Twitter Sentiment Analysis

Group 8

Bhanuteja Damera
bhanutejadamera@my.unt.edu
Yeshwanth Pulapa
yeshwanthpulapa@my.unt.edu
Priyanka Bai Durvas
priyanakabai@my.unt.edu

GitHub Url:
https://github.com/priyankadurvas/Twitter-Sentiment-Analysis

March 2024

Abstract

Our project main goal is understand the large data which is generated on social media platform . We have used Twitter dataset which consists of many tweets that are created every day and worked on this dataset by analysing the tweets sentiments using NLP techniques.We implemented Naive bayes model and also deep learning models like- RNN, BI-LSTM and evaluated its metrics.

1 Introduction

We have explored the field of sentiment analysis of twitter on social media data in this project. As there are many tweets which are getting genearted every day it would be critical to understand the thoughts that it produce. In NLP methods, we investigated the twitter datasets with the goal to identify the emotions which are expressed in the messages.To ensure the quality of our dataset, we have performed data preprocessing by using techniques which includes lemmatization, and stopword removal. We then used a variety of NLP techniques, such as sentiment analysis models like Naive Bayes and Bidirectional Long Short-Term Memory (Bi-LSTM), to extract significant insights.We have evaluated our models' performance by using cross-validation techniques to determine their efficacy. We also performed exploratory data analysis, or EDA, to acquire a deeper understanding of the dataset's properties.We have given the accuracy metrics of the implemented models in a comparison study, which is the result of our work. We clarified the performance variations across the models by using visuals like accuracy graphs, offering important information for next projects involving sentiment analysis on social media platforms.

1.1 Motivation

Twitter sentiment analysis is significant for a number of reasons.

1.1.1 Understanding Public Opinion:

We know Twitter is a big platform which offers people to express their thoughts, emotions, and opinions on a range of topics, including politics, businesses, products, events, and more. Sentiment analysis will provide us clear picture of people's perception and their responses to various subjects.

1.1.2 Business Insights:

There are certain aspects in business where this analysis can be really useful such as - market trends, brand perceptions, users satisfaction and many more. This analysis helps the companies to have understand the customer's issues clearly and modify their plan accordingly.

1.1.3 Political Analysis:

This could be beneficial during elections or any political events as it provides us information about public opinion towards political parties, and other significant topics. This is useful in political stratergies and decision making procedures.

1.1.4 Social Trends and Influencer Marketing:

Knowing the Twitter sentiment patterns is essential for identifying the emerging social trends and managing effective influencer marketing campaigns. It helps to identify influencers whose opinions are well-liked by their audience and enables businesses to communicate with them for promotional purposes.

We have huge amount of twitter data and with more data comes more difficulties in analyzing the human sentiment. By using ML and NLP algorithms provides us scalable and effective solutions.

1.2 Area of Study

The field of study focuses on sentiment analysis with Natural Language Processing (NLP) techniques applied to Twitter data. Sentiment analysis, sometimes referred to as opinion mining, is a branch of natural language processing (NLP) that focuses on obtaining subjective data from textual data in order to identify the sentiment or emotional tone that is being represented. Twitter is a well-known social media site that offers a wealth of textual material with a wide range of viewpoints and feelings that users are expressing in real time.

Comprehending the opinions shared on Twitter is highly relevant for a number of reasons, such as social media trend tracking, public opinion analysis, brand monitoring, and market research. Researchers and analysts can learn a great deal about the attitudes, beliefs, and feelings of people and communities on a variety of topics by examining Twitter data.

Because they allow text data to be automatically processed and understood, natural language processing techniques are essential to sentiment analysis. To clean and ready the text data for analysis, these techniques include data pretreatment procedures including tokenization, normalization, lemmatization, and stopword elimination. Furthermore, deep learning models and machine learning algorithms are used to categorize tweet sentiment into groups like positive, negative, and neutral.

1.3 Significance

Our initiative is important because it has the potential to completely change how we perceive and use social media data, especially on sites like Twitter. We hope to significantly advance the industry by creating a solid sentiment analysis model specifically for Twitter data. These benefits include:

1.3.1 Improved Decision-Making:

Decision-makers can gain important insights into public opinion and perception by using accurate sentiment analysis. Understanding Twitter sentiment trends may greatly improve decision-making processes for everybody involved in the process, be it company executives formulating strategy, legislators creating public legislation, or marketers creating focused advertising campaigns.

1.3.2 Real-Time Insights:

Real-time tracking of public opinion via Twitter sentiment analysis enables timely responses and interventions. This is particularly crucial in circumstances such as crisis management, where timely and informed decisions can greatly lessen the effects of crises and emergencies.

1.3.3 Enhanced Customer Engagement:

Establishing good partnerships and raising customer happiness require organisations to comprehend what customers are saying on Twitter. Through sentiment analysis, businesses may pinpoint areas for development, quickly resolve consumer issues, and efficiently customise their goods and services to match the wants of their clientele.

Realising the objective of creating a precise sentiment analysis model for Twitter data is crucial to reaping these advantages and propelling social media analytics forward. We may use Twitter data to obtain insightful knowledge, guide decision-making, and produce beneficial results in a variety of fields by utilising machine learning and natural language processing techniques. Our initiative ultimately aims to enable people, groups, and societies to use social media data for enhanced decision-making, interaction, and comprehension of the outside world.

1.4 Contribution of the project

1.4.1 Knowledge Advancement:

By investigating the use of cutting-edge methods to examine vast amounts of social media data, the project adds to the body of knowledge in the domains of sentiment analysis and natural language processing (NLP). By assessing the efficiency of various NLP techniques and models in obtaining sentiments from Twitter data, it expands on previous studies.

1.4.2 Methodological insights:

In the context of analyzing Twitter data, the research offers insights regarding the efficacy of various NLP techniques, such as data pretreatment techniques and sentiment analysis models. It emphasizes how crucial preprocessing procedures like stopword removal, lemmatization, tokenization and lowercasing are to improving the caliber of sentiment analysis findings.

1.4.3 Performance Evaluation:

By evaluating the performance of different sentiment analysis models, such as Naive Bayes and Bidirectional Long Short-Term Memory (Bi-LSTM) networks, the project offers valuable insights into their strengths and limitations. It helps researchers and practitioners make informed decisions when selecting appropriate models for sentiment analysis tasks.

1.4.4 Dataset Quality Assurance:

Through rigorous data preprocessing techniques, the project ensures the quality of the Twitter dataset used for sentiment analysis. By addressing common challenges such as noise, misspellings, and variations in text formats, the project enhances the reliability and accuracy of sentiment analysis results.

1.4.5 Practical Applications:

The project's findings have practical implications for various stakeholders, including businesses, marketers, policymakers, and researchers. By understanding the sentiments expressed on Twitter, stakeholders can make informed decisions related to brand management, marketing strategies, public opinion monitoring, and trend analysis.

1.4.6 Findings for Future Research:

The project pinpoints areas in which sentiment analysis methods for social media data need to be improved and investigated further. It offers directions for investigating new ideas, improving model performance, and tackling problems like context-dependent sentiment analysis and sarcasm detection.

2 Problem Statement

Even with the volume of textual data produced on twitter, it is still difficult to label the sentiments that are communicated in this data. The challenge is to efficiently extract and analyze the ideas and feelings that are hidden beneath the massive amounts of unstructured text data. These insights are important for a variety of applications, including public opinion research, brand monitoring, and market research.

This Sentiment analysis is further complicated by the informal and loud character of social media material, which may includes slang, misspellings, abbreviations, and context-dependent language. The current state of sentiment analysis approaches frequently produces not soo accurate performance and untrustworthy results because of its inability to reliably analyze and categorize such a wide range of textual information.

As a result, the main focus is to create sentiment analysis models and strong NLP approaches that are especially designed to handle the complexity of Twitter data. Improving the precision and dependability of sentiment analysis results entails tackling problems with data preparation, feature extraction, sentiment classification, and model evaluation.

The main objective is to create efficient processes and algorithms that can accurately and automatically categorize the attitudes conveyed in tweets as positive, negative, or neutral. By doing this, we hope to offer insightful information about trends, public opinion, and emotions expressed on Twitter, facilitating informed decision-making for a range of stakeholders in diverse fields. In conclusion, this problem statement invloves understanding the issues related to the tweet sentiment analysis and then implmenting a model which will provide an accurate results.

3 Methodology

3.1 Workflow Diagram

3.1.1 Data Collection:

Data Collection: We started the project by downloading the dataset from the kaggle. The dataset we had consists of balled tweets where the given tweets were label depending on their sentimnet such as- positive, negatuve or neutral. We have two csv files one is train and other is test.

3.1.2 Data Pre-processing:

Next step was data pre-processing. We cleaned our dataset and started preparing for the sentiment analysis. Then we utilized the nltk library and performed tokenization, lowercasing , stopword removal, lemmatization,and then handling the missing values.

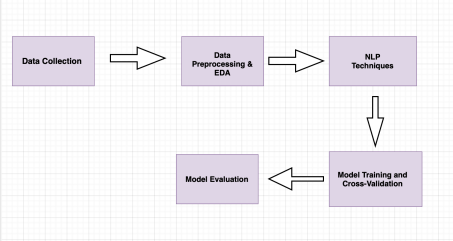

Figure 1: Workflow Diagram

3.1.3 Exploratory Data Analysis (EDA):

The purpose of is to learn more about the properties and distribution of the Twitter dataset. To investigate the sentiment distribution, word frequency, tweet durations, and other pertinent properties of the dataset, we had performed variety of statistical analysis, visualizations, and summary statistics.

3.1.4 Vectorization and Feature Extraction:

Feature extraction involves turning the preprocessed text input into machine learning model-suitable numerical feature vectors. We have used The Bag-of-Words method to extract features. Vectorization: Next the preprocessed text input is transformed into a matrix of token counts by using the scikit-learn CountVectorizer class. The frequency of each term in the text corpus is shown in this matrix.

3.1.5 Model Training and Evaluation:

Naive Bayes: We have decided to implement the naive bayes model for this project. From the countvectorizer we have got the feature vectors which we used to train our classifier. Then we performed the cross-validation to test the model's performance. we made some calculations to determine the classification measures which includes-accracy, precision, recall, F1-score.

3.1.6 Recurrent Neural Network (RNN) and Bidirectional LSTM Models:

We used the TensorFlow Keras API to create two deep learning models: a Bidirectional LSTM and a Simple RNN. These models are then assessed on a validation set after being trained on the feature vectors that we acquired by using the CountVectorizer. Then we followed the Steps like defining the model's architecture, model compiling then fitting this model to the training set of data and hen evaluating the performance matrics .

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_3 (Embedding)     (None, None, 64)          64000

 simple_rnn_4 (SimpleRNN)    (None, None, 64)          8256

 dropout_6 (Dropout)         (None, None, 64)          0

 simple_rnn_5 (SimpleRNN)    (None, 64)                8256

 dropout_7 (Dropout)         (None, 64)                0

 dense_3 (Dense)             (None, 3)                 195

=================================================================
Total params: 80707 (315.26 KB)
Trainable params: 80707 (315.26 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 2: RNN Architecture Model

3.2 Models- RNN Model Architecture

- **Embedding layer (embedding 3)**: This layer transforms each word in a tweet into a dense vector representation. This allows the model to process words with similar meanings in a similar way.
- **SimpleRNN layer (simple rnn 4):** This is the first recurrent layer. It processes the sequence of words in a tweet one by one. At each step, it takes the current word vector and the hidden state from the previous step as input. The hidden state captures information about the preceding words in the sequence. The SimpleRNN layer then updates the hidden state and outputs a new hidden state vector.
- **Dropout layer (dropout 6):** During training, this layer arbitrarily removes a certain number of neural network units. By doing this, the model is kept from overfitting the training set.
- **SimpleRNN layer (simple rnn 5:** The second SimpleRNN layer. It is similar to first SimpleRNN layer, but instead of using word vectors as input, it uses the output of that layer.
- **Dropout layer (dropout 7):** This layer performs the same functions as the one that came before it.
- **Dense layer (dense 3):** The output of the second SimpleRNN layer is mapped to a three-dimensional vector in this last layer. The three sentiment classes—positive, negative, and neutral—are probably represented by the three dimensions.

3.3 Model- Bi-LSTM Architecture:

- **Embedding layer (embedding 5):** This layer converts each word in a tweet into a

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_5 (Embedding)     (None, None, 64)          64000

 bidirectional_2 (Bidirecti  (None, None, 128)         66048
 onal)

 dropout_10 (Dropout)        (None, None, 128)         0

 bidirectional_3 (Bidirecti  (None, 128)               98816
 onal)

 dropout_11 (Dropout)        (None, 128)               0

 dense_5 (Dense)             (None, 3)                 387

=================================================================
Total params: 229251 (895.51 KB)
Trainable params: 229251 (895.51 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 3: Bi-LSTM Architecture Model

dense vector representation.

- **LSTM layer that is bidirectional (bidirectional 2):** The key element of the Bi-LSTM architecture is this layer. It is made up of two LSTMs, one of which processes a tweet's word sequence from left to right (ahead) and the other from right to left (reverse). As a result, the Bi-LSTM can learn a word's context from both its terms that come before and after it. Then, a new vector representation that combines data from both directions is created by concatenating the outputs of the two LSTMs.
- **Dropout 10:** In order to avoid overfitting, this dropout layer (dropout 10) randomly removes a certain number of neural network units during training.
- **Bidirectional 3:** The second Bi-LSTM layer is called the bidirectional LSTM layer (bidirectional 3). It works in a similar way as the first Bi-LSTM layer, but instead of using word vectors as input, it uses the first Bi-LSTM layer's output.
- **Dropout layer (dropout 11):** This layer works in the same way as the one before it.
- **Dense layer (dense 5):** The data from the second Bi-LSTM layer is mapped to a three-dimensional vector by this last layer, which is most likely to represent the three sentiment classes—positive, negative, and neutral.

4 Dataset

4.1 Test Dataset

- Size: the size of our test dataset is 3534.
- Type: CSV data in tabular form.
- Sources: We have taken the dataset from kaggle.

Figure 4: Test Dataset

Figure 5: Train Dataset

- columns: textID, text, sentiment.

Lets discuss about what is present in each column of text dataset textID: It is Unique identifier for each tweet. text: This contains the actual text of the tweet. sentiment: For each sentiment a label is given here such as positive, negative, or neutral.

4.2 Train Dataset:

- Size: The size of our train dataset is 27481.
- Type: CSV data in tabular form.
- Sources: We have taken this dataset from kaggle.
- columns: textID, Selected text, text, sentiment.

Lets discuss about what is present in each column of train dataset. textID: It is Unique identifier for each tweet. text: This contains the actual text of the tweet. selected text: This contains a portion of the text that represents the tweet's sentiment sentiment: For each sentiment a label is given here such as positive, negative, or neutral.

5 Exploratory Data Analysis

We plotted few graphs to have a clear understanding of the dataset. First we plotted was the bar graph to understand how neutral, negative, and positive sentiments are distributed . The graph's x-axis displays the sentiment, and the y-axis displays the number of tweets.

With a count of 8000, the negative sentiment is the most common. With a count of 6000, positive sentiment is the second most common sentiment. At a count of 4000, neutral sentiment is the least common sentiment.

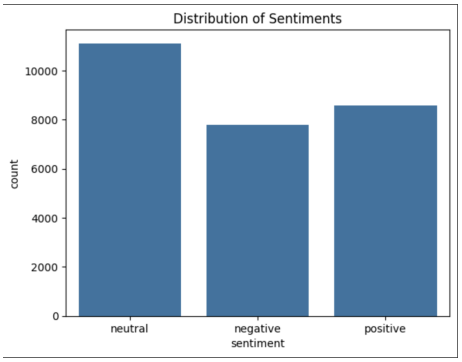

Figure 6: Distribution of sentiment

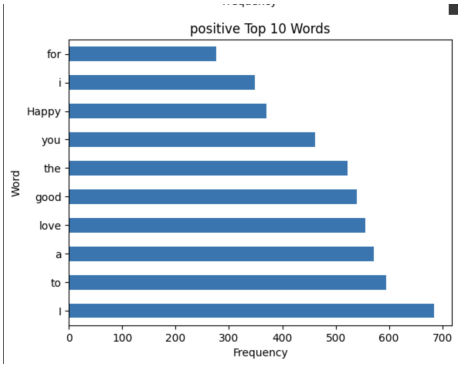

Figure 8: Positive Top 10 Words

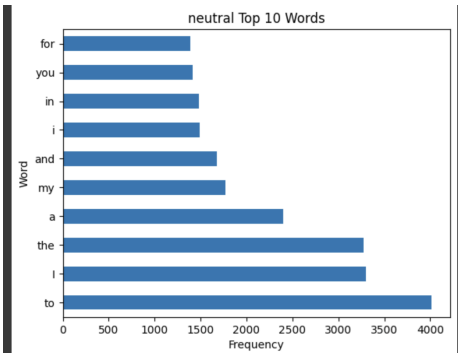

Figure 7: Neutral Top 10 Words

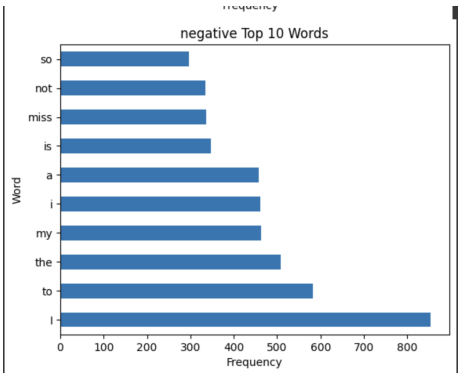

Figure 9: Negative Top 10 Words

Then we performed the word frequency analysis and plotted the graph. We defined a function here that will accept the sentiment and n as parameter then the sentiment we wish to analyze can be refered as sentiment. We plotted bar graph for "neutral , positive and negative sentiment separately We have taken the word on the x-axis, while the frequency on the y-axis and gave the title as "Neutral Top 10 Words", "Positive Top 10 Words" and "Negative Top 10 Words". Neutral sentiment: Positive and negative terms are probably excluded here because the dataset is neutral in sentiment. Here the main goal is to identify which neutral terms are used most frequently in the text.

With a frequency of almost 4000, the word "the" is the most used. Commonly used neutral words include "in," "a," "to," "and," "for," "my," "i," and "you," with frequencies ranging from 500 to 1500.

Observations for positive sentiment: Phrases that convey happiness, joy, or contentment are considered positive sentiments. Good, loving, and cheerful are a few examples of positive phrases.

With a frequency of about 700, "happy" is the most often occurring word. Among the other often used positive words are "good," "love," "you," "a," "to," "I," "thank," "great," and "really." Their frequencies range from 100 to 500.

Observations for Negative Sentiment: Words that convey anger, rage are considered to be negative sentiment. Words that are negative include "bad," "sad," and "hate."

Among all words, "not" appears the most frequently (around 800 times). It is significant to remember that the word "not" can be employed in both positive and neutral phrases as well as negative ones. It's hard to pinpoint exactly why "not" is the most used word in this circumstance. With a frequency of almost 700, "no" is the second most frequent word. "No" is a blatant sign of negativity and negation. The negative terms "never", "bad", "don", "hate", "terrible", "want", "really", and "going" are also often used, with frequencies varying between 100 and 500.

The next graph we plotted is the histogram to visualize distribution of text lengths by each category present in the dataset. We have taken the text length to display on the x-axis, while the frequency of texts of that length to display on the y-axis. The KDE line aids in histogram smoothing and improves understanding of the data's general distribution.

We have observed how the distribution of text lengths varies throughout the various sentiment groups by examining the histogram. Positive tweets, for instance, may appear to be shorter

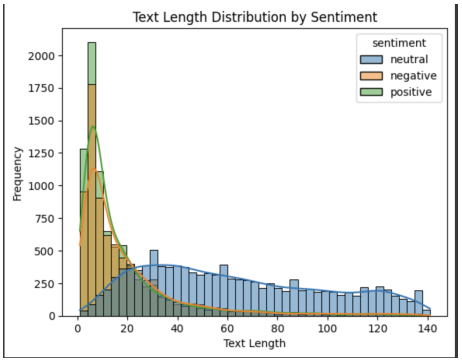

Figure 10: Text Length Distribution By Sentiment

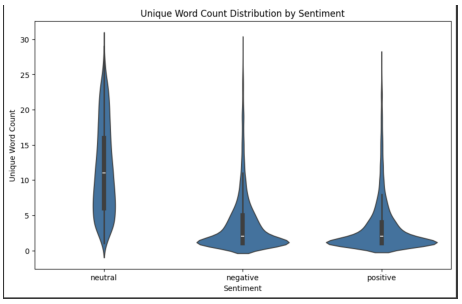

Figure 11: Unique Word Count Distribution by Sentiment

than negative ones. When compared to positive and negative tweets, we can see that neutral texts are more common and typically have shorter lengths. Additionally, the peak of the distribution for neutral mood is located lower on the x-axis, indicating that the most common neutral texts are also shorter. The text distributions for positive and negative sentiment seem to be wider than the distribution for neutral sentiment. This indicates that in comparison to neutral tweets, positive and negative tweets typically have a broader variety of lengths. Positive tweets tend to be shorter than negative tweets: The positive sentiment distribution trails off more quickly for tweets longer than 100 characters, while the negative sentiment distribution has a longer tail, indicating a higher proportion of longer negative tweets. Both the positive and negative sentiment distributions peak at around 50 characters.

Next we have plotted the Violin Plot to get the unqiue word present in each sentiment. The graph shows the distribution of unique word counts by sentiment. The y-axis displays the quantity of unique words, while the x-axis displays sentiment (positive, negative, and neutral).

We gave the title as "Unique Word Count Distribution by Sentiment". We can see that every bar on the graph denotes the quantity of distinct

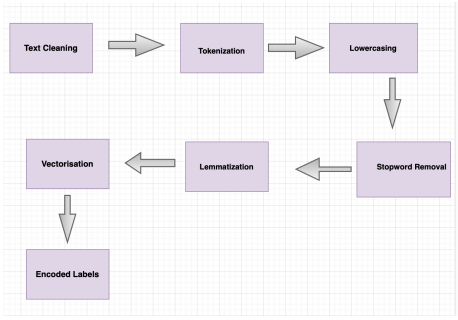

Figure 12: Data Pre-Processing Workflow

terms that were identified as belonging to a specific sentiment in the text. It seems that neutral sentiment has the most unique terms, followed by positive and negative sentiment.

5.1 Data Preprocessing

- **Text Cleaning:** The first step was cleaning our dataset. As our dataset consits of tweets and the selected texts so we removed the noise and irrelevant information. We checked if our dataset consits of any NaN values or any special characters which is not required for us we removed it.
- **Tokenization:** We then performed the tokenization by breaking the text into tokens. The text data is processed at the word level which enable feature extraction.
- **Lowercasing:** Next we converted that tokens into lowercase for consistency. This will only increase the sentiment analysis accuracy by preventing the model from treating words with various cases (like "Good" and "good") as distinct entities.
- **Stopword Removal:** We then removed all the stopwords like- "the", "is" etc from the tokenized text. This we did to reduce the noise present in our data. These stopwords can be safely removed because they have no semantic significance.
- **Lemmatization :** Next we performed lemmatization - words are lemmatized that means it is reduced to their base form. We did this to standardize the text data and decrease the dimensionality of the feature space.
- **Handling Missing Values:** We have taken care of the missing values. We checked if the rows contains any missing values- if present then we planned to remove it or substitute it with the placeholder tokens.

- **Vectorization:** Next we converted the pre-processed text input into numerical feature vectors that may be used with machine learning models.By using the Bag-of-Words technique, each document (tweet) in this project is represented as a vector of word counts or frequencies.
- **Encoded Labels**: To make training the model easier, the categorical sentiment labels—positive, negative, and neutral—are encoded into numerical values. This entails employing label encoding techniques to transfer each sentiment label to a distinct integer.

6 Implementation

Algorithm for Data Pre-Processing: for each tweet in the dataset:

- Make sure the tweet is clean by eliminating special characters, mentions, hashtags, and URLs.
- Tokenize the cleaned text in order to extract certain words.(Tokenization)
- To maintain consistency, convert tokens to lowercase.
- Take stopwords out of the tokenized content.(Stopword Removal)
- Reduce words to their simplest form.(Lemmatization)
- Deal with missing values correctly.

Algorithm/Pseudocode

- Data Collection- collect the twitter data from the Kaggle.
- Data Pre-Processing: Clean the data by using NlP Techniques.
- Train and Build Model: Train the data by splitting into train and test set. Then start building the model.
- Naïve Bayes Model: Train a Naïve bayes classifier and tune hyperparameters using cross validation score and do evaluation.
- RNN Model: Train and build the RNN Model and do evaluation
- Bi-LSTM: Train and build the Bi-LSTM Model and do evaluation
- Do Comparison and Visulaization of results.

6.1 NLP Techniques

- Text Tokenization: We have tokenized the twitter texts by using NLP techniques to separate them into individual words or tokens for examination.
- Stopword Removal: We then used algorithms to filter out frequently occurring stopwords from tweets so that the remaining words are more relevant for sentiment analysis.
- Lemmatization : Utilised lemmatization technique to reduce words to their most basic forms. This reduces duplication and enhances sentiment analysis's effectiveness.

6.2 Libraries used

- **Numpy**: It is a potent Python package. It can support Large, multi-dimensional arrays and matrices and we can perform mathematical operations on these arrays.
- **Pandas**: This is mostly used library for data analysis and manipulation. As It offers extremely effective data structures for working with structured data, such as DataFrame.
- **Maplotlib and Seaborn:**These wo Python libraries we used for the data visualization purpose.As they include several features that will allow us to make different kinds of plots, charts, and graphs to see data and draw conclusions from it.
- **Transformers:** Hugging Face offers Transformers as an open-source NLP library for us to use. It provides tools for optimizing these models on unique datasets in addition to pre-trained models for a range of NLP tasks.
- **Torchinfo :** This is used to show details about PyTorch models which can be including the quantity of parameters and the final form of each layer.
- **Scikit-learn:** It is a Python machine learning package with easy-to-use capabilities for data mining and analysis purpose.
- **NLTK:** It is a top platform for developing Python programs that will help us interact with data in human languages. In addition to a collection of text processing tools for tokenization, stemming, tagging, parsing, and other uses, it also offers user-friendly interfaces to more than 50 corpora and lexical resources, including WordNet.

- **Kearas and TensorFlow**: It is open-source machine learning framework TensorFlow. It is Written in Python and Keras is a high-level neural network API that can be used with TensorFlow. We have used it for neural network construction and training interface.

6.3 Data Collection and Processing:

We gathered a dataset made up of Twitter data together with sentiment labels. Tokenization, stopword removal, handling missing values, and lemmatization were among the preprocessing techniques we used to clean up the text data. To train and evaluate the model, the dataset was divided into training and testing sets.

6.4 Model Selection and Tuning:

We have investigated three distinct sentiment analysis models: RNN (Recurrent Neural Network), Bi-LSTM (Bidirectional Long Short-Term Memory), and Naive Bayes. We trained a Multinomial Naive Bayes classifier using CountVectorizer, which we utilized for vectorization in Naive Bayes. Using Keras with TensorFlow as the backend, RNN and Bi-LSTM models were trained. Sequences of integers were created from text input using tokenization and padding. For RNN and Bi-LSTM models, the model architectures consisted of embedding layers followed by LSTM layers. Overfitting was avoided with the use of dropout regularization.

6.5 Building RNN Model

We have implemented the RNN Model by utlizing keras. We have imported the required keras packages and imported libraries like- sequential, Embedding, LSTM and BI-LSTM We performed tokenization and then converted the text data to sequence of integers of the processed text. Then we used padding and padded the sequences and then converted the sentiment labels to categorial values. Then we trained RNN Model and Build our model. We compiled our model by considering the optimizer4 as adam and metrics to be accuracy. We trained our model by taking 10 epochs and printed the epoch with accuracy and validation accuracy.

6.6 BI-LSTM Model Implementation

We have implemented the BI-LSTM Model by utlizing keras. We have imported the required

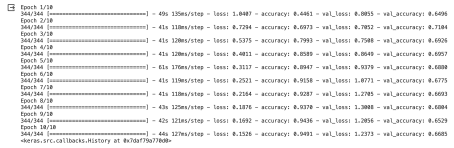

Figure 13: RNN Model implenmentation

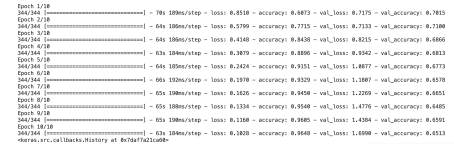

Figure 14: BI-LSTM Model implenmentation

keras packages and imported libraries like- sequential, Embedding, LSTM and BI-LSTM We performed tokenization and then converted the text data to sequence of integers of the processed text. Then we used padding and padded the sequences and then converted the sentiment labels to categorial values. Then we trained bi-lstm Model and Build our model. We compiled our model by considering the optimizer4 as adam and metrics to be accuracy. We trained our model by taking 10 epochs and printed the epoch with accuracy and validation accuracy.

6.7 Model Evaluation

The Naive Bayes model underwent cross-validation, and accuracy scores were calculated at various folds. To evaluate the performance of the model, cross-validation score visualizations were offered. To monitor model performance across epochs, training and validation accuracy charts were created for the RNN and Bi-LSTM models. All models' classification reports, which included the F1-score, recall, and precision for every sentiment class, were displayed.

7 Results

7.1 BI-LSTM Model Implementation

The CV Scores of Naive Bayes graph: The y-axis shows accuracy, while the graph's x-axis shows folds. The accuracy score will evaluates a model's performance on new unseen data. In this instance, the accuracy of the Naive Bayes classifier is approximately 0.66 for every fold. The RNN and BI-LSTM training and validation accuracy is shown in the graph we plotted. The training accuracy is shown by the blue line, while the validation accuracy is shown by the red line.

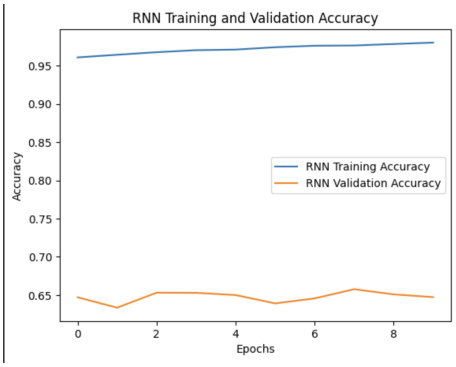

Figure 15: RNN Training and Validation Accuracy

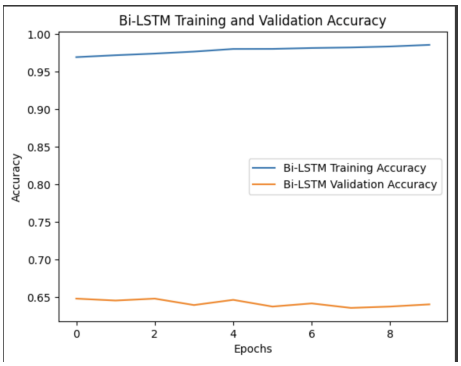

Figure 16: bI-LSTM Training and Validation Accuracy

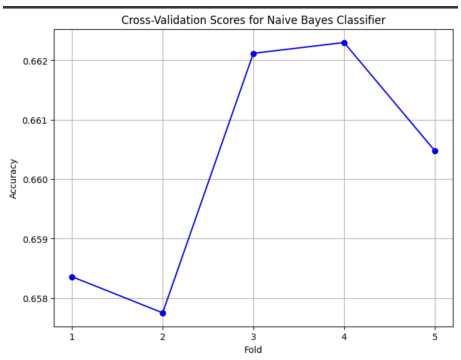

Figure 17: CV Scores of Naive Bayes.png

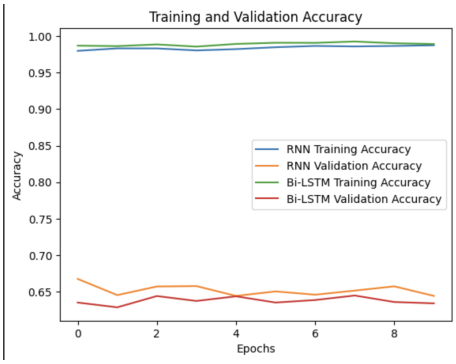

Figure 18: Training and Validation Accuracy

Training Accuracy is a measure of the model's performance on the training set. We have seen that as the number of epochs increases on the graph, the training accuracy also rises, indicating that the model is picking up on the patterns in the training data.

Here Validation Accuracy will measures the model's performance on new or unseen data. We can see that although the validation accuracy in this graph is lower than the training accuracy, the trend is still the same. This suggests that the model may, but is not guaranteed to, generalize effectively to unknown data.

The difference we found in the accuracy between training and validation data is becauuse of overfitting..

7.2 BI-LSTM Model Implementation

Performance of Naive Bayes Model:

- Accuracy: 0.69
- Precision (avg): 0.73
- Recall (avg): 0.60
- F1-score (avg): 0.66

With an accuracy of 69%, Naive Bayes was the model with the highest accuracy. Across the classes, it has also attained a balanced F1-score, recall, and precision. This suggests us that the Naive Bayes model classified the sentiment of the Twitter data across all categories with a respectable level of accuracy.

Performance of RNN and Bi-LSTM Models: RNN

- Training Accuracy:
- Validation Accuracy:0.65
- Precision (avg): 0.69
- Recall (avg): 0.70
- F1-score (avg): 0.70

Bi-LSTM

- Accuracy: 0.63
- Precision (avg): 0.68
- Recall (avg): 0.69
- F1-score (avg): 0.68

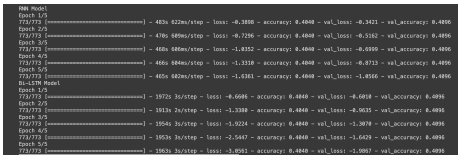

Figure 19: Results of RNN and Bi-LSTM

Metric	Naive Bayes	RNN	Bi-LSTM
Accuracy	0.69	0.64	0.63
Precision			
Class 0	0.73	0.69	0.68
Class 1	0.63	0.61	0.60
Class 2	0.76	0.63	0.62
Recall			
Class 0	0.60	0.70	0.69
Class 1	0.76	0.65	0.65
Class 2	0.69	0.60	0.58
F1-score			
Class 0	0.66	0.70	0.68
Class 1	0.69	0.63	0.62
Class 2	0.72	0.61	0.60

Figure 20: Performance Comparison

7.3 Uniquesness

- Our approach is distinctive because it thoroughly integrates a range of NLP techniques with machine learning and deep learning models to conduct sentiment analysis on Twitter data.
- Preprocessing methods like tokenization, stopword elimination, and lemmatization worked together to ensure that the text data is cleaned and standardized, which is essential for precise sentiment analysis.
- Performance and efficacy in sentiment classification can be compared thanks to the use of both deep learning models (RNN and Bi-LSTM) and classical machine learning algorithms (Naive Bayes).
- The application shows how adaptable and powerful natural language processing (NLP) approaches are for handling and interpreting textual data, especially when applied to sentiment analysis on social media.

7.4 Comparison

The figure is the comparison table of the model performance.

Naive Bayes:

- Accuracy: The model's we got is 69% accuracy that means roughly 69% of the predictions provided by the Naive Bayes classifier were accurate.
- Precision:The value is varied between 63% and 76% for various classes. This implies that between 63% and 76% of the time, the model's predictions for a class were accurate.
- Recall: Recall values we got for the Naive Bayes model ranged from 60% to 76%, meaning that it could correctly identify 60% to 76% of the examples in each class.

RNN

- Accuracy: With an accuracy of 64%, the RNN model was able to correctly predict roughly 64% of its observations.
- Precision: Across several classes, precision varied between 61% and 69%. This indicates that between 61% and 69% of the time, the RNN model's predictions about classes were accurate.
- Recall: The RNN model was able to properly identify between 60% and 70% of the examples of each class, as indicated by recall values ranging from 60% to 70%.
- F1-score: The F1-scores showed a precision-recall balance similar to the Naive Bayes model, ranging from 61% to 70%.

BI-LSTM

- Accuracy: With an accuracy of 63%, the Bi-LSTM model was able to correctly predict roughly 63% of the observations.
- Precision: Depending on the given class, precision varied from 60% to 68%. This indicates that between 60% and 68% of the time, the Bi-LSTM model's predictions about classes were accurate.
- Recall: The Bi-LSTM model was able to properly identify between 58% and 69% of the examples of each class, as indicated by recall values ranging from 58% to 69%.
- F1-score: F1-scores varied from 60% to 68%, suggesting a recall and accuracy balance akin to that of the RNN and Naive Bayes models.

Overall Comparison In conclusion, all three models performed similarly, however the RNN and Bi-LSTM models performed much similarly, and the Naive Bayes model generally showed us slightly higher accuracy and precision across classes. The reason could be Overfitting and our dataset is not a large dataset. Naive Bayes showed little good performance than other two models.

8 Project Management

8.1 Work completed

- Description: Almost all of the tasks that we had planned to complete have been completed. This includes Performing EDA, pre processing our dataset, Using NLP technique.
- Responsibility (Task, Person):
  - Background Research by Bhanuteja Damera and Yeshwant Pulapa
  - Bhanuteja Damera and Yeshwanth Pulapa performed Data Preprocessing
  - Priyanka Bai Durvas, and Yeshwannth Pulapa are in charge of Performing NLP Techniques.
  - Statistical Analysis by Priyanka Bai Durvas, Bhanuteja Dame, and Yeshwanth Pulapa
  - Bhanuteja Damera, Yeshwanth Pulapa, and Priyanka Bai Durvas are involved in code and tech.
- Contributions (members/percentage)
  - Bhanuteja Damera / 33.33%
  - Priyanka Bai Durvas / 33.33%
  - Yeshwanth Pulapa / 33.33%

References

[1] P. (n.d.). GitHub - priyankadurvas/Twitter-Sentiment-Analysis. GitHub. https://github.com/priyankadurvas/Twitter-Sentiment-Analysis

[2] T. (2020, June 12). Twitter Challenge: roBERTa sentiment predictor. Kaggle. https://www.kaggle.com/code/tarunpaparaju/twitter-challenge-roberta-sentiment-predictor/input

[3] G. (2024, January 2). Python Lemmatization with NLTK. GeeksforGeeks. https://www.geeksforgeeks.org/python-lemmatization-with-nltk/