RNNs for Fake News Classification & Sentiment Analysis

Maisha Maliha
Department of Computer Science
University of California, San Diego
mmaliha@ucsd.edu

Abstract

This paper uses Recurrent Neural Networks to build a model that can detect fake news by reading texts. The "fake news"trend is an extremely pressing issue in our lives currently. Media can alter facts and present the public with untrue facts and that can affect the reader's opinions. Its effects can be something as simple as an online shopper buying a product advertised by false information or something like the 2016 elections (which is said to be altered by fake news and social media). A kaggle news classification dataset was used to train our model and achieve an accuracy of 96%. The same model was then run with another kaggle twitter sentiment analysis data to perform sentiment analysis on texts. The model had an accuracy of 78% and can detect if a text or tweet is POSITIVE, NEGATIVE, or NEUTRAL.

1. Introduction

The term fake news became popular during the 2016 election in America when President Trump got elected [1]. "Fake news" was used to describe the misleading articles with incorrect information and facts that got published to attract more views and make more money [1]. There were many articles that flagged the social media giant Facebook for spreading fake news. A lot of those articles claimed "Facebook's failure on fake news and polarized politics got President Trump elected." [2]. False news has the ability to distort the media landscape through misinformation and hyperbole [2]. In 2020 as the world is fighting a pandemic, Facebook was again blamed for spreading misinformation on COVID-19 as 40% of the coronavirus related articles on Facebook were said to be incorrect [2]. Facebook has said that they have started a feature where users can flag fake news [1]. They are always trying to come up with new ways to distinguish between the false rumors. It is clear that an automated model could make this job easier and help the spread of misinformation.

The dataset we use to train our model is from kaggle's fake news classification challenge. The dataset had the article in the 'text' column and the 'label' column showed if the article was true or false (where a 0 means false and 1 means true). Our Model used 11 layers including 2 LSTM layers to reach an accuracy of 96% in only 8 minutes. More details on the model will be given in the experiment part of the paper.

This model can be used to do sentiment analysis on texts too. We used the model on a kaggle dataset that had data scraped, with the twitter api. The tweets were classified as positive(4), negative(0) or neutral (2). Our model had an accuracy of 78% which is not bad considering the large size of the data and the limited time we had. Sentiment Analysis is very useful when we want to know people's opinion on a certain matter or product or movie. Our brains can analyze texts subconsciously. When we read "I love San Diego", we know this means something positive

RNNs for Fake News Classification & Sentiment Analysis

Maisha Maliha
Department of Computer Science
University of California, San Diego
mmaliha@ucsd.edu

and when we read "The current situation saddens me", it falls in the negative zone. But if our computers could do this for us then we don't have to read an entire book to find out if the book has mostly positive words, or negative words. We don't have to search our twitter feeds and look at every tweet to see the public's opinion on some matter.

2. Related Works

I. Shlok Gilda in their paper **"Evaluating Machine Learning Algorithms for Fake News Detection"** used TF-IDF and bi-gram frequency and PCFGS (probabilistic context free grammars) to predict and classify news articles. They train a lot of models like random forests, Stochastic Gradient Descent, Support Vector Machine, Gradient Boosting, Bounded Decision Tree, and Baseline and then compare their accuracy rates. They concluded that SGD models were the best models out of the ones they tested with. The reason for that is that SGDs "outperform on precision while retaining a high recall" [3]. Their work was different from our approach as we used a RNN model. Our average accuracy rates were higher too. But this could also be due to working with a different dataset.

II. Geetika Gautam and Divakar Yadav used NLTK to classify a naive bayes model, Maximum entropy and Support Vector Models and Semantic Analysis to train their twitter dataset. Their dataset size was 18340, which was smaller than the dataset we used. They used preprocessing techniques to improve the efficiency of the models. When comparing the accuracy rates, Semantic Analysis had the highest rate of 89.9. The Naive Bayes model worked better than the maximum entropy and SVM [4]. The accuracy rates achieved in this paper were higher than the ones from our experiment.

3. Data

I. The first dataset we used for our fake news classifier was found in one of kaggle's machine learning competitions. It had three files, train.csv, test.csv and submit.csv which was the file we can submit in kaggle to get a score. The training data had 20800 entries and the testing data had 5200 data. To filter the data we got rid of blank lines and all the punctuations. Then the words were converted to numbers by a process called embedding where each word was represented with a unique 4 digit number [5]. The preprocess process was very similar to the process in the article"How to build a recurrent neural network to detect fake news."

II. The second dataset was from Kaggle's and is probably the largest twitter

RNNs for Fake News Classification & Sentiment Analysis

Maisha Maliha
Department of Computer Science
University of California, San Diego
mmaliha@ucsd.edu

sentiment analysis dataset on kaggle. The dataset contains 1,600,00 tweets which were extracted using the twitter API. The tweets have been classified as negative (with a score of 0), neutral (with a score of 2), and positive (with a score of 4). The score was in the target column of the dataset where it had a score. The cleaning process for this data was more extensive due to the large size of it. We followed a cleaning process from one of the notebooks on kaggle [6]. We used the natural learning tool kit to filter our texts and used a word2vec model to turn our texts to a vector that can be embedded. The label scores were changed to NEUTRAL, NEGATIVE, POSITIVE labels and a threshold score was made for the labels.

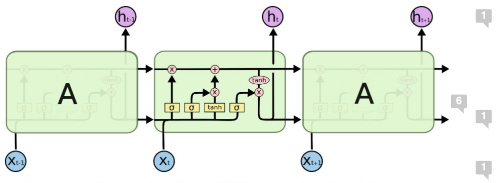
Method

I. The model we built had eleven layers. The first layer was an embedding layer of a vector size of 300. The convolution layer's vector went like **[1 0 1]** and as it swept past the words it gave the middle word a 0 value and the embedded word values for the words that got multiplied to 1 [5]. Layer 2 was a dropout layer of 0.3 (to prevent overfitting) and then a dense activation layer using the relu function. Then we had a 1D MaxPooling layer with a pool size of 4. Layer 4 and 5 were LSTM layers. A LSTM block has an input gate, forget gate, output gate. LSTMs are a special kind of RNN and are able to learn long time dependencies [7]. LSTMS have a repeating structure like RNN but each of those repeating cells have a different structure. The next layer was another dropout layer with a rate of 0.3 and a 512-dense layer that takes the output from the LSTM layer and the last layer was a 1-cell dense layer that was activated using the sigmoid function.

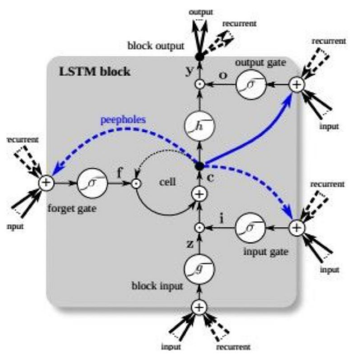
II. The same exact model was used for the sentiment analysis data but the only difference was that the first embedding layer. The sentiment analysis used an embedding layer where the maximum length of each sentence is 300 and the vector size is also 300.

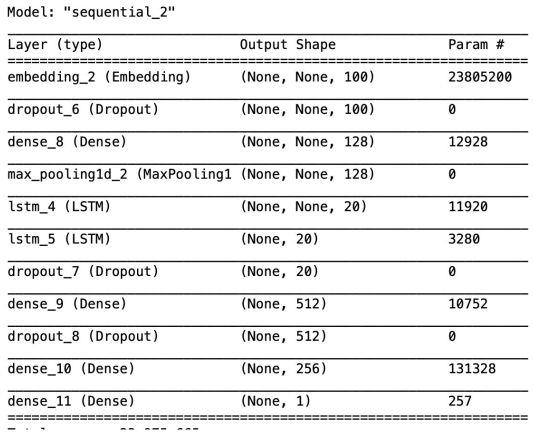
RNNs for Fake News Classification & Sentiment Analysis

Maisha Maliha
Department of Computer Science
University of California, San Diego
mmaliha@ucsd.edu



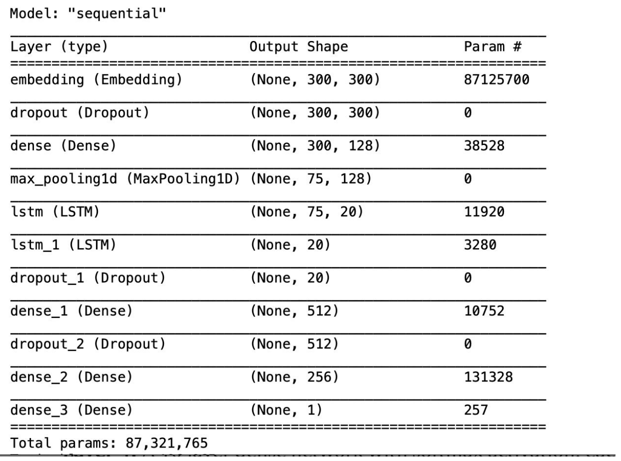
The repeating module in an LSTM contains four interacting layers.





```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, None, 100)         23805200
_________________________________________________________________
dropout_6 (Dropout)          (None, None, 100)         0
_________________________________________________________________
dense_8 (Dense)              (None, None, 128)         12928
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, None, 128)         0
_________________________________________________________________
lstm_4 (LSTM)                (None, None, 20)          11920
_________________________________________________________________
lstm_5 (LSTM)                (None, 20)                3280
_________________________________________________________________
dropout_7 (Dropout)          (None, 20)                0
_________________________________________________________________
dense_9 (Dense)              (None, 512)               10752
_________________________________________________________________
dropout_8 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_10 (Dense)             (None, 256)               131328
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 257
=================================================================
```

Figure: fake news classifier model details



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 300, 300)          87125700
_________________________________________________________________
dropout (Dropout)            (None, 300, 300)          0
_________________________________________________________________
dense (Dense)                (None, 300, 128)          38528
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 75, 128)           0
_________________________________________________________________
lstm (LSTM)                  (None, 75, 20)            11920
_________________________________________________________________
lstm_1 (LSTM)                (None, 20)                3280
_________________________________________________________________
dropout_1 (Dropout)          (None, 20)                0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               10752
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 257
=================================================================
Total params: 87,321,765
```

Figure: sentiment analysis classifier

Experiments:

I. The original classifier from towradsdatascience was changed in a few ways to achieve higher accuracy. We changed the dropout rate of some of the layers and the activation layer and we set the epoch to 8. We ran our model with a lot of different optimizers available in keras like Nadam, SGD, Adagrad and etc to check which gave us the highest accuracy rate. Adam gave the highest average accuracy of 0.961. The original author of the article got an accuracy of 0.9041 [5] but our model was able to get 0.961 in only 8 mins of run time.

Maisha Maliha
Department of Computer Science
University of California, San Diego
mmaliha@ucsd.edu

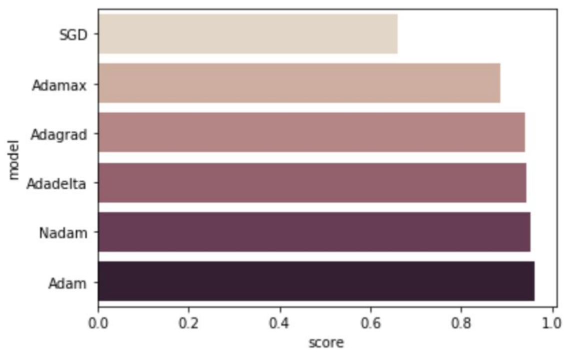


Figure: the accuracy rate of the different optimizers

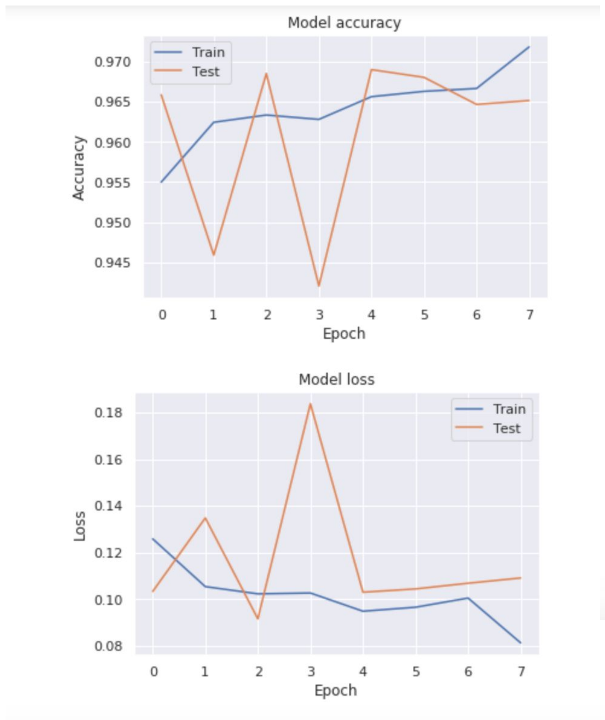


Figure: The accuracy rates and loss rates of the news classification model

II. For the sentiment analysis we only ran our model with 2 epochs because of the large size of data. Each run would take about an hour. But we were able to get an accuracy of 78%. With more time in hands, we can get even better accuracy rates and a more efficient model.

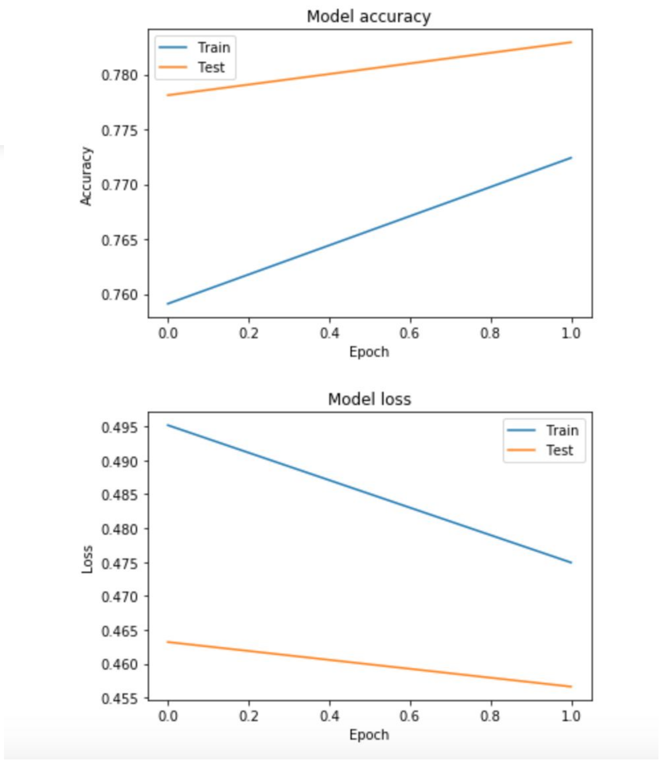


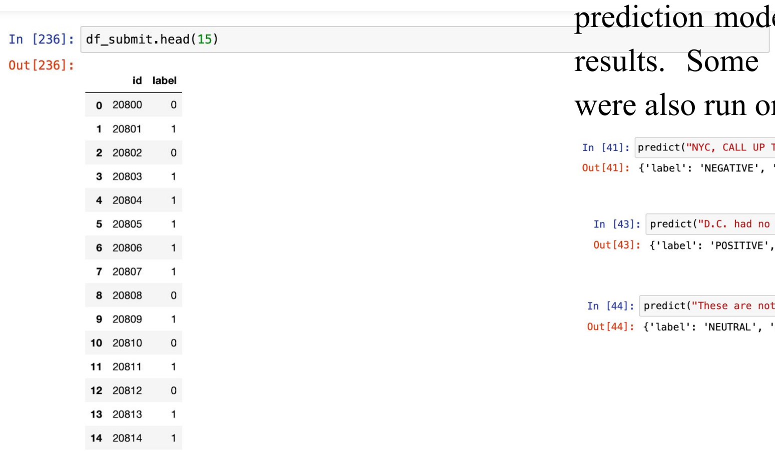
Figure: The accuracy rates and loss rates of the sentiment analysis model

RNNs for Fake News Classification & Sentiment Analysis

Maisha Maliha
Department of Computer Science
University of California, San Diego
mmaliha@ucsd.edu

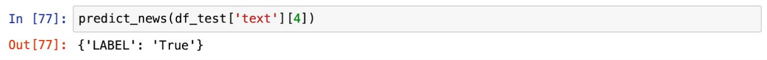
Results

I. We can see our models in action in these pictures. The first output shows the correct classification of the news articles and the next picture shows our classification model predicting if the news is true or fake.

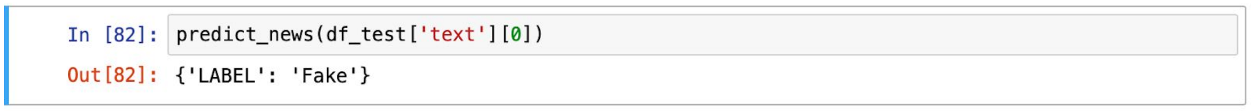


```
In [236]: df_submit.head(15)
Out[236]:
```

	id	label
0	20800	0
1	20801	1
2	20802	0
3	20803	1
4	20804	1
5	20805	1
6	20806	1
7	20807	1
8	20808	0
9	20809	1
10	20810	0
11	20811	1
12	20812	0
13	20813	1
14	20814	1

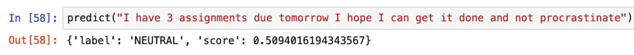


```
In [77]: predict_news(df_test['text'][4])
Out[77]: {'LABEL': 'True'}
```

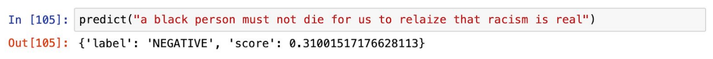


```
In [82]: predict_news(df_test['text'][0])
Out[82]: {'LABEL': 'Fake'}
```

II. To show some simple uses of our sentiment model in action, we can predict these simple sentences.

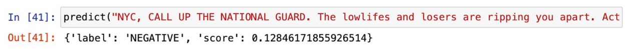


```
In [58]: predict("I have 3 assignments due tomorrow I hope I can get it done and not procrastinate")
Out[58]: {'label': 'NEUTRAL', 'score': 0.5094016194343567}
```

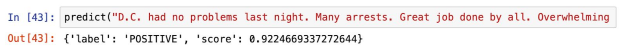


```
In [105]: predict("a black person must not die for us to relaize that racism is real")
Out[105]: {'label': 'NEGATIVE', 'score': 0.310015171766628113}
```

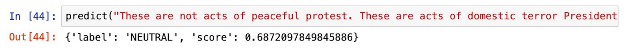
To test the sentiment model further, we ran it with another kaggle dataset that had President Trump's tweets. We selected a random of thousand tweets from that dataset and ran our prediction model on those tweets and plotted the results. Some of the President' recent tweets were also run on the model to test.



```
In [41]: predict("NYC, CALL UP THE NATIONAL GUARD. The lowlifes and losers are ripping you apart. Act
Out[41]: {'label': 'NEGATIVE', 'score': 0.12846171855926514}
```



```
In [43]: predict("D.C. had no problems last night. Many arrests. Great job done by all. Overwhelming
Out[43]: {'label': 'POSITIVE', 'score': 0.9224669337272644}
```



```
In [44]: predict("These are not acts of peaceful protest. These are acts of domestic terror President
Out[44]: {'label': 'NEUTRAL', 'score': 0.6872097849845886}
```

Maisha Maliha
Department of Computer Science
University of California, San Diego
mmaliha@ucsd.edu

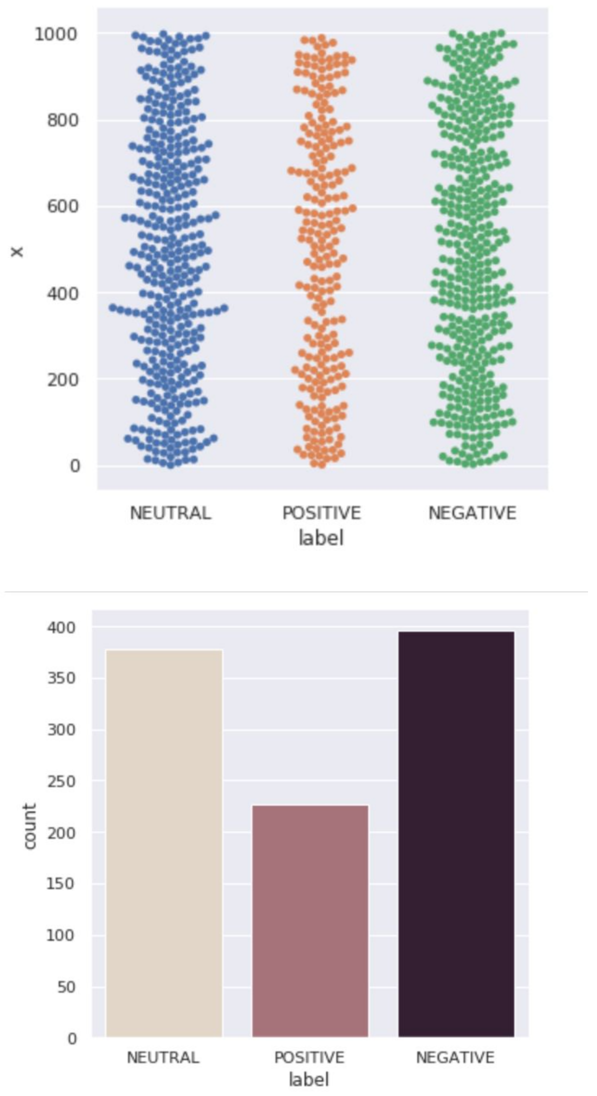


Figure: thousand random tweets of President Trump classified according to their sentiments

Conclusion:

The results we found in this paper look very promising. The model we built can be used in a lot of natural language classifications. The fake news classifier is very useful in our current scenario where more websites are trying to avoid spreading false information. The sentiment analysis classifier can be used to build safe internet spaces. The internet culture these days has trends like “stan culture” and “cancel culture” where fans “cancel” a celebrity for saying or doing something wrong by sending them hate messages. The sentiment classifier can be used to filter those negative and hateful comments. This classifier can be used to take down what are known as “bot accounts” on twitter. Bot accounts are when someone programs an account to tweet acting like an actual person. This can be done to affect polls or opinions on a topic. Right now, as we are witnessing social media being the most important outlet in spreading useful information in the black lives matter movement, we realize how important it is for social media to be a platform where we spread correct information. The models we worked with in this paper can be used to do just that.

RNNs for Fake News Classification & Sentiment Analysis

Maisha Maliha
Department of Computer Science
University of California, San Diego
mmaliha@ucsd.edu

Bonus Points:

- *use of a large dataset (1,600,00 tweets classified according to sentiments)*
- *used the same RNN model for two purposes: news classification and sentiment analysis*

References

[1] Facebook's failure: did fake news and polarized politics get Trump elected?

[2] Facebook to tell millions of users they've seen 'fake news' about coronavirus

[3] https://ieeexplore.ieee.org/abstract/document/6897213

[4] https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8305411

[5] How to build a recurrent neural network to detect fake news

[6] Twitter Sentiment Analysis

[7] Understanding LSTM Networks