Predicting the Sentiment of Tweets using Long Short Term Memory Recurrent Neural Networks

Maxwell Lee

Viterbi School of Engineering, University of Southern California,

Los Angeles, California 90089, USA

(Dated: January 19, 2022)

Abstract

In this report, we will detail the process of creating a Long Short Term Memory (LSTM) Recurrent Neutal Network for the purpose of sentiment analysis of tweets. After processing the data and training the neural network, we are able to get an overall accuracy of 70% on a testing data set. Further, the errors our model makes are usually in a less problematic direction (ex. extremely negative tweets being labeled as simply negative). We can be confident that deploying this model can gather useful insights if we gathered similar tweets and performed this analysis.

I. INTRODUCTION

For this task, we have been asked to create a model to classify the sentiment of tweets about the ongoing Covid-19 pandemic. We are given 37,041 tweets and their associated sentiment on a scale from 0 to 4, where 0 is extremely negative and 4 is extremely positive. After reseraching the task, we have decided to pursue using a Long Short Term Memory recurrent neural network. This recent development in neural network research has shown to be extremely useful in areas of pattern recognition, which sentiment analysis certainly falls under. With this, we are able to achieve great results and a useful model we can be confident will succeed on further real world tweets.

II. DATA EXPLORATION

The first step in the data exploration process was to simply read a good amount of the tweets. This provided some insight into what pre-processing steps may be necessary to achieve our goal. These will be outlined in greater detail in Section III.

The next step was exploring the distribution of sentiment labels. The concern is that if very few tweets have a certain label, it may be difficult for our model to learn these labels. However, we have a reasonably even distribution of labels. The label with the fewest number of associated tweets is the extremely negative label, however this label still has roughly 5,000 associated tweets, only half of the positive label which has the greatest amount of associated tweets at roughly 10,000. This should pose no major issue for our task. However, if we wish to deploy this model on real world tweets, this should be a major consideration. If this data set was artificially created to ensure a relatively even distribution, that could greatly impact performance to a real world application.

III. DATA PREPROCESSING

Data processing is arguably the most important part of this task. We start by removing unnecessary text from the tweets. The first step is to replace all links (starting with www. or https://) with the word URL to signify the inclusion of a link. We do a similar process for a tweet tagging a user, where the @ is replaced with "at_user." The next step is to remove hashtags. We then replace escape and new line characters with spaces for consistent separation between two words. Next, we make all words be in lowercase, again for consistency. Finally, we remove all punctuation as it is not particularly relevant to sentiment. The text is now in an acceptable format to be tokenized.

The next step is to tokenize the words. This essentially means replacing each word with a number. We could add a step of either stemming or lemmatization to try to give forms of the same word the same tokens, however it is our hope that our word embedding process will achieve a similar result. We decide to only give the 2,500 most common words a token, as any less frequency than that will likely not be useful to our model.

We then create a vector for each tweet, where the word in the tweet is replaced with the assigned token. We then pad these sequences so that each vector is the same length and can be easily processed by our neural network.

It should be noted that at some point during this process we could make use of a word embedding to give similar words and similar tweets similar vectors. Instead, we decided to include this process as part of the learning task for our neural network.

The final step is to create training, validation, and testing data sets. Importantly, all of these sets are distinct with no overlap. The breakdown is 72% of total tweets used for training, 8% used for validation, and 20% used for testing.

IV. MODEL SETUP

As mentioned before, we will be using Long Short Term Memory as our model of choice. This has extremely strong pattern recognition capabilities, as it makes connections with inputs that are further back in "time." In our case, that time variable is the word position of the tweet. While a normal recurrent network may only make connections with the surrounding words, LSTM can make these connections through the entire sentence.

Additionally, we will be training a word embedding as a part of the training task. This will ideally map similar words to have similar vector representations, that are then placed in sequence and fed to the LSTM. It has been shown that for many tasks, an embedding vector of size between 100-500 is normally sufficient, so we chose an embedding dimension of 128. After the embedding phase we implement a dropout layer of .4.

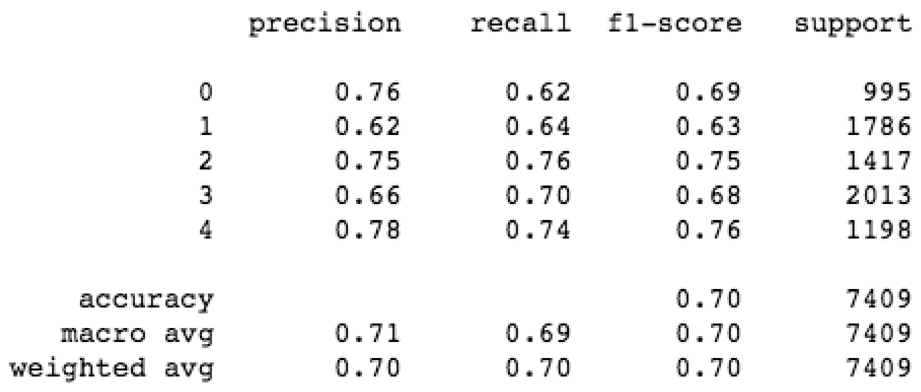
After the dropout layer, the vectors are passed to the LSTM layer. This has a recurrent dropout layer of .2 as well as an overall dropout layer of .2. We again choose an output dimension of 196 as that has been found to be a reasonable output dimension. Finally, these results are passed to a dense layer, which condenses the vectors into 5 dimensional vectors, representing the 5 labels. We use a softmax activation for this dense phase. The model is compiled using categorical cross-entropy as the loss function and adam as the optimization method.

We use a batch size of 32 and train the data for a maximum of 20 epochs. We have an early stopping criteria that if for three consecutive epochs the validation categorical cross-entropy does not improve, the training is halted. On my processor, each epoch takes roughly 3 minutes to train. This means training should take a maximum of roughly an hour, however the early halting stopped the process after 10 epochs, resulting in a training time of roughly 30 minutes.

V. MODEL EVALUATION

With relative ease, we are able to achieve an accuracy score of 70% on the testing set. While this score is impressive and a great achievement, the underlying results appear even better.

For each of our labels, the precision, recall, and F1 scores are all greater than 60%. This indicates that our model is fairly robust and not strongly biased for or against any individual label. The full classification report can be seen in Figure 1



```
              precision    recall  f1-score   support

           0       0.76      0.62      0.69       995
           1       0.62      0.64      0.63      1786
           2       0.75      0.76      0.75      1417
           3       0.66      0.70      0.68      2013
           4       0.78      0.74      0.76      1198

    accuracy                           0.70      7409
   macro avg       0.71      0.69      0.70      7409
weighted avg       0.70      0.70      0.70      7409
```

FIG. 1. The classification report on the testing set

The confusion matrix tells an equally encouraging story. Our model's incorrect guesses tend to be rather unproblematic. A problematic error would be classifying an extremely negative tweet as positive, or something similar to that effect. An unproblematic error would be labeling a positive tweet as either neutral or extremely positive. While of course we prefer correct classifications, we would also prefer our errors to be closer to truth. The confusion matrix seen in Figure 2. The code used to generate this plot was found at [1].

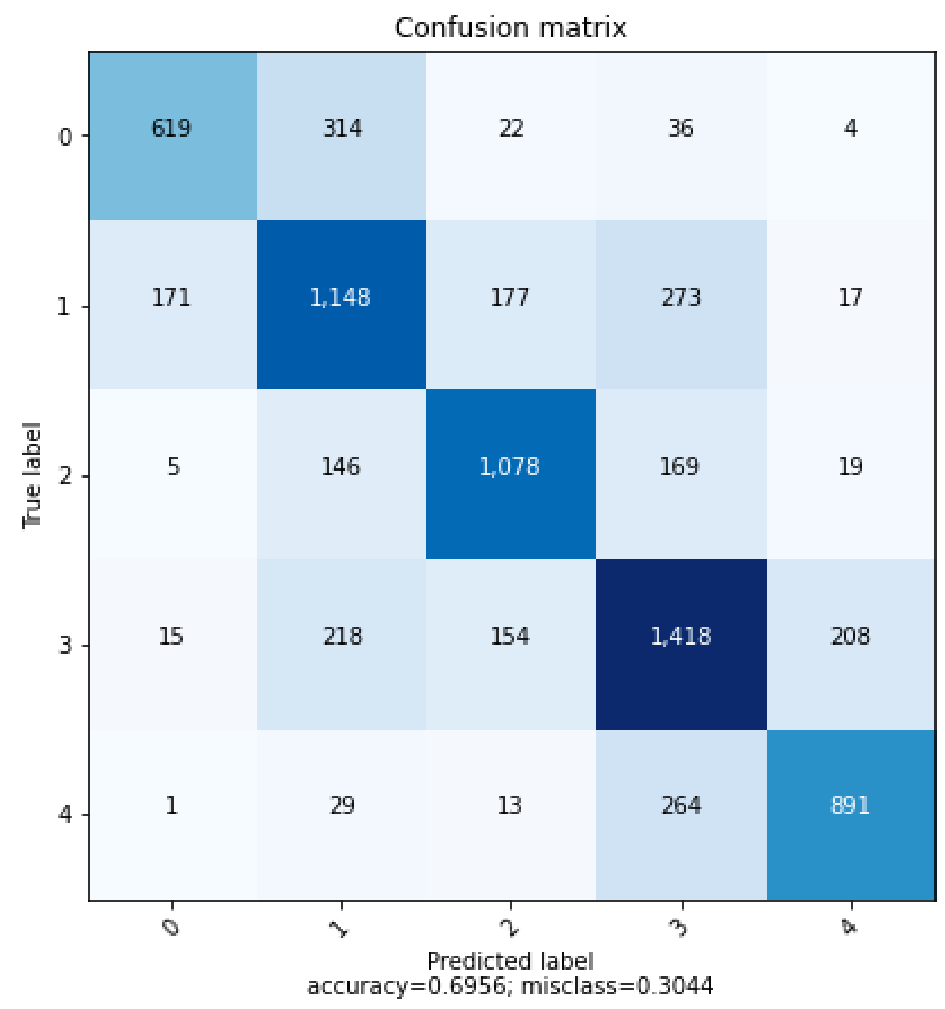


FIG. 2. The confusion matrix of the testing set

From this, we can see that while positive tweets are classified as negative a bit more frequently than we would like (more so than extremely positive), overall our confusion matrix looks very good. Overwhelmingly the prediction is correct and when it is wrong it is not dramatically wrong.

VI. CONCLUSIONS

With the given data, we are able to create an extremely successful model with a neural network that makes use of the latest developments in AI pattern recognition research. Additional steps would need to be taken before using this model to classify tweets in the real world. Our data set is rather limited, as it captures a specific moment in time about a specific topic. This model may not generalize well to tweets in a different time or about different topics. However, if our data set is a representative sample of all tweets about Covid-19 in this time frame, our performance on the testing data set should give us great confidence in our ability to classify the sentiment of those tweets.

[1] sklearn plot confusion matrix with labels (2019).