Harmful Content and Sentiment Analysis

Dikchhya Palikhe

INTRODUCTION

The purpose of this project is to understand text classification models and examine whether sentiment correlates with hate speech/offensive language labels. While hate speech and offensive language are often assumed to carry negative connotations, this project investigates whether that assumption holds true in the data. In addition to this, a simple web application was developed using Flask, allowing users to enter any text and receive both hate speech/offensive and sentiment labels for it.

DATA

Three different datasets were used for this project, and duplicates were removed from all of them.

- hs_data.csv: This data was obtained from Github. The columns 'tweet' and 'class' were used to train the hate speech detection model. The column 'class' contained categorical values: 0 (hate speech), 1 (offensive language) and 2 (neither).

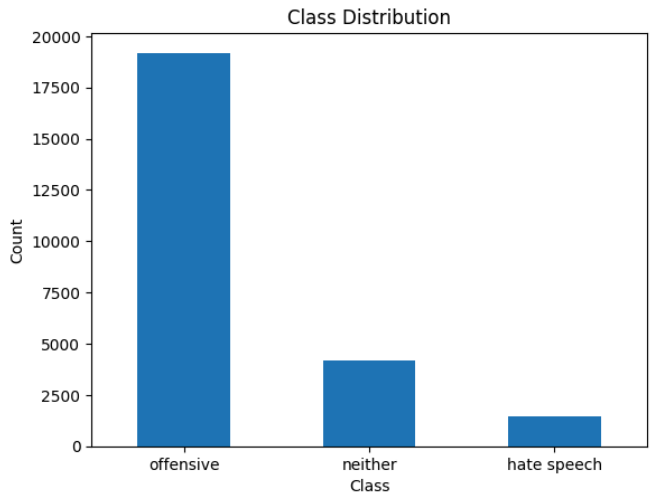

Figure 1

- sentiment_data.csv: This data was obtained from Kaggle. The columns 'text' and 'sentiment' were used to train the sentiment analysis model. The column 'sentiment was converted into categorical values: 0 (negative), 1 (neutral) and 2 (positive).

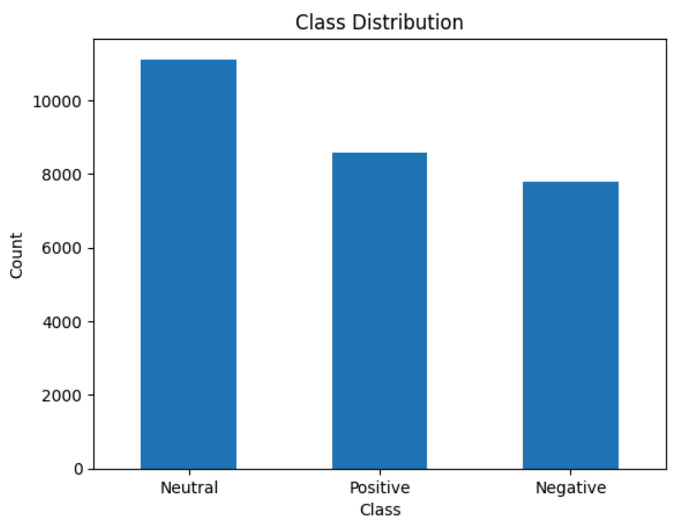

Figure 2

- dataset.csv: This data was also obtained from Kaggle. The columns ‘target’ and ‘text’ were used where ‘text’ was used to get predictions from the two models. The ‘target’ column contained the sentiment labels (0 = negative, 2 = neutral, 4 = positive) but none of the texts were classified as neutral. Therefore, the sentiment analysis model was used to regenerate the labels.

APPROACH

Text Classification

For both the hate speech detection and sentiment analysis models, the scikit-learn library was used to run Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Random Forest classifiers. The models also included a cleaning function to remove extra spaces, usernames and URLs from the data. Cross-validation was then applied to identify the best performing algorithm.

Chi-SquareTest

After generating labels using the best-performing algorithm, a chi-square test was conducted to examine whether sentiment correlates with hate speech/offensive language labels. The null hypothesis stated that there is no relationship between the labels, while the alternative hypothesis stated that a significant relationship exists.

Flask App

Flask, a Python web framework, was used to create a simple web application that ran the classification algorithms, allowing users to enter text and view the labels generated by the models.

RESULTS

After cross-validating Logistic Regression, SVM, KNN, and Random Forest classifiers for both models, SVM achieved the highest accuracy for the hate speech detection model, while Random Forest achieved the highest accuracy for sentiment analysis. SVM works by finding a boundary that optimally separates different classes in the dataset. Hate speech and offensive texts often contain specific words that clearly distinguish them from other texts. This may be why SVM performed best for hate speech detection. In contrast, sentiment labels are more complex. Random Forest is less prone to overfitting because its final prediction is obtained by averaging the predictions of multiple individual trees. This may have been the reason for its higher accuracy in sentiment analysis.

The chi-square test between sentiment and hate speech labels resulted in a p-value of 0.0, which occurred because the value was extremely small and rounded down by the computer. This supports the assumption that the two labels are correlated. Similarly, another chi-square test was conducted between the sentiment labels and the ‘target’ column of dataset.csv, which also produced a p-value of 0.0, indicating a significant correlation between the variables.

The Flask app was able to classify new texts, but there is definitely room for improvement. Texts containing negative words such as ‘hate’ or ‘don’t like’ were classified as negative, while texts with positive words were classified as positive. However, when both positive and negative words appeared in the same text, as in ‘The customer service was terrible but the food was good,’ the model assigned either a positive or negative label instead of neutral. Similarly, texts containing profanity were often classified as neutral, this might be because the training data was censored. Although the hate speech detection model also has room for improvement, it appears to perform better than the sentiment analysis model.

LIMITATIONS

The models were built using scikit-learn, a traditional machine learning library. While effective, these models are generally less accurate and powerful than deep learning approaches (PyTorch and TensorFlow). For sentiment analysis, a censored dataset was used. Other available sentiment datasets contained multiple languages, and filtering English text with libraries such as langdetect or fastText was too time-consuming. As a result, the censored dataset was used, which has affected the predicted labels. Therefore, future improvements could address these limitations by exploring more advanced libraries and higher-quality datasets.