NLP Disaster Tweets Using RNNs

Assignment #8

Michael Venit

Practical Machine Learning, MSDS 422

Background on the Data

For the purpose of this assignment, the data that was provided contained text in the form of tweets. These tweets can be categorized as either disaster or non-disaster messages based on specific diction and uniqueness in a given tweet.

Data Preparation

As it pertains to the data preparation stage, I took it upon myself to alter the training data so that the columns ‘id’ and ‘location’ were dropped as they contained too many missing values for them to be useful in classification. I then altered the ‘keyword’ column so that null values became the string ‘unknown’. I then added some new features to help in the exploration of the attributes of tweets that are specifically about disasters. These new features included word count, character count, average word length, unique word count and stop word count just to name some. The data was then cleaned by removing punctuation, words with less than 4 characters and non-alphabetic characters from the text. I then created a dictionary of contractions and other phrases that translate to said contractions to ensure that meaning was not lost when tokenizing the text. I then tokenized the text in order to properly use word embeddings. This is when I utilized pretrained words from GloVe to construct the aforementioned embeddings for the models that would be created. I then used two LSTM RNN (with LeakyRELU activation) and inserted the word embeddings created to classify our data. The difference between the LSTM models is that the first is an LSTM model with regularization and dropout, while the second model takes the output of the first layer and feeds that directly into the next layer. This is called a dual LSTM model.

Results/Evaluation

The first LSTM model performed well as I enacted early stopping. It generated an accuracy score of 83.8% while having a validation accuracy of 81.7%. As for the dual LSTM model, I thought it would perform better. This model had an accuracy score of 82.4%, which was worse than the first LSTM model. However, validation accuracy was higher as it produced a validation accuracy of 82.6%. I was not satisfied with these results which is when I explored other models to increase accuracy.

Model Performance and Recommendation

As it pertains to performance, the models performed similarly as they generated Kaggle scores of 80.3% and 80.5% respectively. My recommendation is to use the dual LSTM model, while continuing to adjust the number of hidden layers in order to increase the accuracy of such a classification model.