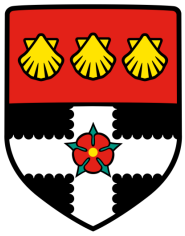

University of Reading

Department of Computer Science

Fake News Detection with Neural Networks

Conor Mehring

Supervisor: Varun Ojha

A report submitted in partial fulfilment of the requirements of
the University of Reading for the degree of
Bachelor of Science in *Computer Science*

April 29, 2021

Declaration

I, Conor Mehring, of the Department of Computer Science, University of Reading, confirm that all the sentences, figures, tables, equations, code snippets, artworks, and illustrations in this report are original and have not been taken from any other person's work, except where the works of others have been explicitly acknowledged, quoted, and referenced. I understand that if failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly.

Conor Mehring
April 29, 2021

Abstract

This report covers the testing of the improvements to fake news detection of LSTM's, GRU's, Bidirectional LSTM's, and Bidirectional GRU's over the classical methods. They are trained on a data set and evaluated on a test set. There is also further testing on articles from a scrape to attain articles not in either set.

Keywords: Neural Network (NN), Long Short Term Memory Neural Network (LSTM), Gated Recurrent Unit (GRU), Bidirectional, Real World Data, BERT, Natural Language Processing (NLP)

Report's total word count: 10711

Acknowledgments

My acknowledgment goes to my supervisor Varun Ojha. Without his help I would not of had a project idea or a basis of how to approach it.

Contents

Chapter 1

Introduction

In the modern age finding reliable news sources is not always an easy task. There are some out there who spread fake news. There are mean reasons people spread fake news, some are from possible profits from lying about a product, another could be a joke stating that a celebrity has died when in reality they have not, or just someone being miss-informed. As a consequence you can't always believe what you read online no matter how reliable the source is. It seems then that it would be a good idea for a tool to be created that can help users detect if an article is fake.

1.1 Fake News Detection

There are many possible ways to achieve this. A very simple method would be to use a dictionary that has each word with an attached truthfulness, however such a method would be poor as words can have multiple uses dependent on their placement in a sentence or the grammar used around them. An example is the difference between 'Let's eat, Kids' and 'Lets eat kids'. One declaring to some children that food is ready and the other that the children are the food. To avoid this, you could have different entries for words dependent on what comes before them, but then such a dictionary would be excessively large and take a huge amount of time to search. Such a basic method is not usable, but it is similar to the classical method of a Naive Bayes Classifier.

1.1.1 Classical Methods

To achieve any form of modern Fake News Sentiment Analysis first some example classical methods are needed to so that a baseline can be made to compare the modern methods to. As stated above, a Naive Bayes Classifier is an example of such a method. A variation of this is used, called a Multinomial Naive Bayes Classifier along with: Logistic Regression, Random Forest Classifier, and a Linear Support Vector Classifier. Without them there is no context for improvement for the chosen modern methods.

1.1.2 Modern Methods

Picking the modern methodologies is the hanging point for this project, so choosing the correct methods will be important. With that in mind, the chosen methodologies should be chosen from know methods which have been proven to show good results within the task of sentiment analysis. LSTM's and GRU's seem like a good choice as they are relatively new compared to

the classical methods, and are popular within the field of sentiment analysis. Moreover to be considered are there Bidirectional methods, as they could result in better classification.

1.2 Summary of contributions and achievements

With this project, I have implemented the usage of neural networks, specifically that of LSTM's, GRU's, and their Bidirectional variants. The purpose of this is to find a more suitable way of detecting fake news. For this I have had to implement a system to pre-process articles, train models, and test them on the training data set and articles from the web, not within the training data set. There has also been research into the understanding of overfeeding, and methods implemented to avoid the problems caused by it.

Chapter 2

Literature Review

To be able to complete the task of creating an improved methodology for fake news detection, first I will need to find correct examples of fake news detection, modern methods to replace them that will be suitable for the task of sentiment analysis, also a data set to train both the classical and modern methods on, and finally a way of testing the train methods on real world data that is not in the original data set.

2.1 Classical Methods

These classical methods consist of currently used methodologies for fake news detection. They are needed to be able to set a baseline to compare the chosen modern methods against, other wise there will be no way to see if the modern methods are an improvement. The following links cover the classical methodologies used in throughout this project [1, 2].

From these sources I chose the Logistic Regression, Random Forest Classifier, Linear Support Vector Classifier, and variant of the Naive Bayes Classifier the Multinomial Naive Bayes Classifier. From [1], the models are trained on the title of the data, rather than than the main text. Due to this some of the classification results may be different, as by using the main text the complexity of the problem has been increased. These methods where chosen as they are non-neural network methodologies, which should be a better contrast to the neural network methodologies within the chosen Modern methodologies.

2.2 Modern Methods

Since the classical methods have been found I need to find suitable modern methods that will improve on the classical methods. There are a lot of different metrics however that could be considered as an improvement in regards to machine learning. One metric could be the learn time, however in the case of fake news detection it is not relevant, as once the learning has been completed on the predication time matter. Speaking of predication time, it is an important metric to cover. If a method take too long, no matter how good its classification is, then it cannot truly be called an improvement as if it was intended as a product for the real world such time matters to people. The time I will set as a maximum is 10 seconds, however I am hoping for predictions to take much closer to 1 second or lower.

The most important metric is the accuracy of the classification as it is the focus of the output. Even if the AI is much faster than the classical methods, if its classification is very

poor, then it also could not be considered as an improvement. For a modern methods to be considered an improvement, it should have better classification accuracy than that of the classical methods.

2.2.1 LSTM: Bidirectional and Not

With the above metrics, a good choice would be that of a Long Short Term Memory Neural Network, otherwise know as an LSTM. LSTM's are a form of Recurrent Neural Networks (RNN) and where created to work with sequence prediction problems [3]. Fake news detection as a task, takes a sequence, that is the news articles mains text, and with that predicts a sentiment of the article which defines a Many-to-One task withing the scope of sequence prediction problems. Since this task matches the design of an LSTM, then it seems like a good idea to use it in this project. There will also be usages of the Bidirectional variant of the LSTM as they have been known to be an improvement over the regular LSTM.

2.2.2 GRU: Bidirectional and Not

Another possible choice is a Gated recurrent unit or GRU. GRU's have a similar premise to that of an LSTM's, but their structure is more simple, making it more computational efficient. Due to their similarity, the GRU is also suited for Many-to-One sequence prediction problems. In the paper [4] it was determined that GRU's and LSTM where better than those of the regular RNN, but it was not concluded if one was better than the other. Due to this conclusion, both the LSTM and GRU will be used throughout this project along with their Bidirectional variant for the modern methods.

2.3 Data Set

To be able to train any neural networks at all, there needs to be a data set with news articles and their sentiments. These sentiments need to represent that and the article is either true or false. This sentiment can be represented with words or with numbers. Such a data set would also need to have a large number of entries for a multitude of different websites with a balance of true and false data that matches that of the real world. This is required, as if you had an equal number of true and false articles it would not be a true representation of the real world and could skew the classification if it was ever used in the real world. Collecting such a data set would require a huge amount of time, as first you would need to find an article, read through the article, do background research in some cases, and with all this find its sentiment. As a consequence, there is not enough time to find and define such a data set, thusly it would be wise to use a pre-classified data set. With that in mind, the following set was used.

2.3.1 Kaggle

The chosen data set is from a website called Kaggle, and can be found here [5]. The data set contains news articles with the following information: URL, Header, Main text, and the assigned sentiment. It is in the form of a CVS with $\approx$4000 news articles each in the previously stated format. Below is an image of the data set on Kaggle.

This data set has a distribution of true to false of 1865:2120 meaning they are $\approx$14% more fake articles than truthful articles in this data set. As fake news is less common than that of truthful news this could be a problem, so there will be a test on an altered version of the data set with a much reduced number of fake news occurrences which accuracy will be

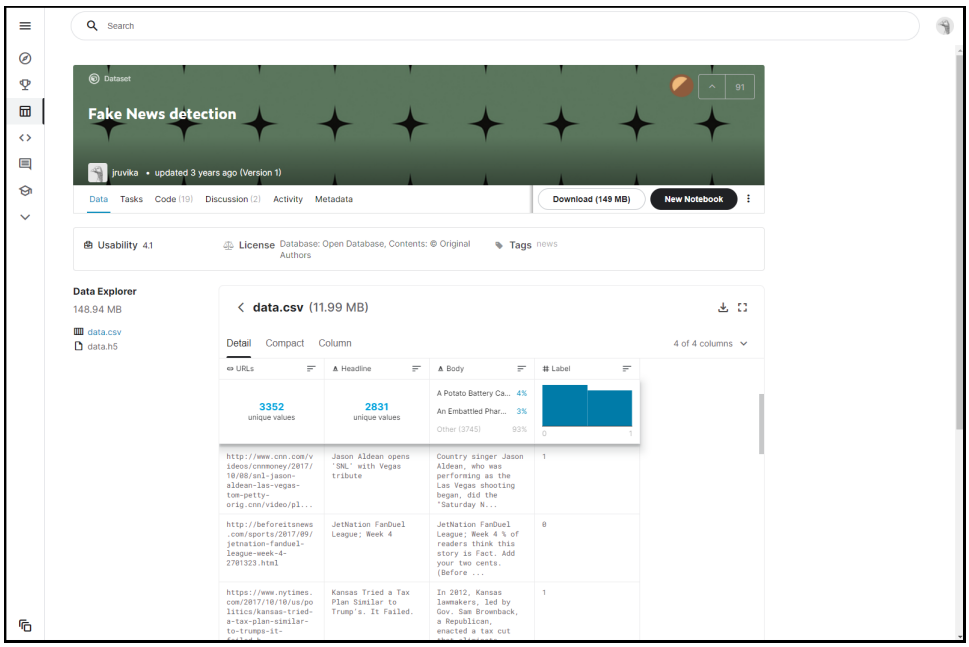

Figure 2.1: Kaggle Data Set

compared to that of the unaltered data set. As this data set is not created by me and is off of the internet, there will also have to be considerations into the fact that the sentiment of some of their articles could be incorrect. This could occur due to the data set creator having unknown political bias's, or they could of simply been miss-informed causing the sentiment to be incorrect. As checking all of the articles would approximately take the same amount of time as finding the articles myself, secondary verification should be made in the case of real world data not within the data set.

2.3.2 Real World News Articles

To achieve testing on other real world data not within the original data set, there will need to be a tool created that can collect the data from different news articles, extract the main text from them, tokenize the data, then feed the data to the best trained neural networks, and finally have a human verify the results. With this, first there will need to be created trained neural networks, of which the best ones with the classification accuracy should be chosen.

2.4 Summary

The purpose of this project is to try and find improved methods for fake news detection over the classical methodologies. This improvement will be based off of how accurately the Neural Networks can predict the sentiment of a news article, but they will also need to be able to achieve this with a prediction time that is reasonable.

Chapter 3

Primary Concepts

For the chosen Modern methodologies, there is a need to explain them at a deeper level, cover there basis, and how they work. To do this, I will use the following resources [6]. Firstly, before I cover the LSTM's and GRU's I need to cover the RNN, which is what these models are based on.

3.1 RNN

RNN or Recurrent Neural Networks are a form of sequential neural network. This means that each time it is passed and input, it does so one at a time. They have a simple structure with an input at the bottom, the hidden state from the left, and the output to the right. The hidden state is the output from the previous unit and it used to pass on predictive information on the previous unit. Both the input and hidden state are sent through a Tanh activation to produce the output.

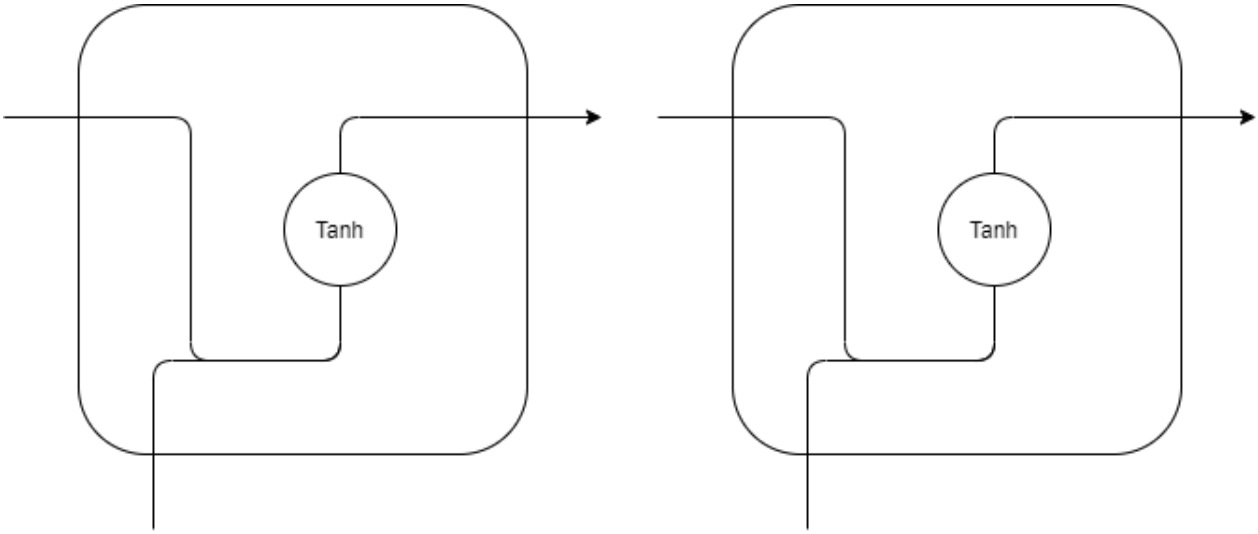

Figure 3.1: Unfolded RNN

The above Figure 3.1 shows an unwrapped model. To train one of these models you need to pass each input sequentially into each unit, and once the unit creates its output, it then passes the output to the next unit along with the next input. The problem with these models is that they have short-term memory loss. If you where to give the the task to predict the next word in this sentence: "Jim went to Africa last summer. While there he visited an animal

reserve, as saw some amazing animals. Jim has recommend to Sam to go to ..." with the task to replace the "..." with "Africa", it will fail as by the time it has got to the end of the sentence the model has forgotten the word "Africa".

3.2 LSTM

LSTM's where designed to resolve this problem by the use of gates. LSTM's have 3 gates: the forget gate, the input gate, and the output gate. The purpose of these gates to to deiced what information is kept and what is not. With the addition of these the LSTM has a better short term memory retention and could perform the task set for the RNN. There is also an addition input of the cell state to add additional memory retention. They are taught in the same manner as an RNN. Below is the general structure of an LSTM.

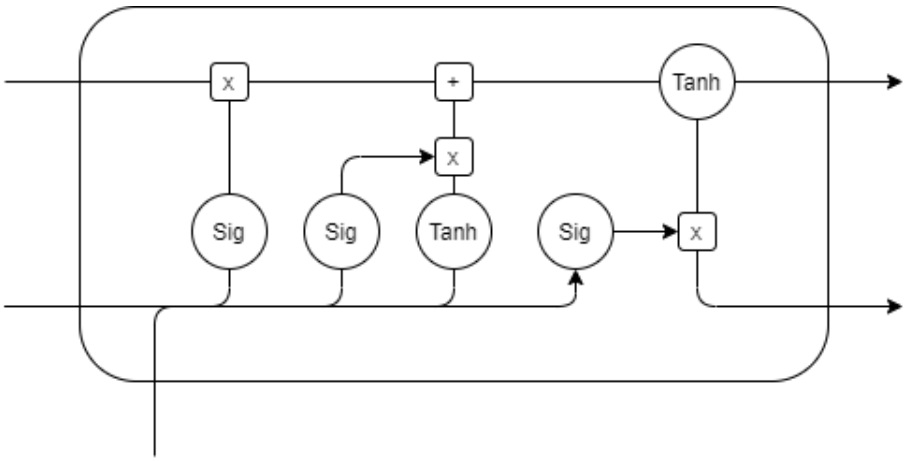

Figure 3.2: Unfolded RNN

3.3 GRU

The GRU is a more modern version of the LSTM. It was designed to reduce the number of gates in the LSTM to improve the computational efficiency. It only has two gates: the reset gate and the update gate. It also loses the cell state and instead passes the information through the hidden state. They are taught in the same manner as an RNN. Below is the general structure of an GRU.

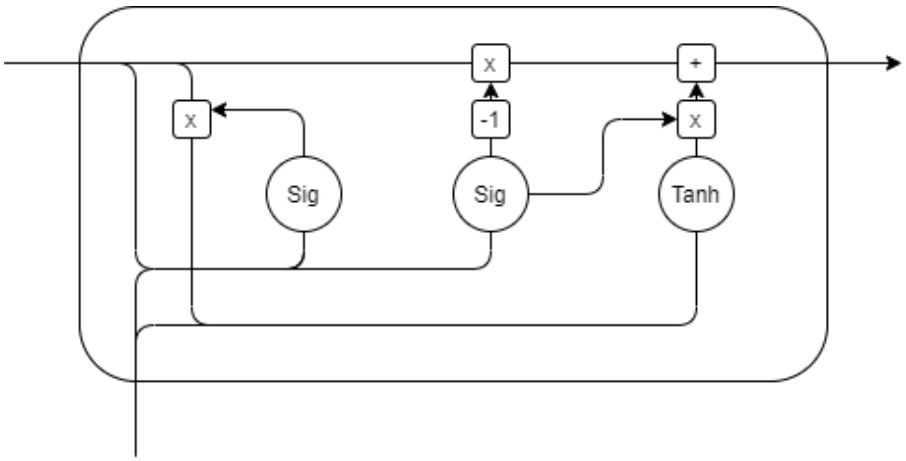

Figure 3.3: Unfolded RNN

3.4 Bidirectional

There is a further improvement to the these models which is making them Bidirectional. This means that while they are having there forward pass they also have a simultaneous backwards pass, presenting the data in the opposite order. This helps with very ling inputs as the models see both ends ans thus don't forget anything from either.

Chapter 4

Methodology

The task here is to take the main text from a news article, feed it to a trained neural network, and to receive a True or Fake classification from the neural network. To achieve this first there needs to be training data containing the main article text and its sentiment. This is extracted from the Kaggle Data Set and stored in a excel file. This excel file will need to be loaded, pre-processed, and the split into the train and test sets. This training data will need to be feed to both the Classical Methods and Modern Methods, and the test data is to be used to test the train NLP methods on the prediction accuracy. There will be multiple training cycles with different parameters to try and achieve the best classification possible by all methods.

Once the models have been trained, they and there tokenizer will need to be saved. These can then be loaded into the article scraper to test the trained neural networks on data that is not within the training data set. This is an important part as it can show if there were any problems with the balance of the training data. If there is, then adjustments can be made to the training data, and new neural networks can be tested on this altered data.

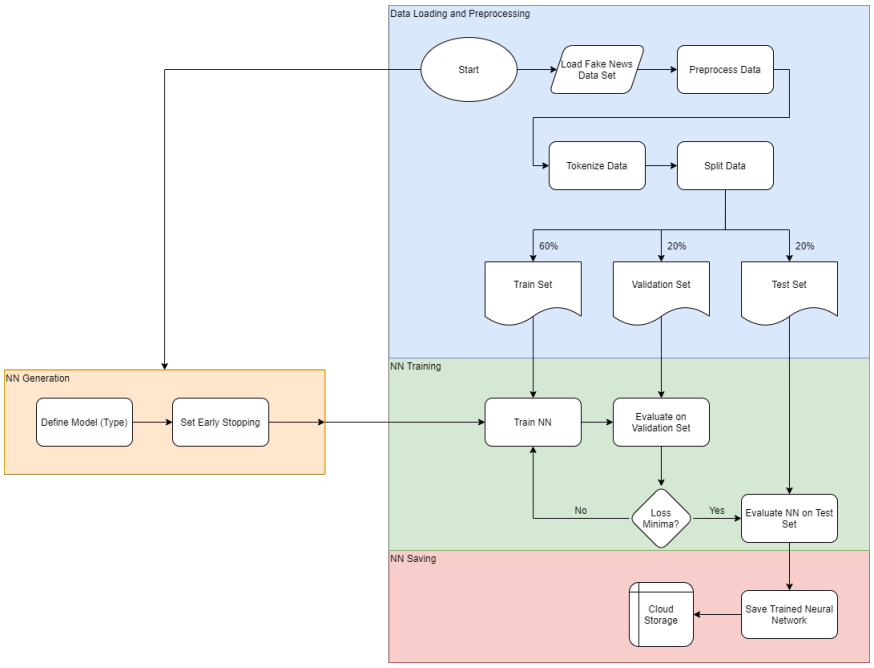

Figure 4.1: Training Structure

The above Figure 4.1 shows the general structure of the training methodologies, where the blocks show the different stages of the training. The following sections cover the training methodologies in more detail and expand upon how these tasks will be completed.

4.1 Data Pre-Processing

There will need to be data pre-processing, once the data is loaded, to avoid any additional tokens. For instance if you have the sentence "This is a 'test' sentence." and tokenize it, there will be the tokens "This", "is", "a", "'test'", and "sentence." as the tokenizer separates each token by the space character. From this you can see that rather than having the the "test" token separated from the "'" there is a combined token. As the word "test", and "'" will have their own tokens, there is also now an extra token for "'test'". To avoid this there will be padding of spaces around "'" character, as well as other characters like the period at the end of the word "sentence".

Without this pre-processing there will be extra tokens, whenever there is a word followed by another character without a space. These extra tokens add ambiguity to the data set, because even though you could have the tokens "'test'" and "test" they fundamentally mean the same thing, making it harder for the NLP methodologies to be able learn the problem. It would also increase the size of the tokenizer, which is not beneficial.

4.2 NLP Methodologies

There are two main classes for the machine learning methodologies, the Classical Methods, and the Modern Methods. Both of them are given the same training and test data to make sure that there is fair testing for both groups. This will result in a more consistent comparison between each methods, giving better validation to any perceived improvements by the modern methods.

All models will need to first be defined, as without this first step no training can occur. This will also involve the setting of their parameters which will need testing to find the best setup for each method, so that they can get the best possible classification. For the modern methods there will also need to be early stopping defined to make sure that the models achieve the best possible classification.

With the models defined they will then be trained. The classical methods are passed the training data, which will of been pre-processed and tokenized, using the fit function, which will train them. The modern methods have a few more parameters for learning. They need their batch size, epochs, callbacks, and validation split to be defined. Once the modern methods are trained, they will be saved for later use.

Once the Classical and Modern methods have been trained, they will also need to be evaluated. The test data will be used here by passing it to each model to predict the test data's sentiment. With this a comparison of the prediction from the models against the real sentiment from the test data. The metrics that will be displayed are the Accuracy, Precision, Recall, F1-Score, a Confusion Matrix, and the classification from a random test sample for a specific view of the classification.

4.2.1 Real World News Articles

As well as training the NLP Methodologies and checking them against the training data, there will also be a system for checking them against real world data not in the data set. This is an important stage of the testing as there could be certain problems with the data set, such as imbalance, or that the fake news within the training data could be extreme examples of fake news that would be obviously fake neglecting articles that have a more subtle fakeness.

Such a system could only be used on trained models, and rather than testing it on all of the models, only the best versions should be used, as testing models that have poor classification on the training set will not do any better than they do on the training data. The best models will be determined from the training data in the final stages of training.

Such a system for testing on News Article not within the data set would require a system that can scrape the data from the News Article, tokenize the data, and then get the prediction from the trained models. The results will need to be evaluated by a human, as the News Articles are not within the data set meaning that they don't have labels.

4.3 Summary

The designed system will have to be able to tokenize the training data and the articles from the scraping tool. With the tokenized training data the models will be trained and evaluated. The scraped news articles will be used as further evaluation for the best models, and will also be a good evaluation of the chosen trained data set.

Chapter 5

Implementation

The following Implementation section will cover the two main stages of the project; Training and Real World Data testing. The first uses the Kaggle data set and applies some pre-processing sets before the data is used from training the defined models. Once the models are trained there tokenizer and the models themselves are saved to be used in the Real World Data testing. This second stage will load the models and there tokenizer, get the HTML of an article, apply the pre-processing to the article, tokenizer the article, and the get a prediction from each model that is human validated. This chapter covers the implementation of both stages along with their substages.

5.1 Data Pre-Processing

To view some code, view Appendix A Data Pre-Processing. Firstly, the training data needs to be loaded into the correct file location, in this case the sample data file in Google Colab, to allow the program to load the training data. Once the data is correctly loaded, it can be read from the file location using the pandas library read excel function storing the data in a Data Frame. The data that is contained within the excel file is the main text from the article and its sentiment.

Once the data has been loaded it can now be pre-processed. The pre-processing purpose here is to reduce the number of tokens that would be created by the raw data. This is done by padding none-alphabetical characters with space's before and after them. For instants, the sentence "This is a 'test' sentence." becomes "This is a ' test ' sentence .". Without this pre-processing the word test would have a token for test, the symbol ', and then the combined token of 'test'. By adding the padding, there is no token needed for 'test', reducing the overall number of tokens. This method is applied to all of the articles text.

Once all of the articles have been processed, the tokenizer is then created. The purpose of the tokenizer is to convert the alphabetical human readable data into numerical machine readable data. This is done by first defining the tokenizer, and the fitting the tokenizer to the data sets text. With this, the tokenizer can read what characters there are and then assign a number to each of the characters. Next the text is then feed into the tokenizer and the result is saved.

The tokenized data is then padded. Padding means that all of the articles are made to be the same length. This can be done in two ways. First the max length of an article is set to the length of the longest article. Secondly, a max length could be defined. Once this max

length is decided, the data is either padded with zeros at the end to match the max length, or the end of the data is chopped off if the overall length of an article is longer than the set length. Both methods are used here, with the second method using a max length of 500 characters.

Once the data has been tokenized, the data now needs to be split into the train and test sets. The development set will be made later using the validation split from the modern methodologies. The train and test split is 80% and 20% respectively. This is done by using the Train Test Split function from the library Sklearn Model Selection. The data is now ready for training, and the tokenizer is saved for later use, in conjunction with the modern methods.

5.1.1 Classical Methods

All classical methods are trained in the same way. To view some example code for the Logistic Regression view, Appendix A Classical Example. The first part of the training requires that the models are defined. When the models are defined, each of them gets there required parameters, which are different depending on the type of model. Once the model is defined it can then be trained. This is done by calling the model with the fit function and then passing the function the training data containing both the tokenized text and there sentiments. There is also a timing function used so that the training time can be tracked.

5.1.2 Modern Methods

All modern methods are trained in the same way and have the same generation. The difference between them is when the model type is declared and if it is Bidirectional or not. To view some example code for the Bidirectional LSTM view, Appendix A Modern Example. When changed from Bidirectional the Bidirectional tag is removed, giving a none Bidirectional variant of the model.

Each of the models is first defined as a Sequential model, and have an Embedding layer added after. Next comes defining which model it is. There will either be a Bidirectional with an LSTM or GRU, or it will just be an LSTM or GRU. Next added is the Dropout and Dense layer, where the latter is set to Sigmoidal activation. Next, the model is compiled with the Adam Optimizer, the Binary Cross Entropy loss function, and the Accuracy metric to be displayed. The model is also set to summary so that when training occurs, statistics on the models learning can be displayed.

Once the model has been defined there Early Stopping conditions are defined.The Early Stopping is important as it can stop the learning at the right point, but has to be set to not stop too soon. If you overfeed a model on the same data, it can reduce its classification on any other set, as it will instead of finding general features within the data set that pertain to a certain class, instead find specific features within the data set that may not be covered in other data. For this early stopping, it monitors the val loss, which is the loss calculated from the development set, and checks when this val loss is at a minima. There is also some Patience within some of the tests as it can stop the early stopping from stopping the learning too soon. This can occur if the model has a dip in its learning early on, but if left to continue would improve. There is also the Restore Best Weights set to true so that the best version of the model throughout the training can be loaded, rather than the last, which could of dipped from the best, especially if there is some Patience.

Once the Early Stopping parameters have been defined the models are now ready to be trained. This uses the model dot fit function from Tensor Flow. This function takes as inputs the training data for both the tokenized text and sentiment, the batch size, the number of epochs, the validation split, and the Early Stopping parameters that where defined prior. The batch size determines the number of samples per-iteration where an iteration predicts on the batch, and then does the backpropagation to learn the problem [7]. The epochs are the number of times that the training will full go through the data set. So, if the epochs are set to 20, then the model will have its first cycle where it gets trained on the data set and when it finish's, it starts a new cycle. This will occur 20 time, or until the model meets the Early Stopping parameters. The validation split determines how much of the trained data is used for validation. In the case of this project it was set to 20%. The validation data is used to evaluate how well the model is learning the problem over the epochs, and also as a stopping parameter in the Early Stopping. With all of this the models are trained, and once there training is complete, they are save for later use.

5.1.3 Test Evaluation

Once the models have been trained, they are then evaluated using the test data. To view some code, view Appendix A Classical Example or Modern Example. Firstly, the test data is passed to the trained NLP methods and the resulting predictions are stored in a list. The first metric uses is the accuracy, using the Metrics Accuracy Score function which is passed the actual sentiment of the test data, and the predicted data from the trained NLP method. This accuracy is then displayed as a decimal, between 0 and 1 to be interpreted as a percentage.

Following the accuracy metric, for each of the classes true and fake there is shown the Precision, Recall, F1-score, and the Support. From [8] the Precision is the percentage of class classifications that are correctly classified, the Recall percentage of the total number of classifications of the class, and the F1-score is a balance of the Precision and Recall in one score. The support is the number of articles for the class.

Next is a confusion matrix. Confusion matrices are a very useful way to visual evaluate the results of the classification of a trained model. As we have two classes for the models to predict, the confusion matrices will be a 2x2.

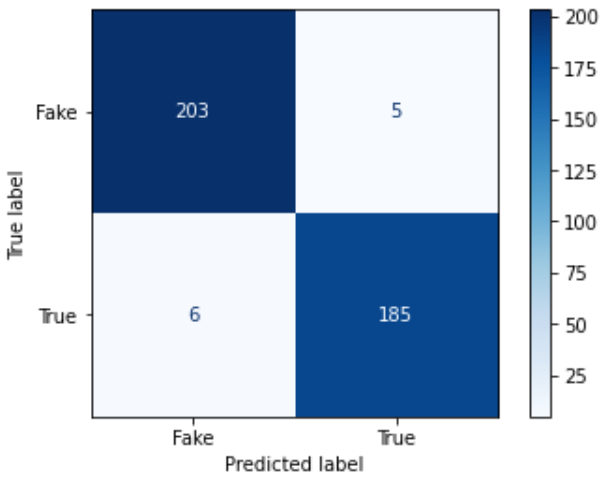

Figure 5.1: Confusion Matrix

The above Figure 5.1 is an example Confusion Matrix. It shows the diagonal from top left to bottom right. The top left is the correctly article correctly classified as fake, and the bottom right the articles correctly classified as true. The true predication's are where the classification of the models matches that of the data set. With this it can be seen that there where 203 correct fake predication's and 185 correct true predication's. Following the correct predictions are the incorrect predictions shown as the diagonal from top right to bottom left. The top right shows the true prediction's that should of been fake, and the bottom left shows the fake prediction's that should of been true. With this, we can read that here there where 6 fake prediction's that should of been true, and 5 true prediction's that should of been fake.

5.1.4 Real World News Articles

The task here was to make a tool that when passed a URL could get the main text from the article, tokenize it, and then feed it to the trained NLP models. In the first stage of scraping the news article the library bs4 was used as it can grab the data from the website in HTML form. With this, the main text can be grabbed using the following tags: "p","h2","figcaption","a href". These tags are HTML tags that separate different sections of text, and using the Body Find All function on these tags, the main text can be extracted. The "p" tag is for paragraphs, the "h2" tag is for subheadings, and the "figcaption" and "a href" tags are for text around images. Once the text has been gathered, the input sanitisation, which is the same as for the training and can be viewed in Data Pre-Processing within the Implementation Chapter. Code for this can be viewed in Appendix B Article Scrapping.

Following the data gathering comes the tokenization. This will use that associated saved tokenization which was made with the trained models during their training. This specific tokenization is needed, as it matches the inputs of the trained models. The tokenization will first need to be loaded into the correct file location so that it can be loaded into Google Colabs memory, and a variable is declared to be the loaded tokenizer. Once it has been loaded, the article is tokenizer, and is now ready to be passed into the trained models for prediction's. To view code go to Appendix B Loading Tokenizer.

To use the models first they need to be loaded into Google Colab, and then loaded into variables using the functionality of the Tensor Flow Library. With the models loaded, they can know make a prediction on the article. This is simply done by using the predict function on the model with the tokenized article. Once the model has made its prediction, a human needs to verify the output, as since this article is not from the data set it has not label. To view code go to Appendix B Models.

5.2 Summary

There are two main stages that have been implemented; Training, and Real World Article testing. Each of the stages also has substages. The Training first needs to load the training data, complete the input sanitization to reduce the number of tokens by padding not alphabetical characters. Following this the data needs to be used to first fit the tokenizer, and then be tokenized by the fitted tokenizer. With that, it is then split into the train and test data where training can occur once the models have been defined with there specific parameter's. Once training is complete, the models are save and used in the the second stage with the saved tokenizer.

The second stage involves loading the best trained models along with their associated tokenizer. Next it will grab the HTML from an article and using the previously mentioned tags gets the main text from the articles. With this text the input sanitization is applied and then the article is tokenized by the loaded tokenizer. Next this article is passed to the loaded models for prediction and then the result are validated by a human.

Chapter 6

Results

This chapter will cover the results from testing with different parameters for the models and different applications of the training data set, for each of the models from both the Classical and the Modern Methodologies. These results will be in the form of scatter plot diagrams, with two for each model. The first class will contain the correct classification, and the second the incorrect classifications. These results where gained from each model once they had been trained on the train set, with the evaluation being performed on the test set, displayed in the confusion matrix. For the correct classification, the better classifiers have results that are closer to the top right, and for the incorrect classification they should be closer to the bottom left. If the result are closer to the opposite corner to the good classification per-graph, then they have shown poor classification.

Firstly, each of the models will be evaluated separately to get an understanding of their classification. Following that, all of the Classical Methodologies will be compared and all of the Modern Methodologies will be compare to find the best from each of these sets. Finally, both sets will be directly compared. For the displayed data the only case where the inputs are unsanitized is when they have that label specifically. All other results used sanitized train and test data.

There was stated previously in this document that there would be testing with a more balance version of the data set. When these tests where applied, there was no improvement to the overall classification on the training and test set. There was also a negative impact on the results from the Real World Data Scraper. As these results did not improve classification, there was no extensive testing and thusly are not in the following section. However, from the limited testing it was found that there was no improvement, and overall the the classification was worse on the Real World Data.

6.1 Classical

The results from the Classical Methodologies are going to be used as the baseline for the Modern Methodologies as the purpose of this project is to find new and improved Methodologies. It will also be useful too see how well these Classical Methodologies do with full articles, as in most cases they were used on much smaller amounts of data, such as just the title of the news article. They will also be shown their training and test accuracy, where the test accuracy is the more important, as it is data that the models have not seen before.

6.1.1 Logistic Regression

The Logistic Regression is the first of the Classical Methodologies to be evaluated. It has a best training accuracy of $\approx$ 90% and a best test accuracy of $\approx$ 67%, which is not the worst test accuracy from the Classical Methodologies, but it is not a good classification.

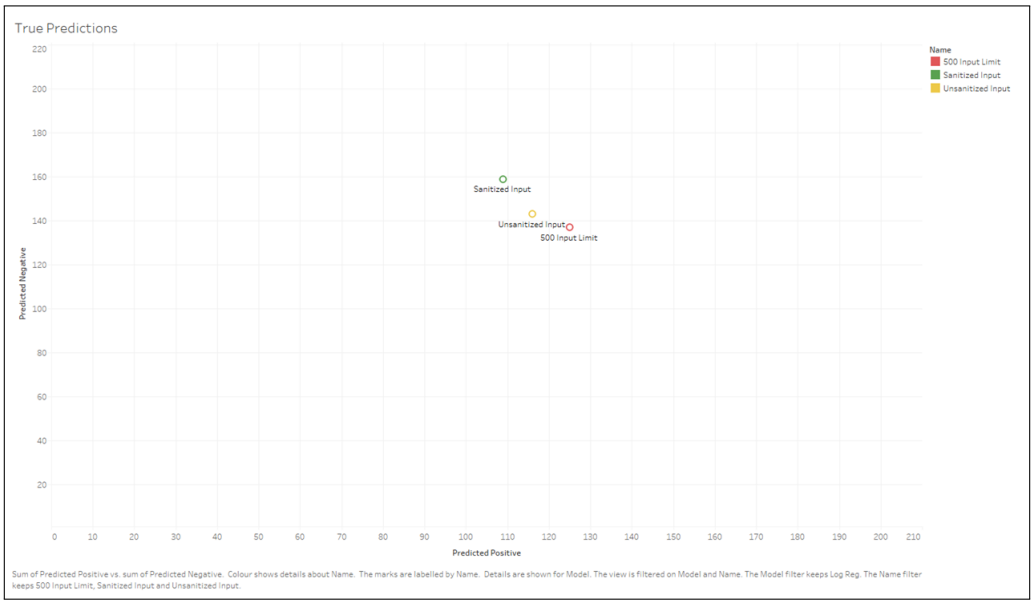

Figure 6.1: Logistic Regression True

The above Figure 6.1 shows the number of correct classification for both the true and fake classes. There are three test models here. The green shows the results from when the input data was sanitized, the yellow for unsanitized, and the red for the 500 word limit. Out of the three, the sanitized has the most number of correct classifications of 268 and with the unsanitized having the lowest number of 259.

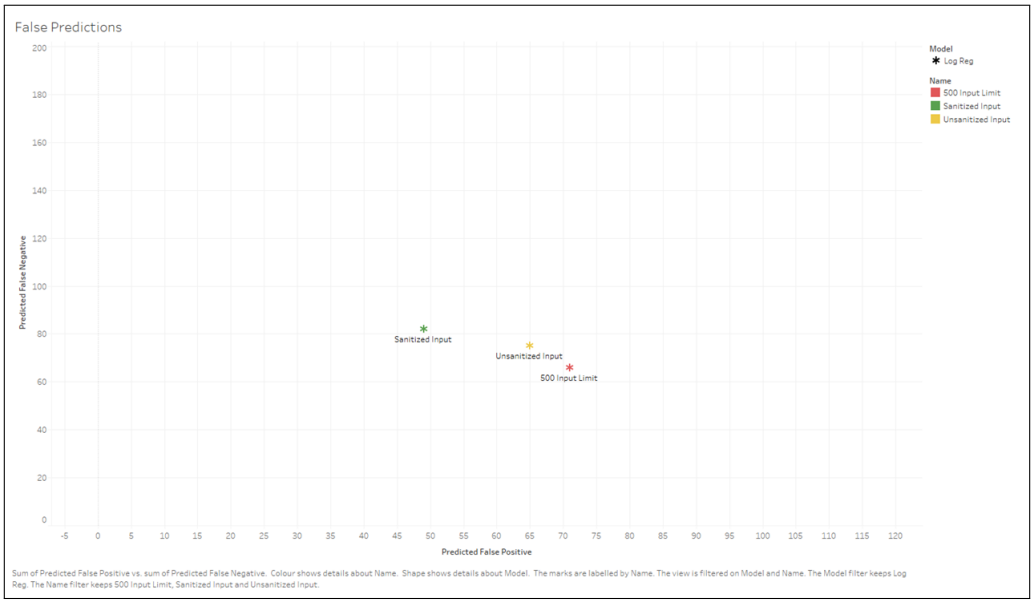

Figure 6.2: Logistic Regression Fake

The above Figure 6.2 shows the number of incorrect classification for both the true and fake classes. There are the same models here as where in Figure 6.1. The sanitized input has the lowest number of incorrect classifications at 131, with the unsanitized having the highest number of inputs of 140. From both of these it can be seen that the sanitized input has the best overall classification, getting both the highest number of correct classifications, and the lowest number of incorrect classifications. Since the sanitized input did the best, it will be used for all later stages of discussion.

6.1.2 Random Forest Classifier

Following the Logistic Regression is the Random Forest Classifier. It out of all of the Classical Methodologies has the best training and test accuracy's of 100% and $\approx$ 84% respectively. This is a large improvement of the test accuracy of $\approx$ 67% for the Logistic Regression. This unlike the Logistic Regression, has some good classification that makes a good baseline to compare the modern methodologies to.

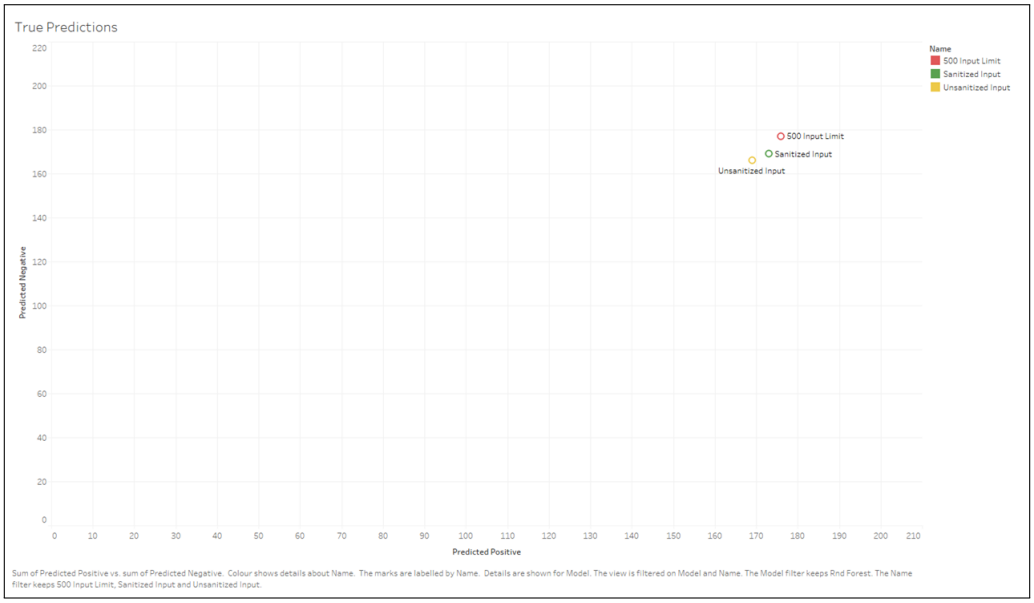

Figure 6.3: Random Forest Classifier True

The above Figure 6.3 shows the number of correct classification for both the true and fake classes. There are three test models here. The yellow is the unsanitized, green is the sanitized, and red is the 500 limited articles. The method with the highest number of correct classifications is the 500 limited articles with 353, and the worst is the unsanitized inputs with 335.

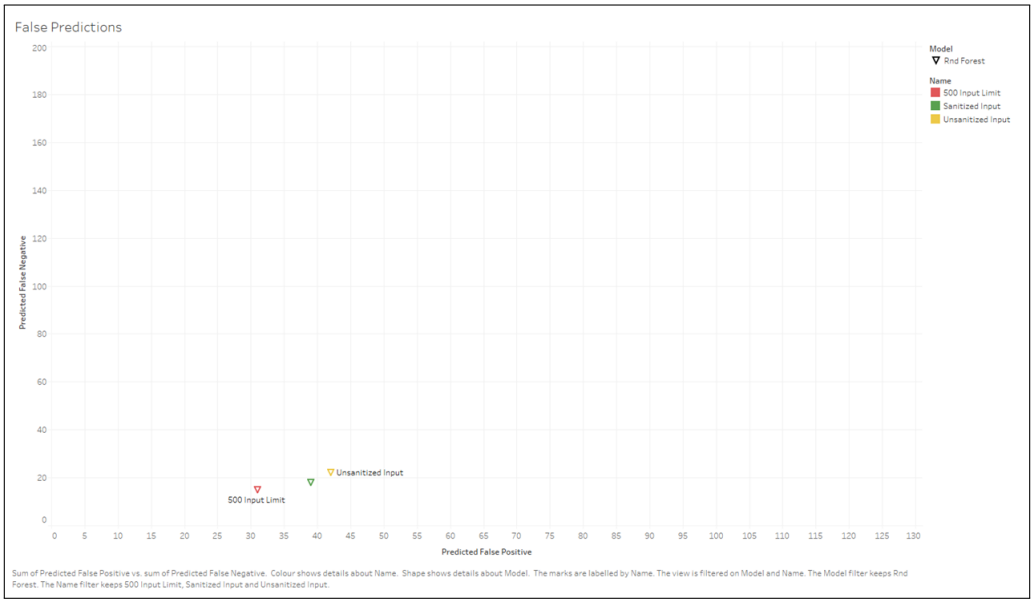

Figure 6.4: Random Forest Classifier Fake

The above Figure 6.4 shows the number of incorrect classification for both the true and fake classes. There are the same models here as where in Figure 6.3. The model with the lowest number of incorrect classifications is the 500 limited articles input with only 46 incorrect classifications, and the highest being the unsanitized inputs with 64 incorrect classifications. Having the highest number of correct classifications and the lowest number of incorrect classifications, the 500 limited articles model has the best classification of the three. As it did the best, the 500 limited articles model will be used in later discussion.

6.1.3 Linear Support Vector Classifier

The third of the Classical Methodologies is the Linear Support Vector Classifier. Unlike the Random Forest Classifier, this methodology did not improve on the Logistic Regression, in fact it did worse with a training classification of $\approx 77\%$ and a test of $\approx 62\%$, whereas the Logistic Regression got a $\approx 67\%$ test accuracy. With an accuracy this low, it is almost a coin toss as to which class it is going to pick.

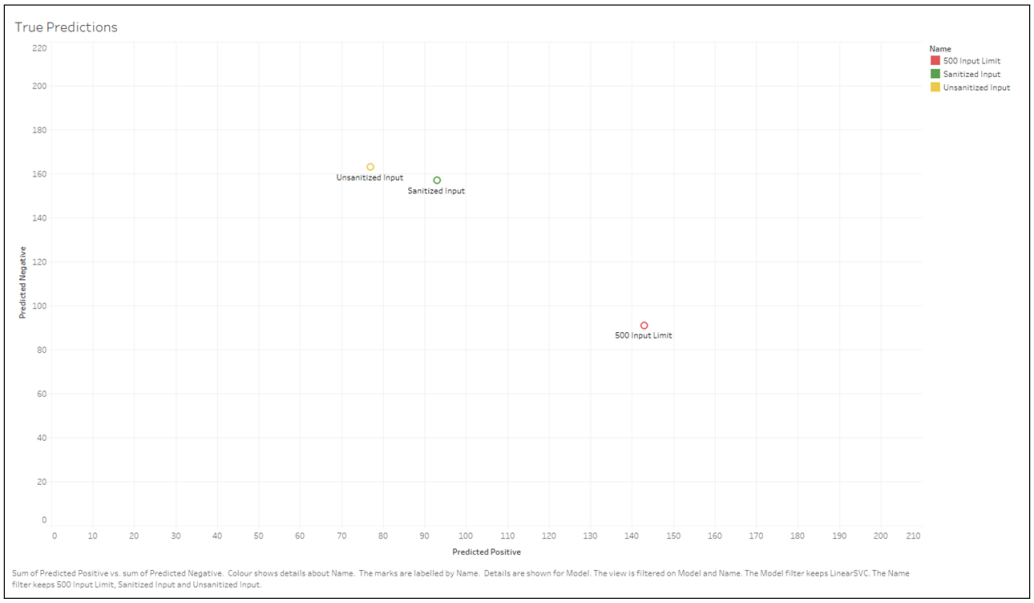

Figure 6.5: Linear Support Vector Classifier True

The above Figure 6.5 shows the number of correct classification for both the true and fake classes. There are three test models here. The yellow is for the unsanitized inputs, green is for sanitized inputs, and red for the 500 limited article. The model with the highest number of correct classification is the sanitized input with 250 correct classifications, and the 500 limited article has 234, making it the lowest.

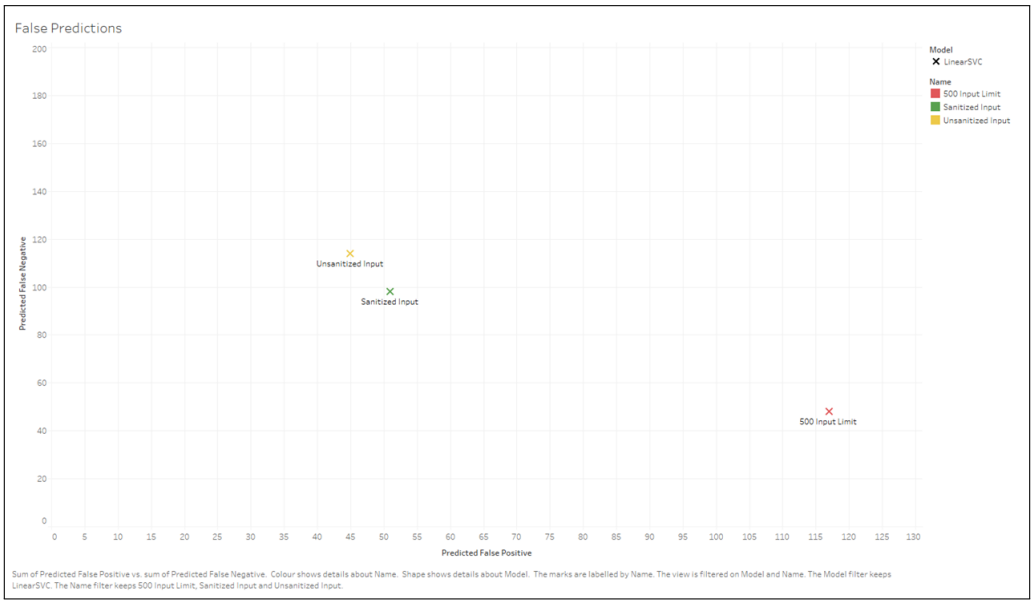

Figure 6.6: Linear Support Vector Classifier Fake

The above Figure 6.6 shows the number of incorrect classification for both the true and fake classes. There are the same models here as where in Figure 6.5. The model with the lowest number of incorrect classifications is the sanitized input with 149 incorrect classifications, and the 500 limited article with 165 incorrect classifications means it has the most incorrect classifications. As the sanitized has the highest number of correct classifications and the

lowest number of incorrect classification of the three version here, it will be used in the further discussions.

6.1.4 Multinomial Naive Bayes Classifier

The final of the Classical Methodologies is the Multinomial Naive Bayes Classifier. Normally you are to leave the best till last, but in this case this model has the worst train and test accuracy of $\approx 69\%$ and $\approx 58\%$ respectively. In the previous model, the Linear Support Vector Classifier, it was stated that it was basically a coin toss, but its even closer with this model, especially saying that the test data leans more to the fake articles.

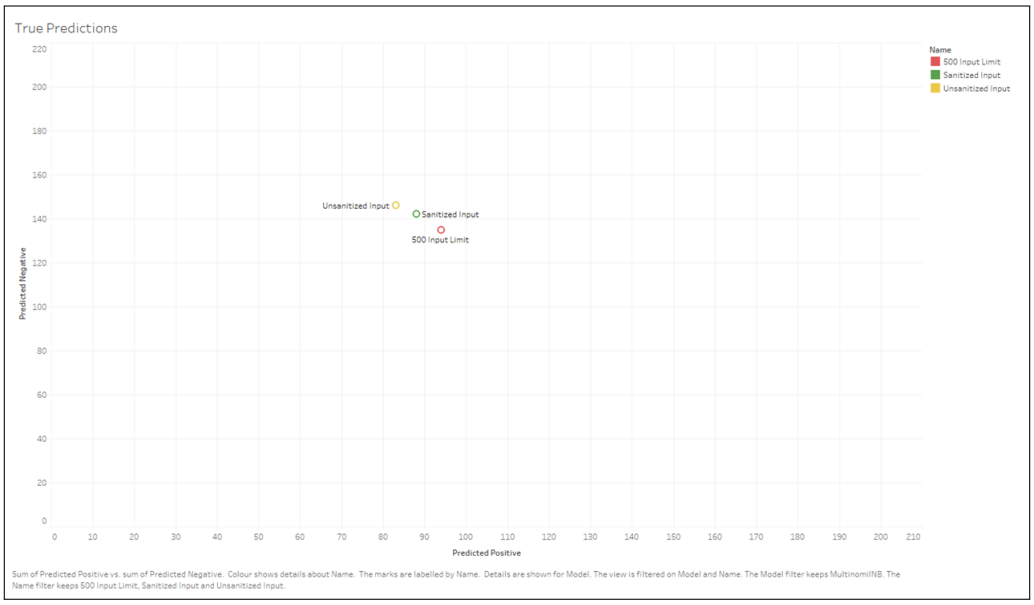

Figure 6.7: Multinomial Naive Bayes Classifier True

The above Figure 6.7 shows the number of correct classification for both the true and fake classes. There are three test models here. The yellow shows the unsanitized inputs, green the sanitized inputs, and the red the 500 limited articles. The model with the highest number of correct classifications was the sanitized inputs with 230, and the model with the lowest was the unsanitized with 228.

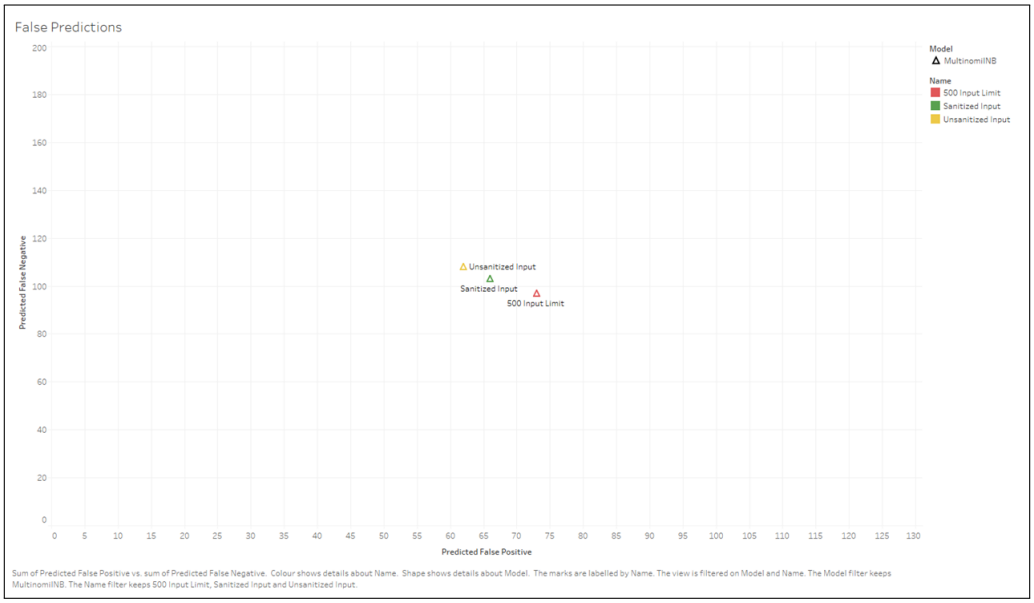

Figure 6.8: Multinomial Naive Bayes Classifier Fake

The above Figure 6.8 shows the number of incorrect classification for both the true and fake classes. There are the same models here as where in Figure 6.7. The model with the lowest number of incorrect classifications is the sanitized input with 169, and the model with the most was the unsanitized with 170. The best out of these three models is the sanitized as it has the highest number of correct classifications and the lowest number of incorrect classifications. Even though the sanitized inputs did the best of these three, all of their classifications where so bad that they where separated by 3 classifications.

6.2 Modern

With the Classical Methodologies out of the way, it is time for the Modern Methodologies. The classification accuracy of these models is the most important as they are meant to be improvements over the Classical Methodologies. They will also be shown their training and validation accuracy, where the validation accuracy is the more important, as it is data that the models have not seen before.

6.2.1 LSTM

The LSTM is the first of the models to be tested, and it has a problem. It can only learn the problem if the article is limited, in the case of the testing with a 500 character limit. The reasons for this will be discussed later, but as a consequence, unless the models are limited to 500 as there input, they fail to learn the problem. This problem also happens in the GRU, but their Bidirectional variants are not effected. Due to this only the 500 limited test results will be shown as the rest group together, for the regular LSTM's and GRU's. With that in mind, they where still tested, which took a long time, with one of the LSTM models taking over 3 hours to learn, making the learning process for all of the models a multi, all-day processes.

With that being said the best test accuracy for the LSTM was $\approx$ 96% and the validation accuracy was $\approx$ 92%. These are some decent classification results, with only $\approx$ 8% of the results not being in the correct class.

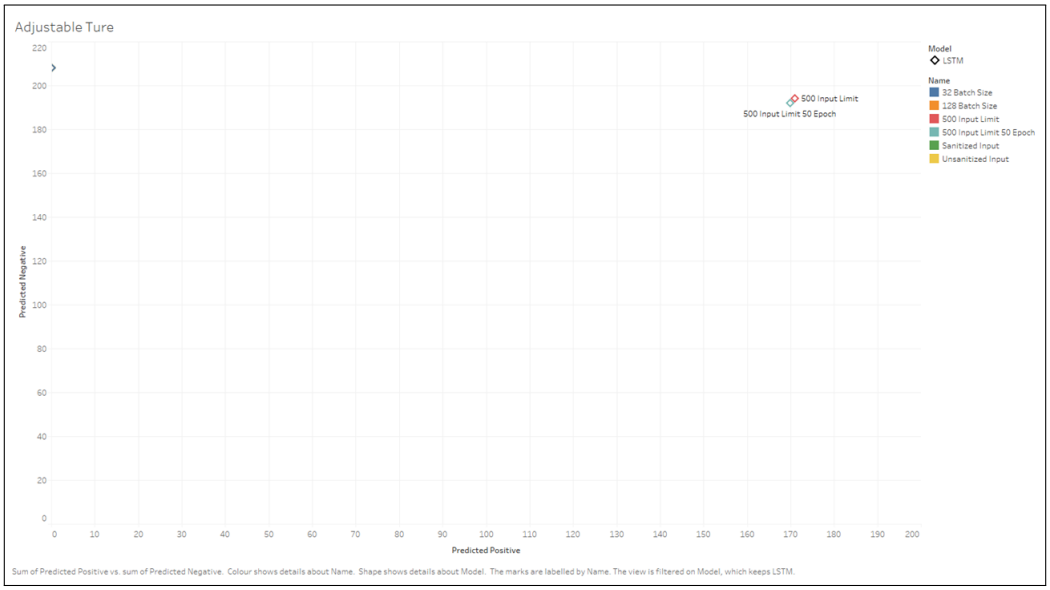

Figure 6.9: LSTM True

The above Figure 6.9 shows the number of correct classification for both the true and fake classes. There are two test models here. Both are limited 500 characters for the articles but blue is set to 50 epochs and red to the default of 20. The 20 epoch model has the highest number of correct classifications of 365, and the 50 has the lower with 262. Both of these results are close, so this could just be due to the difference in the random nature of these learning algorithms.

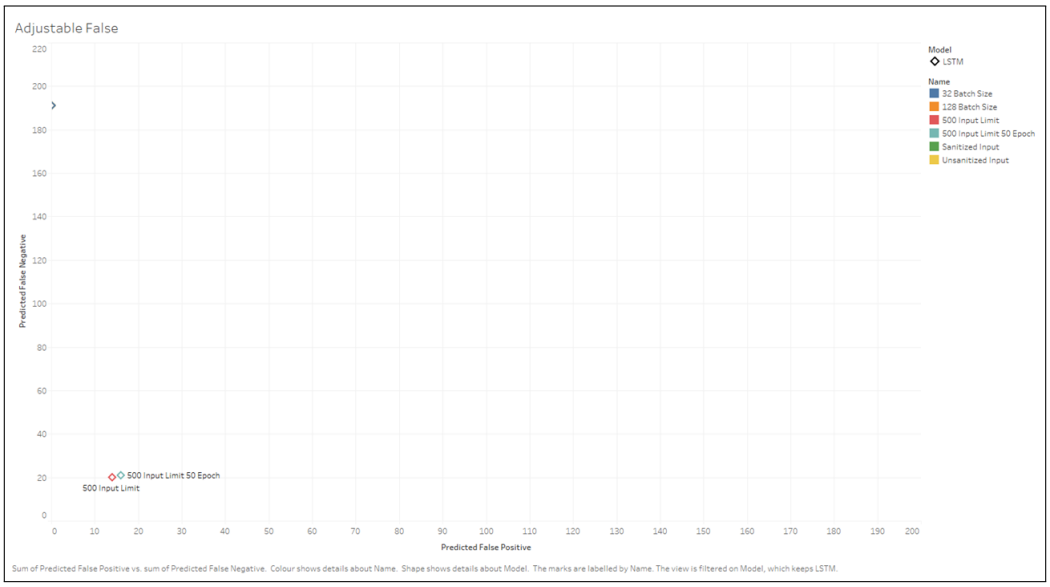

Figure 6.10: LSTM Fake

The above Figure 6.10 shows the number of incorrect classification for both the true and fake classes. There are the same models here as where in Figure 6.9. Here the 20 epoch has the lowest number of incorrect classifications of 35, and the 50 epoch closely follows with 37. Again they are very close, but as the 20 epoch did better overall it will be the model used later on.

6.2.2 Bidirectional LSTM

Unlike the regular LSTM, this Bidirectional of the LSTM does not suffer from the same problem and can learn on data with more that 500 characters. In the case without a 500 limit, the limit is set to the length of the longest article. With that in mind the best test accuracy achieved by this model was 99.97%, and best validation accuracy of 96.66%. These accuracies are shown with 2 decimal places as when compared to the Bidirectional GRU, their accuracies are very close needing the 2 decimal places to pick the better model. This model also improves upon the LSTM which best validation accuracy was $\approx$ 92% making it the best so far.

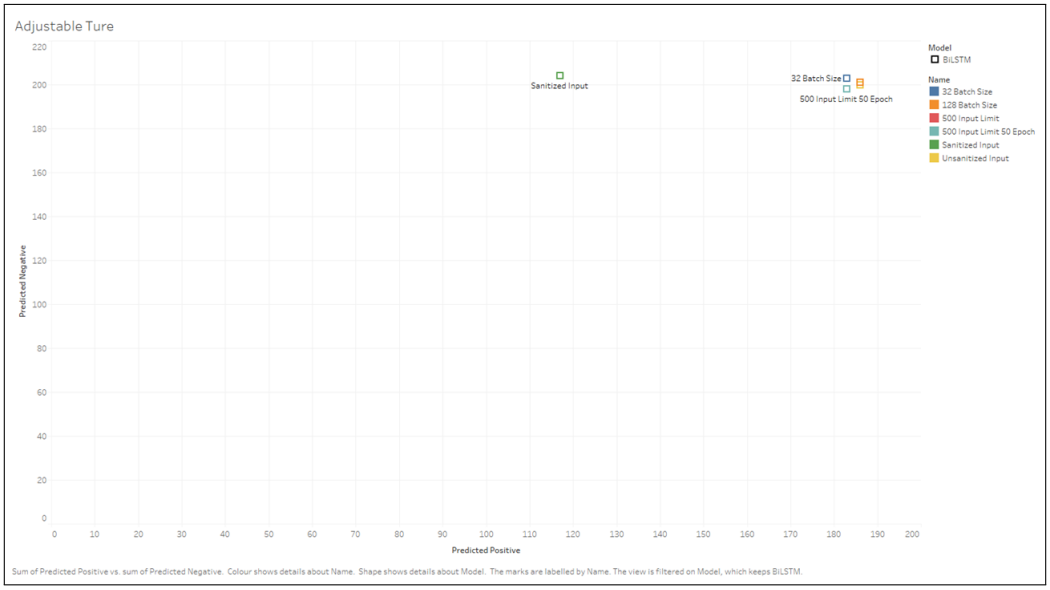

Figure 6.11: Bidirectional LSTM True

The above Figure 6.11 shows the number of correct classification for both the true and fake classes. There are six test models here. The green is the sanitized input, dark blue is the 32 batch size, light blue is the 500 limited with 50 epochs, orange the 128 batch, yellow the unsanitized input, and red the 500 limited with 20 epochs which is covered by the 128 batch as they have the same values. The model with the highest number of correct classifications is the 128 batch and the 500 limited with 20 epochs, both having 387 correct classifications, closely followed by the unsanitized input model. The model with the lowest number of correct classifications was the sanitized with only 321 correct classifications.

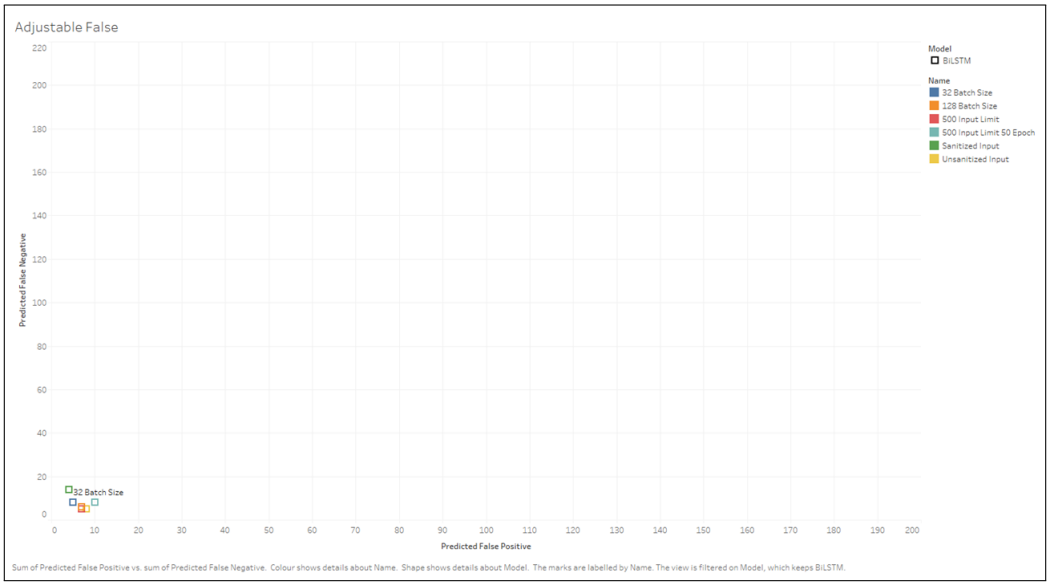

Figure 6.12: Bidirectional LSTM Fake

The above Figure 6.12 shows the number of incorrect classification for both the true and fake classes. There are the same models here as where in Figure 6.11. The model with the lowest number of incorrect classifications is the 500 limited 20 epoch model with only 12 incorrect classifications, and the highest belonging to the sanitized with 18. The reason the 500 limited 20 epochs did better than the others here is most likely due to the fact that when the data was split into validation and training, there is a slight variance between models. With that in mind, there still has to be one declared the best, and in this case it is the 500 limited 20 epochs.

6.2.3 GRU

The third model is the GRU, but as stated in the LSTM section this model only learns the problem when the input is limited to 500. Therefore, any models that are not limited to 500 make a grouping in one of the corners. With that said the best training accuracy achieved was $\approx$ 97% along with a best validation accuracy of $\approx$ 87%. This gives it a similar training accuracy, but not as good as a test accuracy as the LSTM.

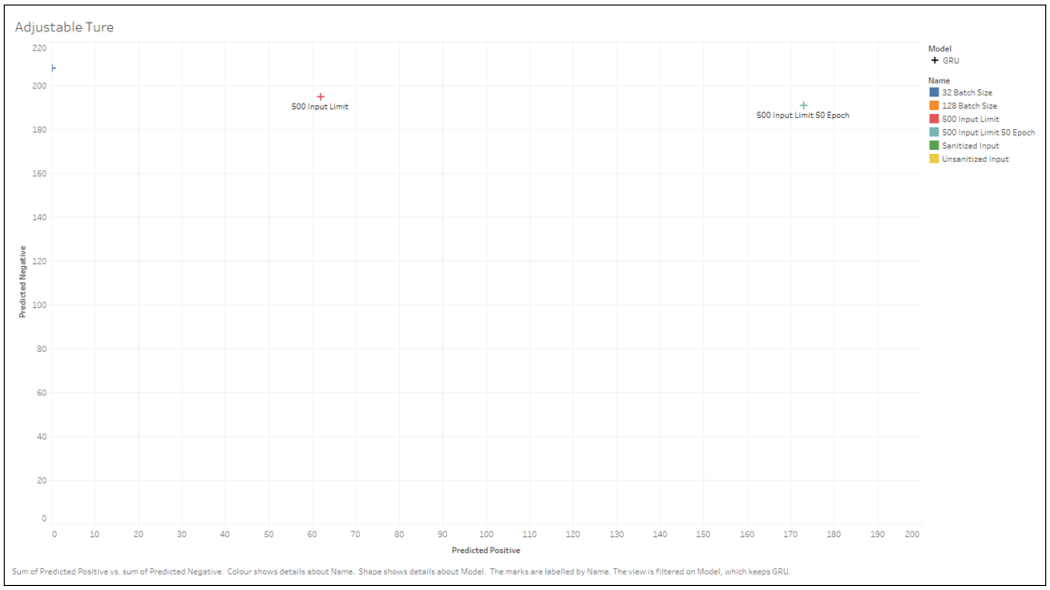

Figure 6.13: GRU True

The above Figure 6.13 shows the number of correct classification for both the true and fake classes. There are two test models here. Both are limited 500 characters for the articles but blue is set to 50 epochs and red to the default of 20. The model with the highest number of correct classifications is the 50 epoch model with 364, and the lower is the 20 epoch with only 257. The large gape between these models suggests that the extra learning time for this model was beneficial.

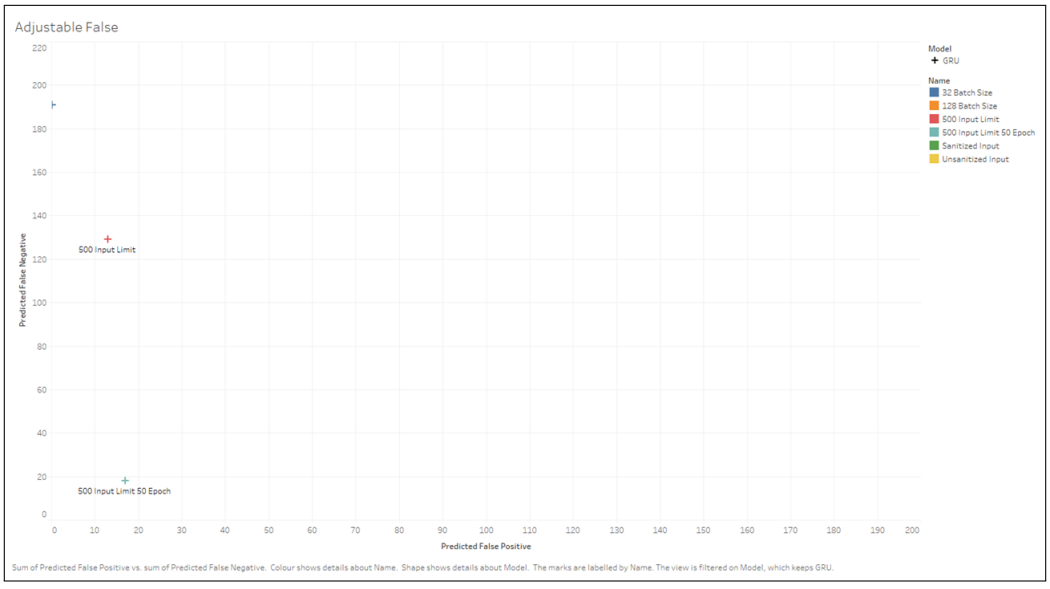

Figure 6.14: GRU Fake

The above Figure 6.14 shows the number of incorrect classification for both the true and fake classes. There are the same models here as where in Figure 6.13. The model with the lowest number of incorrect classifications was the 50 epochs with 35 incorrect classifications, with the 20 epochs having 139 incorrect classifications. With such a large difference between the two models, it is easy to say the the 50 epoch is the better and will be used in later

discussion.

6.2.4 Bidirectional GRU

Following the trend of the Bidirectional LSTM, this model does not need to have its data limited to 500, so it has much more available test data. With a best training accuracy of 99.93%, and a best validation accuracy of 96.94% this model has the best classification of the 4 methodologies.

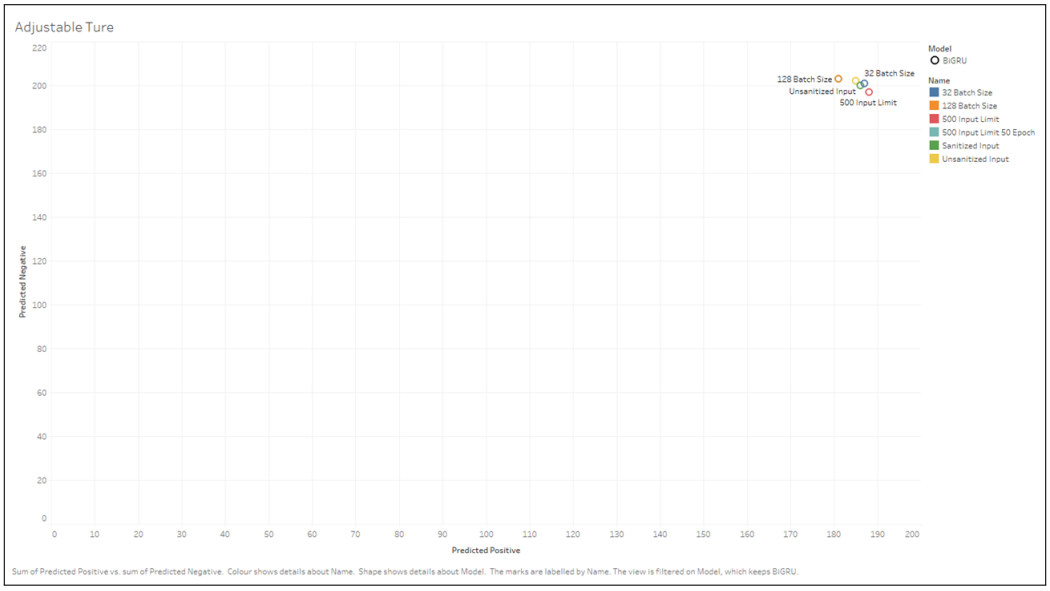

Figure 6.15: Bidirectional GRU True

The above Figure 6.15 shows the number of correct classification for both the true and fake classes. There are six test models here. The first is the 128 batch size which is orange which also covers the light blue of the 500 limited 50 epoch model because they have the same classification result for correct, yellow is the unsanitized, green the sanitized, dark blue the 32 batch size, and red the 500 limited 20 epoch model. The model with the highest number of correct classifications is the 32 batch model with 388 correct classifications, and the lowest is a tie with the 128 batch and the hidden 500 limited 50 epochs.

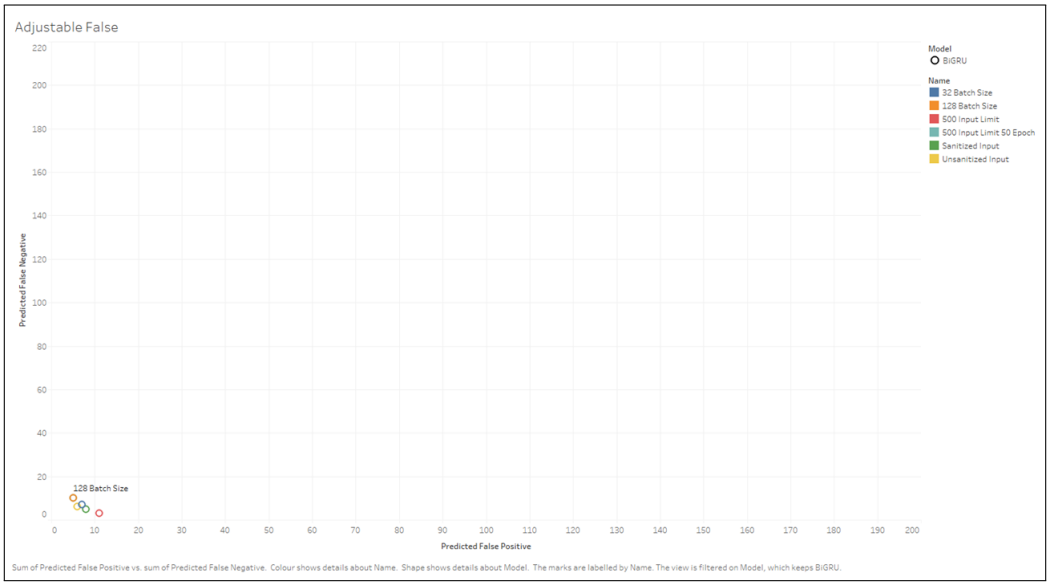

Figure 6.16: Bidirectional GRU Fake

The above Figure 6.16 shows the number of incorrect classification for both the true and fake classes. There are the same models here as where in Figure 6.15. The model with the lowest number of incorrect classifications is the unsanitized input with 12 incorrect classifications, with the highest going to another tie between the 128 batch and the covered 500 limited 50 epochs. As both of the models 32 batch, and unsanitized did the best between these two metrics, it falls to their validation accuracy where the 32 batch model wins with a $\approx 97\%$ accuracy against the $\approx 96\%$ of the unsanitized input model.

6.3 Comparison

This following stage will cover the above results, using the best versions of the models, comparing each model withing there groupings of Classical and Modern, and then final compare the best of the groupings against each other to make a conclusion on the better methodologies.

6.3.1 Classical

Firstly, the Classical models are to be covered. Here the best is easy to discern, as only one of the methodologies did well, which was the Random Forest Classifier. This model had a best test accuracy of $\approx 84\%$ a full 17% better than the next best model Logistic Regression. The other twos classifications were so bad they where akin to a coin toss. The reason most of these results do not compare to that of the literature review is due to that fact that in the past they were classified on the title of the article, and not its main text. This suggest that these models, excluding the Random Forest Classifier, are unable, or at least find it very hard to learn large problems. Since a number of these articles lengths are in the thousands of words, it can make it hard for these models to learn.

With the model of the Multinomial Naive Bayes Classifier, which was the worst classifier within the Classical methodologies, it is easy to understand why. This model works by calculating the probability of a word following another, and then puts it in a set. If your input text is of a large volume, then it is wise to assume that most word combinations will occur

and belong to both classes of true and fake. Due to this, this model gains an ambiguity to its learning, and only in rare cases can find word combinations that belong to one class. Due to these problems, only the Random Forest Classifier was able to learn the problem with any form of good classification, and thusly will be the model for later discussion.

6.3.2 Modern

Following the Classical conversation is the Modern. These models have done well, except in the cases of the LSTM and GRU where there input was not limit. Again this problem will be discussed later on. The best validation accuracy from each of these models are: LSTM 92.34%, GRU 86.63%, Bidirectional LSTM 96.66%, and Bidirectional GRU 96.94%. From this it can be seen that both of the Bidirectional LSTM and the Bidirectional GRU have the best Classification, with the Bidirectional GRU being the better by 2.8%. This variance is not significant enough to say which of the two are better, but for the case of a Combined discussion between the Classical and Modern results the Bidirectional GRU will be chosen as it has the higher test accuracy.

6.3.3 Combined

Finally comes the combined discussion between the Classical and Modern methodologies, where I am happy to say that the Modern methodologies win. The best of the Classical the Random Forest Classifier had a test accuracy of 84% against the best of the Moderns Bidirectional GRU with a validation accuracy of 96.94%. The reason the Classical methods use the test set is because they don't have a validation stage during learning, but there test set serves the same purpose here as the validation set for the Modern methodologies so comparing them is valid.

The regular GRU was the worst of the Modern methodologies but was able to get a validation accuracy of 86.63%, making it be able to compete with the Random Forest Classifier. If the GRU had the best accuracy of the Modern methodologies then they would not be and improvement, but due to the Bidirectional models, and with the Bidirectional GRU having a 12.94% advantage of the Random Forest Classifier I can confidently say that the chosen Modern methodologies out perform the Classical, and thus are an improvement.

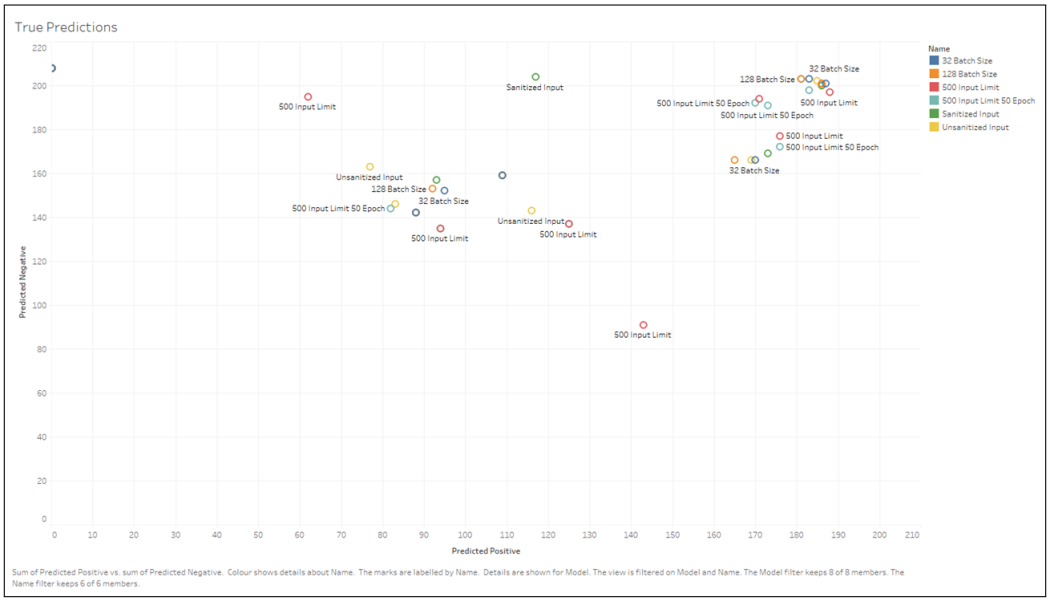

Figure 6.17: Complete Results True

From the above Figure 6.17 if can be seen that the Bidirectional models are the best. They have created a grouping in the top left, meaning they have the most correct classification. The best of the regular LSTM's and GRU's are not far behind these models, with similar classification to the best of the Classical the Random Forest Classifier. The majority of the Classical make a grouping around the center diagonal, with classification accuracy in the range of 50% - 70%. Finally, far off to the left are the non limited LSTM's and GRU's that failed to learn the problem.

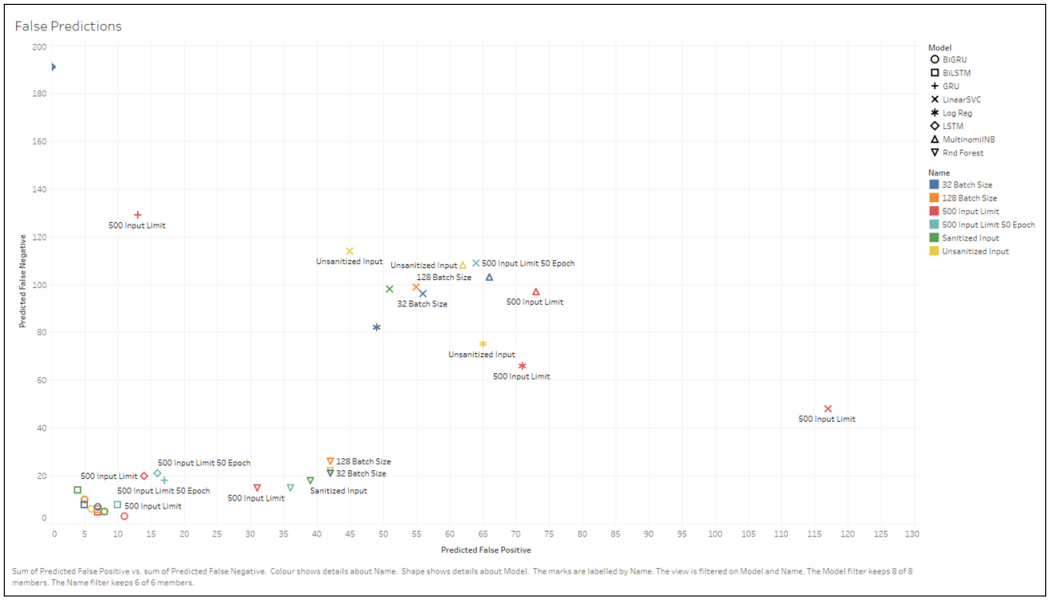

Figure 6.18: Complete Results Fake

From the above Figure 6.17 again the Bidirectional models are the best. They have created a grouping in the bottom right, meaning they have the least incorrect classification. Also the grouping of the Best LSTM's, GRU's, and the Random Forest Classifier can be

seen near the Bidirectional models, but are unable to reach there level of classification accuracy. The majority of the Classical again group in the center diagonal, showing there poor performance, and again the LSTM's and GRU's that failed to learn and be seen in the top left.

There will also be a brief discussion on their learn and prediction times. The Modern methodologies only real down side is there learn time. Some of the models took over 3 hours to learn, however this is not a huge problem. Once the models have been taught, there learn time is no longer of concern. If these models where to be taught on a more powerful machine, then there learn time would be significantly reduce, making there learn time a complete none issue. As for the prediction the Classical methodologies are faster with prediction time less that 0.1 of a second, but the Modern methodologies don't take much longer with the longest taking 0.65 of a second. With times less than a second to predict, it does not significantly tarnish the accuracy of the Modern methodologies, thusly still making them an huge improvement over the Classical methodologies.

6.3.4 Real World Data

For this section the 500 limited with 50 epoch models will be used as they are where the GRU gave its best Classification, as it has the lowest accuracy over all. The reason they are being used together is because they have the same tokenizer, and because it all of them are used they can create a voting system on the results of the of all of the models, hopefully improving the overall classification. The main testing will be done on the following article as it raises an interesting point [9]. This article covers a Russian smear campaign against the Oxford Coronavirus vaccine stating that it will turn anyone who takes it into monkeys. This is obviously not true, but the article states this. The reason this raises a problem is that it is Fake Fake news. News on Fake news. This could mean that the NLP methodologies, may state that this article is fake, even though it is truthful due to it covering a fake new subject.

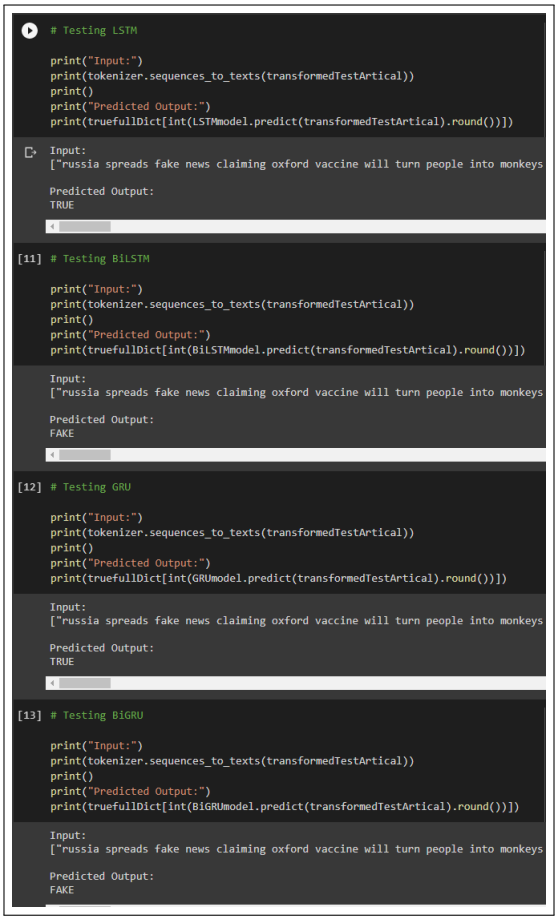

Figure 6.19: Fake News News Predictions

The above Figure 6.19 shows the prediction results from the Fake News News article. The LSTM and GRU both predict that the article is true, but the Bidirectional LSTM and Bidirectional GRU both predict Fake. In this case the regular LSTM and GRU are the correct models, while the Bidirectional LSTM and Bidirectional GRU are incorrect. This is interesting due to the fact that the regular LSTM and GRU, did not get as good classification accuracy as their Bidirectional variants, but where the correct one's here. This suggest that the data set either has no, or few Fake News News articles, and by they LSTM's and GRU's having poorer accuracy allows them to correctly classify Fake News News articles. Due to this any real world implementation, may benefit from a voting system from multiple models, that's combined results are used for any displayed prediction.

6.4 Summary

From these results it can be concluded that the Modern methodologies outperform the Classical methodologies, especially in the case of the Bidirectional variants of the LSTM's and GRU's. The regular LSTM and GRU are only able to learn the problem when there input length is set to 500, but when it does they do outperform the Classical methodologies. The best of the Classical methodologies the Random Forest Classifier has a similar accuracy to that of the GRU, but overall was outperformed by all Modern methods, unless the input was unlimited, in which case only the Bidirectional LSTM's and Bidirectional GRU's learnt the problem and still outperform the Random Forest Classifier.

Chapter 7

Discussion and Analysis

Overall, the Modern methodologies where able to beat the Classical methodologies, and the case of the bidirectional models by a large amount. Due to this it should be considered what made them so good, and why there was a problem with the unlimited length LSTM's and GRU's.

7.1 Modern Pros

The chosen Modern methodologies outclass the Classical methodologies by a large margin, and the Bidirectional even outclass their none Bidirectional variants. This is due to the fact that these models are designed to work with sequence prediction problems [3], which the task of sentiment analysis of news articles, which itself is a Many-to-One problem fits the definition of. Due to this it would of been odd if the chosen Modern methodologies did not beat the Classical methodologies, due to their specific design.

7.2 Modern Cons

Due to the Modern methodologies being Neural Networks, it means that they take a long time to train. In some cases over 3 hours. And because these models are not guarantied to learn the problem, or get the best classification the first time over, it takes a very long amount of time to teach these models. Due to this, it is not guarantied that the resulting models for the Modern methodologies are the best that they could be, as there was not enough time to try out the millions of variations that could of been tested.

There is also the problem of limiting the articles to 500 for the regular LSTM's and GRU's. This may result in them having lower potential accuracy than if they where able to use the whole article. If there was a more refined pre-processing that could remove useless words from the sentences such as the word "the", then the number of tokens could of been reduce, and the LSTM' and GRU's may of then been able to learn the problem.

7.3 Overfeeding Through Length

One of the big problems with the Modern methodologies was with the regular LSTM's and GRU's. These methods could not be given an input larger than 500 words long. This problem was due to overfeeding, as the models where forgetting things within the articles by the end of the sequence. If they were limited to an input size of 500, they could learn those problems

as they did not forget anything. Moreover, the reason the Bidirectional models where not effected by this was due to the fact that through their nature they have a backward pass on the data simultaneously as the forward pass, meaning that the model was seeing both ends, and thus did not forget anything.

Due to this 500 limit, it also suggests that an articles fakeness can be determined near the start of the article. This can be seen due to the fact that all of the models could still learn the problem when this limit was imposed. With that, a follow-up project could occur looking into the structure of these fake news articles, and how quickly the fakeness can be detected, and then possibly improve the classification.

7.4 Summary

To summarise, the Modern methods are able to beat the Classical, unless it is the LSTM' and GRU's when not limited to 500 characters for there input length. This is due to overfeeding these more simple models, resulting in them not learning the problem. Their learning time is of a prohibitive length with some learn time in excess of 3 hours.

Chapter 8

Conclusions and Future Work

From this project it has been shown that the chosen Modern methodologies have been able to outperform and outclass the Classical methodologies, with the best Modern having an accuracy of $\approx$ 97% and the best classical having $\approx$ 84%, giving an $\approx$ 13% improvement to the classification accuracy. With that, although there are some possible improved methods that could be available in the future, that of Googles BERT. Moreover, a new source of fake news could appear and need new predictors from GTP-3.

8.1 Future Work

The following section covers the possible usage of two transformers: BERT and GTP-3. BERT could be used as a future method of fake news detection, while GTP-3 is a way of creating fake news automatically. I will discuss what they are and their potential in the following subsections.

8.1.1 BERT

BERT is a Transformers Created by Google. Its purpose is to speed up the learning time of problems by having a large neural network that already has many problems learnt. With it, you could supposedly get very good classification of fake news, however there is a problem. BERT is a very large program, and as such the default hardware given through the Google Colab service is not powerful enough to run it. A possible way around it would be to reduce each input from an article by splitting each paragraph and from this getting a classification of each paragraph combined to make the total.

8.1.2 GTP3 Generated Fake News

GTP3 is another Transformers but is created by OpenAI. It is capable of producing human like text, which in the future could lead to an advancement in the creation of fake news. Here is an example article [10]. The articles text is all generated by GTP-3, but it is not all from one output. The article was generated by multiple quires where the AI was asked to produce an article on itself, and then combined into a more coherent article by an editor. This technology could be used to make the creation of fake news easier, and in the future could produce fake news, or real news by itself. If this is the case then some testing into the methodologies within this document should be tested against it.

Chapter 9

Reflection

Throughout this project there have been problems along the way. One of the biggest was cause by the non bidirectional LSTM's and GRU's having an overfeeding problem. Original, when I started this project I was only planning to use these two methods as the modern methodologies. I did not immediately find the overfeeding problem until later due to the fact that when starting I was limiting the reports to 500 characters, to reduce the learn time. During the spring term I was starting to show my progress of my work to my supervisor, which meant getting some properly trained models, but when I removed the limit the overfeeding problem became apparent. I was worried at first, as I though that my whole project was going to fail, but after some research I found out about their bidirectional variants which supposedly had better data retention and did not sufferer as badly as the non bidirectional models. With there implementation I found that they where able to complete the classification task, and had better classification than the regular models. With this I learned to improve my research skills in finding the best version, which has help me throughout other coursework.

References

[1] Ali Alwahedi Monther Aldwairi. Detecting Fake News in Social Media Networks. https://www.sciencedirect.com/science/article/pii/S1877050918318210, 2020.

[2] Vaibhav Kumar. Hands-On Guide to Predict Fake News Using Logistic Regression, SVM and Naive Bayes Methods. https://analyticsindiamag.com/hands-on-guide-to-predict-fake-news-using-logistic-regression-svm-and-naive-bayes-me 2020.

[3] Jason Brownlee. When to Use MLP, CNN, and RNN Neural Networks. https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/, 2020.

[4] KyungHyun Cho Yoshua Bengio Junyoung Chung, Caglar Gulcehre. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. https://arxiv.org/pdf/1412.3555v1.pdf, 2020.

[5] Jruvika. Fake News Detection (Data Set). https://www.kaggle.com/jruvika/fake-news-detection, 2020.

[6] Michael Phi. Illustrated Guide to LSTM's and GRU's: A step by step explanation. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21, 2020.

[7] Frank Gaillard Andrew Murphy. Iteration (machine learning). https://radiopaedia.org/articles/iteration-machine-learning?lang=gb, 2020.

[8] Jason Brownlee. How to Calculate Precision, Recall, and F-Measure for Imbalanced Classification. https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/#:~:text=Precision%20quantifies%20the%20number%20of,and%20recall%20in%20one%20number., 2020.

[9] Helena Kelly. Russia spreads fake news claiming Oxford coronavirus vaccine will turn people into MONKEYS - and portrays Boris Johnson as Bigfoot. https://www.dailymail.co.uk/news/article-8845937/Russia-spreads-fake-news-claiming-Oxford-coronavirus-vaccine-turn-people-MONKEYS.html, 2020.

[10] GTP-3. A robot wrote this entire article. Are you scared yet, human? GPT-3. https://www.theguardian.com/commentisfree/2020/sep/08/robot-wrote-this-article-gpt-3, 2020.

Appendix A

Fake News Training

A.1 Data Pre-Processing

```
# Dataset creation

df = pd.read_excel("/content/sample_data/News-Articals.xlsx", nrows=30000)

text = df.text.values.astype('U')

for i in range(0, text.size):
  text[i] = text[i].replace("\n"," \n ")
  text[i] = text[i].replace("."," .")
  text[i] = text[i].replace("!"," !")
  text[i] = text[i].replace("?"," ?")
  text[i] = text[i].replace(","," ,")
  text[i] = text[i].replace("\" "," \" ")
  text[i] = text[i].replace(" \""," \" ")
  text[i] = text[i].replace("\' "," \' ")
  text[i] = text[i].replace(" \'"," \' ")

# Example "This is a 'test' sentence." -> "This is a ' test ' sentence ."
df.text = text

tokenizer = Tokenizer(num_words=200000,filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n',lower=True,split=" ")
tokenizer.fit_on_texts(df.text.values.astype('U'))
x = tokenizer.texts_to_sequences(df.text.values.astype('U'))
x = pad_sequences(x, padding='post', maxlen=500, truncating = 'post')
#x = pad_sequences(x, padding='post', truncating = 'post')

Y = df.polarity
vocab_size = len(tokenizer.word_index) + 1

# Create Train and Test Sets:
X_train, X_test, Y_train, Y_test = train_test_split(x, Y, test_size=0.1, random_state = 24)

# saving
with open('tokenizer.pickle', 'wb') as handle:
    pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

A.2 Classical Example

```
%%time
# Create and Train a Logistic Regression

logreg = LogisticRegression(C=1e9, solver='lbfgs', max_iter=1000)
logreg.fit(X_train, Y_train)
```

```
[ ] # Logistic Regression Test Accuracy

prediction = logreg.predict(X_test).round()

print("Accuracy:", metrics.accuracy_score(Y_test, prediction))

lables = ['Fake','True']
print(metrics.classification_report(Y_test, prediction, target_names=lables))
```

```
[ ] # Logistic Regression Test Confusion Matrix

class_names = ['Fake', 'True']
Y_pred = logreg.predict(X_test)
cm = confusion_matrix(Y_test, Y_pred.round())
cmd = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=class_names)
disp = cmd.plot(cmap=plt.cm.Blues, values_format='')
plt.show()
```

```
%%time
# Testing Logistic Regression

size = (int)(X_test.size/(X_test[:1].size+1))
rnd = random.randrange(1, size)
transformedTestArtical = X_test[rnd-1:rnd]

print("Input:")
print(tokenizer.sequences_to_texts(transformedTestArtical))
print()
print("Expected Output:")
print(Y_test.values[rnd-1:rnd])
print()
print("Actual Output:")
print(logreg.predict(transformedTestArtical))
```

A.3 Modern Example

```
#@title
%%time
# Create and Train a Long Short Term Memory Neural Network (LSTM)

#Return Sequence

BiLSTMmodel = Sequential()
BiLSTMmodel.add(Embedding(vocab_size, 25))
BiLSTMmodel.add(Bidirectional(LSTM(32)))
BiLSTMmodel.add(Dropout(0.5))
BiLSTMmodel.add(Dense(1,activation='sigmoid'))
BiLSTMmodel.compile(optimizer='adam',
  loss='binary_crossentropy',
  metrics=['accuracy'])
BiLSTMmodel.summary()

from keras.callbacks import EarlyStopping
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, restore_best_weights = True)

history=BiLSTMmodel.fit(X_train, Y_train, batch_size=64, epochs=20, validation_split=0.2, callbacks=[es])

#We save this model so that we can use in own web app
BiLSTMmodel.save('BiLSTM.h5')
```

```
[ ]  #BiLSTMmodel = tf.keras.models.load_model('BiLSTM.h5')
```

```
[ ]  # BiLSTMmodel Test Accuracy

     prediction = BiLSTMmodel.predict(X_test).round()

     print("Accuracy:", metrics.accuracy_score(Y_test, prediction))

     lables = ['Fake','True']
     print(metrics.classification_report(Y_test, prediction, target_names=lables))
```

```
[ ]  # BiLSTMmodel Test Confusion Matrix

     class_names = ['Fake', 'True']
     Y_pred = BiLSTMmodel.predict(X_test)
     cm = confusion_matrix(Y_test, Y_pred.round())
     cmd = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=class_names)
     disp = cmd.plot(cmap=plt.cm.Blues, values_format='')
     plt.show()
```

```
[ ]  # Testing BiLSTMmodel

     size = (int)(X_test.size/(X_test[:1].size+1))
     rnd = random.randrange(1, size)
     transformedTestArtical = X_test[rnd-1:rnd]

     print("Input:")
     print(tokenizer.sequences_to_texts(transformedTestArtical))
     print()
     print("Expected Output:")
     print(Y_test.values[rnd-1:rnd])
     print()
     print("Actual Output:")
     print(BiLSTMmodel.predict(transformedTestArtical).round())
```

Appendix B

Real World Testing

B.1 Article Scrapping

```
[ ]  URL = "https://www.dailymail.co.uk/news/article-8845937/Russia-spreads-fake-news-claiming-Oxford-coronavirus-vaccine-turn-people-MONKEYS.html"
     web_page = bs4.BeautifulSoup(requests.get(URL, {}).text, "lxml")

[ ]  criteria = ["p","h2","figcaption","a href"]

     sub_web_page = web_page.body.find_all(name=[n for n in criteria])

[ ]  newInput = ''
     for n in sub_web_page:
       newInput += str(n.getText()) + '\n'

     newInput = newInput.replace("\n"," \n ")
     newInput = newInput.replace("."," .")
     newInput = newInput.replace("!"," !")
     newInput = newInput.replace("?"," ?")
     newInput = newInput.replace(","," ,")
     newInput = newInput.replace("\" "," \" ")
     newInput = newInput.replace(" \""," \" ")
     newInput = newInput.replace("\' "," \' ")
     newInput = newInput.replace(" \'"," \' ")

     print(newInput)
```

B.2 Loading Tokenizer

```
[7]  tokenizer = Tokenizer(num_words=200000,filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n',lower=True,split=" ")
     with open('/content/tokenizer.pickle', 'rb') as pickle_file:
         tokenizer = pickle.load(pickle_file)

     transformedTestArtical = tokenizer.texts_to_sequences([newInput])
     #transformedTestArtical = pad_sequences(transformedTestArtical, padding='post', truncating = 'post')
     transformedTestArtical = pad_sequences(transformedTestArtical, padding='post', maxlen=500, truncating = 'post')
     truefullDict = ["FAKE", "TRUE"]
```

B.3 Models

```
[9] LSTMmodel = tf.keras.models.load_model('/content/LSTM.h5')
    GRUmodel = tf.keras.models.load_model('/content/GRU.h5')
    BiLSTMmodel = tf.keras.models.load_model('/content/BiLSTM.h5')
    BiGRUmodel = tf.keras.models.load_model('/content/BiGRU.h5')

    # Testing LSTM

    print("Input:")
    print(tokenizer.sequences_to_texts(transformedTestArtical))
    print()
    print("Predicted Output:")
    print(truefullDict[int(LSTMmodel.predict(transformedTestArtical).round())])

    Input:
    ["russia spreads fake news claiming oxford vaccine will turn people into monkeys and portrays boris johnson as bigfoot by helena kelly for the daily mail published 01 41 bst 16 october 2020 updat

    Predicted Output:
    TRUE
```