# 

# 

**BT4222**

**Mining Web Data for Business Insights**

**Final Project**

**Group 7**

**Building a Classification System for Fake, Satirical News**

|  |  |
| --- | --- |
| **Name** | **Matriculation Number** |
| Ang Jamais | A0188978W |
| Erica Jane Tem | A0189505W |
| Jay Lew Jie Sheng | A0183426A |
| Justin Julius Chu Qijie | A0183794M |
| Lim Joo Bin | A0182574X |

**Table of Contents**

[**Introduction**](#_p5ua0vrssood) **3**

[1.1 Problem Description](#_pc4heo46b0q) 3

[**Dataset**](#_yk8sggttd3ay) **3**

[2.1 Description of r/TheOnion](#_fvv7csjc8z4v) 3

[2.2 Description of r/NotTheOnion](#_xcjpu6f91xre) 4

[**Analysis**](#_sng8y1846lnv) **5**

[3.1 Exploratory Analysis](#_xhnpjfrfrz1b) 5

[3.2 Pre-processing](#_pz7qf5o7jzu7) 9

[Standardization](#_d85cga9bfqzq) 9

[Lemmatization](#_6gp19v8c7mmm) 9

[Sampling](#_kxs2k4l6e31m) 9

[3.3 Baseline Models](#_b9re1dqeeph7) 9

[3.4 Modelling](#_atnz1uo76bwg) 10

[3.4.1 Gradient Boosting](#_19lfhixirjr7) 10

[3.4.2 Artificial Neural Network (ANN)](#_vy2p2wyb6ewn) 12

[3.4.3 Convolutional Neural Network (CNN)](#_14drpnump2ls) 13

[3.4.4 Recurrent Neural Network (RNN) - LSTM and Bidirectional LSTM](#_5uuz499a7nwa) 14

[3.4.5 Bidirectional Encoder Representations from Transformers (BERT)](#_v3k6cidxp5dc) 16

[3.5 Results](#_wo0vn24o3d7u) 17

[**Insights**](#_bl60ockmk0nv) **17**

[**Further Model Application**](#_bl60ockmk0nv) **18**

[**Potential Application**](#_r5v10vc5vf16) **18**

[**Further Improvements**](#_hef3oxsa9nho) **19**

[**Appendix**](#_bl60ockmk0nv) **21**

# Introduction

## 1.1 Problem Description

In recent years, the spread of satire news and fake news has been gaining traction online, especially so with the aid of the rise of social media and hardware technological advancements. Many people, especially those who are uninformed, have fallen for this satire and fake news that is being spread around. While some harmless satire news brings about a more light-hearted take on current affairs, we still have to be wary about the more harmful fake news that could sow discord between groups of people in our society. Take for example, in the United States of America, the prevalence of harmful fake news has brought about an all-time high[[1]](#footnote-0) in its political divide between the Democrats and the Republicans.

As such, our team has decided to come up with a classification system that will be able to distinguish satire or fake news from actual news using only their headline.

While the definition of satire or fake news may be ambiguous to some, we have decided to turn to Reddit, which is a web forum site categorised into many different groups and categories known as subreddits.

In our project, we have decided to scrape data from r/TheOnion subreddit for fake news and r/NotTheOnion for real news.

# Dataset

## 2.1 Description of r/TheOnion

The Onion is a popular news site that writes satire news on various issues. The subreddit r/TheOnion is a subreddit where reddit users will repost and upload articles that are published on The Onion news webpage.

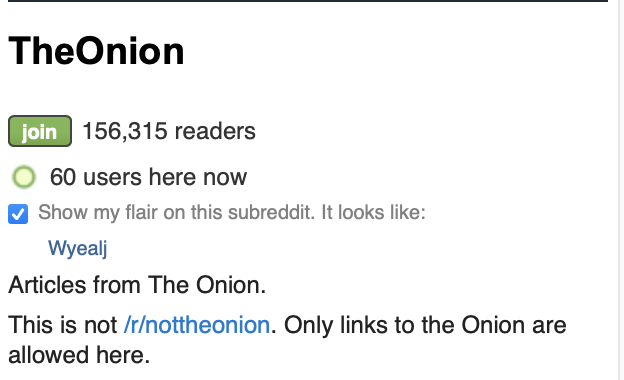


Fig 1: r/TheOnion subreddit page

As r/TheOnion only has a single source of news, the dataset collected from this subreddit will be significantly lesser than that of r/NotTheOnion.

## 2.2 Description of r/NotTheOnion

While there is no news site called Not The Onion, the subreddit r/NotTheOnion colates news from various mainstream news media outlets all around the world. The purpose of this subreddit is to post news articles that sound satire or fake but in actual fact, are real, factual news.

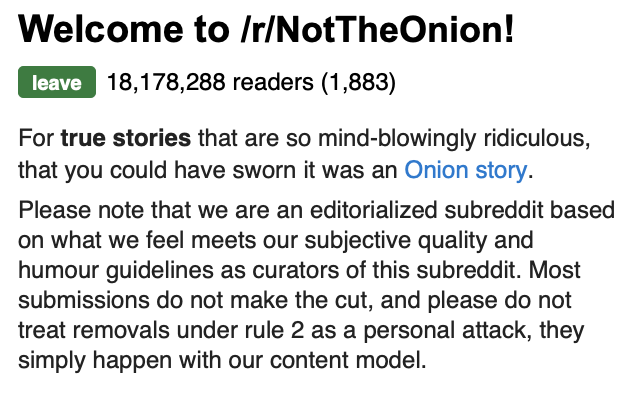


Fig 2: r/NotTheOnion subreddit page

With multiple sources of news being published onto r/NotTheOnion, naturally this subreddit will contain more data compared to that of r/TheOnion.

# Analysis

## 3.1 Exploratory Analysis

As mentioned previously, the number of news sources for r/TheOnion is significantly lesser than that of r/NotTheOnion. On top of that, the subscriber count for r/TheOnion is 156,313, while that of r/NotTheOnion is about 116 times more than r/TheOnion at 18,178,288.

Naturally, the amount of data gathered for r/TheOnion will be lesser than that of r/NotTheOnion within the same timeframe.

|  |  |  |
| --- | --- | --- |
|  | r/TheOnion | r/NotTheOnion |
| Number of posts within the same timeframe | 9,978 | 13,100 |

Using data collected from both subreddits within the same timeframe, the data frame for r/TheOnion has 9,978 rows while the data frame for r/NotTheOnion has 13,100 rows. However, most of these data are duplicates, and after removing the duplicate rows, we managed to arrive at a pretty even amount of data from both subreddits, at around 8,200 rows each.

|  |  |  |
| --- | --- | --- |
|  | r/TheOnion | r/NotTheOnion |
| Number of unique posts within the same timeframe | 8,264 | 8,202 |

Word Cloud

The posts from both subreddits were collated into a word cloud to explore the relative frequencies of certain terms.



Fig 3:Word Cloud of posts within r/TheOnion(left) and r/NotTheOnion(right)

From the two word clouds above, we can see that both r/TheOnion and r/NotTheOnion have pretty similar results with “Man”, “Trump”, and “New” being frequently mentioned in both subreddits. For posts within r/NotTheOnion, current topics like the US elections and COVID-19 are brought up very frequently, with words like “Trump”, “COVID”, “Biden” and “coronavirus” amongst the most frequent words. This is probably due to the nature of real news to report on popular topics nowadays like the US elections and COVID-19. On the other hand, posts from r/TheOnion are more sensationalistic in nature and report on more contentious topics and current affairs in a satire manner.

Title Word Count

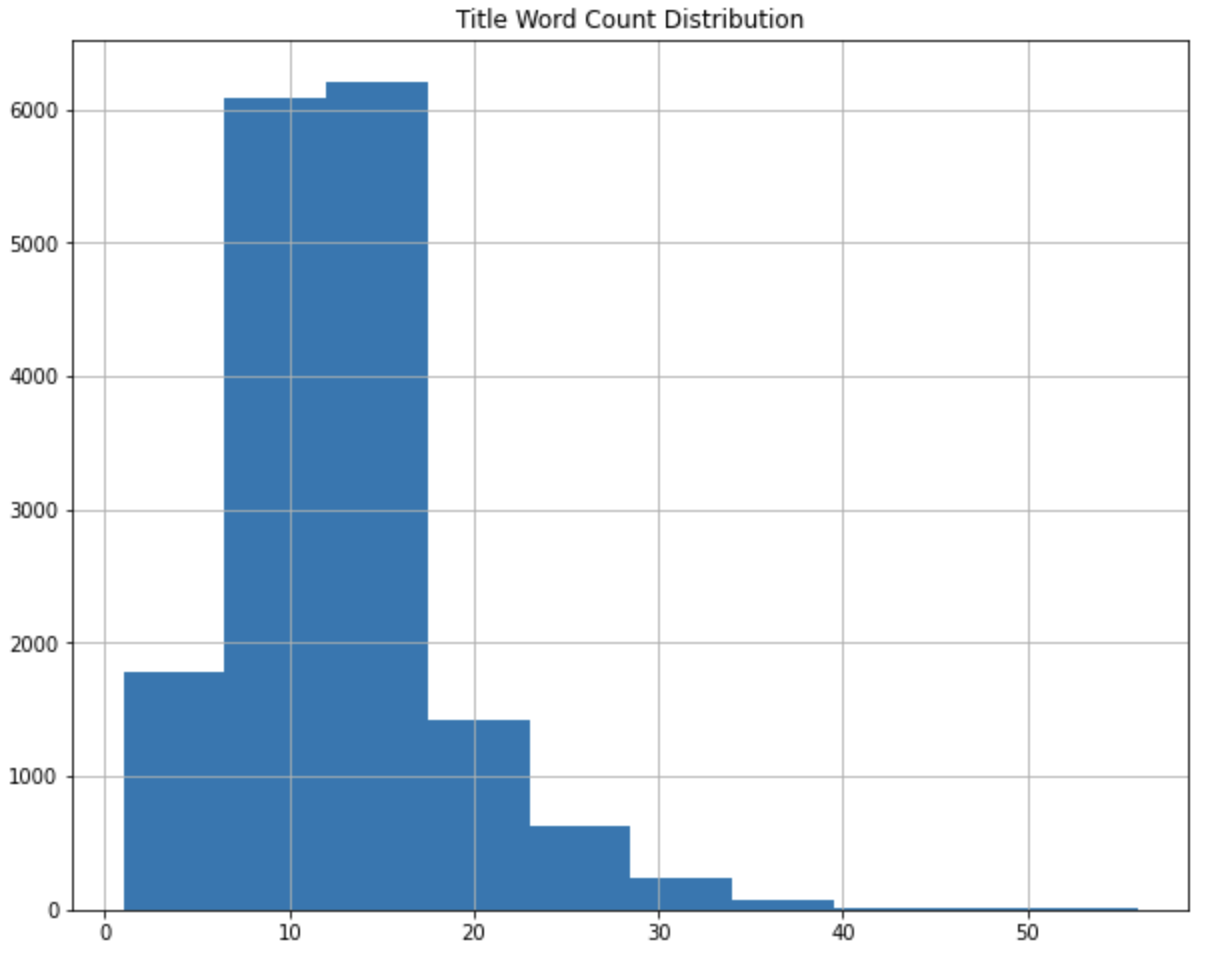
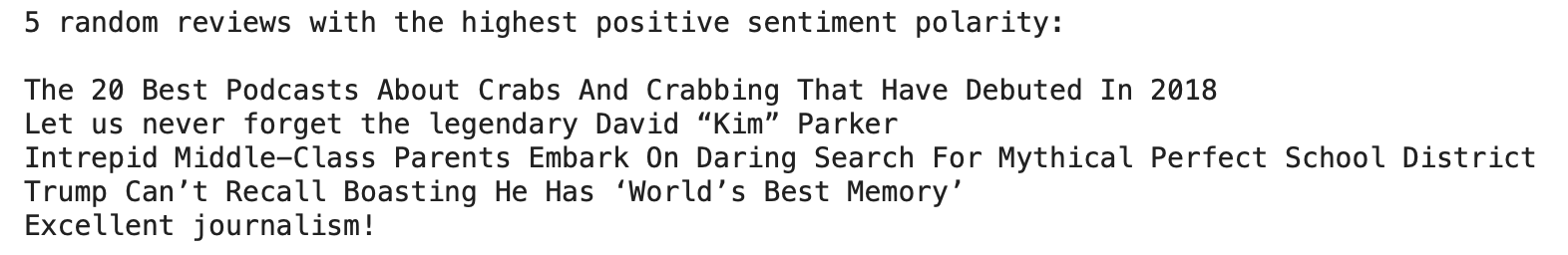


Fig 4: Word count distribution for both subreddits

For both subreddits, the average number of word count for their titles are around 7 to 18 words.

Sentiment of Dataset



Most Common Word Distribution

After removing the stop words from the data, we plot out the distribution of the most common words in the data set.

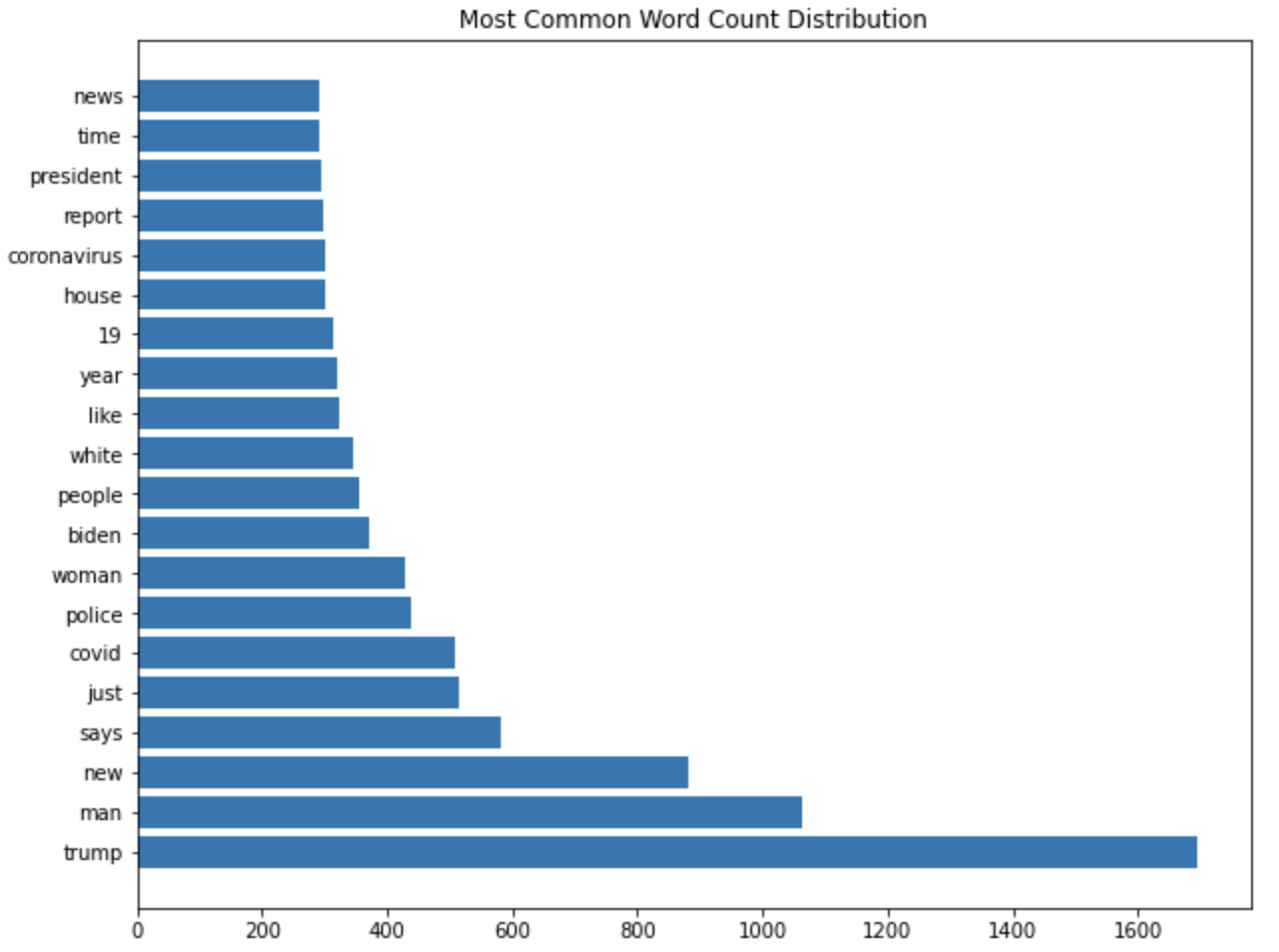


Fig 5: Top 20 common words in our DataFrame

As we can see, “trump” is the most common word in our DataFrame.

Most Common Bigram Distribution

We also plotted out the most common bigram distribution, without stop words.

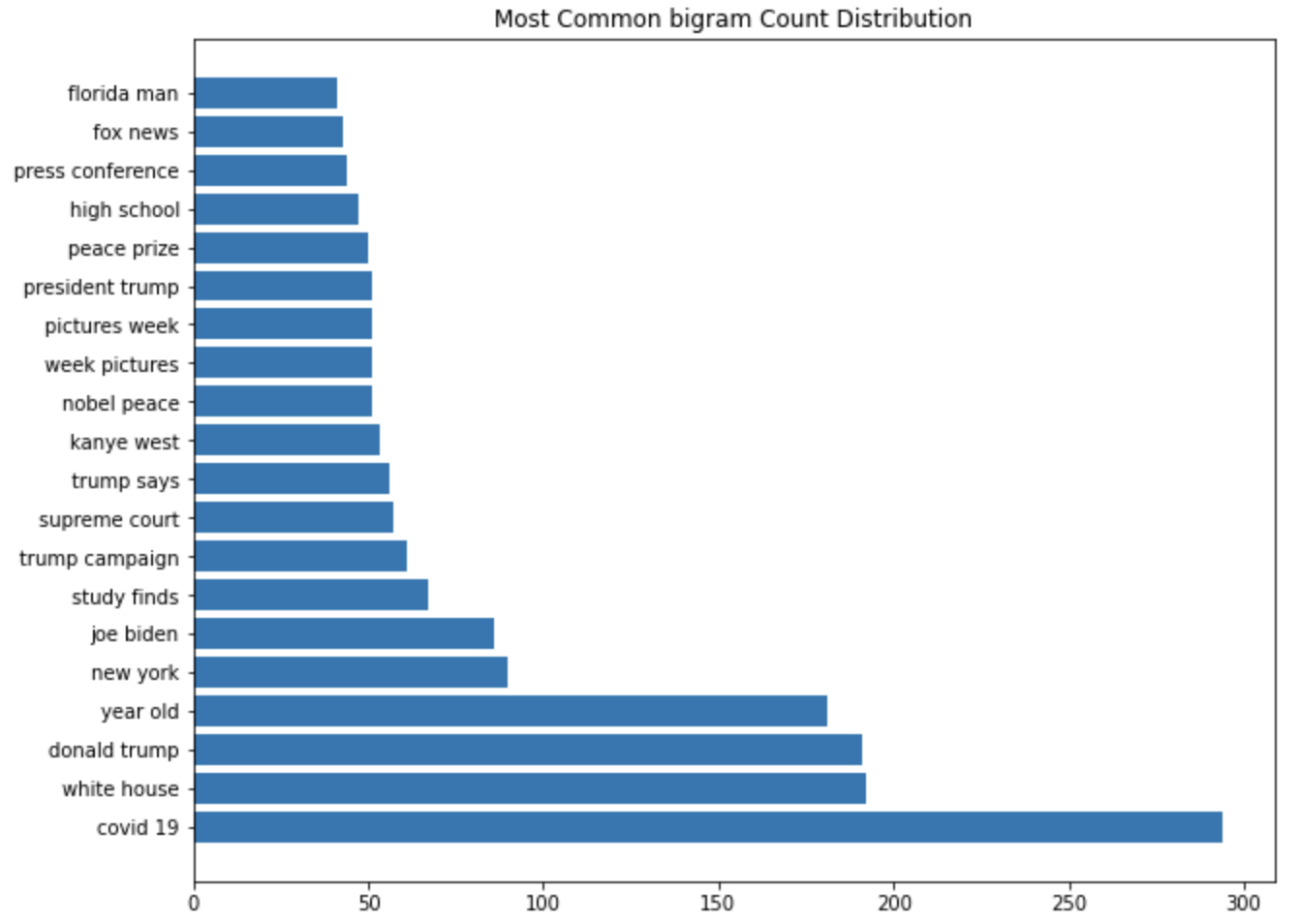


Fig 6: Top 20 common bigrams in our DataFrame

The most common bigram in this case is “covid 19”.

Most Common Trigram Distribution

Taking one step further, we also plotted out the distribution for trigrams.

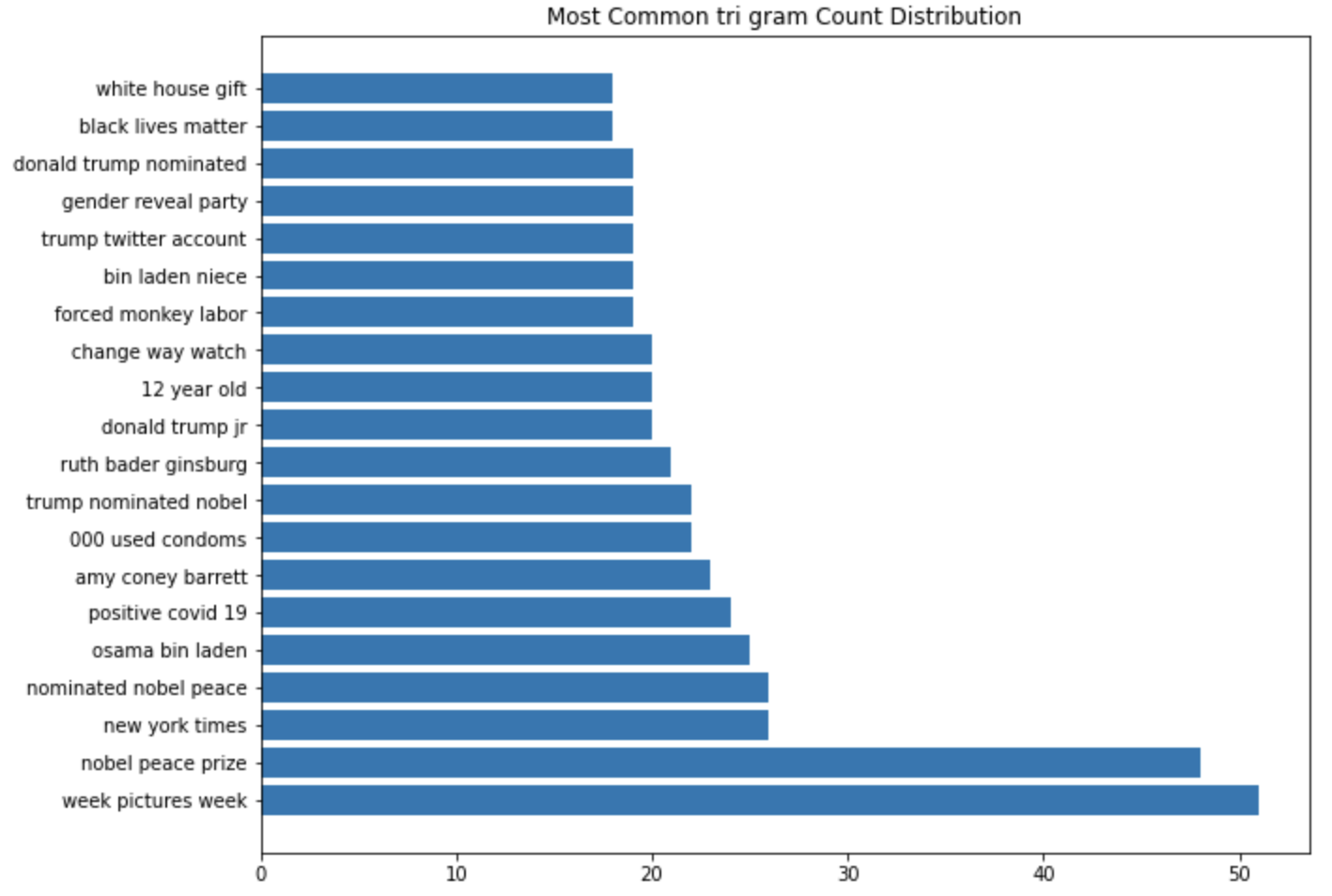


Fig 7: Top 20 common trigrams in our DataFrame

With the distribution of most common words, bigrams and trigrams plotted out and given the time frame of our data set, the most common word, bigram and trigram makes sense as the more recent popular topics are the 2020 US Presidential Elections and COVID-19. The words, bigrams and trigrams used are reflective of that.

Part-of-speech Tagging

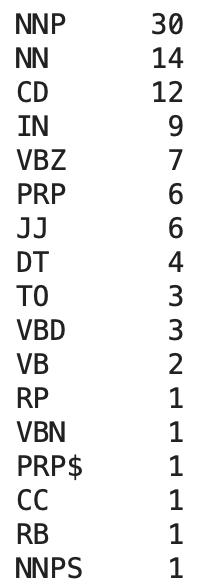


Fig 8: Most frequently used POS-tag

The most common part of speech used are the proper nouns and nouns. This again makes sense because news articles tend to appeal to the masses by having names of well-known people and or places, to increase their readership.

## 

## 3.2 Pre-processing

### Standardization

To reduce variance in our dataset, we have to make sure that the dataset is standardized. The standardization is carried out via removal of accents, removal of punctuations, removal of stop words, as well as the conversion to lowercase.

### Lemmatization

Using NLTK’s Wordnet Lemmatizer, the data from reddit was cleaned. Lemmatization helps remove inflectional endings of words and return the words to its base or dictionary form. For instance, words like “eats”, “ate”, “eaten” will be changed to “eat”. Lemmatization brings context to the words as such we decided to go with lemmatization rather than Stemming. As such, with lemmatization, the words in the title of the subreddit posts will be grammatically the same.

### Sampling

As we do not have an extensive set of data to work on, one way to increase the data size is through up sampling. We carried this out by randomly shuffling the dataframe, and then performing a random train test split on it.

Shuffle

By shuffling, we will have a more general and less overfitted model due to the increase in variance.

Train Test Split

Furthermore, by performing a train test split randomly, the variance will further increase, providing a more general model as well.

## 3.3 Baseline Models

|  |  |  |  |
| --- | --- | --- | --- |
|  | CV Score | Training Accuracy | Testing Accuracy |
| Random Forest | 0.747 | 0.997 | 0.770 |
| Logistic Regression | 0.787 | 0.963 | 0.797 |
| Naive Bayes | 0.797 | 0.927 | 0.801 |

Our group has decided to use Random Forest, Logistic Regression, and Naive Bayes as baseline models to have a general sensing of how our own classifier models should fare in terms of testing accuracy. As such our models aim to be better than the testing accuracies shown in the table.

## 

## 3.4 Modelling

### 3.4.1 Gradient Boosting

**What is it:**

Gradient boosting is a boosting method-based Machine Learning model, which iteratively learns from each of the weak learners to build a strong model through ensembling. It is able to optimize 3 main functions: regression, classification and ranking. For the purpose of this dataset, we will be utilising Gradient Boosting for classification of the news headlines.

Gradient Boosting utilises an iterative functional gradient algorithm. By minimizing a loss function by iteratively choosing a function that points towards the negative gradient, each iteration would gradually reduce the loss function and result in a combined, stronger model made from weaker models.

**Why we used it:**

We utilised Gradient Boosting initially as it is well suited for binary classification machine learning problems.

**How we implemented it:**

We first pre-processed our text headline data by using Keras’ Tokenizers to convert the text headlines to sequences. Thereafter, we initialized the model and conducted 5-fold GridSearch to determine the optimal hyperparameters of the number of estimators, the learning rate and the maximum depth.

Parameters Tuned and Analysis:

1. Maximum Depth

Maximum Depth refers to the depth and number of splits of the tree. The purpose of this is to control over-fitting as models with a higher depth would lead to a model learning relations very specific to that sample.

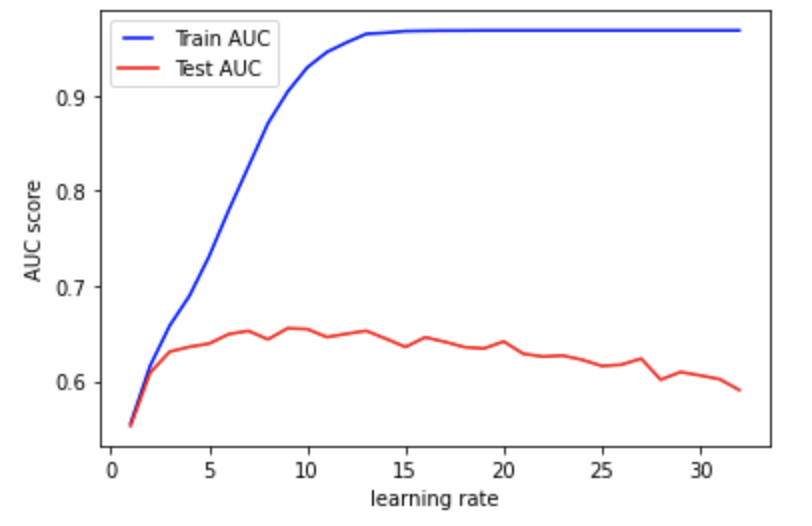


Fig 9: Area under ROC for training and validation set against maximum depth

From the plot, we see that the model overfits for depth values that are larger. We opted to select a maximum depth of 5.

1. Learning Rate

Lower values of Learning Rate are considered to make the model more robust to specific characters of the tree

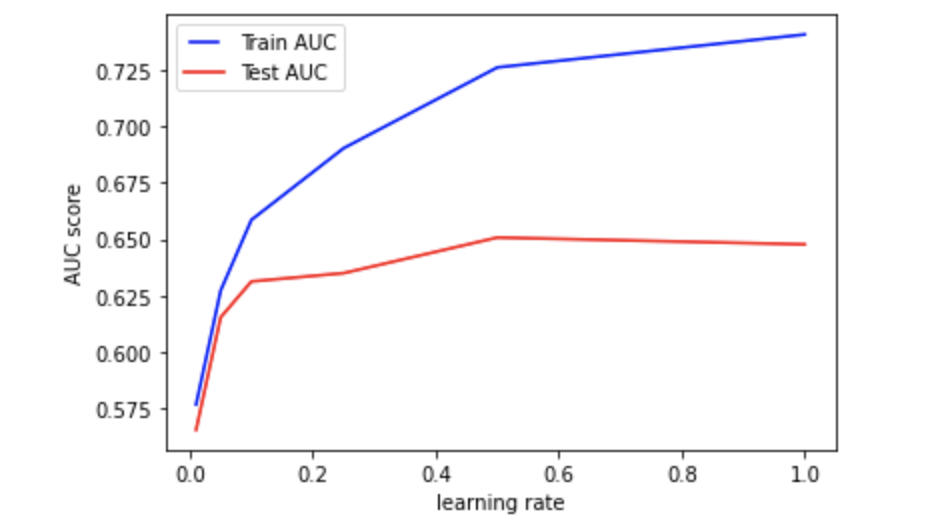


Fig 10: Area under ROC for training and validation set against learning rate

We selected a learning rate of 0.5 to minimize overfitting in the model.

1. Number of Estimator

This refers to the number of sequential trees to be modelled. At a higher number of trees, the model becomes more robust but can lead to overfitting. As a result, this should be tuned together with the CV.

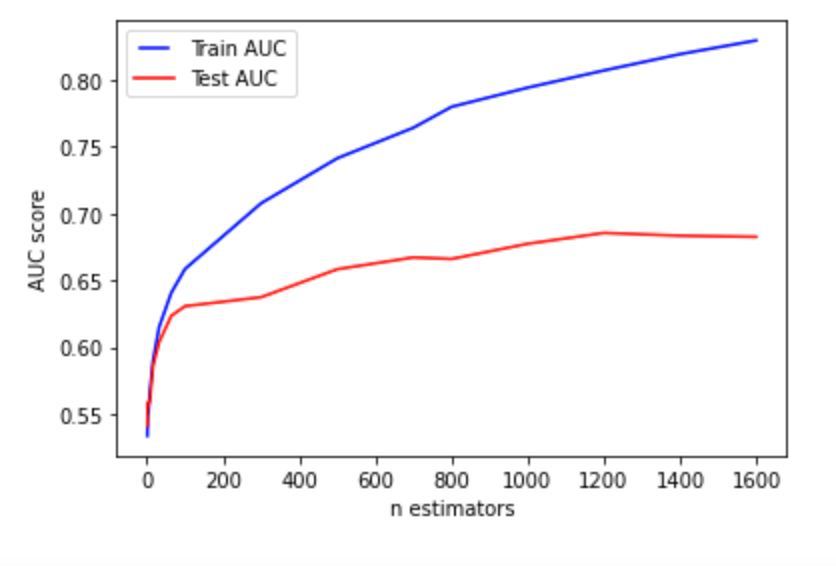


Fig 11: Area under ROC for training and validation set against n\_estimators

We selected the number of estimators to be 1200.

With the final tuned parameters (Max Depth=5, Learning Rate=0.5, Number of Estimators=1200), our final result gave us an accuracy of 0.685 on the validation testing set.

Upon analyzing the results, we determine that the Machine Learning model of Gradient Boosting was not well-suited for this particular classification problem. As such, we decided to turn our attention towards Deep Learning models instead.

### 

### 3.4.2 Artificial Neural Network (ANN)

**What is it:**

This neural network we use is the basic form of neural networks. It is a common and simple architecture known as the feedforward network. I.e there are no backwards or inter-layer connections.

**Why we used it:**

While other neural networks may be more appropriate for this use case, we believed we should start first with a simple ANN to serve as a benchmark. Then after model evaluation, determine what other types of neural networks to experiment with and how to go about doing so. However, we are aware of the limitations of this neural network such as the vanishing and exploding gradient. We are also aware that it cannot capture sequential information which might negatively impact our use case. After initial experimentation with our ANN, we implement a CNN and RNN to overcome these limitations.

**How we implemented it:**

We implement a basic 2 hidden layer model. Then we simply run a grid search to find the optimal parameters for batch size and epoch, optimization algorithm, network weight initialization, hidden layer activation function, dropout rate and number of hidden layer neurons.

## 

Fig 12: Model summary for ANN model

### 

### 3.4.3 Convolutional Neural Network (CNN)

**What is it:**

CNN is a Deep Learning algorithm which is widely used in computer vision and image classification. It is able to extract certain features from pictures and use them in a neural network. However, we are also able to use CNN in sequence processing, and in this case, text analysis. This is because of its ability to detect specific patterns.

**Why we used it:**

While CNN has been proven effective for images (2D), we are able to apply the same concepts with text analysis (1D).With CNN, the convolutional layers are able to detect specific patterns in the headlines from the dataset and using those patterns, we will be able to detect fake news from actual news. With enough training, through each convolutional layer, more complex text patterns will be detected by the model.

**How we implemented it:**

For this model, we used the convolutional layers that Keras offers. Since this is a text dataframe, it is sufficient for us to use the one dimensional layer, which is Conv1D. From this Conv1D layer, we are able to vary the number of filters, size of the kernel, and also the type of activation layer. This layer should be in between the Embedding layer and the MaxPooling1D layer. The MaxPooling1D layer reduces the dimensional complexity while at the same time keeps the significant information of the convolutions layers. This helps to reduce the high amounts of computation in the CNN, thus making sure that the model is not overfitted.

After which, the matrices are flattened into vectors. The Dense layer is added, followed by the Dropout layer, and finally the Dense layer again.

The Dropout layer randomly sets the units in the hidden layer to 0 at each update of the training phase. This is a form of regularization and helps to prevent a model from overfitting.

The final Dense layer has 2 output nodes with a sigmoid activation function for binary classification.

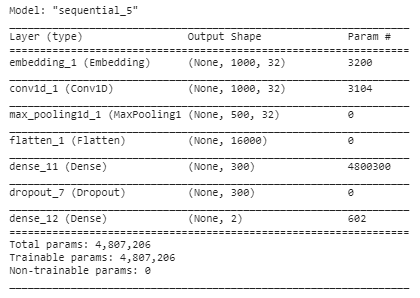


Fig 13: Model summary for CNN model

However, CNN works best with training sets that are large, because the larger the data set, the easier it is for the layers to find a specific pattern after many rounds of training.

### 3.4.4 Recurrent Neural Network (RNN) - LSTM and Bidirectional LSTM

**What is it:**

Recurrent Neural Network is a generalisation of a feedforward neural network that has internal memory. RNN performs the same function for each input of data while the output is dependent on the computation of the previous one. This makes the model repetitive in nature. However, unlike other feedforward neural networks, RNNs use their internal memory to process the sequence of inputs and hence inputs are all related to each other. This makes RNN ideal for natural language processing purposes and in our project, classification of subreddit posts.

In our project, we specifically used Long Short Term Memory (LSTM) networks as it makes it easier to remember past data. The problem of vanishing gradients of other RNN is solved using LSTM.

**Why we used it:**

In our project, we wanted to classify news articles. LSTM’s ability to capture the meaning of words in close proximity made an ideal choice for our classification problem. LSTM only captures important context of words in the sentence and passes the relevant information down the chain of sequences to help make predictions and accurately classify the headlines.

**How we implemented it:**

We used Google’s News Word2Vec model to create the word embeddings for our model. We tokenized and used paddings on our sequences for both the train and test dataset, then we assigned the tokens with the respective embeddings from the Google News Word2Vec model.

Each sequence is passed into the RNN. The embeddings matrix is passed to the embedding layer. The output is then passed to the LSTM layer. Only the output from the last LSTM cell is taken and passed to a dense layer and then dropout. Lastly, the output is passed to the final dense layer.

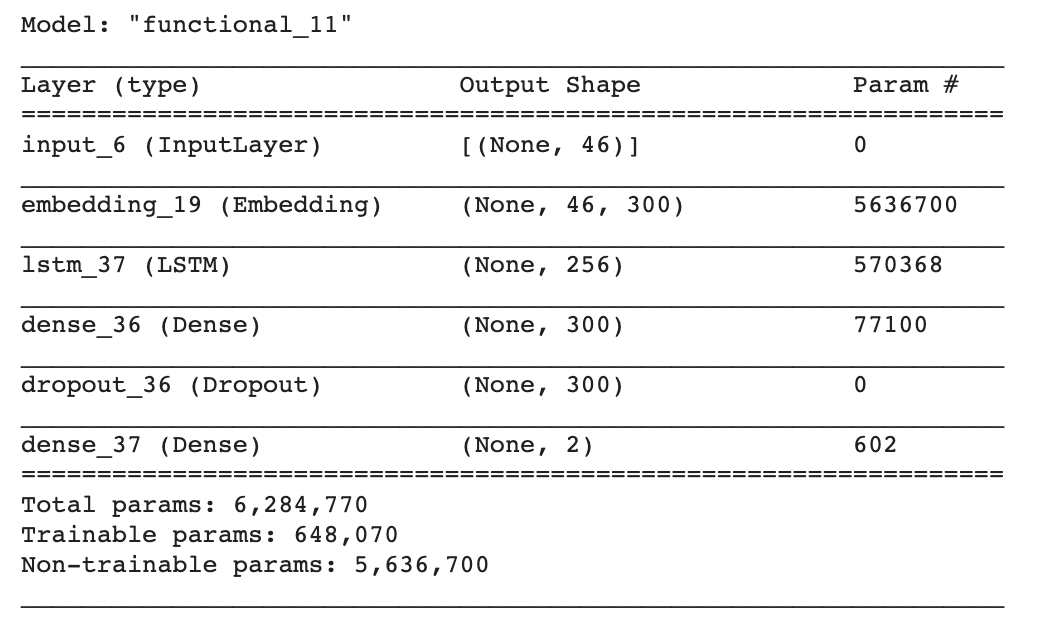


Fig 14: Model summary for RNN model using LSTM

Something interesting to note from the model summary is that there is a significant number of non-trainable parameters, and this is due to BatchNormalisation layers whose mean and variance vectors are updated via updates instead of backpropagation.

In addition, upon further reading, we found the bidirectional LSTM could be used for our problem. Given our problem, we have the entire sequence at the time of classification and hence we can use context from both directions and to guide the model’s prediction. Considering that our sequences are tokenized, bidirectional LSTM could fetch better results.

For our bidirectional LSTM, we also used a different word embedding in hopes to improve its accuracy. For a more accurate comparison, we used the same Word2Vec embedding and similar hyperparameters such as: number of LSTM nodes, drop out rate and optimiser when training the new model. However, we added an additional hidden layer for the bidirectional model.

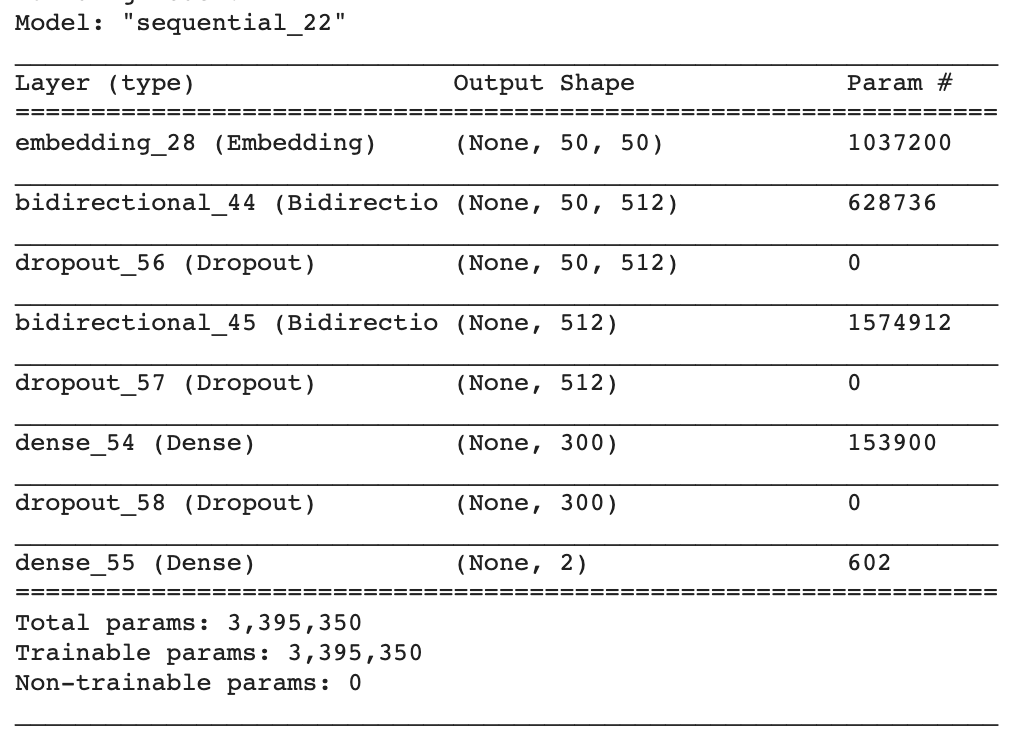


Fig 15: Model summary for RNN model using Bidirectional LSTM

From the results obtained (as seen from the results table below), we see that bidirectional LSTM does indeed obtain slightly better results as compared to LSTM. However, we are still unsatisfied with the current results and hence, we looked into other models and preprocessing methods that could better classify text headlines.

### 3.4.5 Bidirectional Encoder Representations from Transformers (BERT)

**What is it:**

BERT is a NLP framework developed by Google and relies on the transformer architecture to train bidirectional representations by jointly conditioning on both left and right context. Similar to Bidirectional LSTMs, the BERT model is made to learn from words in all positions in the sentence, and is better able to differentiate words based on context.

**Why we used it:**

Traditional NLP methods which use word embeddings like Word2Vec are unable to capture all meanings of words, especially as they are used in different contexts. Even with the inception of Bidirectional LSTMs, some context might not be able to be captured. In a Bidirectional LSTM model, there are separate LSTMs each for the forward and backward sequence, used to predict the next word in the sequence for each direction. However, the Bidirectional LSTM is unable to look at both ways at the same time, which BERT is capable of doing due to its masked language model.

In detecting whether a news article might be satirical in nature, the context of certain words matter a lot. In satire, the words used have heavy contextual meanings, and are meant to be emphatic. Hence, a model like BERT that is able to capture the context of a word in a sentence would be useful for our use.

BERT also uses the transformer architecture and relies on self-attention for encoding and decoding. Through transfer learning using the pre-trained BERT model, we are able to modify the model to fit our specific task, while achieving state-of-the-art results that BERT promises.

**How we implemented it:**

By simply adding an output layer with sigmoid activation function to the BERT model, we are able to fine-tune the model to fit our specific task. We used a lighter version of BERT, DistilBERT, due to its advantage in model size and training time. DistilBERT is reported to run 60% faster than BERT while preserving over 95% of BERT’s performances. To prevent overfitting, we fine tuned the model using a small learning rate and a lower number of epochs.

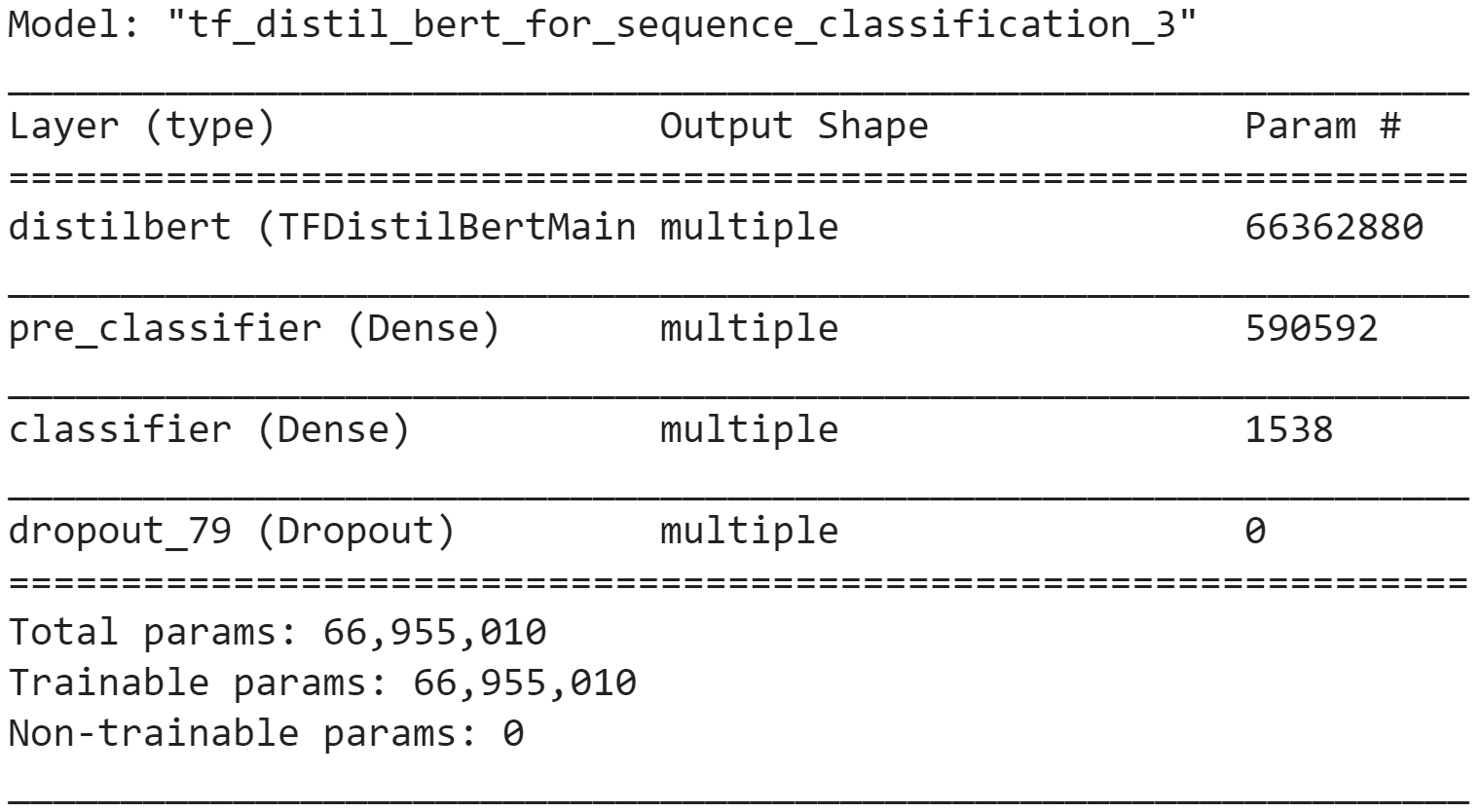


Fig 16: Model summary for BERT

Compared to the original BERT-base model, which has 110 million trainable parameters, DistilBERT only has ~66 million trainable parameters. This makes the training for DistilBERT much faster than that of BERT.

## 3.5 Results

For each neural-network based model, we compared the training and validation accuracy and loss at every epoch to decide the best number of epochs to use for training such that the model is not overfitted. (see Appendix A)

The performance of the tuned models on the test set are as shown below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Accuracy | F1-Score | Precision | Recall |
| Gradient Boosting | 0.685 | 0.690 | 0.67 | 0.70 |
| ANN | 0.838 | 0.839 | 0.839 | 0.839 |
| CNN | 0.827 | 0.828 | 0.828 | 0.827 |
| RNN (LSTM) | 0.780 | 0.780 | 0.780 | 0.780 |
| RNN (Bidirectional LSTM) | 0.807 | 0.802 | 0.822 | 0.792 |
| BERT Transfer Learning | 0.905 | 0.913 | 0.892 | 0.935 |

# Insights

**Machine Learning Insights**

To determine which model is best able to capture the context of a word in a sequence, it is clear from the results that a bi-directional model outperforms a single directional LSTM model or a convolutional network.

Additionally, it is possible to pre-process using BERT for word-embeddings and thereafter utilise various classification models to explore the impact on the classification ability.

**Data Insights**

Given that our data consists of text headlines from The Onion/Not The Onion which are satirical in nature, the data would not be as reflective of real world headlines with more conventional language style. Our rationale behind the selection of this dataset was to utilise the satire style of language as a definite, defining classification between true and fake headlines. This is in comparison to real world headlines where its truth is ambiguous due to the presence of various facts and figures.

# Further Model Application

With the aforementioned Data insight in mind, we sought to explore whether our model could perform on actual real world headlines despite the difference in language styles.

We further tested the model on a separate Dataset of Fake and Real news articles obtained from, but the performance was not very good, only achieving 0.56 accuracy. This led us to the conclusion that even though our model performs well using data obtained from r/TheOnion and r/NotTheOnion, it is not very generalizable to news articles as a whole.

Understandably, given the difference in language styles between conventional headlines versus satirical-style headlines, the results on the actual real-world headlines is not as impressive as that on the satirical-style headlines. Furthermore, it is insufficient to distinguish fake and real news only based on the style of writing, and factual checks should be made as well. Certain fake news sources can be written to imitate the style of legitimate reporting.

# Potential Application

Given the prevalence of fake news and information in this day and age, we see immense potential in this model and its possible applications. Fake news and information is not only limited to news headlines; it can come in the form of information via text shared between loved ones or messages posted on social media. Through this model, we hope to help the masses in improving their fake news literacy.

1. Scam

The model could be further trained to detect potential scam messages.

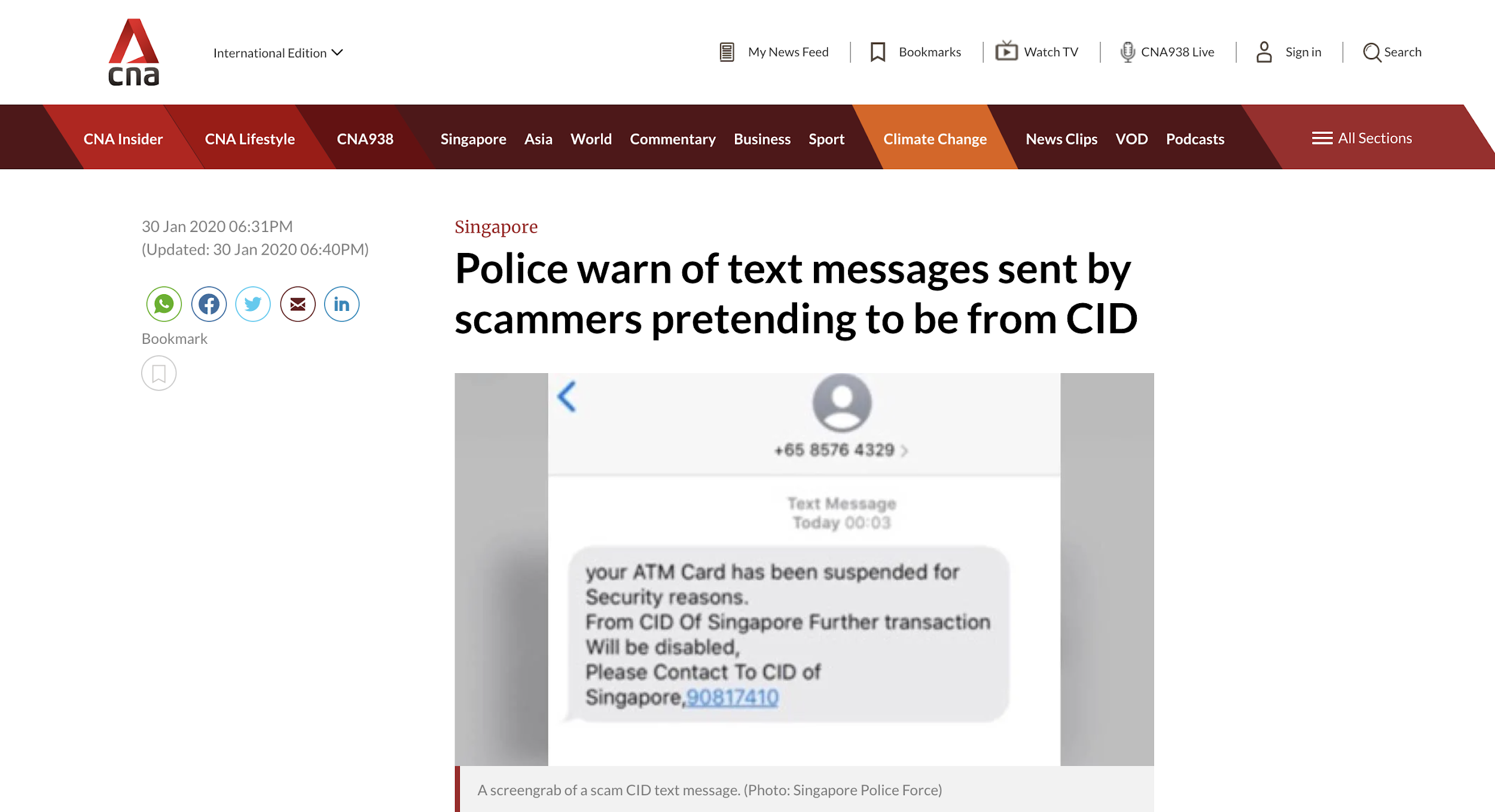


Fig 17: A police advisory to public of scam activity

Ubiquitous use of communication devices in all age groups has led to the most vulnerable within our society falling prey to scam messages. In 2019 alone, online impersonation scams accounted for more than $38 million dollars lost to scam. With this text classification model, we would be able to train it to detect typical scam language style and alert the user thereafter.

1. Social Media

Fake news detection is not new on social media; majority of popular social media sites such as Twitter have started flagging out misleading or fraudulent information on tweets. However, this intervention is not yet rolled-out on the large scale to all users within the platform. Through combining this classification of tweets with a classification on the bot-like nature of an account, we foresee that social media sites would be able to better detect and report both fake and misleading information as well as abusive or dangerous content.

# Further Improvements

From the insights and further model application, we have some additional improvements that we could apply in the future.

Increasing the Scope of the Model

1. Context

As the dataset that we utilised only consisted of headline titles, this may have restricted the model’s ability to understand and classify the article. For instance, the model may be classifying and relying more on a superficial individual words or relations within the headline. As an extension, we could do a secondary model classification of the article of the news through sentiment analysis.

1. Time

Since our data only consisted of The Onion/Not the Onion posts of the time frame September 2018 to November 2020, majority of the data is skewed towards headlines of the major world events within that two years; The US Presidential Elections and COVID-19. This is as corroborated with the data exploration that we conducted on our data which signaled out keywords such as COVID and Trump as significant within the dataset.

Due to this small time frame, the model could be biased towards certain keywords or topics and may not be able to classify headlines of other topics. Going forward, data could be collected from different time frames to expose the model to varying and differing topics for the model to significantly improve its ability to classify more conventional real-world headlines.

1. Author

Apart from simply analyzing the headline, we could also also conduct a secondary classification model to detect the credibility of the author of the message. For instance, on social media sites like Twitter, the model would be able to classify the author of the tweet based on the author and distinguish between real users versus bots or radically-inclined individuals. For news, this could be in distinguishing between more credible new sources and less credible ones like The Sun or Fox News.

Through combining the classification of the headline and the classification of the credibility of the author of the message, it would provide a more comprehensive classification to correctly filter real world headlines and news.

1. Language Styles

As the dataset is satirical in nature, it does not work as effectively with conventional real world headlines. As such, we could train our model on a larger dataset (for instance, the dataset that we utilised for further model applications) which has a larger variation of tones and language styles. This would ensure that the model is effective on real world headlines and able to improve real-world classification of information.

# Appendix

**Appendix A**

Training and validation accuracies/loss against number of epochs

This appendix describes the changes in training and validation accuracies and losses of neural network based models as the number of epochs is increased. This will be used to determine the optimal number of epochs for use for training the models such that the model is not overfitted on training data.

**ANN**

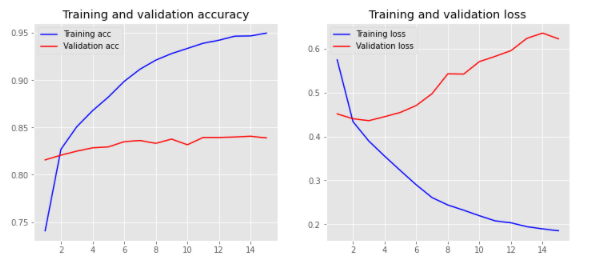


Fig 18: Training/Validation accuracy(left) and loss(right) for ANN

The optimal number of epochs to use for training is **5**.

**CNN**

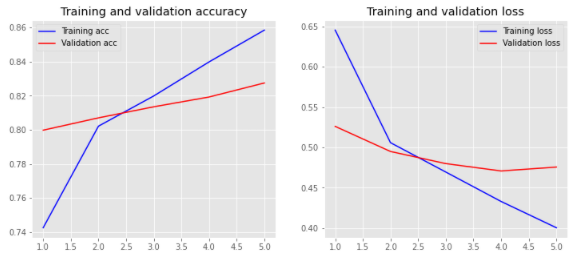


Fig 19: Training/Validation accuracy(left) and loss(right) for CNN

The optimal number of epochs to use for training is **5**.

**RNN - LSTM**

****

Fig 20: Training/Validation accuracy(left) and loss(right) for RNN with LSTM

The optimal number of epochs to use for training is **5**.

**RNN - Bidirectional LSTM**

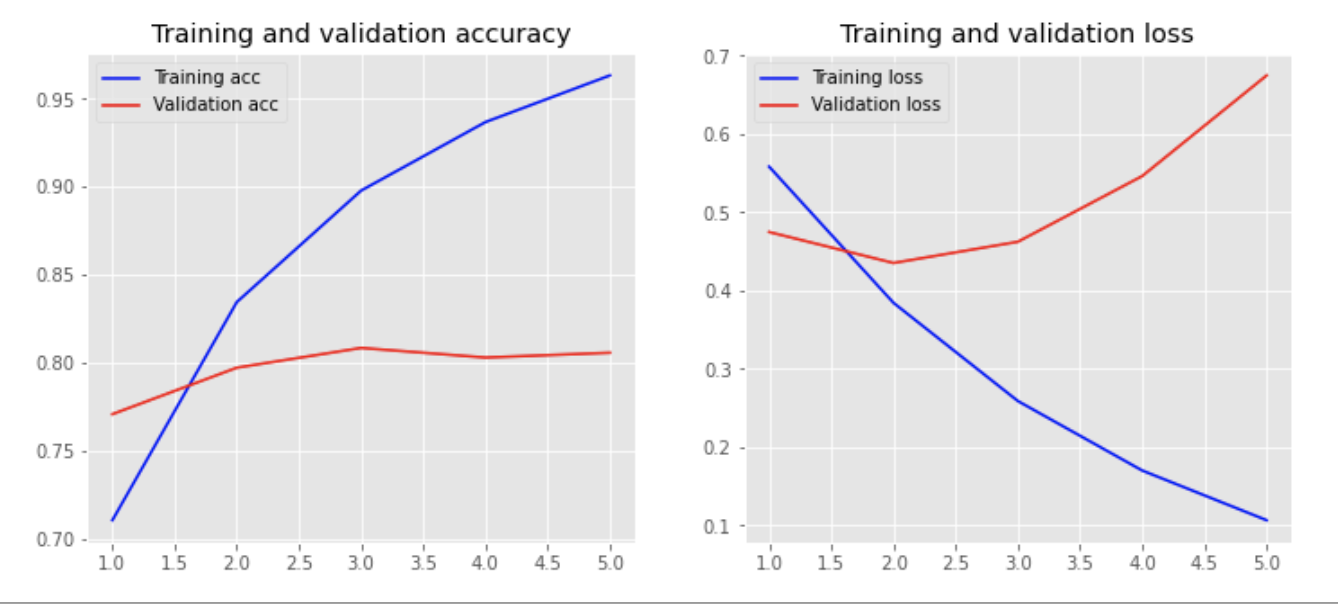
****

Fig 21: Training/Validation accuracy(left) and loss(right) for RNN with Bidirectional LSTM

The optimal number of epochs to use for training is **3**.

**BERT**

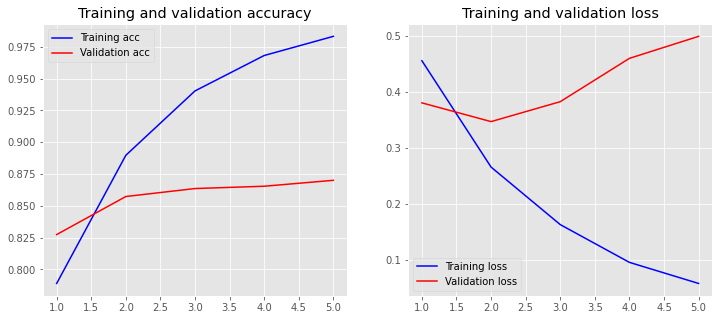


Fig 22: Training/Validation accuracy(left) and loss(right) for BERT Transfer Learning

The optimal number of epochs to use for training is **2**.

**Appendix B**

Each group mate contributed equally towards the project by exploring the model fine tuning for every model. Since we had 5 models, each of us was assigned a model and worked on it accordingly. Then, the results were collated and combined together to gather insights.

The work distribution was as such:

|  |  |
| --- | --- |
| Model | Group mate |
| Gradient Boosting | Erica |
| ANN | Justin |
| CNN | Jay |
| RNN | Jamais |
| BERT | Joo Bin |

1. <https://www.pewresearch.org/politics/2014/06/12/political-polarization-in-the-american-public/> [↑](#footnote-ref-0)