CONSTRUCTOR UNIVERSITY

Data Mining

Final Project

Fake News Detection

Under the Supervision
of
Adalbert Wilhelm

Made By

Sejal Anish Maniyar

TABLE OF CONTENTS

Executive Summary

1. Introduction

2. Problem Statement

3. Objective

4. Literature Survey

5. System Architecture

6. Text Collection

7. Importing the Libraries

8. Preprocessing and Cleaning

9. Text Preprocessing

10. Exploratory Data Analysis

11. WordCloud

12. Data Visualization

13. Feature Extraction

14. Splitting of the Data

15. Training the Model

16. Results

17. Conclusion

18. References

Executive Summary

The issue of authenticity of information has long been a problem affecting businesses and society in both print and digital media. With social networking websites, the reach and effects of information spread occur at a significantly faster pace and can cause distorted, inaccurate, or false information to have a tremendous potential to cause real-world impacts within minutes for millions of users. In response to public concerns about this problem, approaches have been developed to mitigate the issue.

To develop a machine learning model that can classify news as fake or real, several tasks must be performed in the following order: data collection and analysis, preprocessing the data, text feature extraction, using different classification algorithms, selecting the best classification algorithm and feature extraction method, classifying the news as fake or real, and deploying the model.

The dataset used for this project was obtained from Kaggle and consists of two CSV files containing nearly 23,481 fake news and 21,417 true news articles. The files include columns for the title, text, subject, and date the news was published. To prepare the data for analysis, several pre-processing steps must be performed, including creating the target column for both fake and true news, concatenating the title and text columns of news, converting the date columns to datetime format, and appending the two datasets to create a single file.

Once the data is prepared, text pre-processing is performed to convert all letters to lower case, remove numbers, punctuations, accent marks, white spaces, and stop words. Stop words are commonly occurring words in a language that don't carry much meaning on their own. Removing stop words can help reduce the dimensionality of text data and improve the performance of machine learning models that rely on textual data. Feature extraction is performed using the CountVectorizer class from the scikit-learn library, which converts the text into a bag-of-words representation, counting the frequency of each word, and then computes the frequency of each n-gram to return the top n most frequent n-grams. The parameter g specifies the size of the n-gram.

Overall, this project aims to develop a machine learning model that can accurately classify news articles as fake or real, helping to mitigate the spread of distorted, inaccurate, or false information. The approach involves data collection and analysis, preprocessing, text feature extraction, and the use of various classification algorithms to select the best approach for classifying news articles. The project also utilizes various data visualization techniques to help understand the text data and select the most relevant features for the classification model.

1. Introduction

Social media facilitates the creation and sharing of information that uses computer-mediated technologies. This media changed the way groups of people interact and communicate. It allows low cost, simple access and fast dissemination of information to them. Majority of people search and consume news from social media rather than traditional news organizations these days. At one side, where social media have become a powerful source of information and bringing people together, on the other side it also put a negative impact on society. Look at some examples herewith;

Facebook Inc's popular messaging service, WhatsApp became a political battle-platform in Brazil's election [1]. False rumors, manipulated photos, de-contextualized videos, and audio jokes were used for campaigning. These kinds of stuff went viral on the digital platform without monitoring their origin or reach.

A nationwide block on major social media and messaging sites including Facebook and Instagram was done in Sri Lanka after multiple terrorist attacks in the year 2019 [2]. The government claimed that "false news reports" were circulating online. This evident the challenges the world's most powerful tech companies face in reducing the spread of misinformation.

Such examples show that Social Media enable the widespread of "fake news" as well. The news disseminated on social media platforms may be of low quality carrying misleading information intentionally. This sacrifices the credibility of the information. Millions of news articles are being circulated every day on the Internet – how one can trust which is real and which is fake? Thus incredible or fake news is one of the biggest challenges in our digitally connected world. Fake news detection on social media has recently become an emerging research domain [3]. The domain focuses on dealing with the sensitive issue of preventing spread of fake news on social media.

Fake news identification on social media faces several challenges. Firstly, it is difficult to collect fake news data. Furthermore, it is difficult to label fake news manually. Since they are intentionally written to mislead readers, it is difficult to detect them simply on the basis of news content [4]. Furthermore, Facebook, Whatsapp, and Twitter are closed messaging apps. The misinformation disseminated by trusted news outlets or their friends and family is therefore difficult to be considered as fake. It is not easy to verify the credibility of newly emerging and time-bound news as they are not sufficient to train the application dataset [4]. Significant approaches to differentiate credible users, extract useful news features and develop authentic information dissemination system are some useful domains of research and need further investigations.

2. Problem Statement

The authenticity of Information has become a longstanding issue affecting businesses and society, both for printed and digital media. On social networks, the reach and effects of information spread occur at such a fast pace and so amplified that distorted, inaccurate, or false information acquires a tremendous potential to cause real-world impacts, within minutes, for millions of users. Recently, several public concerns about this problem and some approaches to mitigate the problem were expressed.

The sensationalism of not-so-accurate eye-catching and intriguing headlines aimed at retaining the attention of audiences to sell information has persisted all throughout the history of all kinds of information broadcast. On social networking websites, the reach and effects of information spread are however significantly amplified and occur at such a fast pace, that distorted, inaccurate, or false information acquires a tremendous potential to cause real impacts, within minutes, for millions of users.

3. Objective

To achieve our goal of developing machine learning model to classify news as fake or real, we need perform following tasks in the same order as stated.

-Data Collection and Analysis
-Preprocessing the data
-Text feature extraction
-Using different classification algorithms
-Taking the best classification algorithm and feature extraction method
-Classifying the news as fake or real.
-Deploying the model

4. Literature Survey

In Today's world, anybody can post the content over the internet. Unfortunately, counterfeit news gathers a lot of consideration over the web, particularly via web-based networking media. Individuals get misdirected and don't reconsider before flowing such miseducational pieces to the most distant part of the arrangement. Such type of activities are not good for the society where some rumors or vague news evaporates the negative thought among the people or specific category of people[1]. As fast the technology is moving, on the same pace the preventive measures are required to deal with such activities. Broad communications assuming a gigantic job in impacting the general public and as it is normal, a few people attempt to exploit it. There are numerous sites which give false data. They deliberately attempt to bring out purposeful publicity, deceptions and falsehood under the pretense of being true news. Their basic role is to control the data that can cause open to have confidence in it. There are loads of case of such sites everywhere throughout the world .Therefore, counterfeit news influences the brains of the individuals. As indicated by study Scientist accept that numerous man-made brainpower calculations can help in uncovering the bogus news.

Fake news detection is made to stop the rumors that are being spread through the various platforms whether it be social media or messaging platforms, this is done to stop spreading fake news which leads to activities like mob lynching, this has been a great reason motivating us to work on this project. We have been continuously seeing various news of mob lynching that leads to the murder of an individual; fake news detection works on the objective of detecting this fake news and stopping activities like this thereby protecting the society from these unwanted acts of violence.

The digital news industry in the United States is facing a complex future. On one hand, a steadily growing portion of Americans are getting news through the internet, many U.S. adults get news on social media, and employment at digital-native outlets has increased. On the other, digital news has not been immune to issues affecting the broader media environment, including layoffs, made-up news and public distrust.

Half of Americans (52%) say they have changed the way they use social media because of the issue of made-up news. Furthermore, among the Americans who ever get news through social media, half have stopped following a news source because they thought it was posting made-up news and information. At the same time, about a third (31%) of social media news consumers say they at least sometimes click on news stories they think are made up. So there is need to stop the fake news spreading.

5. System Architecture

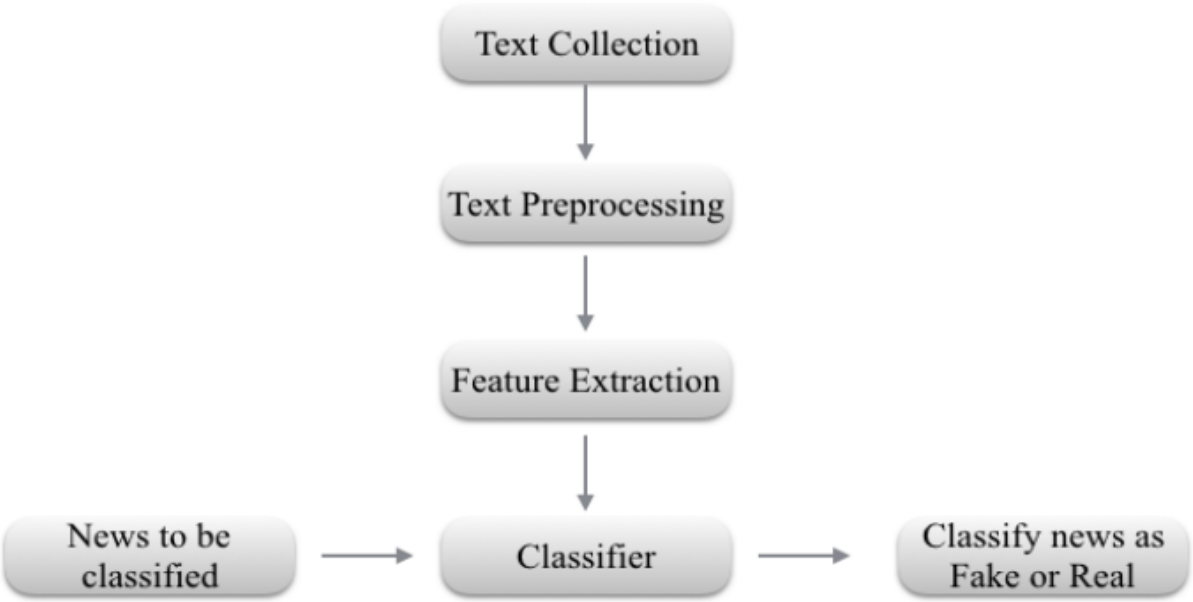

Figure 1. System Architecture

6. Text Collection

The dataset was taken from Kaggle.

Link:-https://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset

This metadata has 2 CSV files where one dataset contains fake news and the other contains true/real news and has nearly 23481 fake news and 21417 true news.

Description of columns in the file:

title- contains news headlines
text- contains news content/article
subject- the type of news
date- the date the news was published

7. Importing the Libraries

- **numpy**: A library for numerical computing with support for large, multi-dimensional arrays and matrices.

- **pandas**: A library for data manipulation and analysis, providing easy-to-use data structures and data analysis tools.

- **seaborn**: A visualization library based on matplotlib that provides a high-level interface for creating informative and attractive statistical graphics.

- **matplotlib.pyplot:** A plotting library for creating static, animated, and interactive visualizations in Python.

- **nltk:** A natural language processing library that provides tools and resources for processing and analyzing human language data.

- **sklearn:** A machine learning library that provides tools for classification, regression, clustering, and other machine learning tasks.

- **wordcloud:** A library for creating word clouds, which are visual representations of text data where the size of each word corresponds to its frequency.

- **bs4**: A library for web scraping that provides tools for parsing HTML and XML documents.

- **keras:** A high-level neural network library that provides a simple interface for building and training neural networks.

- **tensorflow:** An open source machine learning framework for building and training neural networks.

8. Preprocessing and Cleaning

We have to perform certain pre-processing steps before performing EDA and giving the data to the model.

- Creating the target column: Let's create the target column for both fake and true news. Here we are gonna denote the target value as '0' in case of fake news and '1' in case of true news
- Concatenating title and text of news: News has to be classified based on the tile and text jointly. Treating the title and content of news separately doesn't reap any benefit. So, let's concatenate both the columns in both datasets
- Converting the date columns to datetime format: We had links and news headlines inside the date column which can give us trouble when converting to datetime format. So let's remove those records from the column.
- Appending two datasets: When we are providing a dataset for the model, we have to provide it as a single file. So it's better to append both true and fake news data and preprocess it further and perform EDA

9. Text Preprocessing

1. After a text is obtained, we start with text preprocessing. Text preprocessing includes:

- Converting all letters to lower case
- Removing numbers
- Removing punctuations, accent marks
- Removing white spaces
- Removing stop words

2. Stopwords are commonly occurring words in a language that are often excluded from the processing of natural language text because they don't carry much meaning on their own. Examples of stopwords in English are "the", "a", "an", "and", "in", "of", etc. Removing stopwords can help reduce the dimensionality of text data and improve the performance of machine learning models that rely on textual data.

DATA CLEANING

In [19]:

```
def strip_html(text):
    soup = BeautifulSoup(text, "html.parser")
    return soup.get_text()

#Removing the square brackets
def remove_between_square_brackets(text):
    return re.sub('\[[^]]*\]', '', text)
# Removing URL's
def remove_between_square_brackets(text):
    return re.sub(r'http\S+', '', text)
#Removing the stopwords from text
def remove_stopwords(text):
    final_text = []
    for i in text.split():
        if i.strip().lower() not in stop:
            final_text.append(i.strip())
    return " ".join(final_text)
#Removing the noisy text
def denoise_text(text):
    text = strip_html(text)
    text = remove_between_square_brackets(text)
    text = remove_stopwords(text)
    return text
#Apply function on review column
df['text']=df['text'].apply(denoise_text)
```

Fig 2. Data Cleaning Code Snippet

This code defines a series of functions to pre-process textual data, and then applies them to a pandas DataFrame df on the column named text.

The strip_html() function removes HTML tags from the input text using BeautifulSoup library.

The remove_between_square_brackets() function removes any text that is between square brackets (e.g. [some text]), using a regular expression.

The remove_stopwords() function removes commonly occurring words, called stopwords, from the input text. The list of stopwords is defined somewhere else in the code as stop.

The denoise_text() function applies the above three functions in sequence to pre-process the input text.

Finally, the apply() function is used to apply the denoise_text() function to each row of the text column in df.

10. Exploratory Data Analysis

It assigns a binary category to each of the dataframes 'true' and 'false' (presumably indicating whether a news article is true or false), with 1 assigned to the 'true' dataframe and 0 assigned to the 'false' dataframe.

Then, the seaborn library is used to create a countplot of the 'category' column in the original dataframe 'df', which presumably combines both the 'true' and 'false' dataframes. The countplot displays the frequency count of each category (0 or 1) in the form of a bar graph.

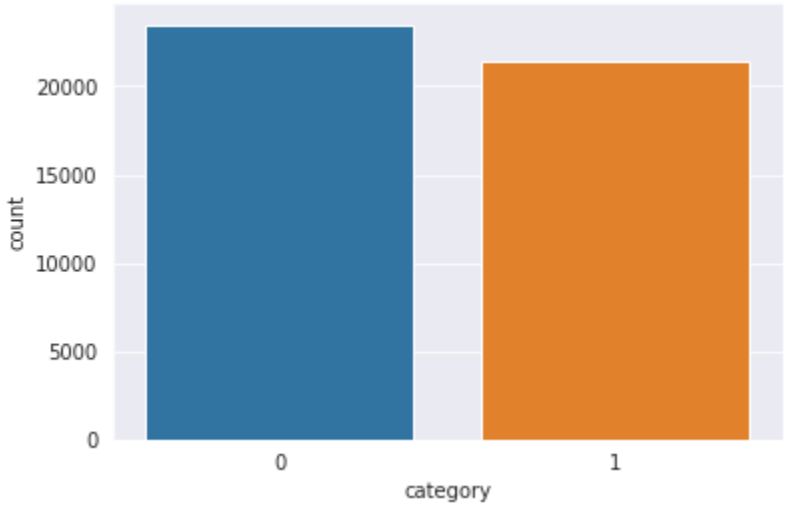

Fig 3. Histogram plot by Category

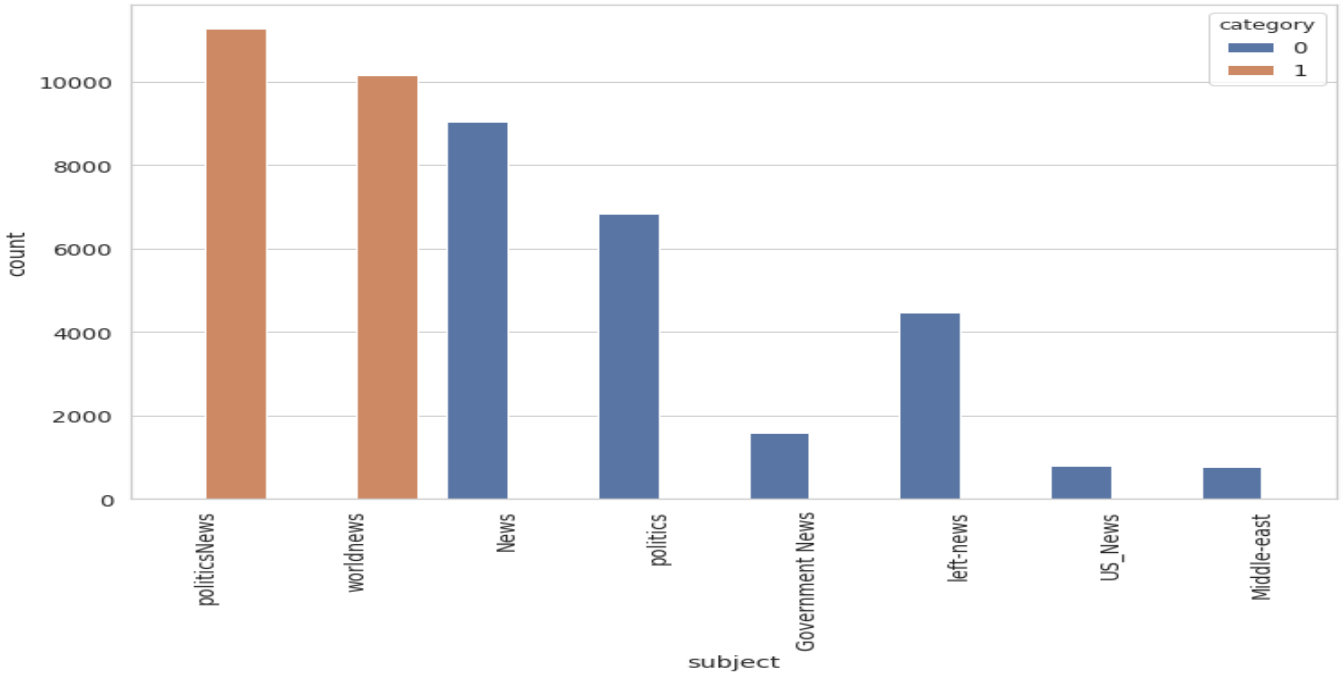

Fig 4. Countplot of Subject by category

It calculates the frequency count of unique values in the 'subject' column of the dataframe 'df' using the value_counts() function.

The seaborn library is then used to create a countplot of the 'subject' column in the 'df' dataframe, with the count of each unique value split by the 'category' column (presumably indicating true or false news articles). The resulting plot shows the frequency of each subject for both true and false news articles, with different colors representing the different categories.

11. WordCloud

A word cloud is a visual representation of text data that uses word frequency to depict the importance of different words in a dataset. It is a popular tool in natural language processing and machine learning for visualizing the most frequently occurring words in a given corpus of text. Word clouds are often used to quickly identify the most important topics or themes in a text, and can be useful in various applications, such as sentiment analysis, topic modelling, and data visualization. In machine learning, word clouds can be generated using algorithms that analyze the frequency of words in a text corpus and then display the results in a graphical format, with more frequently occurring words being represented with larger font sizes or more prominent positions in the visualization.

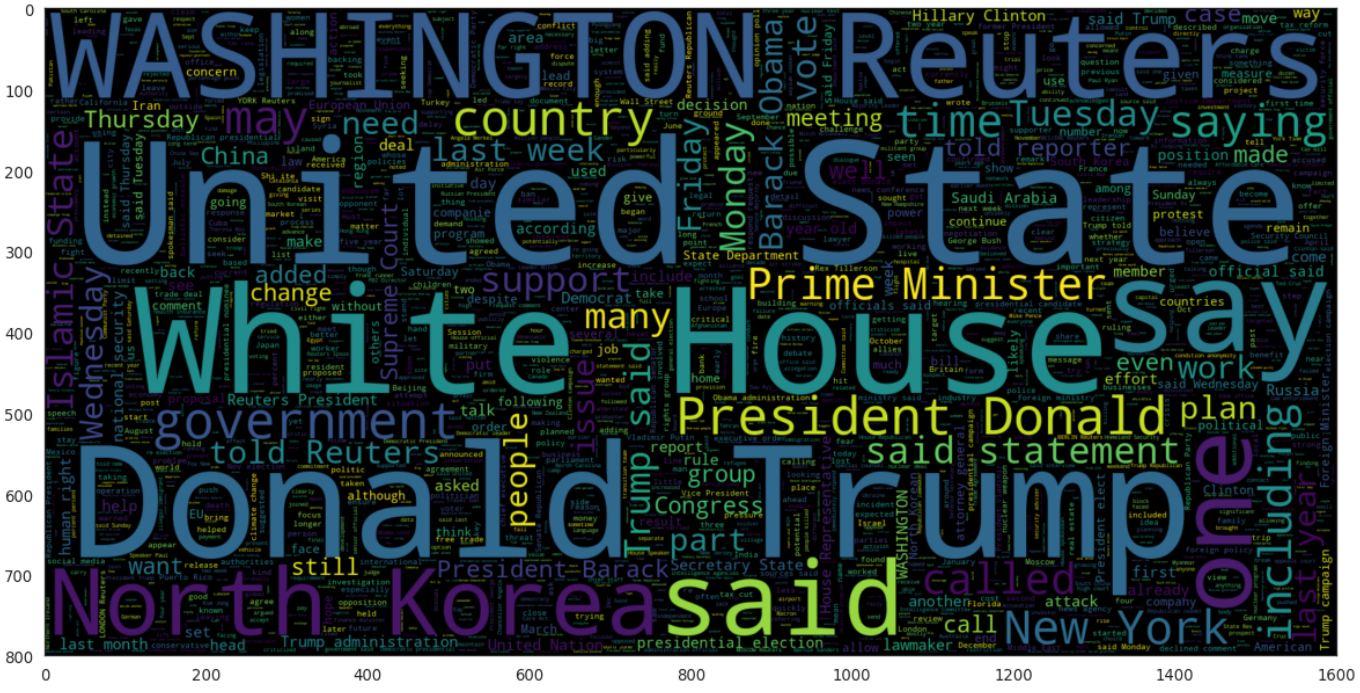

Fig 5. WORDCLOUD FOR REAL TEXT (LABEL - 1)

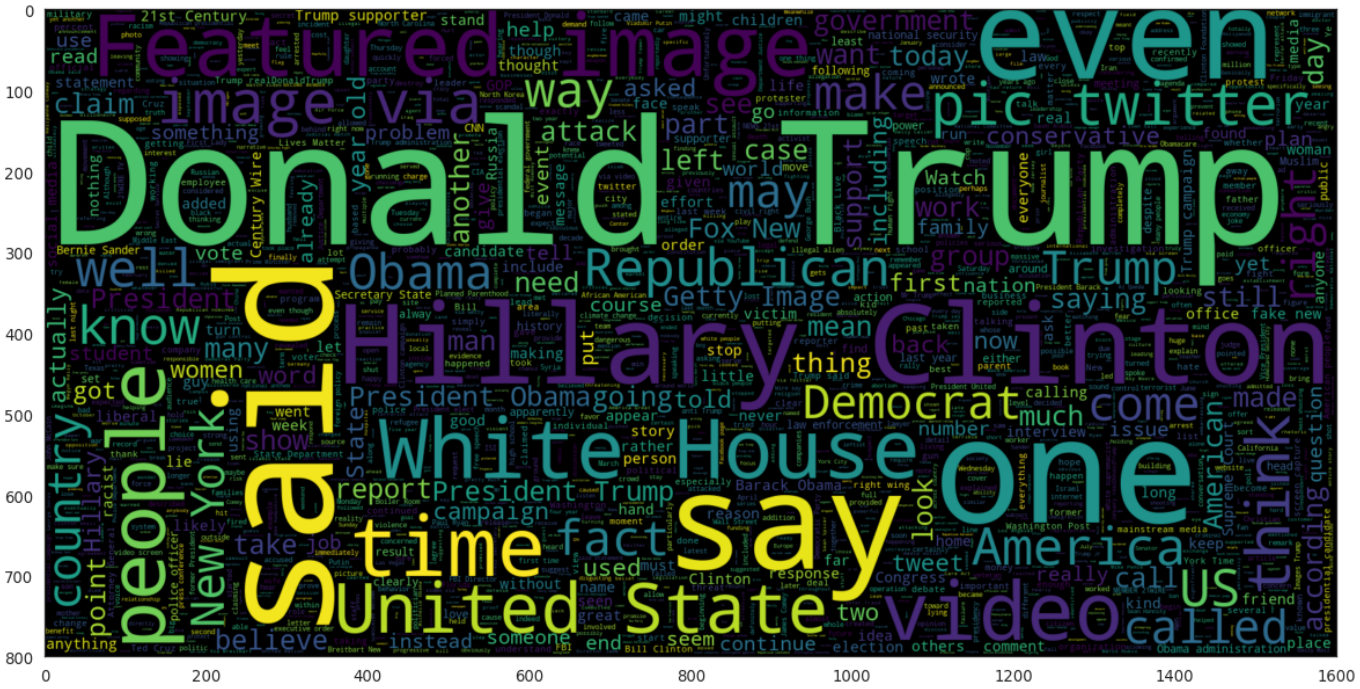

Fig 6. WORDCLOUD FOR FAKE TEXT (LABEL - 0)

12. Data Visualization

Histogram plot of the character length distribution for two categories of text data: original text and fake text. It first creates a figure with two subplots using the subplots function, specifying the number of rows and columns of subplots (in this case, 1 row and 2 columns) and the size of the figure.

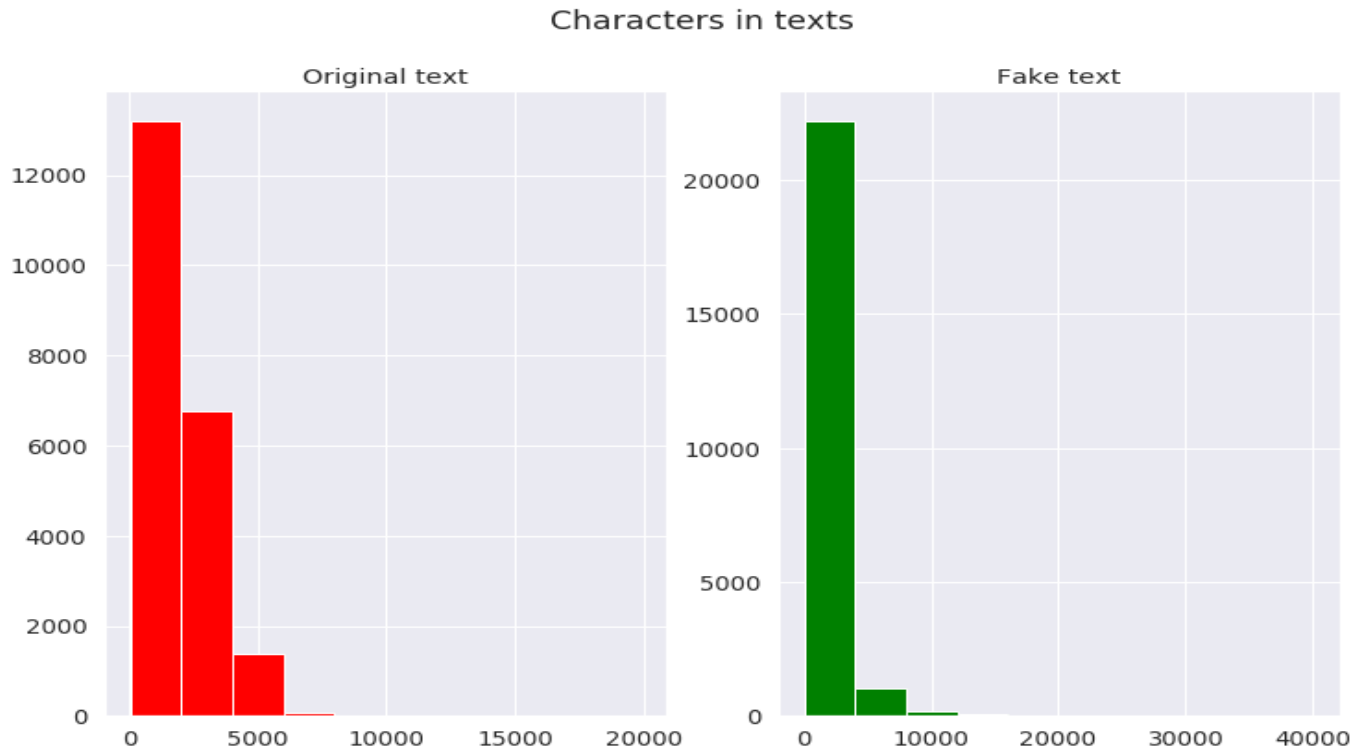

Fig 7. Characters in texts

A visualization that compares the word length of two types of texts - real and fake. The text data is assumed to be stored in a pandas dataframe, where the 'text' column contains the actual text, and the 'category' column is a binary indicator (0 or 1) that specifies whether the text is real or fake.

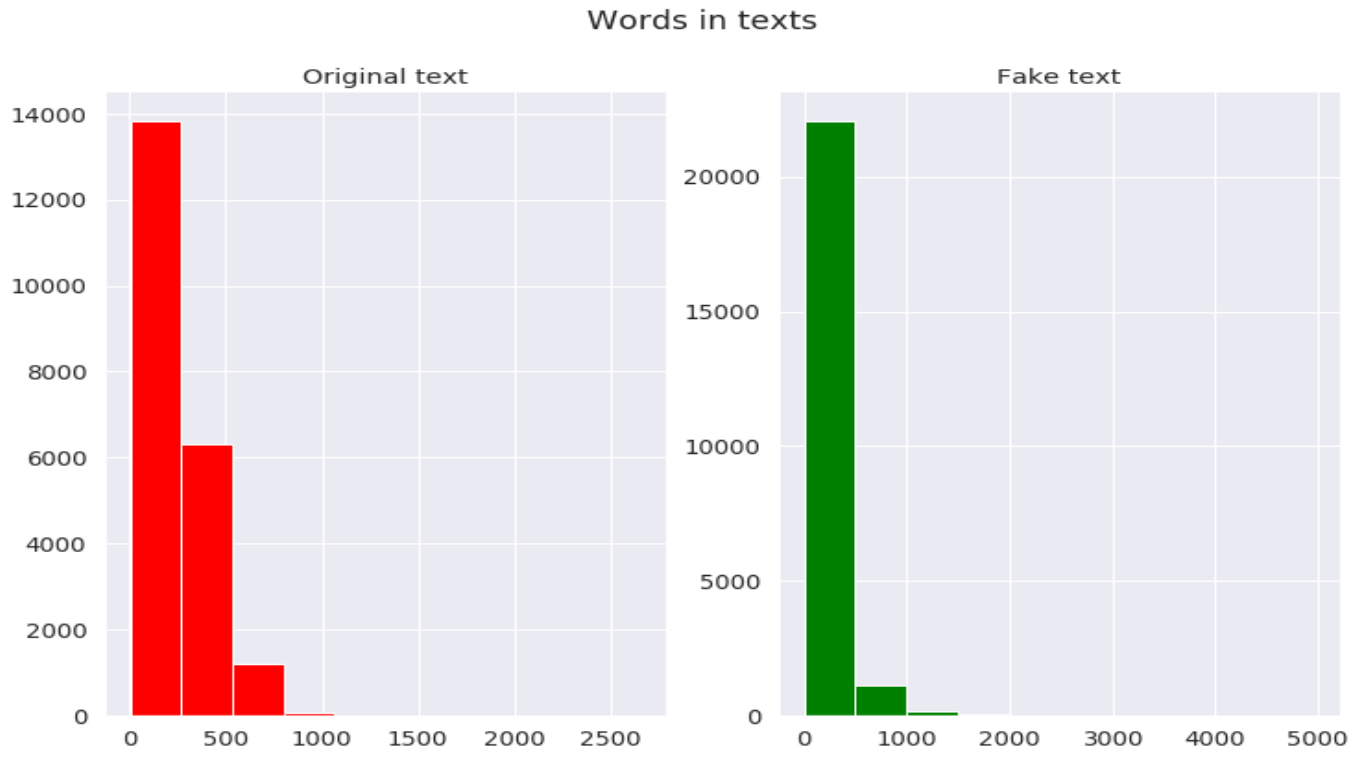

Fig 8. Words in texts

This visualization below compares the average word length of two types of texts - real and fake. The text data is assumed to be stored in a pandas dataframe, where the 'text' column contains the actual text, and the 'category' column is a binary indicator (0 or 1) that specifies whether the text is real or fake.

Then, the average word length is calculated for each text using the map() and mean() functions, and a histogram of the distribution of the average word lengths is plotted on each subplot using the distplot() function from seaborn. The color of the histogram for the real text is set to red, and the color for the fake text is set to green.

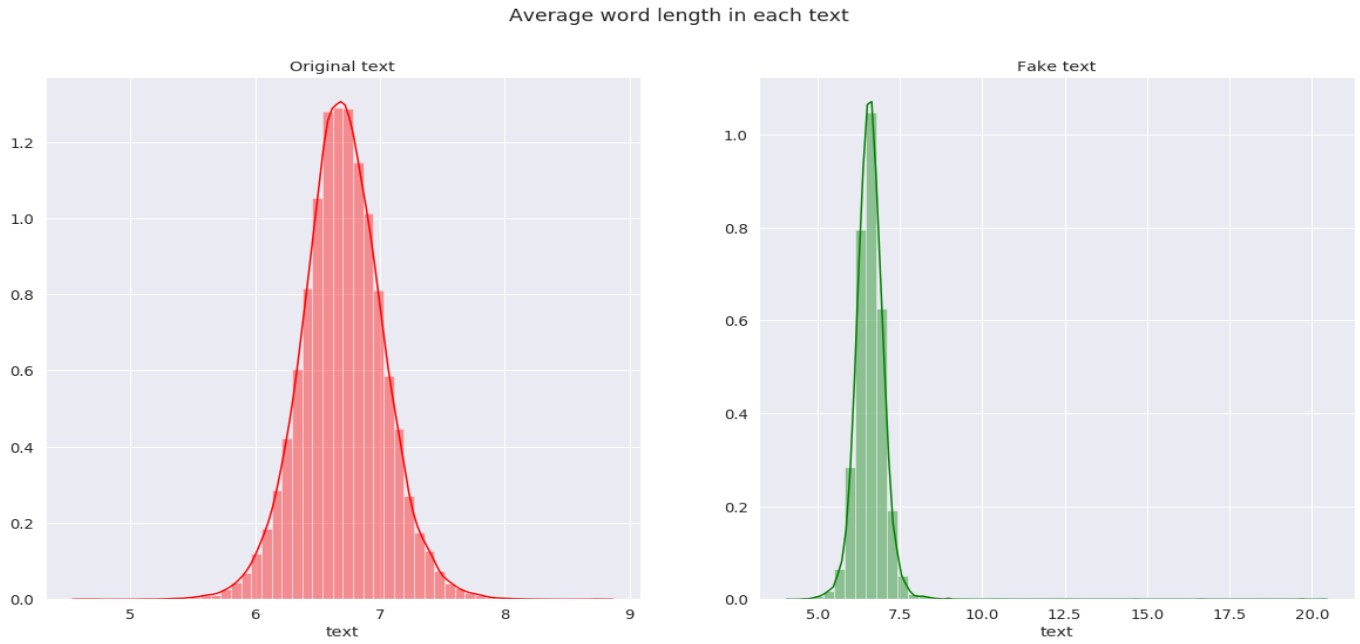

Fig 9. The average word in each text

13. Feature Extraction

A function is defined called get_corpus that takes in a list of strings (text), and returns a list of all the words in the text. The function iterates through each string in the input list, splits it into words, and adds each word to the output list.

13.1 CountVector

The function that takes a corpus of text and returns the top n n-grams (sequences of n words) in the text. It uses the CountVectorizer class from the scikit-learn library to convert the text into a bag-of-words representation, which counts the frequency of each word. It then computes the frequency of each n-gram and returns the top n most frequent n-grams. The parameter g specifies the size of the n-gram.

13.2 Unigram Analysis

A bar chart is created that displays the top 10 most common unigrams (single words) in a text corpus, using the seaborn library for visualization. The x-axis of the barplot displays the frequency counts (values) of each unigram, and the y-axis displays the unigrams themselves (keys).

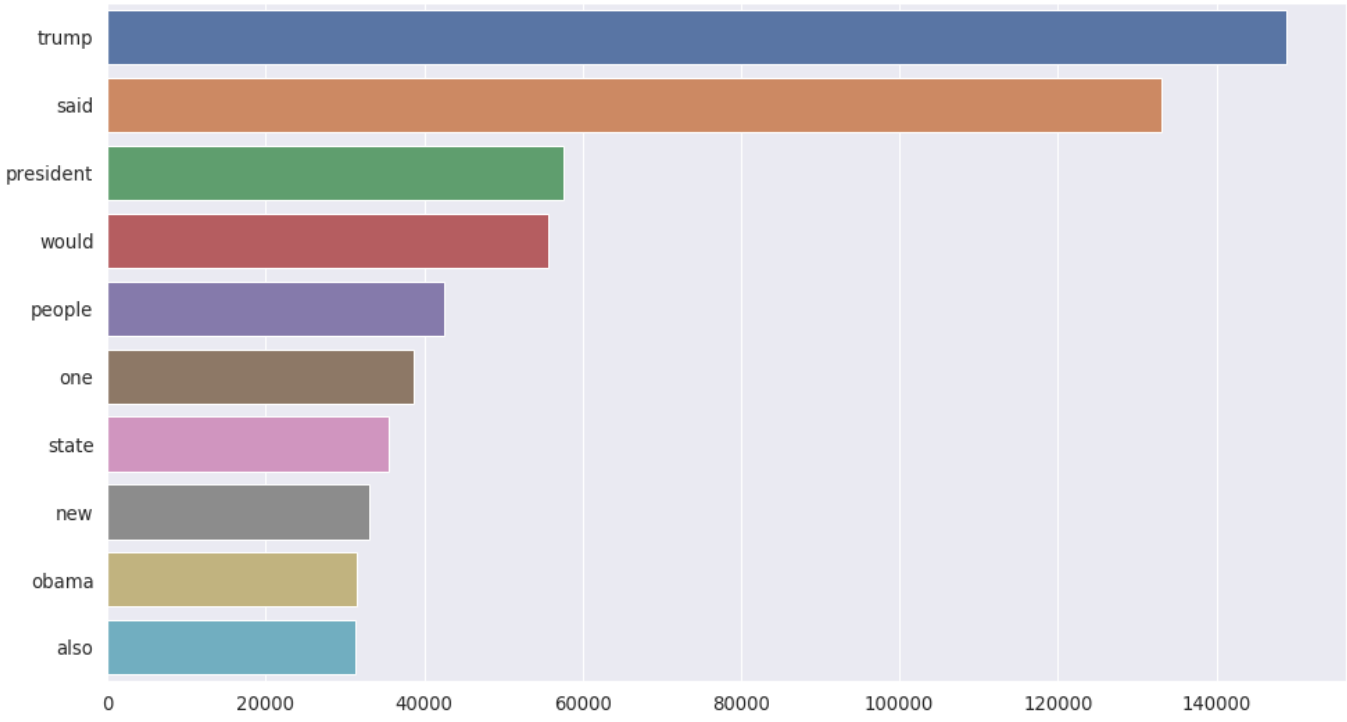

Fig 10. Unigram Analysis

13.2 Bigram Analysis

This creates a bar plot visualization of the most common bigrams (pairs of adjacent words) in a text corpus. It calls a function called 'get_top_text_ngrams', which presumably takes in the text data from a pandas dataframe 'df' and returns the most common bigrams in the text. The function returns a dictionary of the most common bigrams and their frequencies.

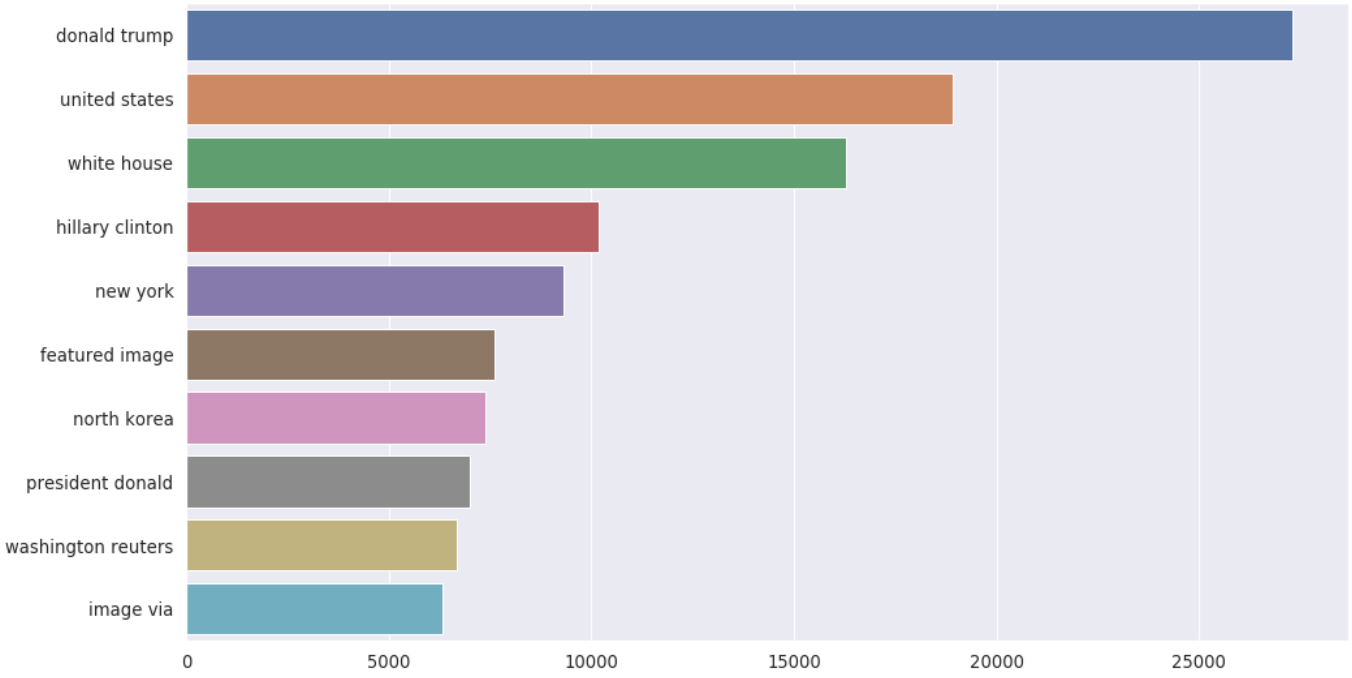

Fig 11. Bigram Analysis

13.3 Trigram Analysis

This creates a bar plot visualization of the top 10 most common trigrams (i.e., three-word sequences) in a text dataset stored in a pandas dataframe 'df'. Then, the function get_top_text_ngrams() is called with arguments 'df.text' (the text column of the dataframe), '10' (indicating to return the top 10 most common trigrams), and '3' (indicating to return trigrams). The result is stored in a dictionary called 'most_common_tri'.

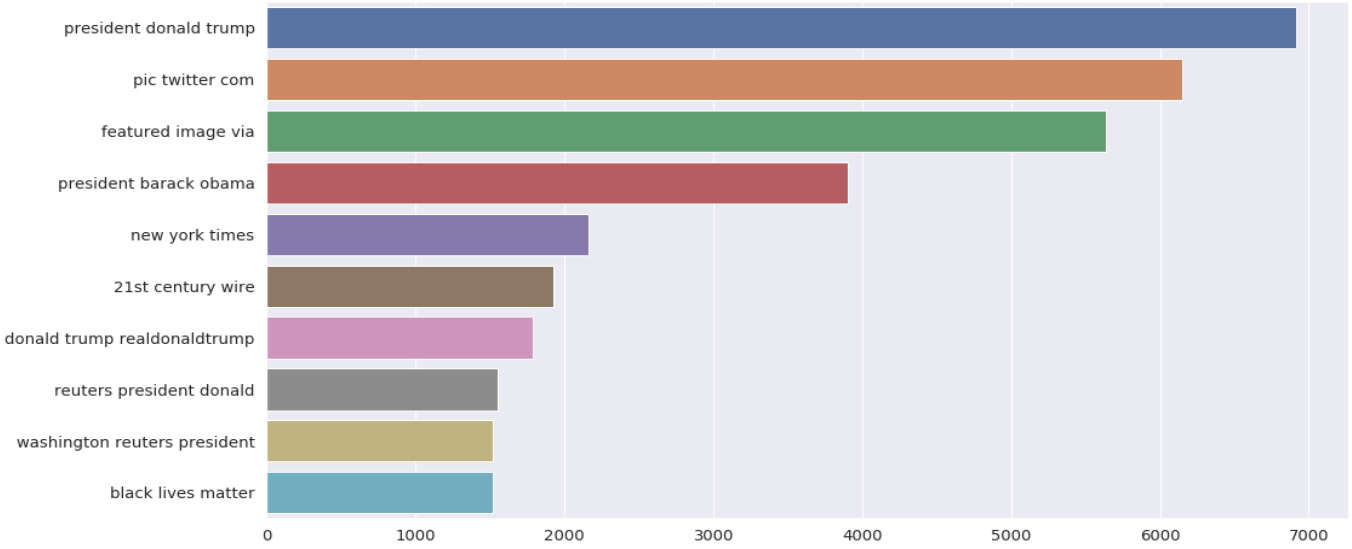

Fig 12. Trigram Analysis

14. Splitting of the Data

We are splitting the data stored in a pandas dataframe 'df' into two sets - a training set and a test set - for use in a supervised machine learning model.

The independent variable, 'text', is assigned to the variable 'x', and the dependent variable, 'category', is assigned to the variable 'y'.

The train_test_split() function from the scikit-learn library is used to randomly split the data into a training set (x_train and y_train) and a test set (x_test and y_test).

14.1 Tokenizer

This prepare the text data for a neural network model. It uses the Keras Tokenizer class to convert the text into sequences of integers and then pads those sequences to a fixed length using the sequence.pad_sequences() method. This is necessary because neural networks require fixed-length inputs. The resulting x_train and x_test arrays can then be used as input to the neural network model.

14.2 GloVe method

GloVe (Global Vectors for Word Representation) is an unsupervised machine learning algorithm used to create dense vector representations (embeddings) of words in a corpus. GloVe aims to capture the meaning of words by analyzing their co-occurrence patterns in the corpus.

The algorithm works by constructing a word-context matrix, which contains information about the frequency of co-occurrences of words in a given context. The matrix is then used to compute a weighted least squares regression, which results in a low-dimensional vector space where words with similar meanings are clustered together.

The resulting embeddings can be used in various NLP tasks, such as text classification, sentiment analysis, and machine translation, among others.

GloVe method is built on an important idea, You can derive semantic relationships between words from the co-occurrence matrix. Given a corpus having V words, the co-occurrence matrix X will be a V x V matrix, where the i th row and j th column of X, X_ij denotes how many times word i has co-occurred with word j. An example co-occurrence matrix might look as follows.

	the	cat	sat	on	mat
the	0	1	0	1	1
cat	1	0	1	0	0
sat	0	1	0	1	0
on	1	0	1	0	0
mat	1	0	0	0	0

Fig 13. Co-occurrence Matrix

14.3 Word Embedding

It is a representation of text where words that have the same meaning have a similar representation. In other words it represents words in a coordinate system where related words, based on a corpus of relationships, are placed closer together. Word embeddings are in fact a class of techniques where individual words are represented as real-valued vectors in a predefined vector space. Each word is mapped to one vector.Each word is represented by a real-valued vector.

15. Training the Model

The model is based on LSTM (Long Short-Term Memory) architecture, which is suitable for processing sequential data such as text. The code sets the batch size to 256, meaning that the model will process 256 samples at a time during training. The number of epochs is set to 10, which determines how many times the model will iterate over the entire training dataset.

The size of the word embeddings used in the model is set to 100. Word embeddings are representations of words in a vector space that capture their semantic meaning. The model uses pre-trained word embeddings, which are loaded into the model using the weights parameter.

The learning rate of the optimizer is set to 0.01, which controls how much the weights are adjusted during training. The learning rate is reduced using the ReduceLROnPlateau callback, which reduces the learning rate by a factor of 0.5 if the validation accuracy does not improve for 2 epochs.

The model architecture consists of an embedding layer, two LSTM layers, and two dense layers. The embedding layer maps the input sequences of words to their corresponding word embeddings. The LSTM layers process the sequences of embeddings and extract relevant features. The dense layers perform the final classification based on the extracted features. The output of the model is a probability score, which is converted to a binary label using a sigmoid activation function.

During training, the model is evaluated on both the training and testing datasets using the evaluate() function, which returns the loss and accuracy of the model. The history of the training process is stored in the history variable, which can be used to plot the learning curves and analyze the performance of the model.

```
In [50]: print("Accuracy of the model on Training Data is - " , model.evaluate(x_train,y_train)[1]*100 , "%")
         print("Accuracy of the model on Testing Data is - " , model.evaluate(X_test,y_test)[1]*100 , "%")

         33673/33673 [==============================] - 108s 3ms/step
         Accuracy of the model on Training Data is -  99.99703168869019 %
         11225/11225 [==============================] - 36s 3ms/step
         Accuracy of the model on Testing Data is -  99.86637234687805 %
```

Fig 14. Accuracy of the Model

Visualizing the training and testing accuracy and loss of a machine learning model across 10 epochs using the Python library, Matplotlib.

The training accuracy and validation accuracy plots should show a steady increase in accuracy over the epochs, while the training loss and validation loss plots should show a steady decrease in loss over the epochs. The plot shows the training and testing accuracy as the number of epochs increases during model training. The x-axis represents the number of epochs, while the y-axis represents the accuracy score.
If there is a close gap between the testing and training accuracy, it suggests that the model is performing well on both the training and testing data, and it's not overfitting or underfitting.

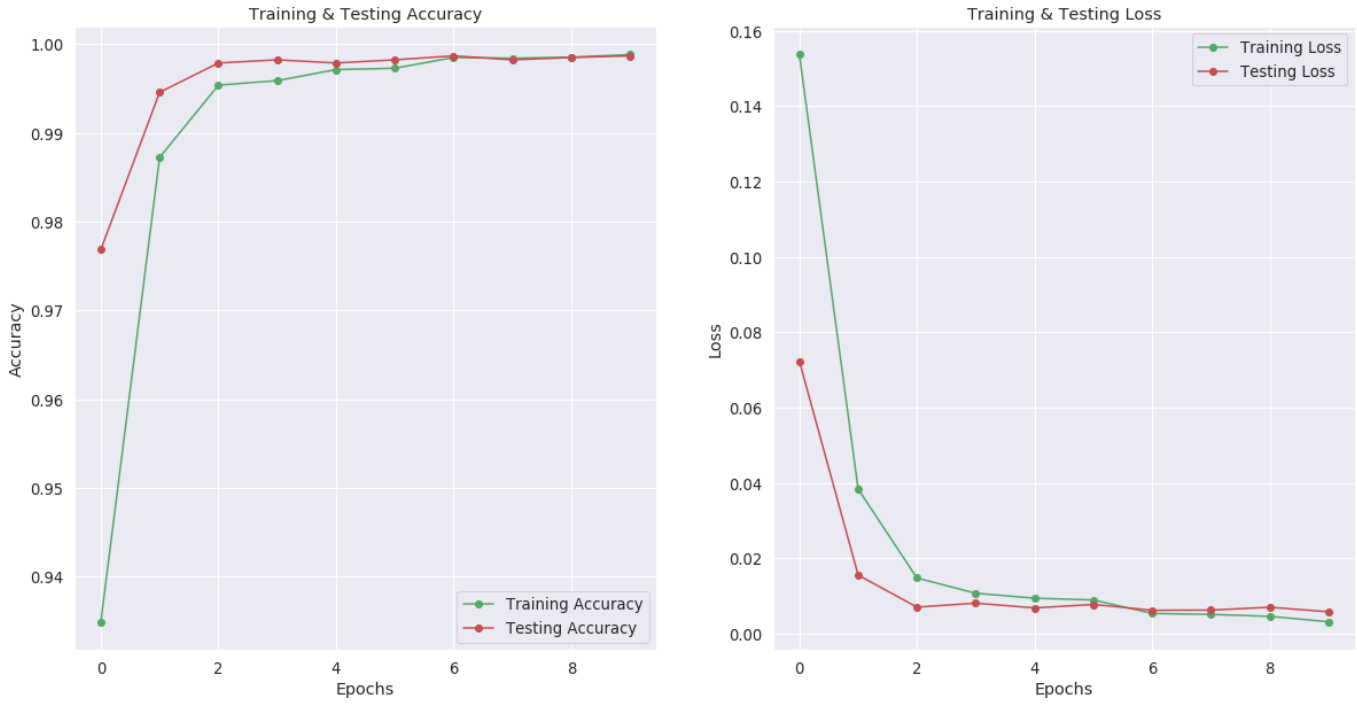

Fig 15. Plots of Training & Testing Accuracy and Loss

We shave a close gap between the testing and training accuracy, it suggests that the model is performing well on both the training and testing data, and it's not overfitting or underfitting.

16. Results

16.1 Classification Report

In machine learning, a classification report is a performance evaluation metric that provides a comprehensive report of the model's performance for each class in a classification problem. It is commonly used to evaluate the predictive accuracy of a classification model by summarizing the precision, recall, and F1 score for each class.

The classification report contains various metrics such as precision, recall, F1-score, and support for each class. Precision measures how often a positive prediction is correct, while recall measures how well the model can detect positive instances. The F1-score is the harmonic mean of precision and recall, providing a balanced score that takes both metrics into account. Finally, the support refers to the number of samples in each class.

```
In [53]: print(classification_report(y_test, pred, target_names = ['Fake','Not Fake']))
```

```
                precision    recall  f1-score   support

          Fake       1.00      1.00      1.00      5858
      Not Fake       1.00      1.00      1.00      5367

      accuracy                           1.00     11225
     macro avg       1.00      1.00      1.00     11225
  weighted avg       1.00      1.00      1.00     11225
```

Fig 16. Classification Report

16.2 Confusion Matrix

A confusion matrix is a table used in machine learning to evaluate the performance of a classification model by comparing the predicted and actual labels of a test dataset. It shows the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) for each class in the classification problem.

To get better results, we want the values of TP and TN to be as high as possible, and the values of FP and FN to be as low as possible. A high TP value means that the model correctly identified a positive case, while a high TN value means that the model correctly identified a negative case. A low FP value means that the model didn't incorrectly predict a positive case when the actual class was negative, and a low FN value means that the model didn't incorrectly predict a negative case when the actual class was positive. In other words, we want the model to minimize false positives and false negatives to achieve high accuracy and precision in the classification.

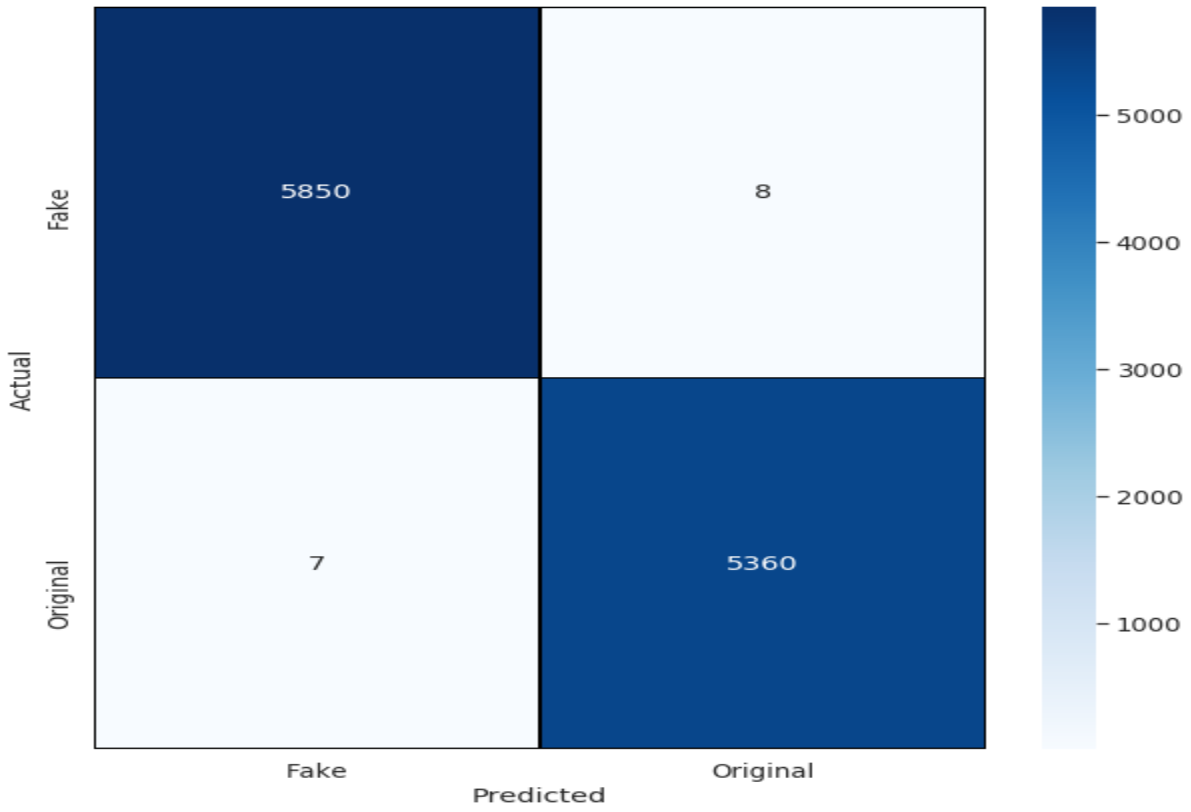

Fig 17. Confusion Matrix

Our confusion matrix shows that we have the values of TP and TN as high as possible, and the values of FP and FN to be as low as possible.

17. Conclusion

In conclusion, the proposed deep learning model shows promising results in accurately classifying news articles as real or fake, using GloVe word embedding and LSTM architecture. The high accuracy, confirmed by the confusion matrix and plots, suggests that this model can be helpful in addressing the issue of misinformation in the news media. Further experiments and analysis will be required to explore the potential of deep learning models with attention for automatic credibility analysis of news. Overall, this work contributes to the development of effective tools and methods for mitigating the spread of distorted or false information in society.

18. References

[1]Reuters World News, October 21,2018,https://www.reuters.com/article/us-brazil-election-whatsapp-explainer/facebooks-whatsapp-flooded with-fake-news-in-brazil-election-idUSKCN1MU0UP,last accessed 2019/07/13

[2] CNN Business April 21, 2019, https://edition.cnn.com/2019/04/21/tech/sri-lanka-blocks-social-media/index.html, last accessed 2019/07/13.

[3] Fake News Challenge, http://www.fakenewschallenge.org/, last accessed 2019/07/13.

[4] William Kai Shu, Amy Sliva , Suhang Wang , Jiliang Tang and Huan Liu. (2017). “Fake News Detection on Social Media: A Data Mining Perspective”, SIGKDD Explorations: 19(1).

[6] GloVe: Global Vectors for Word Representation, https://nlp.stanford.edu/projects/glove/, last accessed 2019/07/13.

[7] Hassan and A. Mahmood. (2018). “Convolutional Recurrent Deep Learning Model for Sentence Classification.” IEEE Access 6:1394913957

[8] Hochreiter, J. Schmidhuber. (1997). “Long Short-Term Memory”, Neural Computation, 9(8):1735-1780.

[9] K. Greff, R. K.Srivastava, J. Koutník, B. R. Steunebrink, J. Schmidhuber. (2015). “LSTM: A search space odyssey.”IEEE Transactions on Neural Networks and Learning Systems.

[10] ChatGPT. (2021). [Description of the specific information or advice provided]. OpenAI. https://openai.com/