Embeddings from Language Mdels (ELMo)

REPORT

Ashmit Chamoli

Introduction

This repository contains an implementation of ELMo architecture using PyTorch. Instructions to use the code have been provided in the README.

Dataset

We use the News Classification Dataset which contains labeled news snippets. First, we pre-train ELMo on this dataset and then we use these embeddings to train an LSTM classifier to classify the news snippets.

Experiments

First we pre-train the ELMo architecture on the next word prediction task and run the model for 16 epochs with the following hyperparameters:

Embedding Size
512

We train 6 embedding models, 3 each for SVD and Word2Vec, varying context size from 2 to 4 with an embedding size of 300 and k=3 for Word2Vec.
We use the same classifier for all the 6 set of embeddings with the following hyperparameters:

Hidden Size	Num Layers	Bidirectional	Hidden Layers	Activation
256	3	True	[128, 64]	tanh

It was observed during the classifier training that the model has difficulty learning with svd embeddings as they are very noise prone. For this reason, the SVD classifier was trained for a more epochs than its Word2Vec counterpart.

Results

Word2Vec

Context size of 3 gives the best result for Word2Vec. Context size of 2 comes close in terms of scores but both 2 and 3 are much better than context size 4. Table 1 shows the scores for different context sizes.

Context Size	Accuracy	Precision	Recall	F1 Score
2	0.906	0.906	0.906	0.905
3	**0.911**	**0.912**	**0.912**	**0.911**
4	0.862	0.875	0.861	0.861

Table 1: Context size of 3 performs the best in all metrics.

One possible reason for context size 4 performing worse than the other context sizes could be the addition of noise in the context for each word, i.e. words with no direct relation with the target word appear in its context. Context size 3 performs better than context size 2 because of the additional information provided in the context without adding much noise. However, the performance upgrade is not substantial (0.5%)

Confusion matrices.

SVD

Table 2 shows the performance metrics for different context sizes.

Context Size	Accuracy	Precision	Recall	F1 Score
2	0.868	0.870	0.867	0.867
3	0.873	0.876	0.870	0.873
4	**0.882**	**0.885**	**0.878**	**0.881**

Table 2: Context size of 4 performs the best in all metrics.

For SVD, context size of 4 performs the best. The expectation is that the performance will increase more as the context size is increased further and then start decreasing.

Conclusion

We observe that Word2Vec embeddings achieve better performance than SVD embeddings. However, word2vec is computationally much mroe expensive and takes more time to train. Another advantage with SVD embeddings is that the time complexity does not increase much with the context size, which enables us to use relatively higher context sizes There is a trade off between accuracy and training time when choosing between SVD and Word2Vec and the choice of model hugely depends on the task at hand as well.