**Comparison of LSTM RNN and Batch Perceptron in classifying Fake News Titles**

Daniel Gural

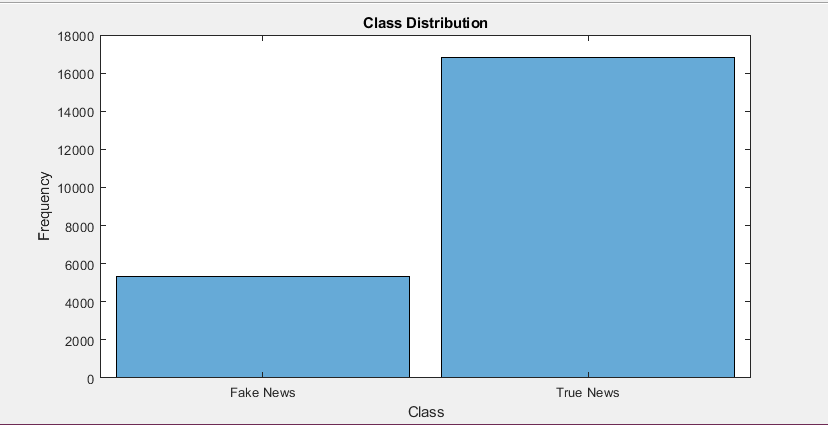
**Abstract**

With news coming from all over the globe at an unprecedented rate, the validity of the source has been a source of major complication in the modern world. Fake news can affect one’s opinion on people, idea, and events. This paper investigates the accuracy of the conventional LSTM (Long Short Term Memory) way of classifying sequences compared to the Batch Perceptron we observed in class. The articles titles were set to a maximum of 21 words in this experiment. After executing the two methods, it was found that LSTM posted an accuracy of 82.2% and the Batch Perceptron had an accuracy of 66.5% on the test data set. LSTM also had a much better F1 score with .8841 to .778 of the Batch Perceptron. This is mostly attributed due to the large amounts of false positives in Batch Perceptron.

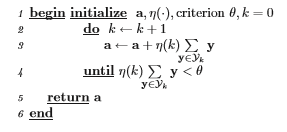
**Data and Detection Methods**

1. *Dataset*

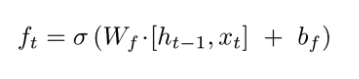
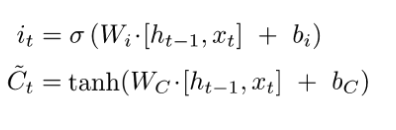
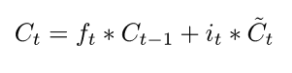
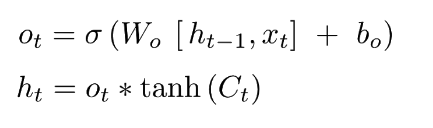
The data set used for the project was retrieved from a dataset titled FakeNewsNet. The data set includes stories from the popular fact checking website GossipCop. The data used contained article titles and their respective labels in order to not add to many dimensions. The data contained in total 22156 articles which was broken up into test and training groups. The data was also unevenly distributed with 16819 “True” articles to 5336 “False” articles. The data is publicly available on the FakeNewsNet Github.



1. *Batch Perceptron Algorithm*

The Batch Perceptron Algorithm used as follows below. The stopping criterion followed was to stop if all data points were classified correctly or after roughly 250 weight updates. The data set was broken up into True and False articles. These groups were further separated in to training and testing groups using and 80/20 split. The data was then preprocessed. The articles titles were turned to all lower case and removed punctuation. Next the y vector was created. In order to capture all interactions of the data, index y(1-21) was populated by the sum of those indexes. Index 22 was set to 1 for bias. Finally, the rest of the indexes were set to ignoring when k =i. This multiplied each index of the 21 articles with the other 20. This created a final length of 441 data points. The a vector matched this and captured 441 weights. A g vector was then found by multiplying the a vector with the y vectors to find if the sample was classified correctly. If not classified correctly, the weights were updated by the total sum. This was repeated over and over as the algorithm converged.

1. *RNN/LSTM*

The recursive neural network was constructed using the MATLAB Deep Learning Toolbox. The networks main layer was the Long Short-Term Memory layer. This layer can detect trends over periods of time in the article titles to more accurately determine the class of the title. The recursive algorithm is broken up to four main steps. The first is a sigmoid forget gate that determines based of and x how much of data should be disposed of. The formula follows that of the one pictured below. The second step is the input gate which is another sigmoid gate. Also an activation function of tanh is used to determine candidate values to be added to the state. This can be seen below. The third step is to update the old cell state with the new cell state. This is one of the major parts of an RNN. This can be seen above in the formula. The final step is to decide the output this based off a sigmoid gate with a tanh activation function. This will also funnel the cell state and the output to the next state.

The network structure of our neural network was an input layer, then a word embedding layer, then the LSTM layer, followed by a fully connected layer, then a SoftMax layer, then ending with a classification layer. The LSTM was used to predict a classifier and not a sequence in this example. In order to prepare the data for the neural network, the data was first tokenized then turned into sequences using the MATLAB commands tokenizedDocument() and doc2sequence(). For the data set, 70% of the data was used for training, 15% for validation and 15% for testing. 10 epochs were used for with a gradient threshold of 1, initial learn rate of 0.01 for training.

**Results**

1. *Batch Perceptron*

The Batch perceptron after roughly 250 updates posted an accuracy of 66.5% on the test set. The algorithm moved very slowly as it was very complex having 441 weights. This caused it to converge to slow unless it had many hours to trained. I chose to run the algorithm for 6 hours on my best computer available. The most telling part of the results for the Batch Perceptron was the confusion matrix. It can be seen clearly that there is many false positives. This greatly affected its F1 score to .771.

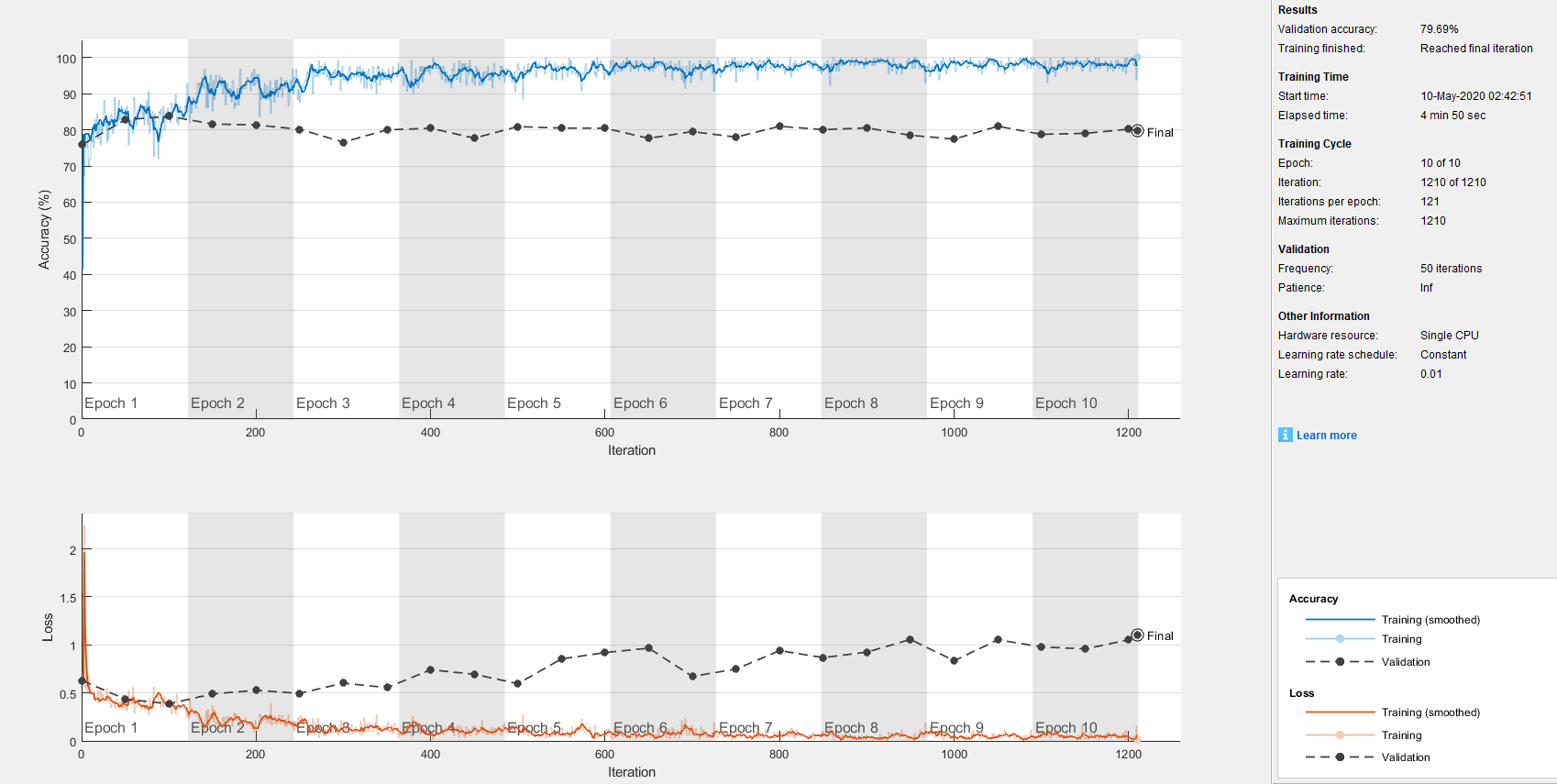
|  |  |
| --- | --- |
| **2593** | **770** |
| **127** | **940** |

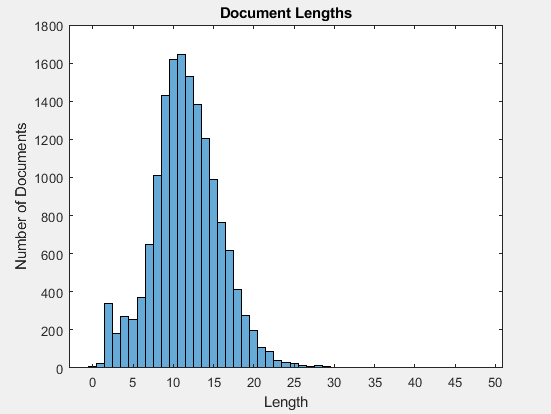
This could be due to the offset of the data sets or an insufficient amount of time to train. Also, the algorithm could struggle with encoded words vs true quantifiable data.

1. *RNN/LSTM*

The RNN LSTM was able to achieve an accuracy of 82.12% on the data set with an F1 score of .8841. The algorithm was able to train very fast compared to the other model. This model also saw a large number of false positives, but much less comparatively than the Batch Perceptron.

|  |  |
| --- | --- |
| **2266** | **256** |
| **462** | **338** |

Below is the training results of the LSTM Model. It shows loss and training accuracy in the two graphs.

Below it can be seen the distribution of document lengths used in the dataset.

**Conclusion**

After the experiment was performed, it can be clearly observed that RNN LSTM is the superior algorithm used to classify Fake News. Not only is the LSTM model much more accurate (82.12% to 66.5%), it is also able to get a better F1 score by a large margin (.8841 to .771). This shows that it is not only accurate but able to not only guess to the bias of the uneven dataset. If I were to perform a similar comparison in the future, I would use a different type of data that is more quantifiable and more balanced data set as well. I am greatly interested in pushing mainstream or conventional models and challenging myself to try to make a better model.

**How to Run the Code**

The code contained in the zip file comes with four matlab files to be run. The first to be run is CSV fixer. This will generate the gossipcopreal table needed to run the other programs. FakeNewsDetector will run a RNN LSTM training algorithm as well as generate graphics such as a few histograms. BPFakeNewsDetector will run the Batch Perceptron algorithm. BPTest will run a test to find accuracy after running BPFakeNewsDetector. The csv file is included in the zip file. Func.m is also included and used but does not need to be run separate.

**References**

1. Mathworks.com. 2020. *Classify Text Data Using Deep Learning- MATLAB & Simulink*. [online] Available at: <https://www.mathworks.com/help/textanalytics/ug/classify-text-data-using-deep-learning.html> [Accessed 10 May 2020].
2. Duda, R., Hart, P. and Stork, D., n.d. *Pattern Classification*.
3. GitHub. 2020. *Kaidmml/Fakenewsnet*. [online] Available at: <https://github.com/KaiDMML/FakeNewsNet/blob/master/README.md> [Accessed 10 May 2020].