Twitter Sentiment Recognition

1st Jeff Matejka
jmmatejk@ncsu.edu

2nd John Mcdonald
jamcdon3@ncsu.edu

3rd Ryan Grainger
rpgraing@ncsu.edu

I. Introduction

Sentiment analysis is a useful tool in understanding peoples expressed opinion on social media. Social media platforms such as Twitter allow people to express their opinion about any topic, usually relevant to current popular trends [7]. Most sentiment analysis currently today use dictionary analysis to indicate whether the text is positive or negative, but the use of neural network techniques such as Recurrent Neural Network (RNN) can provide a more in-depth analysis, recognizing language patterns for more accurate predictions.

II. Dataset

With the use of datasets available on Kaggle of Twitter tweets for positive and negative sentiment classification, our goal was to train a network to be able to classify a tweet as positive or negative. References [3], [4], and [5] contain specific datasets that we used for training the model where each contain thousands to millions of real tweet text and its corresponding positive or negative label.

III. Methodology

A major challenge with working with Tweets is that they do not come in a standardized syntax or format since it is natural language. In order for the network to be able to learn, the tweets must be filtered to only useful and unique words, which are then vectorized into numbers. This is done by normalizing, tokenizing, and vectorizing the data. Adjusting the dataset to be learnable for a neural network is also known as data pre-processing. There are many ways to pre-process written language for neural networks, but we chose to use a straight-forward approach outlined in the next sections.

A. Text Normalizing

In order to normalize tweets, we start by removing any information that is not useful in determining sentiment. To begin, we removed twitter handles from tweets. This removes any twitter account mentioned in the tweet by filtering out any information beginning with the symbol '@' (Figure 1). The next step is to remove any hyper-links in the tweets. This will remove any links to photos or websites that are included in the data (Figure 2). Once handles and links are removed, we remove punctuation and numbers (Figure 3). After removing unhelpful information, we normalize the data itself by first making all characters lowercase. Then we removed 'stopwords' or extremely common and short words such as: 'a', 'the', 'to', etc. that would add to computation but not benefit the accuracy of the model in a significant way (Figure 4).

```
Original tweet:
@FrankieTheSats best interview with mcfly ever! dougie was adorable

Tweet after handle removal:
best interview with mcfly ever! dougie was adorable
```

Fig. 1. Twitter Handle Removal

```
Original tweet:
http://twitpic.com/3ky85 - Me and James hanging out. .. I love him

Tweet after link removal:
- Me and James hanging out. .. I love him
```

Fig. 2. Hyper-link Removal

```
Original tweet:
you'll always be my one and only... [12*23*2007]

Tweet after removing numbers and punctuation:
youll always be my one and only
```

Fig. 3. Remove Punctuation and Numbers

```
Original tweet:
thanks to all the haters up in my face all day

Tweet after removing stopwords:
thanks haters face day
```

Fig. 4. Lowercase and Removal of Stopwords

B. Tokenization and Vectorization

Tokenization is the process of taking a sentence and parsing all the words into 'tokens'. This result is then used in the Vectorization process which takes a tokenized sentence and returns a numerical vector based on the contents of the sentence (Figure 5).

```
Original tweet:
ok thats win

Tweet after Tokenization:

['ok', 'thats', 'win']
Tweet after Vectorization:

[  0   0   0   0   0   0   0  78  16 292]
```

Fig. 5. Tokenization and Vectorization

IV. Model

A. Deep Fully-Connected Neural Network

We initially tested several traditional models to see the performance and if they could successfully classify tweet sentiment. First, using a multi-layer neural network with 50% dropout between layers and using the vectorized tweets as input, we were able to train a network to classify tweets when the dataset contained only short tweets and a short total unique words dictionary, around 31 words long and 8000 words, respectively, in 20 epochs. The figure below shows the network structure, loss, and accuracy performance on a validation dataset.

We found that using this network with a large dataset containing much longer tweets and a huge total unique word count broke the model performance. The most likely cause is that the model is unable to converge since it is unable to remember patterns of words or that the word ordering may be slightly different amongst specific tweets.

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 31, 1000)	2000
dropout (Dropout)	(None, 31, 1000)	0
dense_1 (Dense)	(None, 31, 75)	75075
dropout_1 (Dropout)	(None, 31, 75)	0
dense_2 (Dense)	(None, 31, 50)	3800
dropout_2 (Dropout)	(None, 31, 50)	0
flatten (Flatten)	(None, 1550)	0
dense_3 (Dense)	(None, 100)	155100
dense_4 (Dense)	(None, 3)	303

Total params: 236,278
Trainable params: 236,278
Non-trainable params: 0

Fig. 6. Deep Fully Connected Network

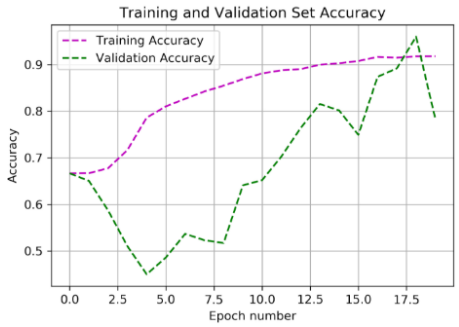

Fig. 7. Deep Fully Connected Network - Accuracy

We tested an autoencoder to 'de-noise' the input vectors, but this failed to produce any usable latent vectors since tweets the model couldn't produce readable text.

B. Final Model: Deep Bi-directional RNN

Final model selection includes a deep RNN that utilizes Bidirectional CuDNNLSTM and GRU for faster straining. This was essential for operating on a larger dataset. Details regarding layers, parameters, and structure is included in the graph below.

An important decision for model selection was the word embedding to build classification on top of. We chose ConceptNet Numberbatch which uses semantic vectors for mapping word meanings, and had proven applications in machine learning problems with high accuracy [8].

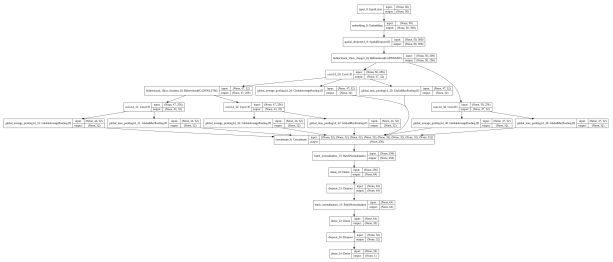

Fig. 8. Deep RNN Structure using CuDNNLSTM and GRU

V. Training and Hyper-Parameter Selection

We ran hyper-parameter testing across 10 epochs to gain a comprehensive understanding of how each variable affected model stability. Note that the batch size was kept at 4096; this was a constraint to keep step size small and training time low. High batch size can be justified, as the training set is large with 990,000 tweets.

Across the graphs we notice the following trends. A higher learning rate noticeably increased validation accuracy. Dropout rate did not affect model accuracy, but a higher value positively affected model stability across epochs. Spatial dropout also had a large impact on model instability, but lower values tended to mitigate variability. A higher number of Dense Units negatively impacted stability and decreased validation accuracy. Finally, a higher number of Units in the LSTM also negative impacted stability and reduced accuracy for the model. Support for this analysis is included in the following graphs. Our finalized hyper-parameter selection is included.

TABLE I
Finalized Hyper-Parameter Selection

Parameter	Value	Description
Learning Rate	10e-3	Rate of updating weights
Dropout Rate	0.3	Rate of ignoring units; regularization
Spatial Dropout Rate	0.4	Rate of ignoring entire features; regularization
Dense Units	64	Number of cells in dropout layer
Units	128	Number of cells of LSTM

VI. Evaluation and Testing

Using our finalized Hyper-Parameters over a larger number of epochs, we see that the model accuracy peaks around 78% with a minimal loss at 0.46. The test evaluation results gave 74.8% accuracy with a .504 loss. We are satisfied with these results, as human analysts tend to agree only 80% to 85% of the time. Due to this variability, our model accuracy was competitive with industry standards [7]. Also, it's important to note the data set we used was based off a large social media

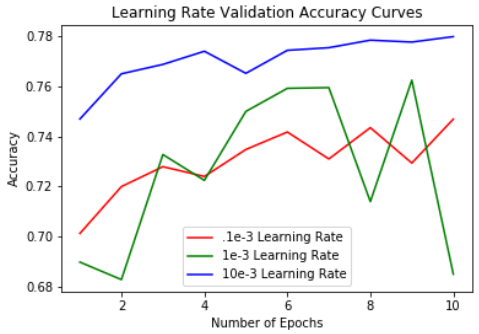

Fig. 9. Learning Rate vs Accuracy

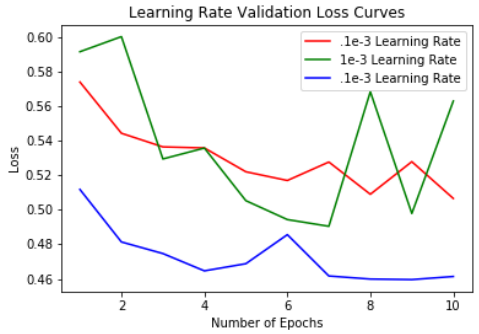

Fig. 10. Learning Rate vs Loss

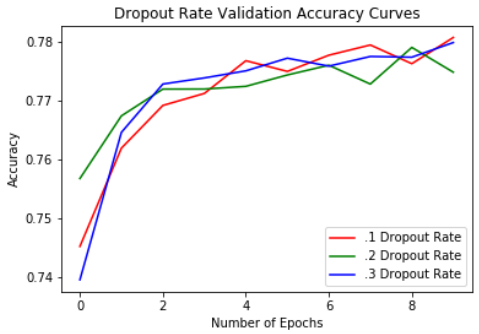

Fig. 11. Dropout Rate vs Accuracy

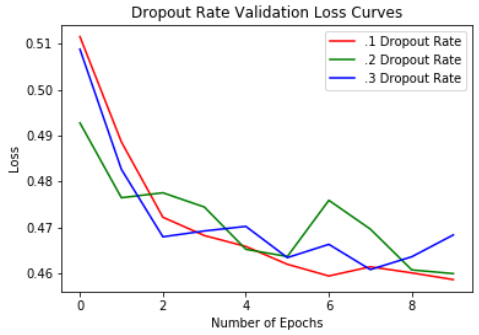

Fig. 12. Dropout Rate vs Loss

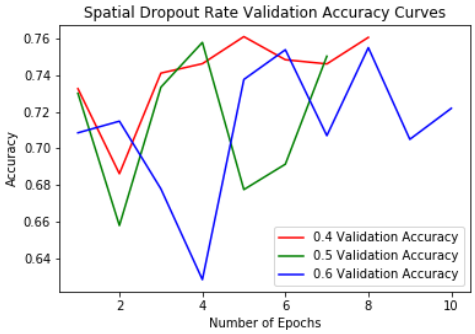

Fig. 13. Spatial Dropout Rate vs Accuracy

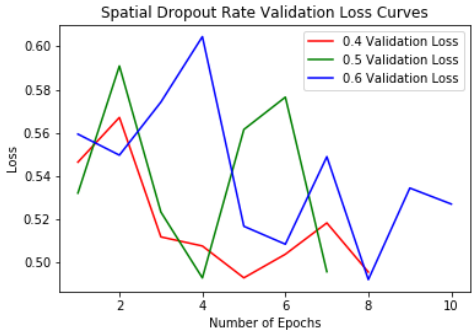

Fig. 14. Spatial Dropout Rate vs Loss

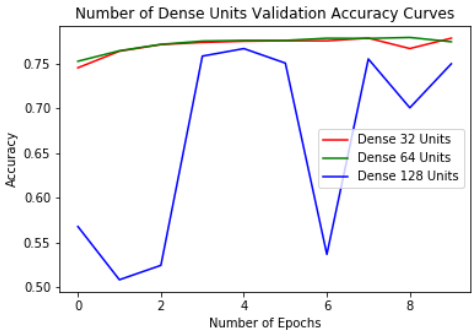

Fig. 15. Number of Dense Units vs Accuracy

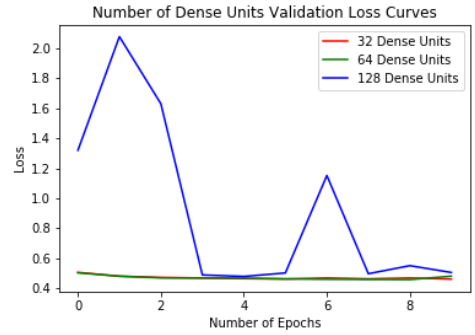

Fig. 16. Number of Dense Units vs Loss

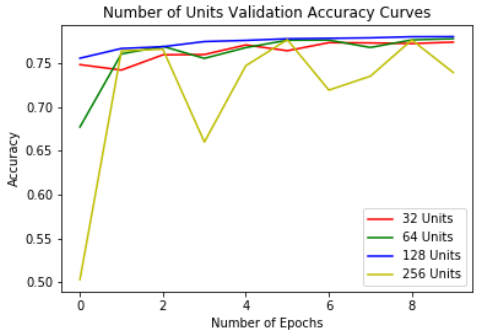

Fig. 17. Units vs Accuracy

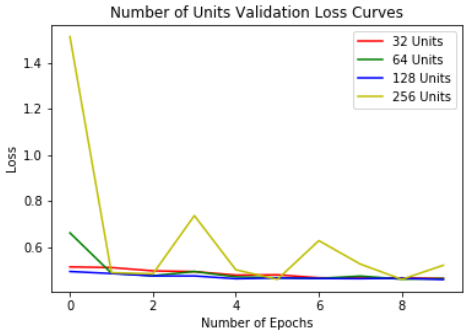

Fig. 18. Units vs Loss

based domain with variable semantics and colloquialisms. Including a more formal domain would likely improve sentiment analysis accuracy.

A potential adjustment to data sets to improve accuracy would be to include a third classification for neutral sentiment. Currently the dataset classifies neutral sentiment as positive, which can create higher losses in model prediction for tweets that do not clearly define emotional context. Alternatively we noticed that, for predictions, adding a round bias in the positive skew provided higher accuracy. This would be something to study in further analysis.

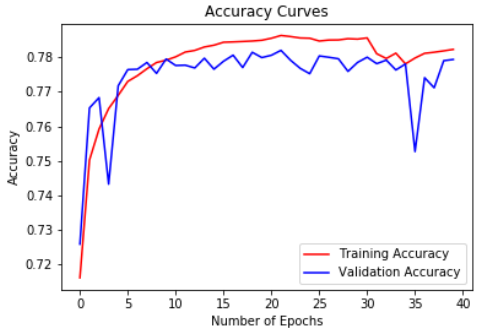

Fig. 19. Final Model Accuracy

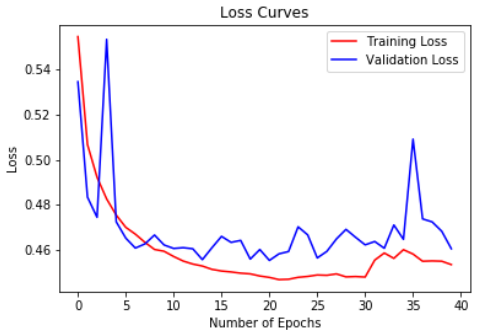

Fig. 20. Final Model Loss

VII. Results

The results show our predictions of sentiment on 320,725 tweets. A 1 indications positive or neutral sentiment; 0 indicates negative sentiment. An example of these predictions are included in Figure 20. Words such as ”rejection” and ”aww” easily skew the model for a negative sentiment, while words like ”Thank you” skew for positive. The harder examples for classification seem to be near the neutral category, and these we predict create most of the loss in our model predictions.

```
Pred : Tweet
0 : hectic crazy monday-ness.  at least its already after 1!
0 : @bisante awww why?  i kinda feel like that right now...
1 : "tomorrows agenda: campus visit/meeting
1 : @Christian0386 thankyouuu!  well it seems to be working! only 9 more followers to go! how long do we have left?
1 : "Did I say 'tires'?  I mean
1 : Should I go to there? Hmmmmmmmmm.....
1 : @graff_king: not sure what it means but I'm thinking its a good thing  Mahalo !!  http://myloc.me/2Ex1
1 : @svyongue but I have this really strong feeling that the Jonas Brothers will be there  maybe that'll convince Heidi &amp  Maggie to go! lol
0 : Rejection comes in many forms and I guess I need to swallow the pill.
1 : @Kellie0309 Okay night!  Hope the kiddos stay asleep for ya  Try not to have nightmares of Joes brows comin to get ya!
```

Fig. 21. Keras Model Predictions on Unlabeled Set

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, Deep learning. Cambridge (EE. UU.): MIT Press, 2016.

[2] “Twitter Sentiment Analysis using Machine Learning on Python,” Pantech Solutions. [Online]. Available: https://www.pantechsolutions.net/twitter-sentiment-analysis-using-machine-learning-algorithms-on-python. [Accessed: 01-Nov-2019].

[3] S. Yadav, “Pre-processed Twitter tweets,” Kaggle, 17-May-2017. [Online]. Available:https: //www.kaggle.com/shashank1558/preprocessed-twitter-tweets#processedNegative.csv. [Accessed: 01-Nov-2019].

[4] “Twitter sentiment analysis,” Kaggle. [Online]. Available: https://www.kaggle.com/c/twitter-sentiment-analysis2/data. [Accessed: 01-Nov-2019].

[5] S. Belkacem, “Twitter: Sentiment Analysis data,” Kaggle, 20-Oct-2018. [Online]. Available:

[6] E. Martinez-Camara, M. Martin-Valdiva, L. Urena-Lopez, and A. Montejo-Raez, “Sentiment analysis in Twitter,” Natural Language Engineering, vol. 20, no. 1, pp. 1–28, 2014.

[7] P. Barba, “Sentiment Accuracy: Explaining the Baseline and How to Test It,” Lexalytics, 28-Aug-2019. [Online]. Available: https://www.lexalytics.com/lexablog/sentiment-accuracy-baseline-testing. [Accessed: 13-Dec-2019].

[8] Commonsense, “commonsense/conceptnet-numberbatch,” GitHub, 11-Dec-2019. [Online]. Available: https://github.com/commonsense/conceptnet-numberbatch. [Accessed: 13-Dec-2019].

Appendix A:

Figure 8: Final Model - Deep RNN Structure using CuDNNLSTM and GRU