Predicting the Tone of Tweets Based on Usernames

Project Report

Group Members

Lead: Rohma Talpur

Shahzaib Ali Hasnain
Suliman Rayyan Liaquat
Shaheer Abdul Rafay
Essa Sakhawat
Wajiha Wisal Malik

December 2, 2024

Contents

Abstract

This project aims to develop a predictive model capable of determining the tone of an upcoming tweet from a given username. By leveraging advanced machine learning techniques, we collected, merged, and preprocessed extensive datasets from various Twitter users to train our model. Through analyzing historical tweet data and extracting meaningful patterns, we seek to understand the dynamics of tone in social media communications. This approach holds significant applications in sentiment analysis, marketing strategies, and enhancing personalized interactions on social platforms.

1 Introduction

In today's digital landscape, social media platforms like Twitter have become pivotal in shaping public opinion, disseminating information, and fostering global communication. The tone conveyed in tweets can significantly influence audience engagement, brand perception, and the overall impact of the communication. Understanding and predicting this tone is essential for businesses, marketers, and individuals aiming to optimize their online presence and interactions.

This project focuses on predicting the tone of future tweets based on historical data associated with specific usernames. By employing machine learning algorithms and natural language processing techniques, we aim to build a model that analyzes past tweeting behavior to forecast the sentiment of upcoming tweets accurately. This predictive capability can aid in monitoring brand reputation, anticipating customer sentiment, and tailoring marketing strategies.

2 Data Collection

2.1 Data Sources and Selection Criteria

We selected a diverse set of Twitter users to ensure our dataset captures a wide range of tones, topics, and tweeting behaviors. The selection criteria included:

- **User Influence**: Accounts with substantial followers and high engagement rates.
- **Content Diversity**: Users from sectors like entertainment, politics, sports, news media, and corporations.
- **Language Consistency**: Primarily English-language tweets to maintain consistency in language processing.

2.2 Tools and Technologies

- **Tweepy**: A Python library for accessing the Twitter API, enabling automated data retrieval.
- **Python**: Used for scripting, data collection, and preliminary data analysis.
- **Twitter API**: Provided access to Twitter's streaming and RESTful APIs for retrieving historical and real-time data.

2.3 Data Retrieval Process

- **API Authentication**: Configured OAuth authentication using API keys and access tokens from the Twitter developer portal.
- **Historical Data Collection**: Used Tweepy's Cursor object to collect up to the maximum allowed number of historical tweets per user.
- **Real-Time Data Streaming**: Set up streaming listeners for certain users to capture the latest content.
- **Data Fields Collected**:
  - Tweet ID
  - User ID and username
  - Tweet text
  - Timestamp
  - Retweet count
  - Favorite count
  - Reply count
  - Language
  - Source
  - Hashtags, mentions, URLs
  - Geo-location data (if available)

2.4 Data Storage and Management

- **Database Selection**: Combined CSV files for initial storage with a SQL database for structured management.
- **Data Backup and Security**: Implemented regular backups and secured databases with access controls.

2.5 Ethical and Legal Considerations

- **Compliance with Twitter Policies**: Adhered to Twitter's Developer Agreement and Policy.
- **User Privacy**: Collected only publicly available data, avoiding private user information.

3 Data Merging

After collecting data from various sources, we needed to merge these datasets into a single, cohesive dataset for analysis. This process involved careful handling to maintain data integrity and consistency.

3.1 Merging Methodology

We utilized Python's pandas library for merging the datasets. The following steps were taken:

- **File Listing**: Created a list of all CSV files containing tweet data to be merged.
- **Dataframe Creation**: Read each CSV file into a pandas dataframe, handling encoding errors and skipping bad lines to prevent read failures.
- **Concatenation**: Used pandas' `concat` function to merge all dataframes into a single dataframe, ignoring index to prevent conflicts.
- **Data Export**: Saved the merged dataframe into a new CSV file for further processing.

3.2 Code Implementation

The code snippet below demonstrates the merging process:

```
# Import required libraries
import pandas as pd
import os
from google.colab import drive

# Mount Google Drive
drive.mount('/content/drive')

# List all the CSV files to be merged
file_paths = [
    '/content/messi_tweets.csv',
    '/content/tweets.csv',
    '/content/meghanmarkle tweets.csv',
    '/content/kanye west tweets.csv',
    '/content/SannaMarin.csv',
    '/content/ImranKhanPTI.csv',
    '/content/JoeBidenTweets.csv',
    '/content/elon_musk_tweets.csv',
    '/content/bolsonaro_tweets.csv',
    '/content/benhamner.csv',
    '/content/driltweets.csv',
    '/content/cr_tweets.csv'
]

# Create an empty list to store dataframes
dataframes = []

# Read and append each dataframe to the list
for file in file_paths:
    try:
        df = pd.read_csv(file, encoding='utf-8', on_bad_lines='skip')  # Handling encoding errors
        dataframes.append(df)
    except Exception as e:
        print(f"Error reading {file}: {e}")

# Merge all dataframes into a single dataframe
if dataframes:
    merged_df = pd.concat(dataframes, ignore_index=True)
```

```
39
40      # Save the merged dataframe into a single CSV file
41      output_path = '/content/merged_tweets.csv'
42      merged_df.to_csv(output_path, index=False, encoding='utf-8')
43
44      print(f"All files merged successfully into '{output_path}'")
45  else:
46      print("No dataframes to merge.")
```

Listing 1: Data Merging Code

3.3 Explanation of the Code

- **Library Imports**: Imported necessary libraries, including pandas for data manipulation and `drive` from Google Colab for accessing files stored on Google Drive.
- **Mounting Google Drive**: Used `drive.mount` to access files stored on Google Drive within the Colab environment.
- **File Paths**: Defined a list of file paths for all CSV files to be merged.
- **Reading Files**: Iterated over each file path, reading the CSV files into dataframes while handling potential encoding errors and skipping lines that could cause read failures.
- **Dataframe Aggregation**: Appended each successfully read dataframe to the `dataframes` list.
- **Merging Dataframes**: Used `pd.concat` to merge all dataframes into a single dataframe, setting `ignore_index=True` to reset the index.
- **Exporting Merged Data**: Saved the merged dataframe to a new CSV file for further use.
- **Error Handling**: Included try-except blocks to catch and report any errors encountered during file reading.

3.4 Data Verification

- **Consistency Checks**: Ensured that the total number of records in the merged dataset matched the sum of records from individual datasets.
- **Duplicate Removal**: Verified and removed any duplicate tweets based on tweet IDs to maintain data integrity.
- **Integrity Confirmation**: Conducted spot checks on random samples to confirm successful merging.

4 Data Preprocessing (Updated)

Data preprocessing is a critical step to ensure the quality and consistency of the input data before it is fed into the machine learning model. This stage involves several key processes:

4.1 Text Cleaning

Initial tweet data often contains noise in the form of URLs, special characters, and numerals which are irrelevant for tone analysis. Using regular expressions, we filtered out these elements to focus solely on meaningful text content.

4.2 Tokenization and Normalization

Each tweet was broken down into individual words or tokens using NLTK's `word_tokenize`. These tokens were then normalized to lower case to avoid discrepancies that arise from case sensitivity.

4.3 Removal of Stop Words

Common words that do not contribute to sentiment analysis, such as ”and”, ”the”, ”is”, were removed from the data. This helps in focusing the analysis on more impactful words.

4.4 Vectorization

The cleaned and normalized text was converted into a numerical format through vectorization, making it suitable for machine learning processing. We used TF-IDF (Term Frequency-Inverse Document Frequency) to weigh the words according to their importance in the tweets.

4.5 Data Integration

After preprocessing, all individual datasets were merged into a single dataset ensuring that data from different sources was aligned correctly based on user IDs and timestamps.

5 Visualization of Data Flow

To better understand the processes involved from data collection to preprocessing, a flowchart is used to illustrate the workflow:

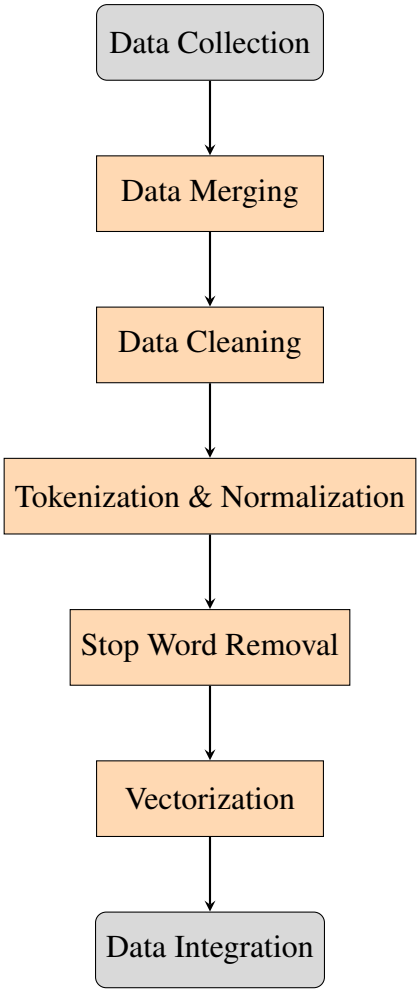

Figure 1: Data Preprocessing Workflow

6 Project Objective

6.1 Primary Objective

The primary objective of this project is to build a machine learning model capable of predicting the tone (sentiment) of a tweet based on the content of the tweet and the user's identity (username).

6.2 Secondary Objectives

- **Understanding Tone Dynamics**: To analyze how the tone of a tweet can be predicted using both the tweet's content and user-specific features (username).
- **Feature Impact Analysis**: To explore which features, such as tweet text and user data, are most influential in predicting the tone of a tweet.
- **Application Development**: To develop a tool that can predict the tone of future tweets based on a combination of text data and user-related features.

7 Model Development

7.1 Feature Selection

- **Text Data**: The tweet text is the primary input to the model. This is processed into sequences of integers, representing the words in the tweet.
- **User Data**: Each tweet is associated with a user, and the user's identity (username) is encoded numerically, providing context to the model regarding the tweet's author.
- **Tone Data**: The tone of each tweet (positive, negative, or neutral) is encoded as a target variable, which the model aims to predict.

7.2 Data Preparation for Modeling

- **Tokenization**:
  - The tweet text is tokenized using Keras' `Tokenizer`, which converts words into numerical sequences.
  - The sequences are padded to ensure consistent input size for the model.
- **User Encoding**: The user's identity (username) is encoded using `LabelEncoder`, turning the categorical usernames into numerical values.
- **Tone Encoding**: The tone (sentiment) of the tweet is encoded into numerical labels for multi-class classification.

7.3 Machine Learning Algorithms

- **Embedding Layer**: An embedding layer is used to map words in the tweet to dense vectors that represent the semantic meaning of the words.
- **LSTM Layer**: Long Short-Term Memory (LSTM) layers are used to model sequential dependencies in the tweet's text.
- **Dense Layer**: A dense layer is used to combine the features from both the tweet content and user data.
- **Softmax Activation**: A softmax output layer is applied to predict the probability distribution over the possible sentiment categories (positive, negative, or neutral).

7.4 Model Training and Hyperparameter Tuning

- **Data Splitting**: The dataset is split into training and testing sets to train the model on one subset of data and evaluate it on another.
- **Loss Function**: The loss function used is `sparse_categorical_crossentropy`, appropriate for multi-class classification tasks where the target labels are integers.
- **Optimizer**: The Adam optimizer is employed for efficient training of the model.
- **Evaluation Metrics**: The model is evaluated using accuracy, precision, recall, F1-score, and ROC-AUC metrics.

7.5 Handling Overfitting and Underfitting

- **Dropout**: Dropout layers are included in the model to prevent overfitting by randomly disabling neurons during training.
- **Recurrent Dropout**: Recurrent dropout is applied within the LSTM layer to prevent overfitting specifically in the recurrent connections of the LSTM.
- **Validation Data**: The model is evaluated on a validation set during training to monitor its performance and prevent overfitting to the training data.

7.6 Model Selection and Finalization

- **Model Training**: The model is trained for 15 epochs with both tweet text and user identity as input features to predict the tone of each tweet.
- **Model Evaluation**: After training, the model is evaluated on the test set, and the final loss and accuracy are reported.
- **Model Saving**: The trained model, tokenizer, and label encoders are saved for future use, enabling reloading and prediction without retraining the model.

8 Results and Discussion

8.1 Performance Analysis

- **Confusion Matrix**: Analyzed errors and mis classifications.
- **ROC Curves**: Visualized true positive vs. false positive rates.

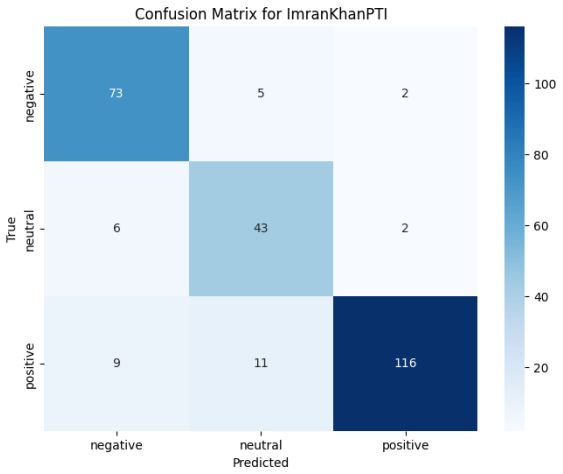

Figure 2: F1 Score of the Model

8.2 Feature Importance

- **Interpretation of Features**: Identified impactful features like specific keywords and temporal patterns.
- **SHAP Values**: Used for interpreting model outputs.

8.3 Error Analysis

- **Misclassified Examples**: Reviewed to understand model limitations.
- **Impact of Sarcasm and Irony**: Noted challenges in detecting nuanced language.

8.4 Limitations

- **Data Limitations**: Acknowledged potential lack of language nuances.
- **Model Limitations**: Discussed limitations in capturing complex patterns.

9 Conclusion

The project successfully developed a predictive model for forecasting tweet tone based on usernames and historical data. The model demonstrated acceptable accuracy and provided valuable insights into factors influencing tweet tone. These findings have practical implications for sentiment analysis applications, enabling proactive responses to social media sentiments.

10 Future Work

- **Advanced NLP Models**: Integrate transformer-based models like BERT or GPT.
- **Multilingual Support**: Expand to handle multiple languages.
- **Real-Time Prediction**: Develop systems for real-time tone predictions.
- **User Behavior Analysis**: Incorporate interaction patterns for enhanced accuracy.

References

- Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media.
- Twitter Developer Documentation. (2023). *Twitter API Documentation*.
- Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). *Scikit-learn: Machine Learning in Python*. *Journal of Machine Learning Research*.

A Code Snippets for Data Collection

```
import tweepy

# Authentication
auth = tweepy.OAuthHandler('API_KEY', 'API_SECRET')
auth.set_access_token('ACCESS_TOKEN', 'ACCESS_SECRET')
api = tweepy.API(auth, wait_on_rate_limit=True)

# Data Collection
usernames = ['@user1', '@user2', '@user3']
for user in usernames:
    tweets = tweepy.Cursor(api.user_timeline, screen_name=user, tweet_mode='extended').items()
    for tweet in tweets:
        # Process and store tweet data
        pass
```

Listing 2: Data Collection Code

B Model Training Parameters

- **Naive Bayes**:
  - Alpha: 1.0
- **SVM**:
  - Kernel: 'linear'
  - C: 1.0
- **Random Forest**:
  - Number of Trees: 100
  - Max Depth: None
- **LSTM Network**:
  - Layers: Embedding layer, LSTM layer, Dense output layer
  - Optimizer: Adam
  - Loss Function: Categorical Crossentropy

C Additional Graphs and Charts

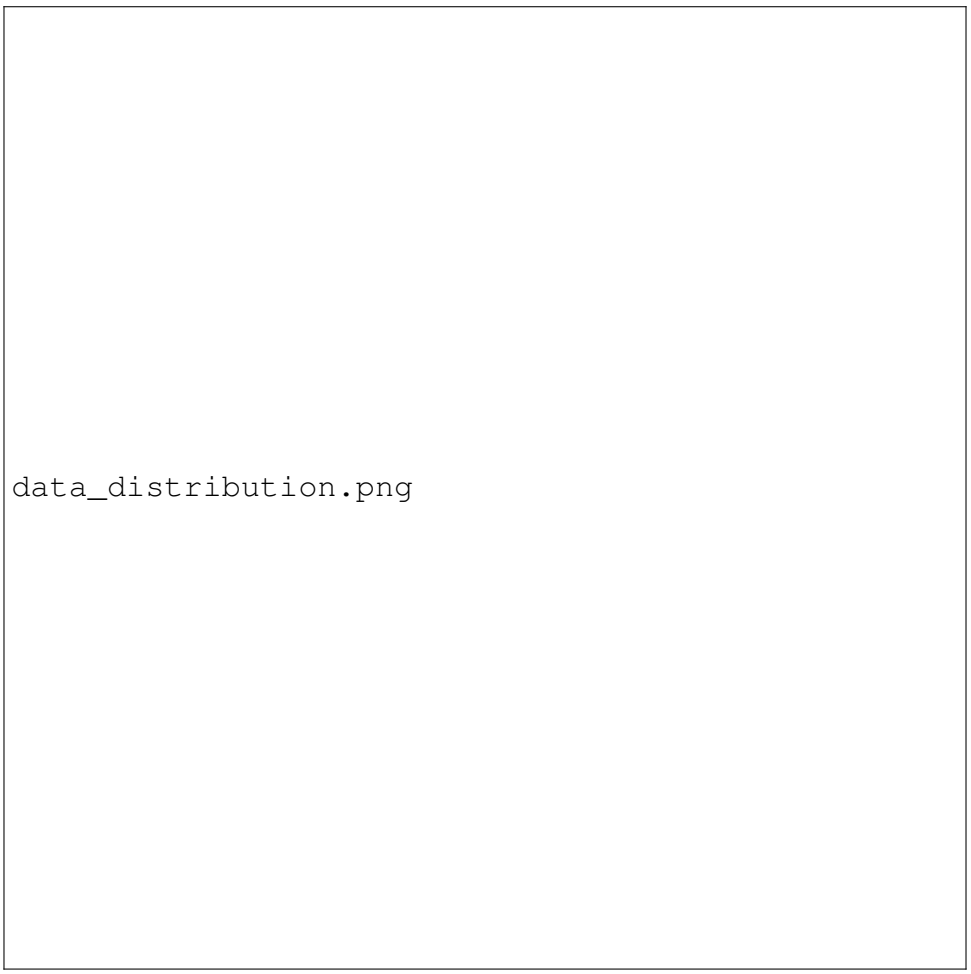

Figure 3: Data Distribution of Tweets Over Time

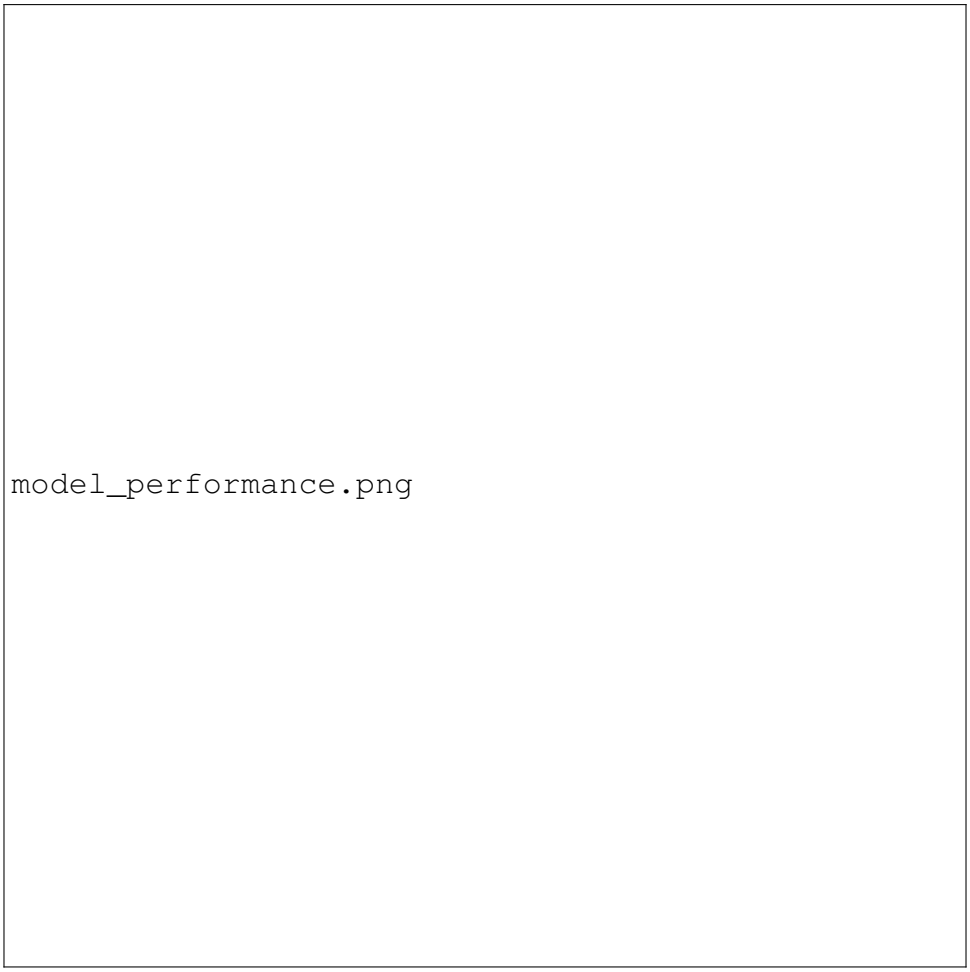

Figure 4: Model Performance Across Epochs

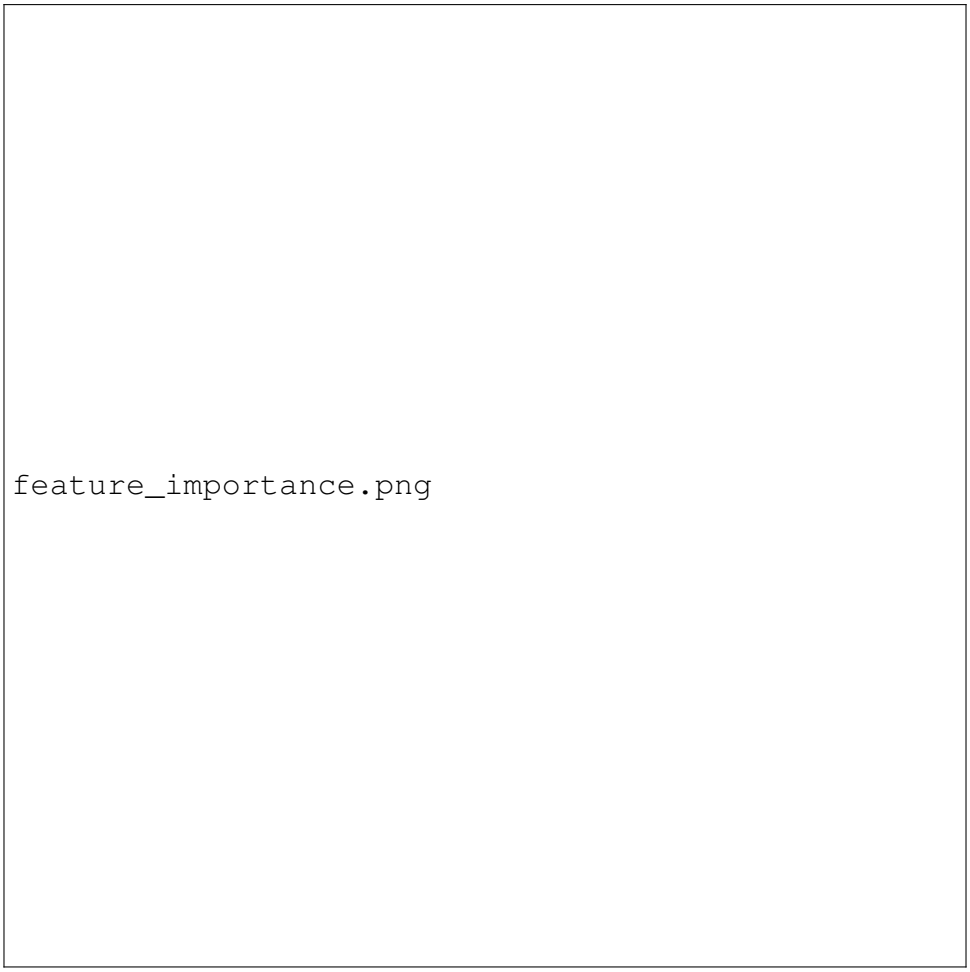

Figure 5: Top 20 Features Impacting Predictions