Comparing pre-trained vs. Learned Embeddings for NLP Tasks

Minjae Lee, Ethan Lumbantobing, Grace Rarer, and Fuad Youssef

Abstract

This paper studies the impact of using pre-trained embeddings as opposed to learned embeddings on the accuracy and training time of 'Naive Bag of Words' (NBOW) natural language processing models. In addition, the impact of 'fine-tuning' these pre-trained embeddings is studied and compared. We found that the use of pre-trained embeddings without modifications to the embedding layer during training significantly reduced training time per epoch as expected, but resulted in drastically lower accuracy. Using pre-trained embeddings with gradient-descent fine-tuning during training results in equivalent training time per epoch to learned embeddings, but allows the model to acheive high accuracy in fewer epochs than randomly-initialized learned embeddings.

1 Introduction

In natural language processing, word embeddings are the vector representations associated with each word or token in a vocabulary. When creating a machine learning model, there are two ways to handle choose the embeddings: learned and pre-trained. With learned embeddings, the model starts out with randomly-initialized vectors for each word in the vocabulary, and learns a usable vector representation through gradient descent. With pre-trained embeddings, algorithms like Word2Vec and GloVe are used to learn a general-purpose set of embeddings that can then be used in any model, potentially removing the need for gradient-descent updates of the embeddings during the training of a specific model.

2 Process

As a simple benchmark natural language classification task, we use the Enron-Spam dataset first used in the paper "Spam Filtering with Naive Bayes - Which Naive Bayes?" (Metsis et al., 2006). Neural Bag-of-Words(NBOW) models, which are based on the average embedding vector of the tokens in each document, work well on this task. To measure the differences between models using pre-trained or learned word embeddings, we created three PyTorch NBOW models for comparison: an NBOW model initialized with random embedding vectors that learns embeddings through training, an NBOW model that uses the GloVe pre-trained embedding vectors without fine-tuning, and an NBOW model using the same GloVe pre-trained embedding vectors with fine-tuning.

The specific GloVe embeddings used were the GloVe 42B 300d embeddings generated from the Common Crawl dataset (Pennington et al., 2014). When we generate the embedding tensor from the Glove 42B embeddings, words in the Enron-spam vocabulary not included in the GloVe vocabulary are mapped to a zero vector. To be comparable to the GloVe-based models, the learned embeddings NBOW model also uses an embedding dimension of 300.

To measure the accuracy and training time of these three model types, we ran 5 trials of ten epochs each training these models, using 90% of the Enron-spam dataset as training data and the remaining 10% as validation data to measure accuracy. These trails were run on a Google Colab runtime with GPU acceleration; to minimize the impact that variations in available Colab resources would have on average training time, we rotate between model types rather than running the same model type five times in a row. To minimize the impact of outliers when evaluating the best achieved accuracy of a model, we take the median of the validation-data accuracy scores over the five trials for each model type.

3 Results

After implementing and training the three models, we ended up with the following results: using pre-trained embeddings compared to learned embeddings results in a roughly 73% reduction in training time per-epoch, which can be observed from the average training time per epoch graph found in Figure 1. However, as can be seen from Figure 2, the cost of this time save is a 22% reduction in accuracy. Fine-tuning these embeddings during the training phase results in the learned embedding and pre-trained embedding models having nearly identical training time per epoch and accuracy.

Figure 1: Model average training time per epoch

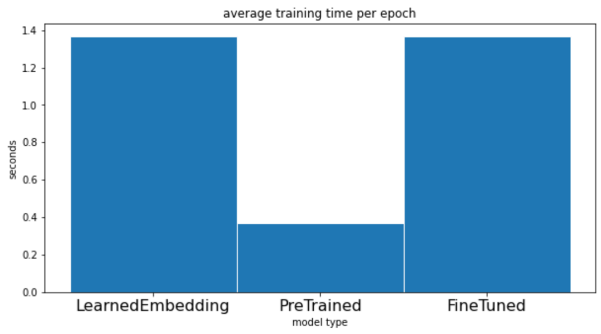
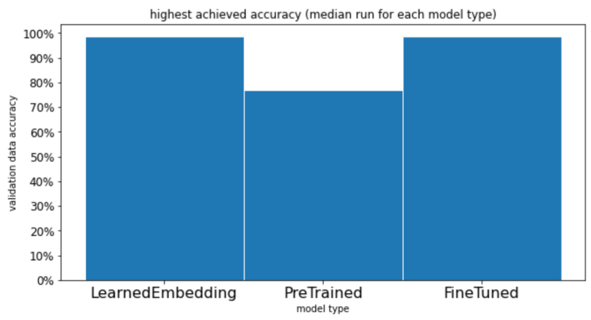


Figure 2: Model highest accuracy achieved



After comparing the accuracy over time for the three models in Figure 3, the fine-tuned pre-trained model was able to reach the same level of accuracy in six epochs as the learned model could in nine.

With these results, the question arose as to why the pre-trained model without fine-tuning performed so poorly. We decided to investigate the existing vocabulary to determine if there were any words that the model with pre-trained embeddings did not recognize. What we found is that 25.51% of the vocabulary in the Enron dataset were not present in the GloVe Common Crawl embeddings that we used for this project. Many of the missing terms appeared to be either typos (e.g. "sofftwaares", "technoiogy", "resuits"), Enron-specific vocabulary and jargon (e.g. "enronxgate", "eastrans", "reliantenergy"), or numbers.

Because our pre-trained embeddings did not include vectors for these tokens, our model was unable to learn nearly a quarter of the words without fine-tuning. This provides a likely cause as to why the pre-trained model that made no modifications to its embeddings during the training phase drastically under-performed in comparison to the other two models that did train its embeddings.

4 Conclusion

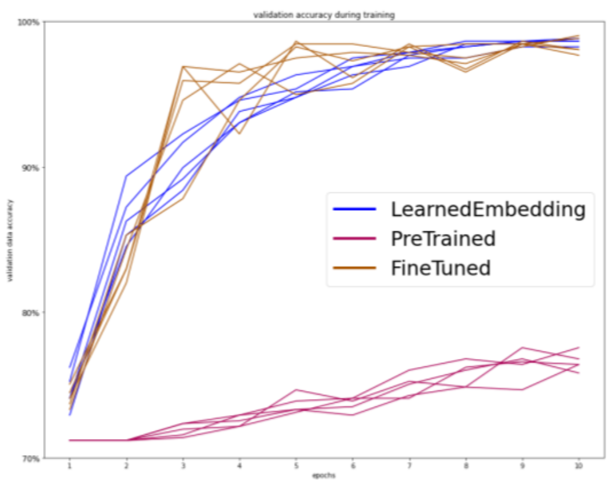
We had expected that using pre-trained embeddings without fine-tuning would greatly reduce training time because, in the learned-embeddings and fine-tuned models, the embedding layer makes up most of the parameters that have to be updated during training. However we had not anticipated how much worse the accuracy achieved by the pre-trained embeddings model would be. The lack of domain-specific vocabulary in the GloVe dataset likely accounts for a large part of the reduced accuracy. Typos are an important signal for noticing spam, and Enron-specific jargon are a valuable signal for detecting non-spam.

Another possible cause of the accuracy difference is simply that the GloVe embeddings are a general-purpose embedding generated by unsupervised learning, whereas then we learn an embedding for this specific task, all features of the embedding vectors will be trained to represent attributes that are meaningful to the task. This allows the model to be "smarter" given the same size of embedding vector.

It makes sense that the fine-tuned model takes fewer epochs to achieve high accuracy compared to the learned-embeddings model, since it starts out with a reasonable embedding layer instead of randomized embeddings that don't reflect any meaningful differences between tokens. However, we see that within 10 epochs the learned embeddings model catches up in terms of accuracy. Therefore for applications like this example where training time is already fairly short, there is little advantage to using the GloVe pre-trained embeddings rather than randomly initialized embeddings. In fact, any time savings that could be achieved by training the fine-tuned model for fewer epochs are far more than made up for by the very slow process of reading and processing the GloVe embeddings into a usable form.

Finally, one important note to keep in mind is that our experiment tested the effectiveness of an embedding pre-trained from the GloVe vocabulary

Figure 3: Model accuracy over epochs



in reference to the Enron-spam dataset. As stated earlier, many of the unknown words were occurrences of typos or vernacular specific to Enron vocabulary. While our tests did prove that pre-trained embeddings were unable to manage a large number of unknown words, the actual difference in accuracy when compared to real-world examples may be overstated in our results. Future work in this field of study can apply pre-trained embeddings to other datasets, where they might perform better (or worse) compared to the Enron-spam dataset. The relationship between the percentage of unknown words in a vocabulary and the performance of a pre-trained embedding is also an area of interest that can be further explored.

References

Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. 2006. Spam filtering with naive bayes – which naive bayes? *CEAS 2006 - Third Conference on Email and Anti-Spam.*

Martín Pellarolo. How to use pre-trained word embeddings in pytorch [online]. 2018.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation [online]. 2014.

5 Group Member Contribution Statement

Grace came up with the initial idea for our project topic; she also wrote the basic learned embeddings NBOW model and the code for running trial experiments and recording, processing, and graphing time and accuracy results, as well as integrating and refactoring the rest of the project code. Fuad was largely responsible for figuring out how to use pre-trained word embeddings and writing the pre-trained embedding NBOW model we used. In addition, he contributed to the first draft of the report. Ethan worked on the first draft of the report and on the editing process. Minjae modified the model to use various pre-trained embeddings and conducted experiments regarding amount and effect of missing vocabularies of pre-trained embeddings. He also helped build and debug the models and investigated the reasons behind the poor performance of the pre-trained embeddings model.