Twitter Sentiment Classification

CS 583 Research Project

Rajit Puzhakkarezhath
673351777
Department of Computer Science
University of Illinois at Chicago

Introduction:

This report describes the process of sentiment analysis performed on tweets related to the 2012 United States presidential election. The dataset used for this task was sourced from Twitter and contains tweets mentioning either Barack Obama or Mitt Romney, the two major candidates of that election. The goal of this project was to build a classification model that could accurately predict the sentiment of tweets as positive, negative, or neutral.

The sentiment analysis was performed using the BERTweet model, which is a pre-trained deep learning model designed specifically for processing text data from Twitter. The model was fine-tuned on the dataset and evaluated on a held-out test set to assess its performance. The performance of the model was measured using various metrics, including accuracy, F1 score, precision, and recall.

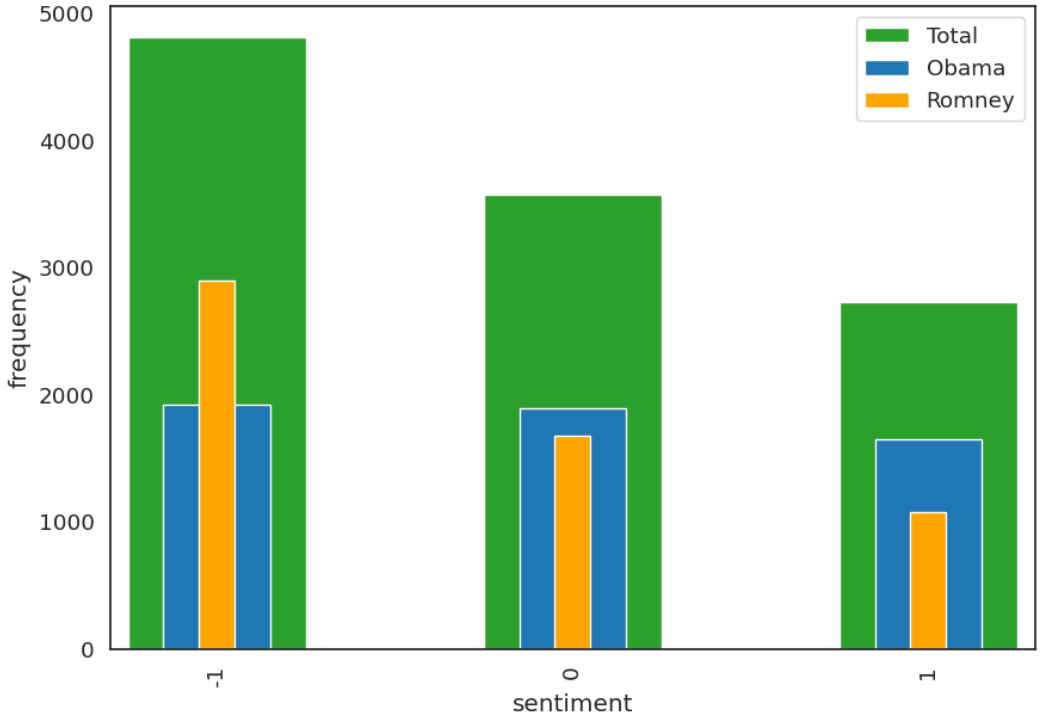
The report details the preprocessing steps performed on the dataset, the training of the BERTweet model, and the evaluation of the model's performance. Additionally, the report includes the predictions made by the model on a separate test set of tweets, and these predictions were saved to output files for further analysis. The following sections provide a more detailed overview of each step in the process.

Data Loading and Preprocessing

The code starts by loading the training data from an Excel file containing two sheets, one for tweets related to Obama and the other for Romney. The preprocess function is then used to preprocess the data. This function keeps only the "Annotated tweet" and "Unnamed: 4" columns, renames them to "text" and "labels", respectively, drops the first row of the data frame, replaces all -1 values in the "labels" column with 2, keeps only the rows with label values of 0, 1, or 2, drops any null values, and returns the resulting data frame. Finally, the function is used to preprocess both sheets, and the resulting data frames are concatenated into a single data frame.

Next, the code splits the data into training and test sets using the train_test_split function from sklearn, with a validation size of 15% and a random state of 42. The training and test sets are then reset to have continuous indices.

On analyzing the data, I found the following distribution of the data. The graph includes the distribution for negative, neutral and positive. The dataset is a little skewed with a greater number of negative tweets but that is expected for twitter.



Model Creation and Training

The ClassificationArgs class from the simpletransformers library is used to define the arguments for the classification model. Several arguments are set here, including the number of training epochs, the batch size, the metrics to use for early stopping, and the output directories for the best and final models. A ClassificationModel is then created with the bertweet-base pre-trained model from the vinai repository on Hugging Face. The model is trained on the training data using the train_model function with the f1_score metric.

BERTweet is a language model developed by the team at the University of California, Santa Barbara. It is based on the popular BERT (Bidirectional Encoder Representations from Transformers) architecture, which uses a transformer neural network to encode text inputs.

However, unlike the original BERT model which was trained on general text data from the internet, BERTweet is specifically trained on tweets. This allows it to better capture the unique characteristics of Twitter language such as hashtags, user mentions, and emojis.

BERTweet uses a large-scale pre-training process on a diverse set of Twitter data to learn contextual embeddings for words, sub-words, and characters. It utilizes a Byte-Pair Encoding (BPE) algorithm to handle out-of-vocabulary (OOV) words and a masked language modeling (MLM) objective function during pre-training to learn contextual word representations. The MLM objective involves masking a random subset of tokens in the input text and asking the model to predict the masked tokens.

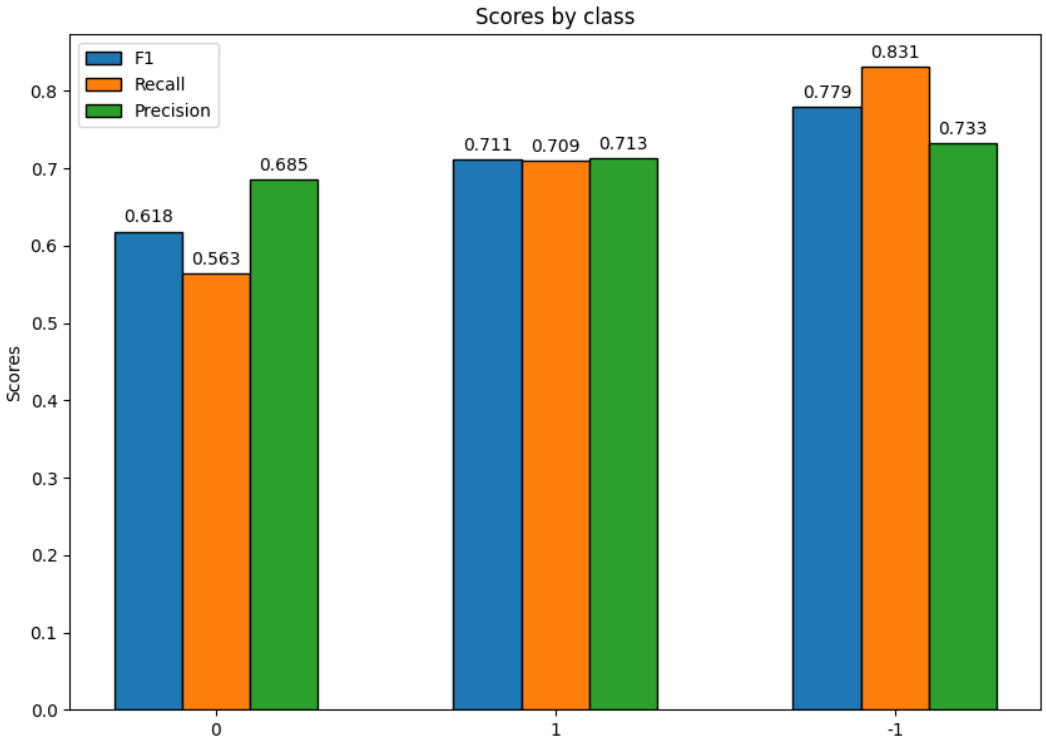
The pre-trained BERTweet model can then be fine-tuned on a downstream task such as sentiment analysis, named entity recognition, or classification by adding an additional classification layer on top of the BERTweet encoder. The entire model can be fine-tuned end-to-end using backpropagation and gradient descent.

Overall, BERTweet has shown to outperform previous state-of-the-art models on several benchmark datasets for sentiment analysis and other NLP tasks on Twitter data.

Evaluation

The resulting evaluation metrics, including accuracy, f1-score, precision, and recall, are printed to the console as well as depicted in a bar-plot shown below. These have been evaluated on 15% of the training data set provided that had been initially separated into train and val dataframes.

The accuracy for the model was `'acc': 0.7152278177458034`.



Test Data Prediction and Export

The code then loads two separate Excel files containing the test data for Obama and Romney, respectively. The data is loaded into data frames, and the predict function of the trained model is used to predict the labels of the tweets. The resulting predictions are written to text files in the format required for the submission of the predictions for each candidate.

Libraries Used

The code imports several libraries used for data loading, preprocessing, training, and evaluation. These libraries include:

os: used for setting environment variables
csv: used for reading and writing CSV files
torch: used for deep learning computations and operations
transformers: used for loading and using pre-trained transformer models
gc: used for garbage collection to free up memory
pandas: used for data manipulation and analysis
seaborn: used for visualization of data distributions and relationships
matplotlib: used for data visualization and plotting
scipy: used for mathematical operations and functions
simpletransformers: used for creating and training the BERTweet model for classification
sklearn: used for splitting data into training and test sets, evaluating the model, and computing evaluation metrics such as f1-score, precision, and recall.

Execution Time

The code may take a significant amount of time to execute, depending on the size of the data and the computational resources available. Training the model may take several minutes on a CPU, while using a GPU can significantly reduce the training time. The evaluation and prediction steps are much faster, and should not take more than a few minutes to complete.