# Embedding Study Report

### Introduction

Pre-trained word embeddings are considered efficient and valuable for improving performance in natural language processing tasks. In this study, we focus on how the pre-trained embeddings can help improve performances for sentiment analysis tasks. Specifically, we want to prove that pre-trained embedding help reduces the ambiguity and uncertainty of unknown tokens in the validation and test set with respect to the training set.

We first built the classification models with the same architecture, one of which is from scratch with embeddings randomly initialized, and the other is with pre-trained embeddings. We built architecture using two layers of bi-directional LSTM with attention.

### Data Preparation

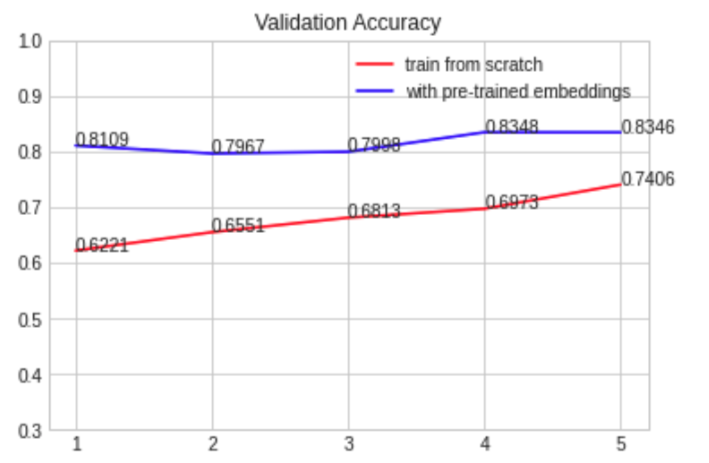
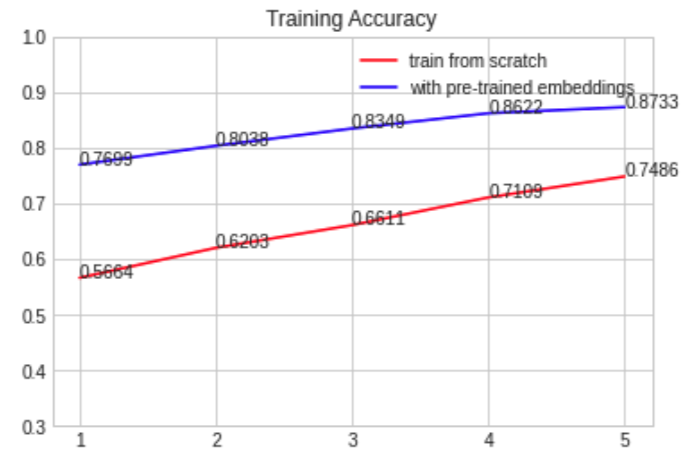
The dataset we used is the extracted SST-2 that contains 9,613 sentences in movie reviews with label “0” or “1” indicating “positive” and “negative”. The training, validation, and test sets have been split, with 6920, 872, 1821 examples respectively. We built the vocabulary from the training set, one with pre-trained word embedding “glove.6B.100d” and one without. We found out that there were 13824 tokens in total, which was reasonably small.

### Build the Model

Then we built the model that has two layers of bi-directional LSTM with attention. We defined the layers of the module. Our layers are an embedding layer, two layers of LSTM, a linear layer, an attention layer, and a layer of dropout to consider regularization. Before we pass our embeddings to the RNN, we packed them, causing our RNN to only process the non-padded elements of our sequence. We then unpacked the output sequence to transform it from a packed sequence to a tensor. The instance of our model class has an embedding dimension of 100, a hidden dimension of 256, and a dropout of 0.5.

For the model trained from scratch, the embedding layer will initialize the weight randomly. While for the model with pre-trained embeddings, we replace the initial weights of the embedding layer to be the “glove.6B.100d” embeddings.

When we trained the model, we used Adam optimizer and define the binary accuracy for evaluation. We trained our model with batch size equals to 64 and 5 epochs. Below is the table that compares the validation and test accuracy using pre-trained word embeddings and training from scratch.



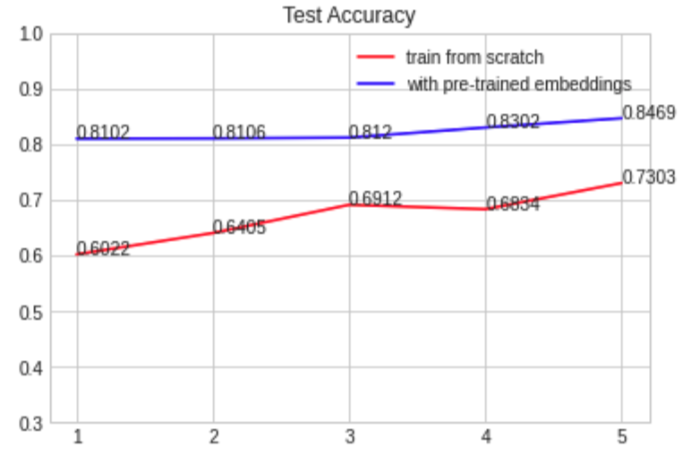


Figure 1. Train, validation and test accuracy on the whole dataset (x-axis: training epoch, y-axis: accuracy)

The accuracy can achieve up to around 84.69% for the model with pre-trained embeddings while only 73.03% for the model trained from scratch. Therefore, we found out that pre-trained embeddings improved accuracy significantly.

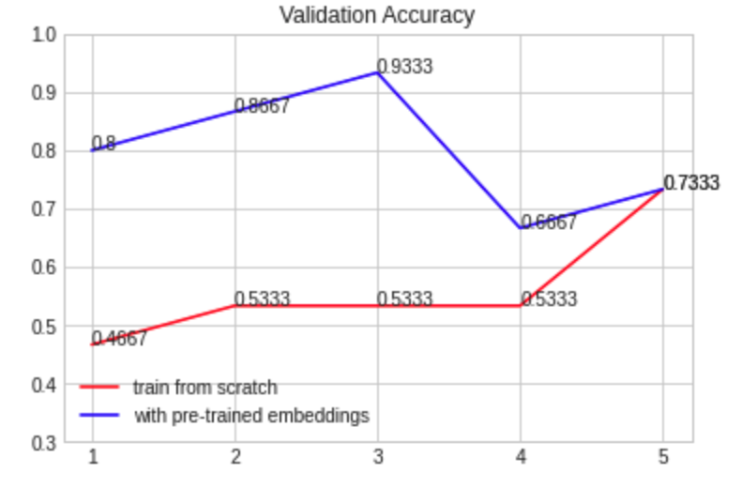
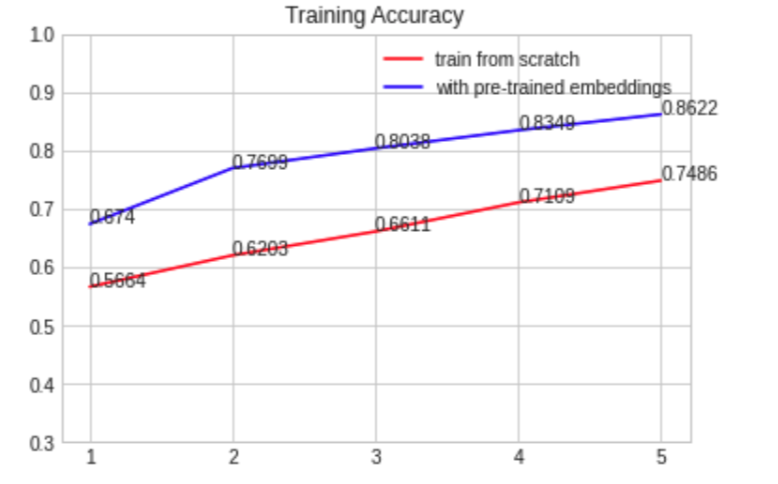
### Unknown Tokens Analysis

Then we examined the dataset and extracted two subsets for each of the validation set and the test set such that one subset has no unknown tokens with respect to the training set, the other subset has many unknown tokens. We then evaluated the models on each of the subsets. Below is the table that shows the statistics of each subset.

|  |  |  |
| --- | --- | --- |
| Subset name | Size (number of examples) | Number of unknown tokens in each sentence |
| val\_few\_unk.csv | 250 | 0 |
| test\_few\_unk.csv | 250 | 0 |
| val\_many\_unk.csv | 15 | > 4 |
| test\_many\_unk.csv | 41 | > 4 |

Tabel 1. Statistics of the Subsets

We figured out that the model trained from scratch performed poorly on the subset with many unknown tokens, while the model with pre-trained embeddings did much better. This is due to the notion of similarity that pre-trained word embeddings have. Below are the performance comparison figures that took the subset of examples with many unknown tokens as validation and test set.



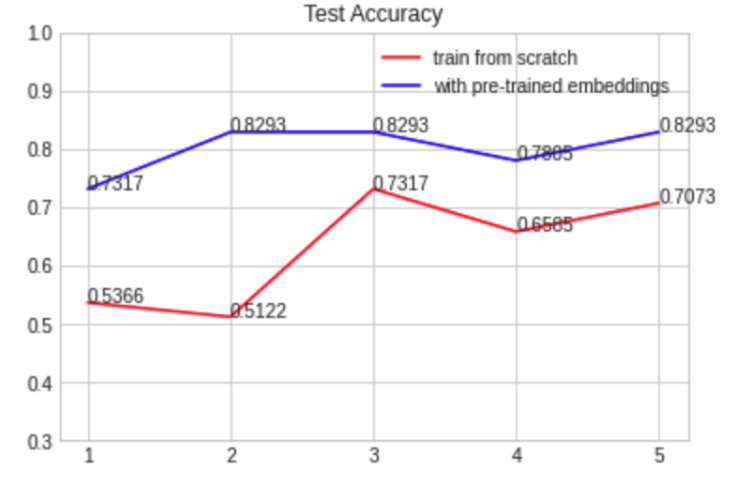
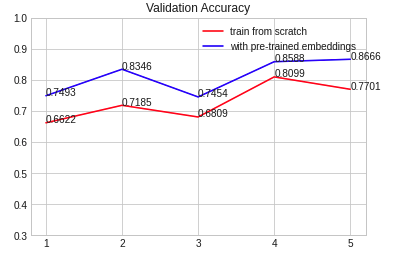
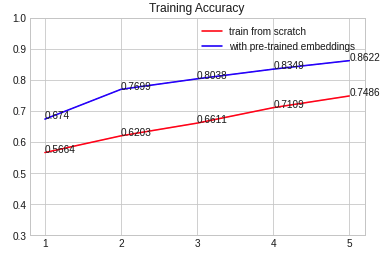


Figure 2. Train, validation and test accuracy on the subset with many unknown tokens in validation and test set (x-axis: training epoch, y-axis: accuracy)

For the subset with no unknown tokens, it is obvious that the pre-trained-embedding model performs better as well than the model trained from scratch. Below are the performance figures.



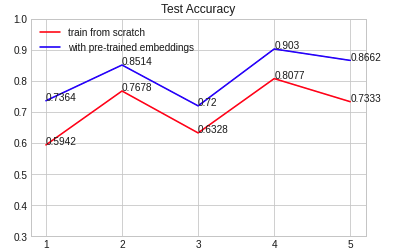


Figure 2. Train, validation and test accuracy on the subset with no unknown tokens in validation and test set (x-axis: training epoch, y-axis: accuracy)

In conclusion, the model with pre-trained embeddings perform generally much better than the model trained from scratch for sentiment analysis. We also showed that the pre-trained embeddings help improve the performance specifically when there are many unknown tokens in the validation and test sets with respect to the training set. In the future, we can analyze the token-level polarity scores to further evaluate the significance of pre-trained word embeddings.