Università degli Studi di Milano

Experimental Project

Project: I Hate You

Text Mining and Sentiment Analysis

Professor Ferrara Alfio

Danial Forouzanfar, 12988A

Danial.forouzanfar@studenti.unimi.it

2023-2024

1. Introduction

With the rise of the social media era, spreading hate and offended content has emerged as a great cause for concern; hence, detecting hate speech has been considered one of the most critical challenges of the modern digital era. Such detection automatically demands powerful NLP techniques and machine learning models. This detection forms a very essential task to be carried out to maintain safe online spaces and keeping all legal and ethical considerations in mind. Real applications include content moderation on social media, monitoring online forums, and assisting law enforcement in identifying potentially harmful content.

This project, focuses on the three-class classification of tweets includes hate speech, offensive language, and neutrals. In this project, two machine learning models are implemented and evaluated in an attempt to predict accurately which class a certain tweet falls into, considering the textual content.

The primary goals of this project are to:

- Develop a preprocessing pipeline to clean and prepare tweet data for analysis.
- Train and evaluate machine learning models to classify tweets into the three categories.
- Compare the performance of different models, namely XGBoost and Support Vector Machine (SVM).
- Explore the impact of feature engineering, such as word embeddings, on model performance.

2. Methodology

The dataset used for this project is Thomas Davidson's Hate Speech and Offensive Language dataset. It contains 24,802 labeled tweets, with features including the tweet text and a label indicating the class. For this project, the dataset was remapped to align with the following labels:

- Neutral: 0
- Hate Speech: 1
- Offensive Language: 2

Each class was downsampled to the same number of samples to address class imbalance and ensure fair representation during training. This balance in the labels is beneficial for classification problem, as it minimizes the risk of model bias toward any particular class (we might have higher accuracy but lower precision and F1 score). After this step, our dataframe contains 4290 rows and 2 columns, "Text" and "Label". The dataset has no missing values across its columns. This means that there will not be a need to handle missing values during preprocessing.

Another good analysis concerning the 'Text' column was the number of characters per text entry. The average number of characters per row was 87.514.

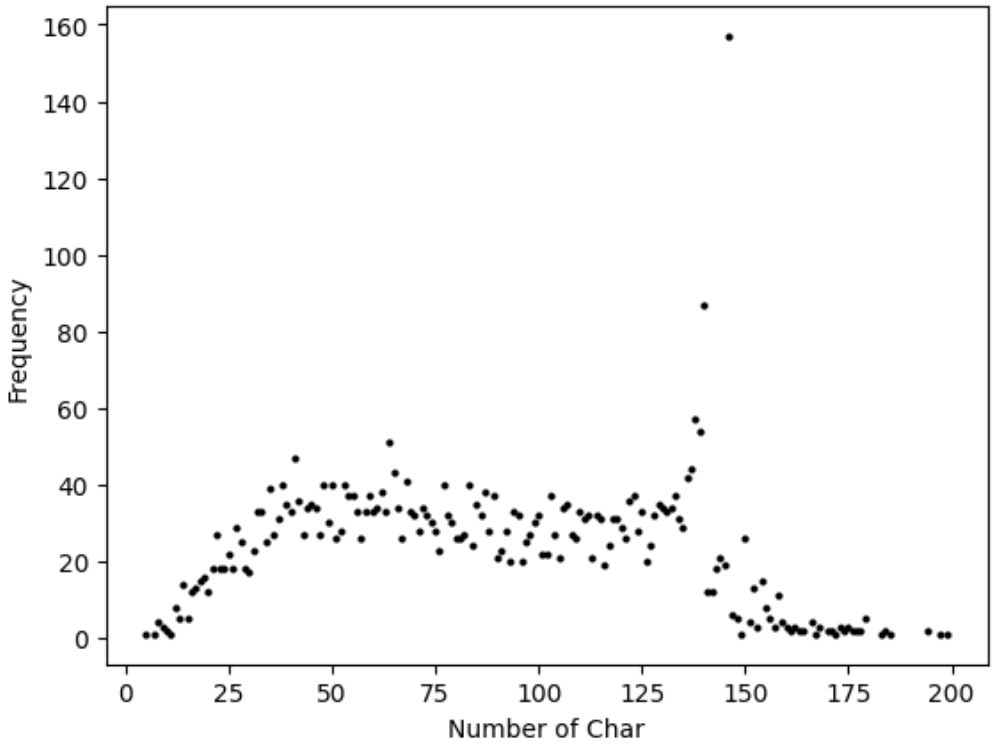

Figure 1. Number of Characters

In figure 1, we can see that there are around 160 texts with a length of 150 chars in our dataset. Minimum of char was 5 and maximum was 199. These statistics highlight the variability in text lengths within the dataset. This can be helpful in deciding on preprocessing steps.

The preprocessing pipeline involved the following steps:

- **Cleaning**: Removal of hashtags, URLs, mentions, and special characters.
- **Text Normalization**: Conversion of text to lowercase, fixing contractions, and converting numbers to words.
- **Tokenization and Stemming**: Splitting text into tokens and applying stemming to reduce words to their root forms.
- **Stop Word Removal**: Remove the common English stop words to focus on meaningful content.
- **Vectorization**: Use of pre-trained Word2Vec embeddings, word2vec-google-news-300, to generate numerical sentence representations.

By representing a word as a vector, we can observe relationships between words. we can see that for instance, with the word "king," we can identify connections to words like " queen," " sultan," " ruler," and others with a specific score.

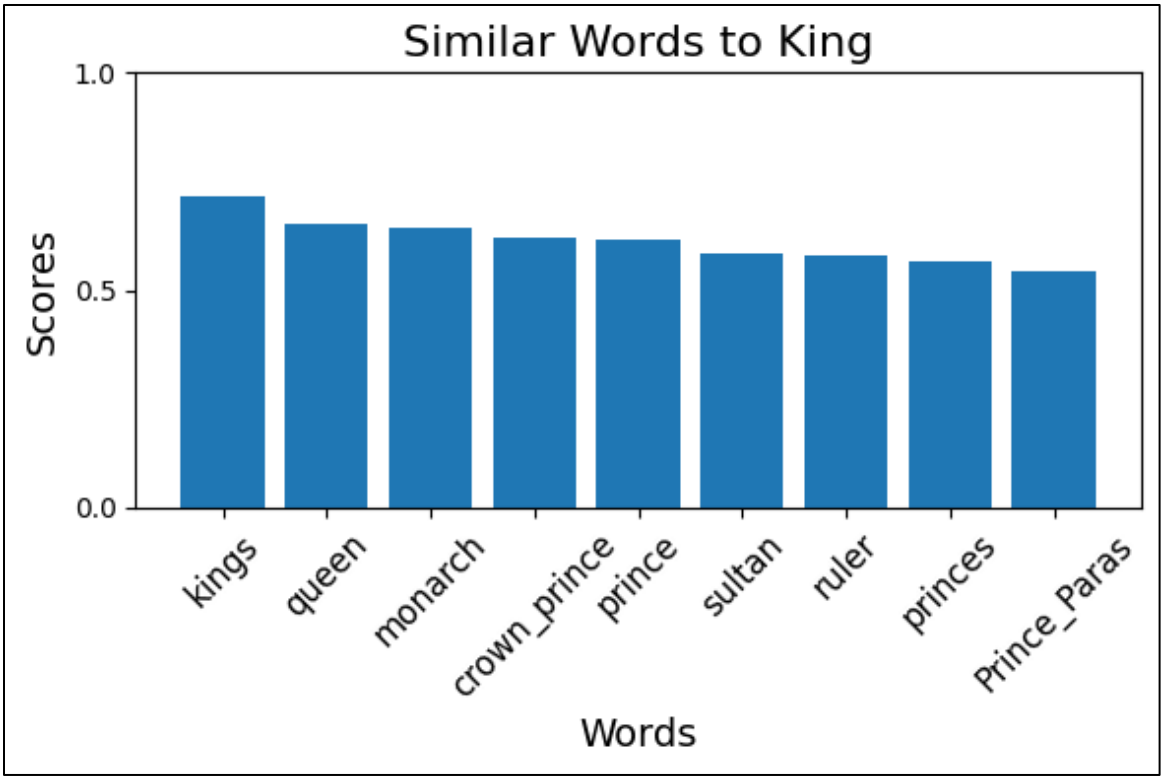

Figure 2. Relation between words

Lastly, the models' performance was evaluated using:

- Accuracy: Overall correctness of predictions.
- Precision, Recall, and F1 Score: To assess performance across imbalanced classes.
- Confusion Matrix: To visualize prediction errors across categories.

These metrics provided a comprehensive understanding of model performance and highlighted areas for improvement.

3. Result

The performance of the models on the test datasets is summarized below:

Table 1. Accuracy of models

Model	Accuracy
XGBoost	72%
SVM	71%

Table 1 shows the accuracy of each model. While XGBoost provided a slightly better prediction, both models perform similarly. A detailed analysis of the models' performance is provided using the classification report and confusion matrix:

XGBoost classification report & confusion matrix:

Classification Report:

	precision	recall	f1-score	support
0	0.72	0.84	0.78	427
1	0.72	0.59	0.65	428
2	0.72	0.73	0.73	427
accuracy			0.72	1282
macro avg	0.72	0.72	0.72	1282
weighted avg	0.72	0.72	0.72	1282

Figure 3. Classification Report of XGBoost model

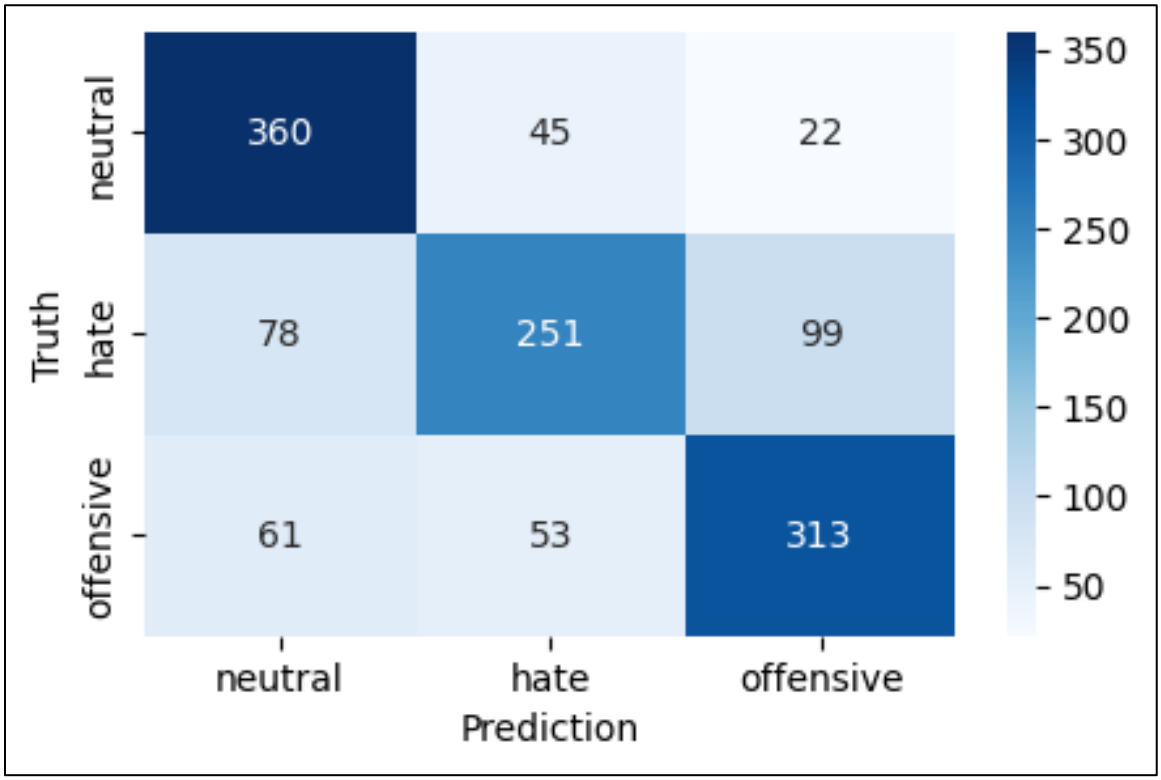

Figure 4. Confusion Matrix of XGBoost model

SVM classification report & confusion matrix:

Classification Report:

	precision	recall	f1-score	support
0	0.71	0.85	0.77	427
1	0.69	0.59	0.64	428
2	0.73	0.70	0.72	427
accuracy			0.71	1282
macro avg	0.71	0.71	0.71	1282
weighted avg	0.71	0.71	0.71	1282

Figure 5. Classification Report of SVM model

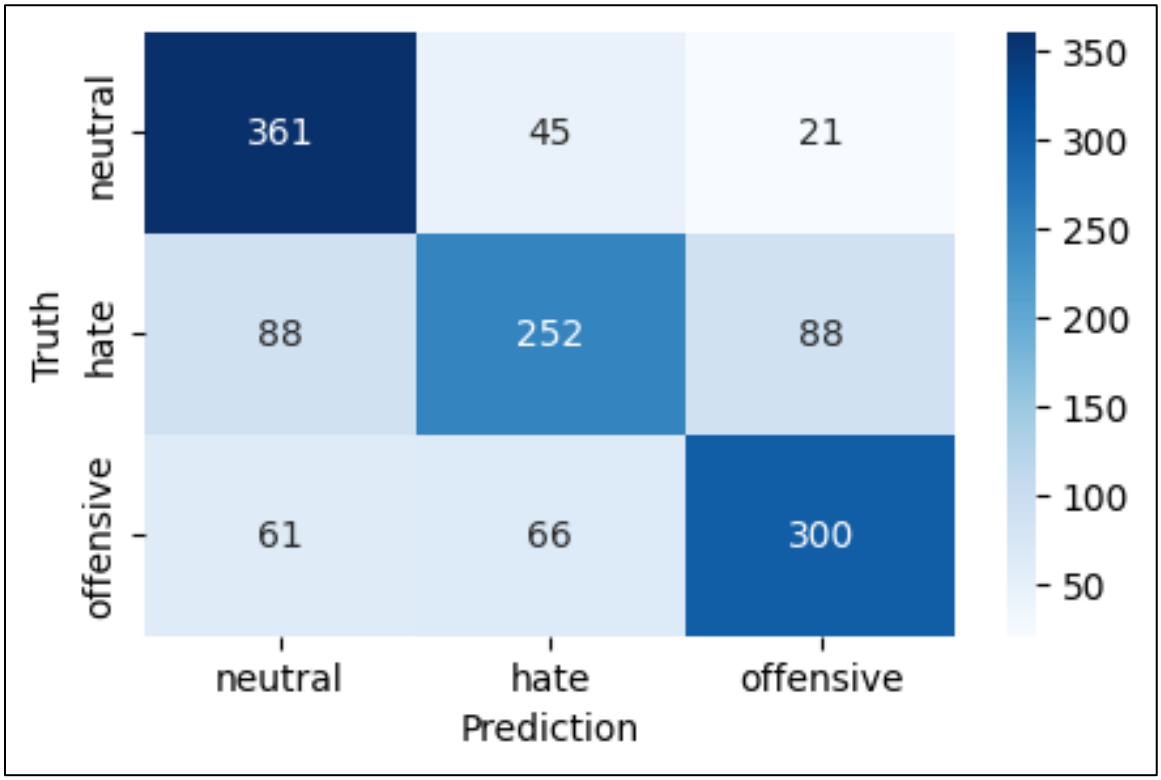

Figure 6. Confusion Matrix of SVM model

4. Conclusion

In this project we used two different models, SVM and XGBoost, and also word to vector technique to detect hate speech and offensive language in social media. Word2Vec embeddings helps us to obtain the vector representations of sentence and also captured the semantic context of tweets and those two ML models helps us do the classifying. In this

project, we saw that XGBoost model outperformed the SVM in most of the metrics, proving to be more efficient for this task.

Improvements to the model from this research could be achieved through advanced NLP techniques, the use of transformers, or contextual embeddings like BERT for improving the performance. Increasing the variety of examples would significantly improve generalization capability.