Sentiment Analysis on Twitter Data

Uponika Barman Roy (251299020)
Electrical and Computer Engineering
Western University
London, Canada
ubarmanr@uwo.ca

Harshit Kohli (251310106)
Electrical and Computer Engineering
Western University
London, Canada
hkohli4@uwo.ca

Shrutam Parmar (251312991)
Electrical and Computer Engineering
Western University
London, Canada
sparma55@uwo.ca

***Abstract*— Internet usage has made social media an integral part of our everyday lives. With the aid of Natural Language Tool Kit (NLTK), sentiment analysis refers to the process of identifying and analyzing a piece of writing to determine whether its sentiment, opinions, views, and emotions are positive, negative, or neutral towards a specific issue, from politics and entertainment to social issues and personal experiences. As a result, sentiment analysis on Twitter data has become an increasingly popular research area, as it can provide valuable insights into public opinions and attitudes towards different issues. Sentiment analysis on Twitter data can have various applications, including brand monitoring, reputation management, customer feedback analysis, and market research. For example, companies can use sentiment analysis to monitor social media mentions of their brand and products and identify potential issues or areas for improvement. Similarly, governments and policymakers can use sentiment analysis to gauge public opinion on different policies and initiatives and adjust their strategies accordingly. However, sentiment analysis on Twitter data also faces several challenges, including the presence of noise, sarcasm, and irony, as well as the potential impact of data biases. For example, Twitter users may express their opinions in different ways depending on their cultural, social, or demographic backgrounds, which can affect the accuracy of sentiment analysis models. Therefore, developing robust and reliable sentiment analysis models that can handle such challenges is crucial for the success of this research area. In the chosen topic, the sentiment analysis is done on public opinions as posted on Twitter. Data is collected from one of the Kaggle competitions namely "Tweet Sentiment Extraction". The train dataset contains 27481 rows and test dataset contains 3534 rows including the tweets in the text format, a selected part of text containing the emotions, the sentiments behind the tweets as the labels (positive, negative, or neutral) and the text id. Next is the data preprocessing where the text data is explored, different preprocessing steps are applied on it which will be discussed in the following sections. Another important step here is to tokenize the tweets and convert the words to lowercase. The following step would be converting the preprocessed tweets into numerical features that can be used by a deep learning model. The models like Bi-directional Long-Short Term Memory (LSTM) and Google's pre-trained BERT are used to predict the part of the text which accurately expresses the correct sentiment. The models are trained and evaluated against the test. Further, the models are trained, and predictions are performed. Different classification performance metrics are used to evaluate the above models.**

Keywords— Natural Language Processing, Sentiment Analysis, Twitter Dataset, LSTM, BERT

I. INTRODUCTION

Twitter is a popular social media platform where people from all over the world share their opinions on a wide range of topics. As a result, Twitter has become a valuable source of feedback for businesses and organizations seeking to understand customer sentiment [1]. The goal of our project is to use natural language processing (NLP) techniques to analyze public tweets and predict the sentiments behind them.

The challenge we face is teaching a machine learning model to recognize the complex and nuanced sentiments that are expressed in everyday language. To do this, we will be using advanced NLP techniques and deep learning models such as Long-Short Term Memory (LSTM) and BERT. These models have been shown to be effective in tasks such as sentiment analysis, and we will be using them to analyze a large dataset of tweets.

The value of sentiment analysis goes beyond simply understanding how people feel about a particular topic or product. By analyzing sentiment, businesses can gain insights into customer preferences and make informed decisions about product development, marketing, and other key areas. In our project, we will be using Python, along with machine learning libraries such as Scikit-learn, Keras, and PyTorch, to implement our models and analyze the data.

II. RELATED WORK

Sentiment analysis has been a popular area of research, particularly in the context of Twitter data. However, sentiment analysis in general can be highly useful in determining any product or services. For instance, in the first survey paper, the authors conducted sentiment analysis on Twitter data from KFC and McDonald's to determine which company has better reviews. The proposed data preprocessing involves removal of URLs, email ids, stop words, emojis and punctuations from the review tweets. The statistical models classify the tweets into positive, negative, and neutral categories. The Maximum entropy model experimented in the paper gives the best results [2].

Similarly, in another paper the COVID-19 Twitter data is used to analyze public sentiments on the pandemic situation. The models implemented are bidirectional LSTM, Google's pre-trained architecture BERT, and CNN. The tweet emotions are classified in three types- positive, negative, and neutral. The authors reported that the BERT model achieved the highest accuracy of 64.1% [3].

While the existing research has provided useful insights, it has some limitations, such as not considering the impact of context on sentiment analysis. In this project, we aim to address this limitation by utilizing deep learning models, such as Bi-directional Long-Short Term Memory (LSTM) and Google's pre-trained BERT model, to accurately predict the sentiment of tweets. Furthermore, we propose

to use the Jaccard similarity score to evaluate the models' performance on imbalanced data.

III. Dataset

The twitter dataset is taken from Kaggle's official competition website. It is important to note that the dataset was collected during a particular period of time and may not be the representative of the sentiments of Twitter users at present or in the future. To ensure unbiased model evaluation, the training and testing datasets have been provided separately by Kaggle. In the training data, there are a total of 27481 rows where each row represents a single tweet. The data contains 4 columns comprising of textID which is a unique ID for each piece of text, text – the text of the tweet, sentiment- general sentiment of the tweet (positive, negative and neutral) and selected_text – it is the part of tweet which exhibits the sentiment of the user [4].

The test data contains 3534 rows of tweets and 3 columns of textID, text and sentiment. The dataset is preprocessed to some extent, but further data cleaning and preprocessing may be required to improve the accuracy of the models. This includes handling misspellings, grammatical errors, and typos that are common in social media posts. In addition to sentiment analysis, this dataset can also be used for other natural language processing tasks, such as topic modeling and text classification.

The pie chart (Fig.1)below shows the distribution of sentiments in the train data set, it is observed that out of three categories of sentiments the neutral sentiment has the highest frequency of the tweets as compared to the positive and negative tweets with 40.45% contribution, rest 31.23% and 28.32% are the positive and negative tweets respectively. Although the size of the dataset is relatively small and may not be enough to train deep learning models effectively. However, with careful feature engineering and model selection, we can still achieve good results.

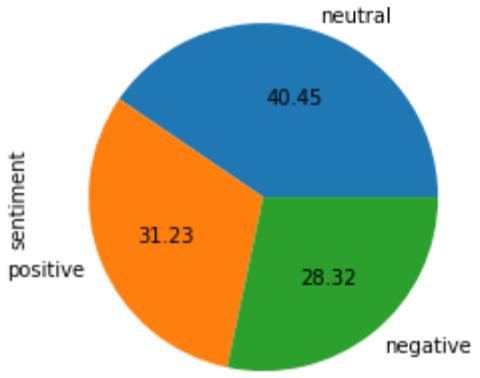

Fig. 1. Sentiment class distribution in the training dataset

IV. Methodology

The data is loaded using Pandas library of python. The next steps are discussed as follows:

A. Exploratory Data Analysis

- *Exploring dataset*- This step uses the functions like info(), describe() and shape() to see the statistical distribution of the data, shape of the data and to gather information about the rows and columns of the data as shown in Fig 2.

	textID	text	selected_text	sentiment
count	27481	27480	27480	27481
unique	27481	27480	22463	3
top	cb774db0d1	I`d have responded, if I were going	good	neutral
freq	1	1	199	11118

Fig. 2. Description of the training dataset

- *Frequency of discrete column categories*- As seen in the last step, the dataset contains a column called sentiment. So, using the function value_counts(), the frequency of each class of the column can be seen.

```
neutral     11118
positive     8582
negative     7781
Name: sentiment, dtype: int64
```

Fig. 3. Frequency of sentiments

- *Finding and fixing Missing Values*- This is an important step at any data preprocessing where the columns containing missing data and its frequency can be found and removed. In this case, the number of nullis found to be only one row, so the entire row is dropped.
- *Data Preprocessing*- In this step, different techniques like estimating the character count, word count, frequency of unique words, removing URLs, special characters and stop words as it does not have any impact on the sentiment. This preprocessing is done by using different regular expressions in the corresponding lambda functions. Lemmatization technique from '*WordNetLemmatizer'* package of '*nltk.stem'* utility was used to reduce the words to their base form.
- *Word Cloud Formation*- Once the data is cleaned, the '*wordcloud'* package is imported and used to form the set of words with maximum frequency and highest importance [5].

Fig. 4. Word Cloud

- With the help of bigrams, trigrams and wordcloud the significant words which exemplify the sentiment are extracted from the tweet data as shown below.

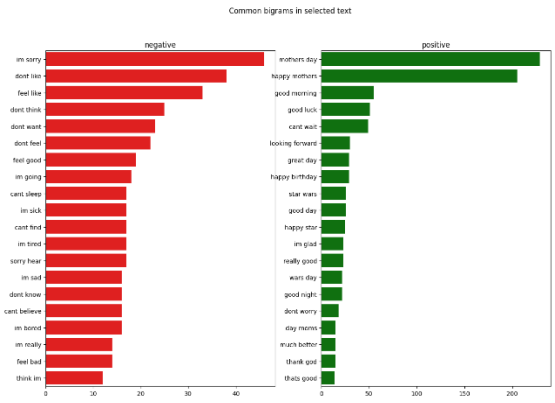

Fig. 5. Biagrams of sentiment

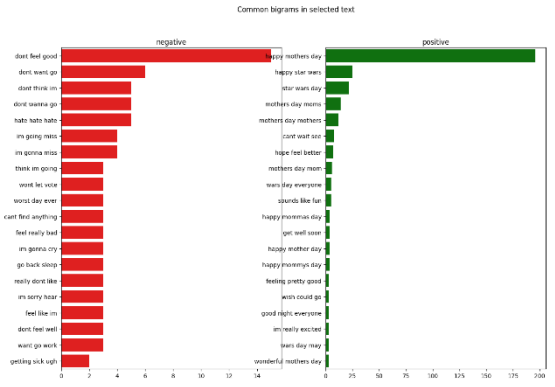

Fig. 6. Triagrams of sentiment

B. *Implementation of Bidirectional LSTM model*

- *Feature Extraction*: The model learns only on the numerical data hence it is imperative to convert the words into vector matrices. For that, the words are tokenized into numerical vectors using '*Tokenizers'* package from Keras *preprocessing* library. Along with tokenization, the '*pad_sequences'* from the Keras *utility* is used to truncate the longer texts and reshape them to maintain scalability in the input data. The tokenized model is then saved in the pickle file for future repetitive usage.

- *Train-Validation Split*: The train dataset is split into train and validation set with 80-20 distribution. After splitting, the training set contains 21984 samples and validation set contains 5497 samples. A complete set of data with 3534 samples is kept aside for testing after the models are trained. It will be discussed in the later section.

- The first model built is the bidirectional LSTM. *Long Short-Term Memory (LSTM)* is a type of recurrent neural network which is used to process sequential data. The bidirectional LSTM processes the data in both forward and backward directions and hence posses the memory which plays a crucial part in NLP [6].

- Prior to the addition of LSTM layer, an *embedding* layer is added from the KERAS package. The objective of using an embedding layer is to receive the input tokens, estimate the Euclidean distance between them and then feed them to the bidirectional LSTM layer. Hence, it mostly works as an interface between the input matrix of vocab and the model.

- In this experiment, the LSTM layer is designed with 8 neurons. A dense hidden layer is defined after the LSTM layer which has 24 neurons and '*relu'* as the activation function. The following layer is the output layer where it is activated with *softmax* function as dataset is multi classified in three sentiments. Below in fig 7, the model summary gives an overview of the architecture used.

- The model is trained with specifications like:

  vocab_size = 50, embedding_size = 32, epochs=10

  learning_rate = 0.1, batch_size = 128

- Once the model is trained, it is used to predict on the test samples.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 5000, 32)          1600

bidirectional (Bidirectiona  (None, 16)                2624
l)

dense (Dense)                (None, 24)                408

dense_1 (Dense)              (None, 3)                 75

=================================================================
Total params: 4,707
Trainable params: 4,707
Non-trainable params: 0
_________________________________________________________________
```

Fig. 7. Bidirectional LSTM model architecture

- The performance of the model is discussed in the result and discussion section.

C. *Implementing the pre-trained BERT model*

- BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained language model developed by Google in 2018.

- BERT is trained on a large corpus of text data using a masked language modeling (MLM) objective and a next sentence prediction (NSP) objective. The MLM objective involves randomly masking some words in the input sequence and training the model to predict the masked words based on the context of the surrounding words. The NSP objective involves feeding the model pairs of sentences and training it to predict whether the second sentence is likely to follow the first sentence in the original text [7].

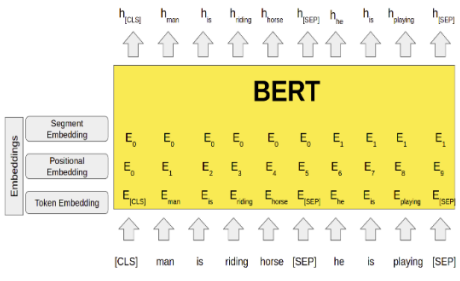

Fig. 8. BERT architecture

- It has become one of the most popular pre-trained language models and has been used in many state-of-the-art natural language processing systems.
- In this experiment, the first BERT model is '*bert-large-uncased-whole-word-masking-finetuned-squad*' which verifies the relativity of the extracted text from the tweet text using Jaccard's similarity score. The configuration of the model is 24-layer, 1024 hidden dimension, 16 attention heads and 336M parameters. This experiment is done to gain confidence on the BERT's selection of text to classify the correct sentiment [8, 9].
- Another BERT model used is '*bert-base-cased'* which is a case sensitive pre-trained base model of 12 layers, 768 hidden units per layer, and 110 million parameters. This model is used to classify the sentiments of the tweets in three categories of positive, negative, and neutral [10].
- There are few preprocessing need to be performed before implementing BERT, in addition to the basic preprocessing steps that are already done [7].
- [SEP] and [CLS] tokens are the markers where the model identifies the end and start of a sentence to understand that it is a classification task [7, 8, 11].
- To avoid the issue of varied tweet lengths, the maximum length of tweet is considered as a fixed length for the model to work on. As shown in fig 9, most of the tweet length is less than 80 but for safer side, the maximum length is set to 100. This is the concept of sequence padding [11].

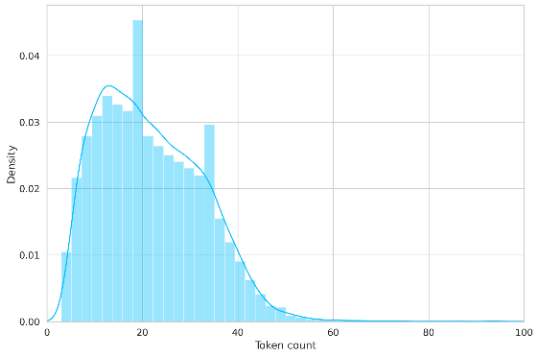

Fig. 9. Length of Tweets

- The next step is to create an array of every tweet in the padded sequence where the pad token is marked as 0 and the real data token is given attention and marked as 1. This is called *attention masking* which is the most important preprocessing needed for a BERT model to train [11].
- All the above steps are wrapped in a class to enter the pyTorch environment where the model will be trained.
- The dataset is now split into train, validation, and test set. From the entire train and test set of data, 80% is assigned to the training set, 10% for the validation and a complete new 10% for the test set. A data loader is created which releases the data in a *batch size* of 16.
- The BERT model is then loaded with a *drop out layer* with a *regularization factor* of 0.3 to avoid overfitting followed by a fully connected output layer.
- The hidden layers of the model are verified as 768. Now the model is ready to train on the data with *Adam optimizer* and a *learning rate scheduler* with *no warmups*. The use of this scheduler is to provide model stability while the learning rates are increased or decreased. With all the mentioned specifications, the model is trained for 10 *epochs*. After training, the model gets a new set of test data to predict, and the result will be discussed in the next section.

V. Result/Analysis

The Bidirectional LSTM model that was studied did not perform good in both training as well as testing phase. In the below figures the loss per epoch and accuracy per epoch for both the training and validation set is depicted.

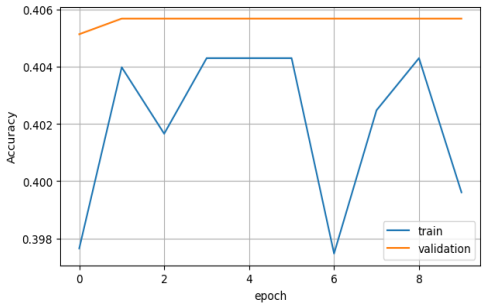

Fig. 10. Accuracy versus epochs

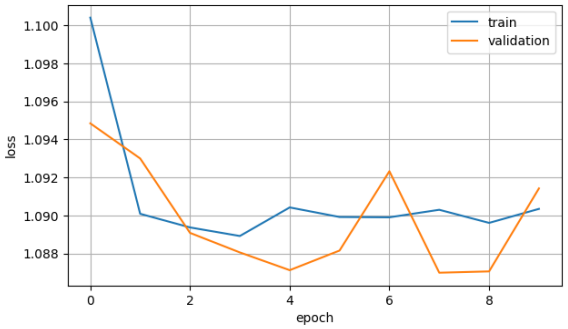

Fig. 11. Loss versus epochs

From the training phase of the model, it is observed that there is no significant change in the training and validation accuracy through out 10 epochs , so it infers that the model failed to learn on the data.

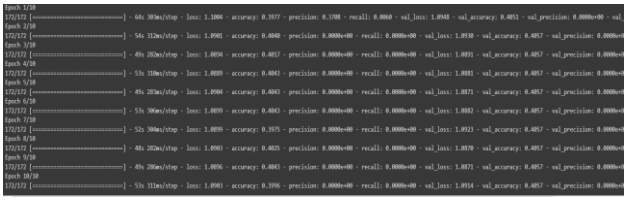

Fig. 12. LSTM Training summary

The metrics like Recall, Precision represent number of correct positives out of all the real positives and the number of correct positives out of predicted positives. And the F1 score is the harmonic mean of the precision and recall values. These metrics help to evaluate the model's performance better as it brings out how many samples are correctly predicted. The classification report shown below represents how bad the model performed in terms of accuracy, precision, recall and F1 score [12].

```
111/111 [==============================] - 14s 112ms/step
              precision    recall  f1-score   support

           0       0.00      0.00      0.00      1001
           1       0.40      1.00      0.58      1430
           2       0.00      0.00      0.00      1103

    accuracy                           0.40      3534
   macro avg       0.13      0.33      0.19      3534
weighted avg       0.16      0.40      0.23      3534

[[   0 1001    0]
 [   0 1430    0]
 [   0 1103    0]]
```

Fig. 13. LSTM – Classification report

From the loss vs epoch and accuracy versus epoch graphs as shown above, it is seen that the training and validation accuracy is not up to the mark. Due to the smaller size of dataset, LSTM fails to learn much from the data. Also, there is no point of hyperparameter tuning of this model as it is severly underperforming. This motivated to look for another model.

The challenges from the LSTM model as discussed above leads to experiment with BERT. In this pre-trained architecture two experiments are done in which the first imported model '*bert-large-uncased-whole-word-masking-finetuned-squad*' extracts the relative sentiment of the tweets that holds emotions.

The Jaccard similarity score is the metric which finds the diversity of two different sample sets. In this case it is obtained as 0.98 which indicates that the selected text is highly accurate.to the sentiment of the tweet. This analysis shows that if the '*bert-large-uncased-whole-word-masking-finetuned-squad'* learns these phrases, there is a higher possibility that the model will be trained accurately and hence will be able to extract the most relevant set of phrases, i.e, selected_text from the text data representing the ground truth sentiment in the next model '*bert-base-cased'*.

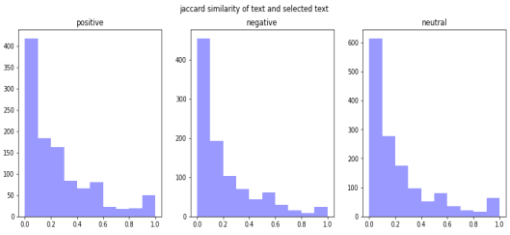

Fig. 14. Jaccard's Similarity Score used in BERT

The '*bert-base-cased'* model results better than the LSTM model. The below figure represents the training history. The training accuracy is reported as 98% where the validation is around 79%. There is still a significant difference in the performance between training and validation, yet it gives much better results than the previous model.

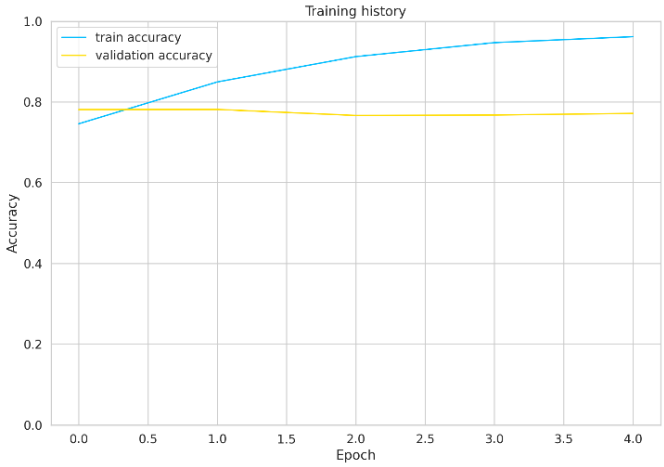

Fig. 15. Accuracy versus epoch graph for BERT model

Further, the model is predicted on the test data set and then evaluated. The recall, precision and F1 also report better prediction than the previous model and this concludes that the BERT model outperforms the Bidirectional LSTM in this experiment of analyzing the twitter data sentiments.

	precision	recall	f1-score	support
negative	0.79	0.78	0.79	782
neutral	0.76	0.77	0.76	1099
positive	0.83	0.83	0.83	867
accuracy			0.79	2748
macro avg	0.80	0.79	0.79	2748
weighted avg	0.79	0.79	0.79	2748

Fig. 16. Classification Report for BERT model

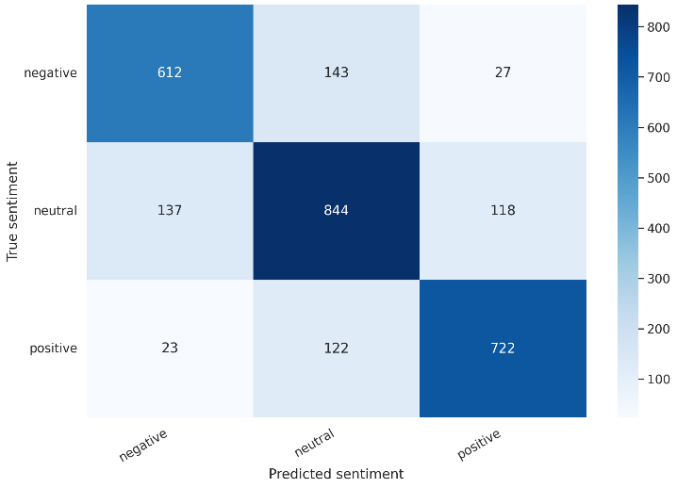

Fig. 17. Confusion Matrix for BERT model

VI. Future Steps

The result obtained so far benchmarks the performance of BERT algorithm. As seen in the training and testing report, the model can perform better. This means tuning of the hyperparameters is required so that it can overcome the performance gap between the training, validation, and testing. Another aspect is yet to be explored that how the pre-trained BERT model trains on the data if there are modifications of existing encoder layers of the model.

VII. Reference

[1] M. Kavitha, B. B. Naib, B. Mallikarjuna, R. Kavitha and R. Srinivasan, "Sentiment Analysis using NLP and Machine Learning Techniques on Social Media Data," 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), Greater Noida, India, 2022, pp. 112-115, doi: 10.1109/ICACITE53722.2022.9823708.

[2] A. Roy and M. Ojha, "Twitter sentiment analysis using deep learning models," 2020 IEEE 17th India Council International Conference (INDICON), New Delhi, India, 2020, pp. 1-6, doi: 10.1109/INDICON49873.2020.9342279.

[3] S. A. El Rahman, F. A. AlOtaibi and W. A. AlShehri, "Sentiment Analysis of Twitter Data," 2019 International Conference on Computer and Information Sciences (ICCIS), Sakaka, Saudi Arabia, 2019, pp. 1-4, doi: 10.1109/ICCISci.2019.8716464.

[4] "Tweet Sentiment Extraction" https://www.kaggle.com/competitions/tweet-sentiment-extraction

[5] Senapati Rajesh, "Twitter sentiment analysis EDA" https://www.kaggle.com/code/senapatirajesh/twitter-sentiment-analysis-eda/notebook

[6] L. Alawneh, B. Mohsen, M. Al-Zinati, A. Shatnawi and M. Al-Ayyoub, "A Comparison of Unidirectional and Bidirectional LSTM Networks for Human Activity Recognition," 2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Austin, TX, USA, 2020, pp. 1-6, doi: 10.1109/PerComWorkshops48775.2020.9156264.

[7] R. K. Kaliyar, "A Multi-layer Bidirectional Transformer Encoder for Pre-trained Word Embedding: A Survey of BERT," 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2020, pp. 336-340, doi: 10.1109/Confluence47617.2020.9058044.

[8]"bert-large-uncased-whole-word-masking" https://huggingface.co/bert-large-uncased-whole-word-masking

[9] M. Besta et al., "Communication-Efficient Jaccard similarity for High-Performance Distributed Genome Comparisons," 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS), New Orleans, LA, USA, 2020, pp. 1122-1132, doi: 10.1109/IPDPS47924.2020.00118.

[10] "bert-base-cased" https://huggingface.co/bert-base-cased

[11] A. Marium and G. S. Mamatha, "BERT Model for Classification of Fake News using the Cloud Processing Capacity," 2021 IEEE 9th Region 10 Humanitarian Technology Conference (R10-HTC), Bangalore, India, 2021, pp. 1-6, doi: 10.1109/R10-HTC53172.2021.9641632.

[12] M. Besta et al., "Communication-Efficient Jaccard similarity for High-Performance Distributed Genome Comparisons," 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS), New Orleans, LA, USA, 2020, pp. 1122-1132, doi: 10.1109/IPDPS47924.2020.00118.

Team Contribution:

Team Member	Contribution
Shrutam Parmar	Exploratory Data analysis on the Twitter Dataset, Data Preprocessing, Report Writing.
Harshit Kohli	Data Preprocessing, LSTM and BERT model implementation. Report Writing.
Uponika Barman Roy	Data Preprocessing, LSTM and BERT model implementation. Report Writing.