Poornima Andukuri

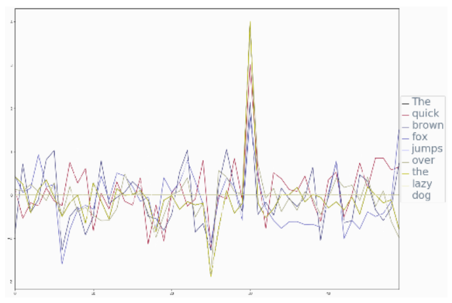
Introduction:

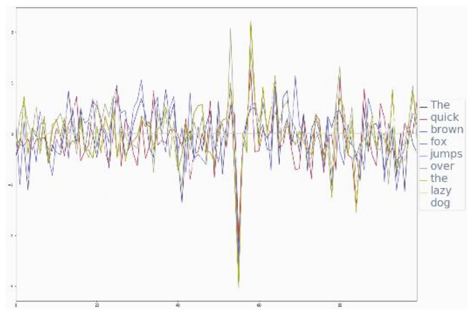
Assignment 8 is to create a sentiment analysis for which we create language models developed with pretrained vectors. We use sequences of words to train language models for classifying movie reviews as thumbs up or thumbs down. Use Tensorflow library to build recurrent neural networks (RNNs) models that uses the word vectors and vocabulary. The steps involved in the analysis are

1. Use two pretrained word embeddings to replace each word in a sentence of movie reviews
2. Use tensorflow to build RNN models
3. Use train data set to fit and evaluate the models using hidden layers.
4. Record the accuracy of models on train and test data sets.

Exploratory Data Analysis:

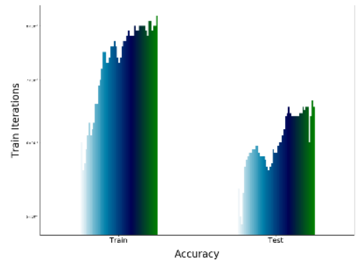
Two pre-trained word embeddings from GloVe (Global Vectors developed by researchers at Stanford University) for Word Representation, to transform the written language content into numeric representation. The encodings are used to transform a sequence of words or a sentence into a sequence of numeric vectors which can derive mathematical models. The sentence represented in the diagram, "The quick brown for jumps over the lazy dog" that contains all the letters in the English alphabet is an example of the data that is used in the model. The data used in the analysis, is a pre-selected sample of 1000 positive and negative movie reviews. The data is merged to a single series with a two-value output.



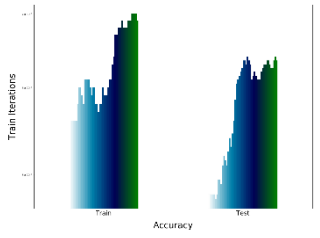


Data Preparation and Overview of Programming:

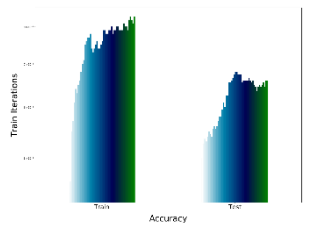
The GloVe embeddings dataset and the movie reviews are downloaded and loaded. The stop words were removed. The positive and negative movie reviews are loaded into a single numpy array with a value of 0 and 1. The data is split into train and test data set using scikit learn train_test_split. The two different RNN models created are BasicRNN, BasicLSTMCell with different number of layers and different learning rate. The two different embedding from 50 and 100 dimensions for words to run these models on the train and test data.



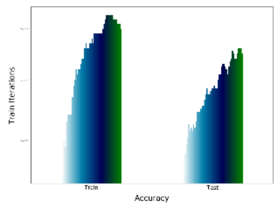
BasicRNN model with 50 dimensions
Train/Test Accuracy: 0.82/0.65



BasicLTSMwith 50 dimensions
Train/Test Accuracy: 0.69/0.63



BasicRNN model with 100 dimensions
Train/Test Accuracy: 0.83/0.66



BasicLTSM model with 100 dimensions
Train/Test Accuracy: 0.70/0.65

Insights & Conclusions:

Both models performed closely on the test set but with a huge difference on the training data set. Based on the accuracy of the models with 50 and 100 dimensions, I recommend using the BasicRNN model with 100 dimensions on the embeddings.

Appendix:

The ipynb notebook and an html version of the notebook along with the output and Kaggle submission scores are included in the submission.