss transformation. The random projection-based techniques mainly address the preservation of some intrinsic structure, in JL's case for example, the l_2 norm between any two vectors is approximately preserved when projected onto the lower dimension. Other techniques, such as those mentioned in [2] and [3] aim to preserve the cosine similarity between any two vectors in the projected space.

Dimensionality reduction of feature vectors in the context of deep learning has been demonstrated before in works such as [4]. Usually, the tasks are classification-based tasks where the input space is very large.

Works like [5] offer different ways to evaluate and compare different word embedding techniques. The techniques are divided into two categories - intrinsic methods, which are based solely on the embedding properties, and extrinsic methods, which are based on performing a downstream task and evaluating the adaptation of the data for the given task.

2 Solution

2.1 General Approach

In this project, we used two different dimensionality reduction techniques - the JL transform and the CSSRPL proposed in [3]. The JL lemma states that given $\epsilon \in (0,1)$ and a set $X \subset R^D$ there exists a linear map $f : R^D \to R^K$ where $K << D$ such that

$$(1-\epsilon)||v-u||_2 \leq ||f(v)-f(u)||_2 \leq (1+\epsilon)||v-u||_2$$

for all $u, v \in X$.

The CSSRPL algorithm is based on the SRP method (signed random projection) which was proposed in [2]. The SRP works as follows:

Let $A \in R^{K\times D}$ be a random linear operator where $\forall a_{ij} \sim N(0,1)$, $K << D$ and let $a, b \in R^D$. Now let $\alpha, \beta \in \{0,1\}^K$ be such that $\alpha_i = \mathbb{I}_{Aa_i>0}$ and $\beta_i = \mathbb{I}_{Ab_i>0}$ (meaning that the projection vectors get one when the product of the vector with the projection matrix row is positive).

Now if we take the Hamming distance between α, β and multiply it by $\frac{\pi}{K}$ we get in expectation the angle between a, b:

$$X = \tfrac{\pi}{K}\textstyle\sum_{i=1}^{K} X^{(i)} \text{ where } X^{(i)} = \mathbb{I}_{\alpha_i \neq \beta_i}\ ,\ \theta_{(a,b)} = \mathbb{E}[X].$$

which is equivalent to:

$$\theta_{(a,b)} = \frac{\pi}{K}||\alpha-\beta||_1$$

In the CSSRPL paper, the authors suggest further improvement to the original algorithm - instead of using the random operator $A \in R^{K\times D}$ where

$\forall a_{ij} \sim N\ (0, 1)$, they offer using a matrix $A' \in R^{K \times D}$ where each column has l random entries which are $\{\pm 1\}$ with probability $\frac{1}{2}$ and the rest of the matrix is 0. The rest of the algorithm stays the same - projecting the vectors, creating the indicator vectors, and computing their scaled Hamming distance.

The paper shows that this is an unbiased estimate of the cosine similarity of any two vectors, and furthermore, this result has a smaller variance than the SRP.

We apply these two techniques to word embedding and compare their performance (regarding training time and accuracy) to the original embedding.

2.2 Design

In the project, we perform two intrinsic evaluation tasks - word similarity and word analogy, and one extrinsic task - neural machine translation.

The word embedding we used in all tasks is the GloVe embedding. We first ran all three tasks with the original GloVe embedding (the 300d version) to get baseline results, and then we applied the two techniques and retrained the models to get the experimental results. In the following subsections, we will elaborate on each task.

Note that all tasks that required GPUs used a single T4 GPU with 16GB.

2.2.1 Word Similarity

In the word similarity task, we took two datasets of word pairs in which their similarities were ranked by different people (10 people for each pair) from 0 - not similar, to 10 - the same word.

The datasets are the WordSim353 dataset [2] and the Rare Words dataset [3]. We took the normalized average score, and for each pair, we calculated the cosine similarity of the embedded words. For the vectors w_x, w_y, the cosine similarity function is defined as:

$$cos(w_x, w_y) = \frac{w_x \cdot w_y}{||w_x||_2 ||w_y||_2}$$

We then used the MSE loss to estimate how well the original embedding performed in this task, and then we applied the two techniques in different dimensions and performed the task again. With the CSSRPL method, we also tried different values of $l = 2, 3, 4$ as the authors suggested these values. In order to decrease the variance, each experiment was conducted 10 times and the average result was taken.

2.2.2 Word Analogy

In the word analogy task, when given a pair of words a and a^* and a third word b, the analogy relationship between a and a^* can be used to find the corresponding word b^* to b. Mathematically, it is expressed as

$$a : a^* :: b : \_\_$$

where the blank is b^*. One example could be:

$$\text{write} : \text{writing} :: \text{read} : \text{reading}$$

The 3CosAdd function, which was proposed in [6] solves for b^* using the following equation:

$$b^* = argmax_{b'}(cos(b', a^* - a + b))$$

We used the Google Analogy Dataset 4 and again created a baseline with the original embeddings. We then applied the two techniques in different dimensions, and performed the task again. Again, we used $l = 2, 3, 4$ with the CSSRPL method. and performed each experiment 10 times to decrease the variance.

2.2.3 Neural Machine Translation

In this task, we built a transformer-based neural network that converted sentences from German to English. We used the GloVe embedding for the English sentences, and a German GloVe embedding which is available at 6.

The model consists of an embedding layer of German sentences, followed by a multi-headed transformer, and lastly a fully connected layer that outputs vectors in the English embedding space. All layer dimensions are either constant (such as the number of encoder-decoder layers) or a linear function of the embedding dimension (such as the number of heads in the transformer and the fully connected layer dimensions'). For example - when the embedding dimension was reduced to half (150 features), the total number of network parameters decreased roughly by a factor of 2.

We trained the model for 10 epochs, where we initialized the embedding layer with the pre-trained original embeddings to get a baseline model. We then transformed the embeddings using the optimal dimensions and value of l that we had found in the intrinsic tasks and initialized the embedding layer of a smaller network with the modified embeddings.

The dataset that was used for this task is the Multi30K dataset mentioned in 5.

3 Experimental results

3.1 Word Similarity

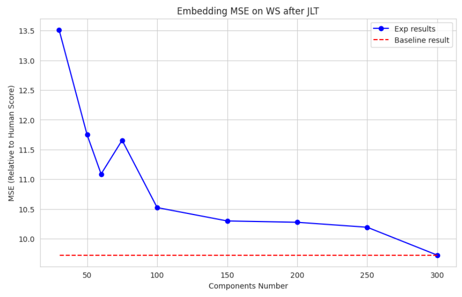
In this experiment, a comparative study of the GloVe embedding similarity scores on different datasets is conducted, with a focus on different dimensions. Our code demonstrates how different models were trained and evaluated, which can be found in 1.

The outcomes of the word similarity task indicate that the JLT and CSSRPL embeddings have a varying impact on different dimensions. Our analysis reveals that in some cases, the reduction of dimensions can result in an improvement in embedding similarity scores, while in others, it may lead to a decrease. These findings suggest that the effectiveness of dimensionality reduction techniques depends not only on the technique itself but also on the characteristics of the input data. Notably, our study demonstrates that CSSRPL embeddings outperform JLT embeddings in terms of cosine similarity preservation, particularly when considering different values of parameter l.

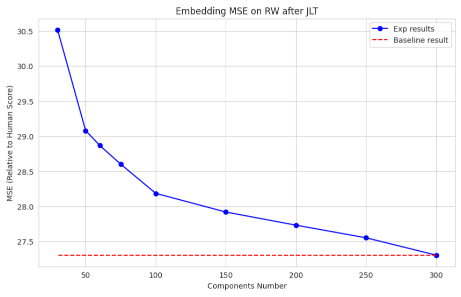
3.1.1 JLT Embedding Experiment

In this section, the Johnson-Lindenstrauss transformation (JLT) is applied to GloVe embeddings, and the cosine similarity of datasets is computed.

We can see in figures 1a and 1b how the mean squared error (MSE) of cosine similarity changes across different components for JLT embeddings, in comparison to the baseline result. We can see that for all the tested dimensions, the MSE loss decreases and as the dimension gets smaller, the information loss is greater and the error increases. On the other hand, decreasing the dimension generally yields shorter runtime.



(a) Embedding MSE on WS after JLT



(b) Embedding MSE on RW after JLT

Figure 1: Comparison of Embedding MSE on different datasets after JLT

We can see in 2a and in 2b the MSE loss and runtime changes as the number of components varies for JLT embeddings, alongside the baseline runtime on the Rare Words Dataset.

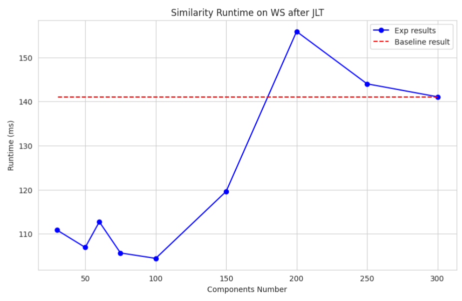
3.1.2 CSSRPL Embedding Experiment

In this section, the CSSRPL transformation is applied to GloVe embeddings, using different dimensions and different l values. We see in 3a and in 3b a set of plots demonstrating how MSE changes with varying components for different l values in CSSRPL embeddings, along with the baseline result.

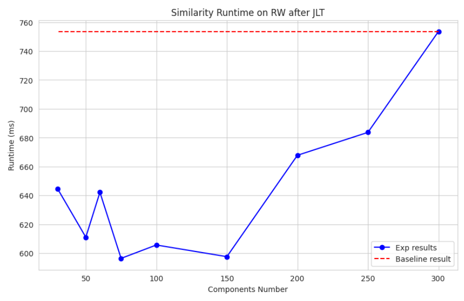
We can see in 4a and in 4b the results of the Rare Words Dataset.

It is clear that $l = 2$ yields the best results which is the value the authors of the paper suggested. Both the MSE score and the run-time decreased using this technique.

When comparing both techniques, the JLT is more sensitive to the dataset, whereas the CSSRPL performs well on both datasets.

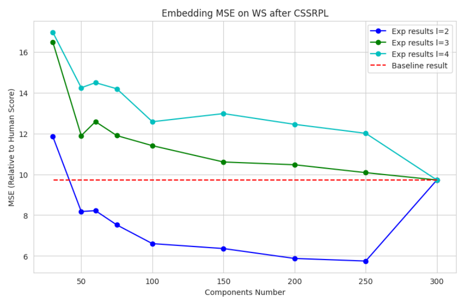


(a) Similarity Runtime on WS after JLT

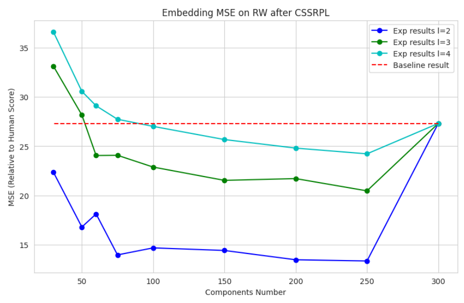


(b) Similarity Runtime on RW after JLT

Figure 2: Comparison of Similarity Runtime on different datasets after JLT



(a) Embedding MSE on WS after CSSRPL



(b) Embedding MSE on RW after CSSRPL

Figure 3: Comparison of Embedding MSE on different datasets after CSSRPL

3.2 Word Analogy

In this experiment, a comparative study of the GloVe embedding Analogy scores on different datasets is conducted, with a focus on different dimensions. Our code provided demonstrates how different models were trained and evaluated, which can be found in 2.

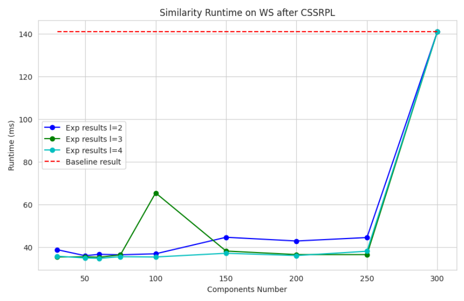
3.2.1 JLT Embedding Experiment

In this experiment, the Johnson-Lindenstrauss transformation (JLT) is applied to GloVe embeddings, followed by the computation of the 3CosAdd accuracy of datasets. We can see the Embedding Accuracy after JLT in figure 5a.

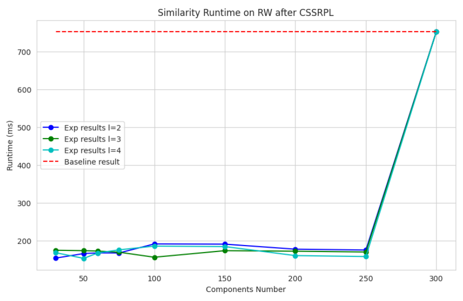
3.2.2 CSSRPL Embedding Experiment

In this experiment, the CSSRPL transformation is applied to GloVe embeddings. We can see the Embedding Accuracy after CSSRPL in figure 5b.

We can see that the baseline result is around 18% accuracy. The authors of [5] noted that most embeddings get a score of less than 30%. In both cases,

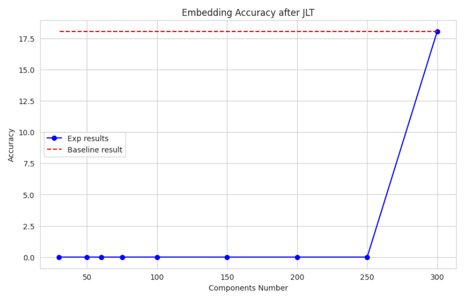


(a) Similarity Run-time on WS after CSS-RPL

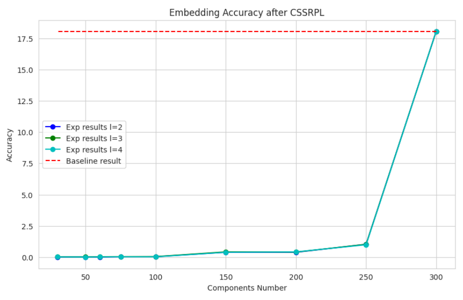


(b) Similarity Runtime on RW after CSS-RPL

Figure 4: Comparison of Similarity Runtime on different datasets after CSSRPL



(a) Embedding Accuracy after JLT



(b) Embedding Accuracy after CSSRPL

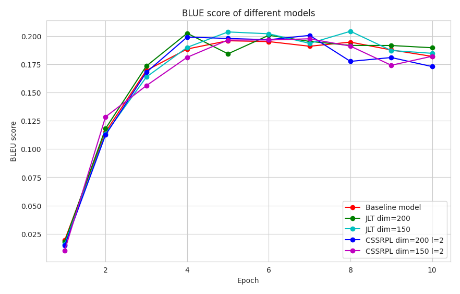
Figure 5: Comparison of Embedding Accuracy after different embedding techniques

we can see that the score of the transformed embedding is much lower than the baseline. A possible reason is the nature of the task - for each quadruple of words, the first three are compared against the entire vocabulary, and the most similar vector is selected. In both techniques the subspace is perturbed, therefore even a slight change in the cosine similarity may result in a different vector getting a higher score.

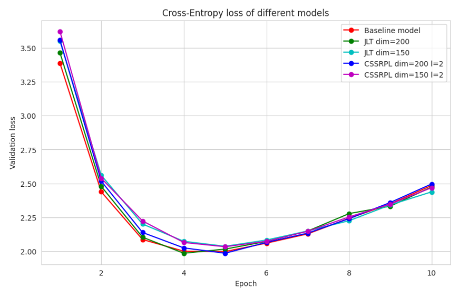
3.3 Neural Machine Translation

In this experiment, a comparative study of various Neural Machine Translation models is conducted, with a focus on different input dimensions and model sizes. Our code provided demonstrates how different models were trained and evaluated. It can be found in 3.

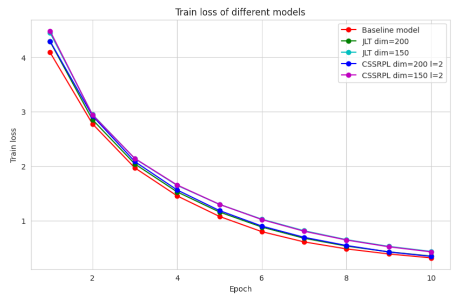
In this experiment the baseline model was trained along 4 other models - two input dimensions were examined (150 and 200) with the two methods. These dimensions were selected according to the results of the intrinsic tasks - they both improve the training duration without decreasing the accuracy significantly. We can see in 6a and in 6b that the modified embedding can yield better results than the baseline on the validation set. In 6c it can be seen that the training loss hasn't significantly changed and in 6d there is a slight improvement in the total training duration.



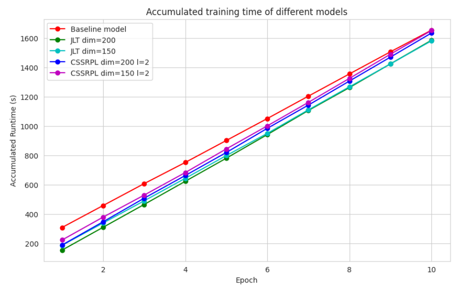
(a) BLUE score



(b) Cross-Entropy loss



(c) Train loss



(d) Accumulated training time

Figure 6: Neural Machine Translation Metrics

Model Type	Dimension	Params (M)	Epochs	Avg. Epoch Duration (s)	Test BLUE Score
Baseline	300	1.94	8	198.94	0.193
with JLT	150	0.79	8	156.99	0.207
with CSSRPL	200	1.13	7	161.24	0.211

Table 1: Model Comparison on Test Set

We can see from the table 1 that the best models take shorter to train (regarding average epoch run-time and an overall number of epochs), are much smaller, and achieve higher BLUE scores.

The outcomes of the neural machine translation (NMT) task illuminate the potential benefits of employing low-dimensional embeddings. Moreover, the experiment highlights that the CSSRPL technique, with carefully tuned parameters, can achieve favorable results in terms of training efficiency and translation quality.

4 Discussion

The findings of our study have broader implications for the field of natural language processing. In high-dimensional spaces, the computational complexity of NLP tasks can hinder efficiency and scalability. The observed trade-off between embedding dimensionality reduction and task performance emphasizes the importance of striking a balance between computational resource utilization and accuracy. Researchers and practitioners can leverage these insights to optimize embedding techniques for specific NLP applications, tailoring the dimension reduction process to match the requirements of the task at hand.

Furthermore, our study serves as a reminder that no single dimension reduction technique is universally superior. The choice of technique and its associated parameters should be informed by a thorough understanding of the data and task characteristics (as seen in the word analogy experiment). While CSSRPL shows better results than JLT, their efficacy may vary across different contexts, making it crucial to conduct thorough evaluations before implementation.

In conclusion, our research contributes to the ongoing discourse on low-dimensional embeddings in NLP by providing empirical evidence of their impact on training performance. The results emphasize the potential benefits of these techniques in terms of computational efficiency and accuracy. However, they also emphasize the need for careful consideration and evaluation when selecting a specific technique and determining the appropriate target dimension. In the future, we encourage further exploration and experimentation with other dimension reduction methods and their potential applications in diverse NLP tasks. Another approach may be incorporating these techniques into inner layers of the model, thus decreasing model size and required resources.

5 Demonstration

5.1 Code

Our code is provided in the following links:

1. Word Similarity Notebook: https://colab.research.google.com/drive/1y-_xS4KbvTZwVv2IYK90TgdbjOq9zcOr?usp=sharing
2. Word Analogy notebook: https://colab.research.google.com/drive/1kD7wtPPEbOMTB80z63_Vsg71iWYkiUKF?usp=sharing
3. Neural Machine Translation Notebook: https://colab.research.google.com/drive/1LI5oALyHUAhMWy544jBXU3NugokEBzRr#scrollTo=navK13KM73xP

5.2 Datasets and Pretrained Embeddings

1. GloVe embedding: https://nlp.stanford.edu/data/glove.6B.zip
2. WordSim353 dataset: https://gabrilovich.com/resources/data/wordsim353/wordsim353.zip
3. Rare Words dataset: http://www-nlp.stanford.edu/~lmthang/morphoNLM/rw.zip
4. Google analogy dataset: https://raw.githubusercontent.com/nicholas-leonard/word2vec/master/questions-words.txt
5. Multi30K dataset: https://github.com/multi30k/dataset/tree/master/data/task1/raw
6. German GloVe embedding: https://int-emb-glove-de-wiki.s3.eu-central-1.amazonaws.com/vectors.txt

References

[1] The space complexity of approximating the frequency moments: https://www.tau.ac.il/~nogaa/PDFS/amsz4.pdf

[2] Similarity Estimation Techniques from Rounding Algorithms: https://www.cs.princeton.edu/courses/archive/spr04/cos598B/bib/CharikarEstim.pdf

[3] Improving Sign-Random-Projection via Count Sketch: https://proceedings.mlr.press/v180/dubey22a/dubey22a.pdf

[4] Training neural networks on high-dimensional data using random projection: https://link.springer.com/content/pdf/10.1007/s10044-018-0697-0.pdf?pdf=button

[5] Evaluating word embedding models: methods and experimental results: https://www.cambridge.org/core/services/aop-cambridge-core/content/view/EDF43F837150B94E71DBB36B28B85E79/S204877031900012Xa.pdf/div-class-title-evaluating-word-embedding-models-methods-and-experimental-results-div.pdf

[6] Linguistic Regularities in Continuous Space Word Representations https://aclanthology.org/N13-1090.pdf