Twitter Sentiment Analysis Project Report

1. Introduction

This report details the process of building and evaluating a sentiment analysis model for Twitter data. The goal of this project is to classify tweets into three sentiment categories: Positive, Negative, and Neutral. Two different approaches were explored: a traditional machine learning model using TF-IDF with Logistic Regression, and a deep learning model using a pre-trained BERT model.

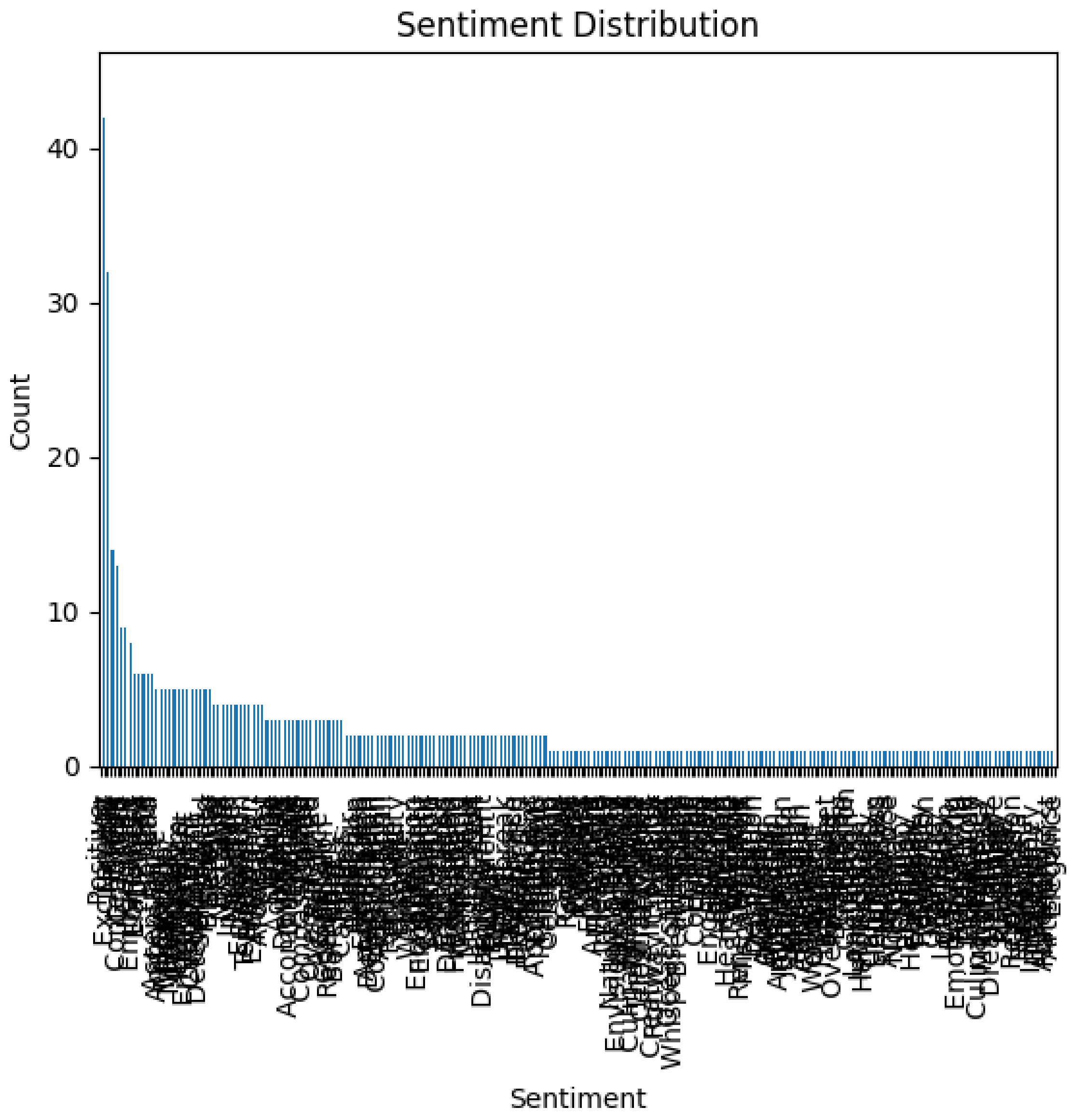
2. Data Loading and Preprocessing

2.1. Data Loading and Initial Exploration

- The dataset used is `sentimentdataset.csv`.
- The initial dataset contained 732 entries.
- The initial exploration of the dataset was done using pandas to understand its structure, identify missing values, and check for duplicates.
- Unnecessary columns like 'Unnamed: 0.1' and 'Unnamed: 0' were dropped.
- 20 duplicate rows were found and removed, resulting in a dataset with 712 entries.
- Data types of 'Retweets' and 'Likes' columns were converted from float to integer, and the 'Timestamp' column was converted to a datetime object.

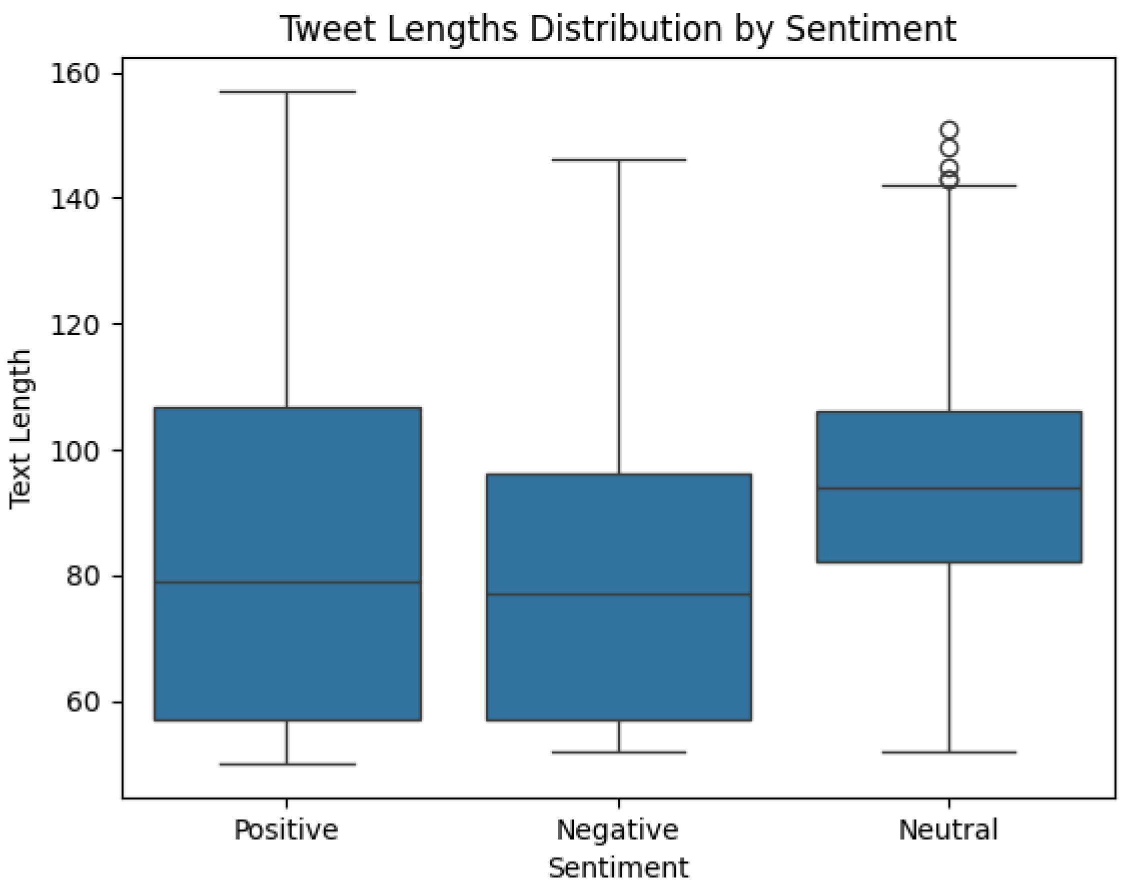
2.2. Sentiment Label Consolidation

- The original dataset contained 279 unique sentiment labels.
- To simplify the classification task, these labels were mapped into three main categories: 'Positive', 'Negative', and 'Neutral'.
- After mapping, the sentiment distribution was as follows:
    - Positive: 430
    - Negative: 160
    - Neutral: 122



2.3. Exploratory Data Analysis (EDA)

- The distribution of tweet lengths was analyzed to see if there is a correlation between the length of a tweet and its sentiment.
- A box plot was created to visualize the distribution of tweet lengths for each sentiment category.



2.4. Text Cleaning

- A text cleaning function was created to preprocess the tweet text. The cleaning process involved:
    - Converting text to lowercase.
    - Removing user mentions (`@username`).
    - Removing URLs.
    - Removing punctuation.
    - Removing emojis.
    - Tokenizing the text.
    - Lemmatizing the tokens using NLTK's `WordNetLemmatizer`.
    - Removing stopwords.

3. Model 1: TF-IDF with Logistic Regression

3.1. Feature Extraction

- The cleaned tweet text was converted into a numerical format using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorization technique.
- The `TfidfVectorizer` from scikit-learn was used with a `max_features` limit of 5000, resulting in a feature matrix of shape (712, 2237).

3.2. Model Training and Evaluation (Initial - Imbalanced Data)

- The data was split into training and testing sets (80/20 split) with stratification to maintain the sentiment distribution.
- A Logistic Regression model was trained on the initial imbalanced data.
- The initial evaluation showed an accuracy of **0.69**.
- The classification report revealed poor performance on the 'Negative' and 'Neutral' classes:
    - **Negative**: Precision=1.00, Recall=0.34, F1-score=0.51
    - **Neutral**: Precision=1.00, Recall=0.08, F1-score=0.15
    - **Positive**: Precision=0.66, Recall=1.00, F1-score=0.80

3.3. Handling Class Imbalance

- To address the class imbalance issue, the `class_weight='balanced'` parameter was used in the Logistic Regression model.

3.4. Model Training and Evaluation (After Resampling)

- The Logistic Regression model was retrained on the balanced dataset.
- The model's performance improved significantly after resampling. The accuracy increased to **0.84**.
- The precision, recall, and F1-scores for the 'Negative' and 'Neutral' classes were much better:
    - **Negative**: Precision=0.84, Recall=0.81, F1-score=0.83
    - **Neutral**: Precision=0.85, Recall=0.44, F1-score=0.58
    - **Positive**: Precision=0.84, Recall=0.97, F1-score=0.90

3.5. Model Saving

- The trained `TfidfVectorizer` and the final `LogisticRegression` model were saved to disk using `joblib` for later use. The saved files are:

  - `tfidf_vectorizer.joblib`
  - `logistic_reg_model.joblib`

4. Model 2: BERT for Sentiment Analysis

4.1. Introduction to BERT

- A more advanced deep learning approach using a pre-trained BERT (Bidirectional Encoder Representations from Transformers) model was also implemented. BERT is a powerful language representation model that has shown state-of-the-art results on various NLP tasks.

4.2. Tokenization and Input Preparation

- The text data was tokenized using the `DistilBertTokenizer` from the `transformers` library.
- The tokenized text was then converted into input tensors (input IDs and attention masks) suitable for the BERT model. The resulting feature matrix had a shape of (712, 768).

4.3. Model Building and Training

- A Logistic Regression model was trained on the BERT embeddings.
- The training data was resampled using `RandomOverSampler` to handle class imbalance.

4.4. Evaluation

- The fine-tuned BERT model was evaluated on the test set. The results showed a high level of accuracy: **0.83**.
- The classification report for the BERT-based model:
  - **Negative**: Precision=0.79, Recall=0.84, F1-score=0.82
  - **Neutral**: Precision=0.55, Recall=0.48, F1-score=0.51
  - **Positive**: Precision=0.92, Recall=0.93, F1-score=0.92

4.5. Model Saving

- The trained BERT-based model was also saved. The file is:

  - `bert_logistic_reg_model.joblib`

5. Conclusion

This project successfully demonstrated the process of building a sentiment analysis model for Twitter data. Both the traditional machine learning approach (TF-IDF with Logistic Regression) and the deep learning approach (BERT) were explored.

- The TF-IDF with Logistic Regression model, after handling class imbalance, provided a good baseline and performed reasonably well with an accuracy of 84%.
- The BERT-based model also achieved a high accuracy of 83%, showcasing the effectiveness of using pre-trained language models for NLP tasks.

The saved models can be used to predict the sentiment of new tweets.