# Mini Project 2- Sentiment Analysis using the Sentiment140 Dataset

Alaina Faisal

## 1 Introduction

Sentiment analysis, is a vital task in the field of natural language processing (NLP) and machine learning, aiming to determine the emotional tone behind a body of text. The Sentiment140 dataset, with its 1.6 million tweets annotated for sentiment, presents a unique opportunity to develop and refine sentiment analysis models. Tweets, due to their brief and often informal nature, encapsulate a wide range of human emotions, making them an ideal candidate for sentiment analysis. The challenge, however, lied in the context-dependent nature of language used in tweets, which included slang, abbreviations, and emoticons, adding layers of complexity to the task.

The sentiment analysis project undertaken with the Sentiment140 dataset involved several critical steps, starting from data preprocessing to model application and evaluation. Preprocessing involved cleaning the data and preparing it in a format suitable for analysis, a step crucial for any NLP task due to the inherent irregularities and noise in raw text data. The project explored various machine learning models, from traditional algorithms like Naive Bayes and Logistic Regression to more advanced deep learning techniques such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. The deep learning models, particularly LSTMs, stood out for their ability to understand the context and sequence of words, reflecting the importance of capturing temporal dependencies in textual data for sentiment analysis.

## 2 Data Preprocessing

The data processing phase was crucial for preparing the Sentiment140 dataset for sentiment analysis. The dataset had a balanced dataset as shown in Figure 1.

The daily trend of sentiment shown in Figure 2 also showed a similarity between both trends each day.

**Data Cleaning and Normalization**

Initially, the dataset underwent a thorough cleaning process to remove noise and standardize the text, enhancing the quality of the input data for subsequent analysis:

- **Special Characters Removal**: Non-alphanumeric characters, including emojis, hashtags, and user mentions (denoted by @username in tweets), were identified and eliminated from the text. This step was accomplished using regular expressions, which allowed for precise pattern matching and removal of these elements, thereby reducing the noise in the dataset.
- **Lowercasing**: All text in the dataset was converted to lowercase. This normalization step was critical for ensuring consistency across the data by eliminating variations in capitalization.

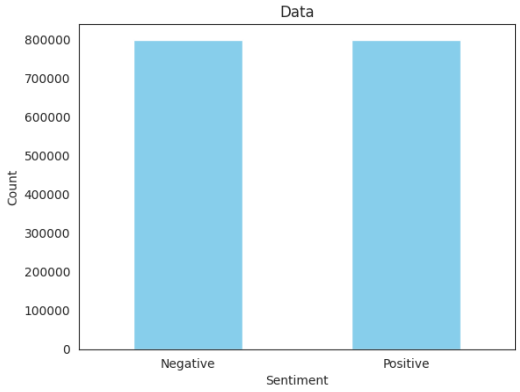


Figure 1: Data Distribution

**Tokenization**

The text of each tweet was then tokenized, breaking it down into individual words or tokens. This process transformed the unstructured text into a sequence of tokens, making it amenable to further processing and analysis.

**Stopword Removal**

Following tokenization, stopwords were removed from the token sequences. Stopwords, which include common words such as "the," "is," and "and," were filtered out due to their high frequency and low informational value in the context of sentiment analysis. This step was essential for focusing the model's attention on words with greater potential to

convey sentiment, thereby improving the efficiency and effectiveness of the analysis.

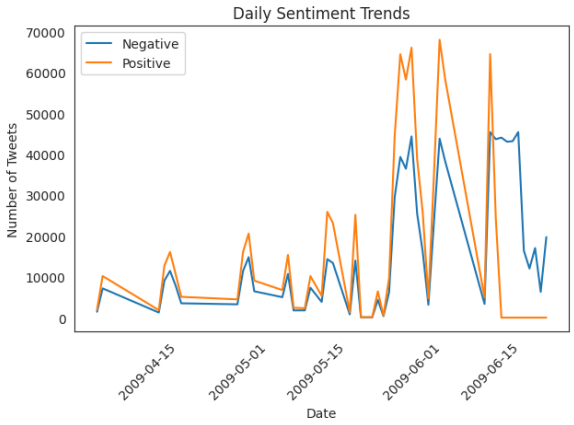


Figure 2: Trend of sentiments

**Stemming/Lemmatization**

The preprocessing pipeline also included a step for stemming or lemmatization, aimed at reducing words to their base or root forms. This process helped in consolidating different morphological variants of a word, ensuring that they are recognized as the same token by the model.

**Feature Extraction**

Finally, the preprocessed text data was transformed into numerical features, making it suitable for input into machine learning models. This transformation involved the use of vectorization technique Term Frequency-Inverse Document Frequency (TF-IDF), which convert text into a matrix of token frequencies or weighted frequencies, respectively. These feature extraction methods are pivotal for translating textual data into a format that can be effectively analyzed by statistical models.

## 3 Models

The sentiment analysis project employed a combination of traditional machine learning algorithms and advanced deep learning models to classify tweets from the Sentiment140 dataset into positive or negative sentiments. This section outlines these methods in detail, discussing their implementation and contribution to the project:

### Naive Bayes Classifier

The Naive Bayes classifier was implemented as one of the initial models for sentiment analysis. This probabilistic model, known for its simplicity and efficiency, assumes independence among the features. In the context of text classification, it calculates the probability of each word given the sentiment and uses these probabilities to make predictions.

Naive Bayes was chosen for its effectiveness in dealing with large datasets and its strong performance in basic text classification tasks, making it a suitable baseline model for sentiment analysis.

**Logistic Regression**

Logistic Regression was employed as a more sophisticated linear model that predicts the probability of a tweet belonging to a particular sentiment category (positive or negative). The model was trained using the vectorized text data, where each tweet is represented as a feature vector**.**

The choice of Logistic Regression was motivated by its ability to handle binary classification problems and provide interpretable results in terms of odds ratios, making it a valuable tool for understanding the impact of individual words on sentiment.

**Convolutional Neural Networks (CNNs)**

A Convolutional Neural Network (CNN) architecture was adapted for text processing by applying convolutional layers to the embedded word representations. The model would use filters to capture local patterns within sequences of words, pooling layers to reduce dimensionality, and fully connected layers for classification.

CNNs were chosen for their ability to capture local dependencies and significant patterns in text data, such as n-grams, which are crucial for understanding the sentiment expressed in tweets.

LSTMs, were implemented to take advantage of their sequential data processing capabilities. LSTMs are designed to address the vanishing gradient problem of standard RNNs, allowing them to capture long-range dependencies in text data effectively.

The use of LSTMs was driven by the need to model the temporal relationships between words in tweets, which is essential for understanding the context and sentiment of the text. Their ability to remember information over long sequences makes them well-suited for sentiment analysis tasks.

## Results and Analysis

The sentiment analysis project aimed to classify tweets into positive or negative sentiments using various machine learning and deep learning models. The performance of each model was critically assessed using metrics such as accuracy, precision, recall, F1-

score, and the area under the ROC curve (AUC).

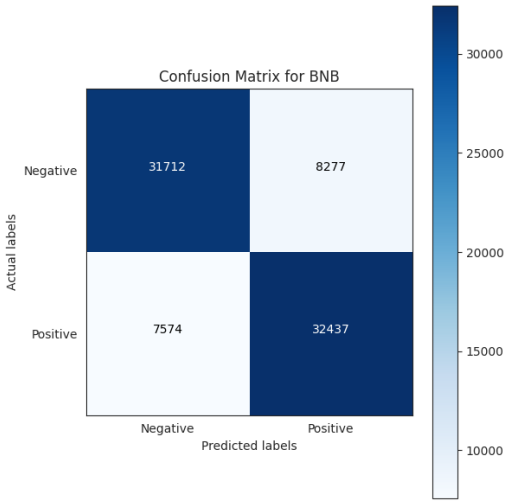


Figure 3: Confusion Matrix for NB

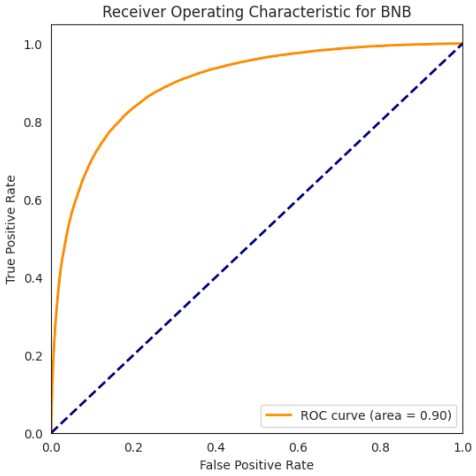


Figure 4: ROC Curve for NB

**Naive Bayes Classifier**

The Naive Bayes classifier yielded a baseline level of accuracy i.e., 80%, with moderate precision and recall scores of 79 and 80. The confusion matrix showed in Figure 3 shows these in terms of the actual and the predicted classifications.

While the Naive Bayes classifier served as a good starting point, its performance highlighted the limitations of assuming feature independence in sentiment analysis, where contextual and sequential information can significantly influence sentiment.

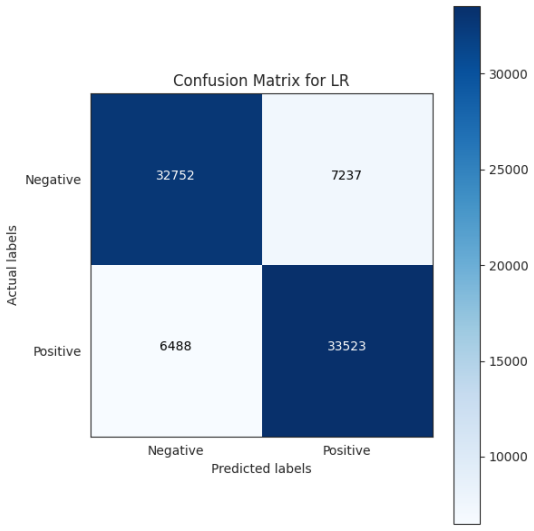


Figure 5: Confusion Matrix for LR

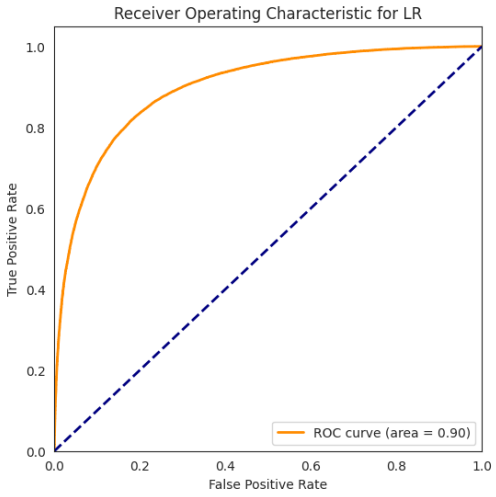


Figure 6: ROC Curve for LR

**Logistic Regression**

Logistic Regression showed an improvement over the Naive Bayes classifier in terms of accuracy and F1-score. The model's interpretability would have allowed for an analysis of feature coefficients, shedding light on the words most strongly associated with positive or negative sentiments. The results are shown in Figures 5 and 6 where ROC has value 0.90 and the results are promising.

**CNN and LSTM:**

The CNN model, adapted for text data, showed improvement in accuracy and F1-score compared to traditional machine learning models. The model's ability to capture local patterns and n-grams within the text would contribute to this enhanced performance.

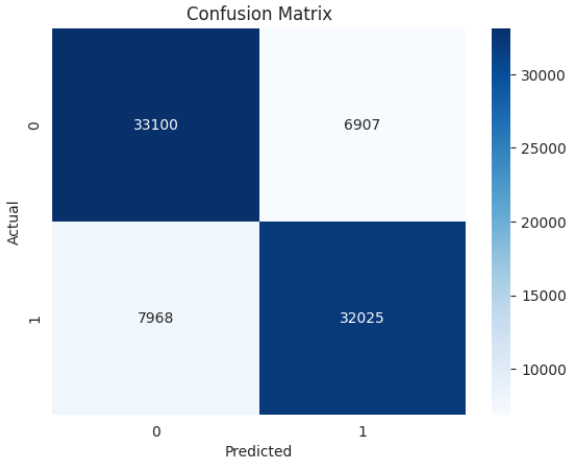


Figure 7: Confusion Matrix for CNN + LSTM

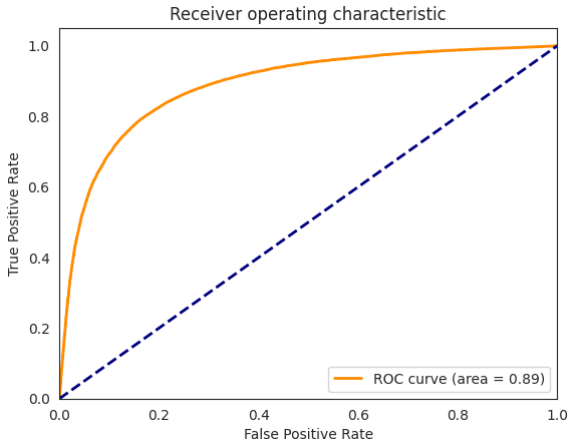


Figure 8: ROC Curve for CNN+LSTM

The success of the CNN model underscore the importance of capturing local contextual information in sentiment analysis. The results also highlight the potential trade-offs between model complexity and interpretability, with CNNs being less transparent than simpler models like Logistic Regression.

LSTM models, in particular, demonstrated strong performance, achieving good F1-scores. Their ability to model long-range dependencies in text makes them well-suited for sentiment analysis.

**Comparative Analysis and Visualization**

**Confusion Matrices**: Confusion matrices for each model provides insights into the true positive, false positive, true negative, and false negative rates, offering a more nuanced understanding of model performance beyond aggregate metrics as shown in the results in previous sections.

**ROC Curves**: The ROC curves and AUC scores are used to compare the models' ability to discriminate between positive and negative sentiments across different threshold settings, providing a comprehensive view of model performance.

**Word Clouds**: Word clouds generated from the coefficients or feature importances of models (where applicable) visually represent the words most strongly associated with positive or negative sentiments, offering an intuitive understanding of the models' decision-making processes. These are shown in Figure 9 and Figure 10.

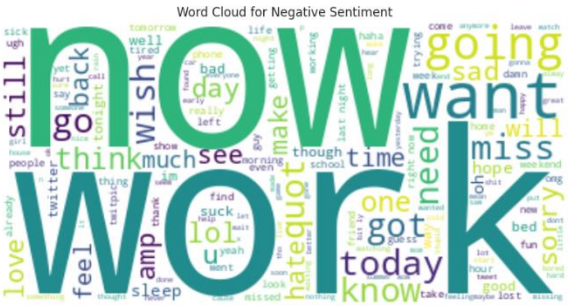


Figure 9: Word Cloud for Negative Sentiment

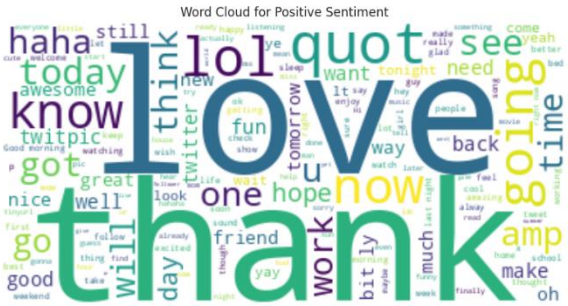


Figure 10: Word Cloud for Positive Sentiment

## 4 Conclusion

The sentiment analysis project on the Sentiment140 dataset demonstrated the effectiveness of various machine learning and deep learning models in classifying tweets by sentiment. The endeavor began with rigorous data preprocessing, including cleaning, normalization, and tokenization, which was crucial for preparing the raw tweet data for analysis. Models such as Naive Bayes, Logistic Regression, CNNs, and LSTMs were applied, with deep learning approaches.

Future research could explore further hyperparameter tuning to refine model efficacy, and the exploration of more complex models, such as Gradient Boosting or advanced neural networks, might offer improvements in accuracy and robustness. Experimenting with ensemble methods and transfer learning from pre-

trained language models could also provide new insights and enhancements in sentiment classification tasks.