# Mini Project: Sentiment Analysis on Twitter Data using XG Boost

Candidate No.: 126739
*Department of Computer Science*
*University of Exeter, Exeter, UK*

***Abstract*—This report presents the implementation and analysis of a sentiment analysis model to classify tweets as hate speech. The classifier achieved 94% accuracy on the test set.**

## I. Introduction

In the recent years, microblogging platforms like Twitter have seen tremendous growth in number of users as well as the data being generated. Many individuals and organisations are mining this data to analyse the sentiments of the users for applications like marketing, politics, and stock price prediction etc. Governments try to find tweets which are against countries' national interests or hamper safety of citizens. Among those, identification and restriction of hate speech is one of the most challenging tasks faced by the governments. In fact, it's not just the governments which want to find hate speech tweets, the platforms themselves want to keep a check on it to keep their image of a free and safe digital space. Hence there arises a need for a model which classifies tweets into positive or negative with the context of hate speech. In this project, we explore a model based on gradient boosted tree to identify hate speech in tweet data.

## II. Problem Statement

The objective of this project is to detect hate speech in tweets. For simplicity, it can be said that a tweet contains hate speech if it has a racist or sexist sentiment associated with it. So, the task is to classify racist or sexist tweets from other tweets.

Formally, given a training sample of tweets and labels, where label '1' denotes the tweet is racist/sexist and label '0' denotes the tweet is not racist/sexist, the objective is to predict the labels on the test dataset [1].

## III. Data

The data consists of two csv files: train.csv and test.csv. The file train.csv contains three columns namely id, label and tweet. 'id' is a unique number for each tweet. 'label' 0 denotes that the tweet is not a hate speech and label '1' denotes that the tweet is racist/sexist. 'tweet' contains the text of the tweet where the tagged user id's are replaced by text user. The train file consists data of 31962 tweets out of which 29720(93%) belong to label '0' and 2242(7%) tweets belong to label '1'. As the data is highly imbalanced, we need to create a classifier which won't be affected by this imbalance. The data is provided by Analytics Vidhya [2].

## IV. Software Specifications

The project was implemented using python language in a jupyter notebook environment on Google Colab [3]. Along with python, the following libraries were used in the project:

- numpy: support for numerical operations.
- pandas: data manipulation and analysis.
- sklearn: tools for predictive data analysis.
- xgboost: optimized distributed gradient boosting library.
- matplotlib: comprehensive library for creating static, animated, and interactive visualizations.
- seaborn: data visualization library based on matplotlib.
- warnings: contains warnings classes.
- re: provides regular expressions support.
- nltk: symbolic and statistical natural language processing for English.

## V. Methods And Experiment Design

In this section, we discuss the specifications of the implementation of the project. The project follows the following stages:

### A. Exploratory Data Analysis

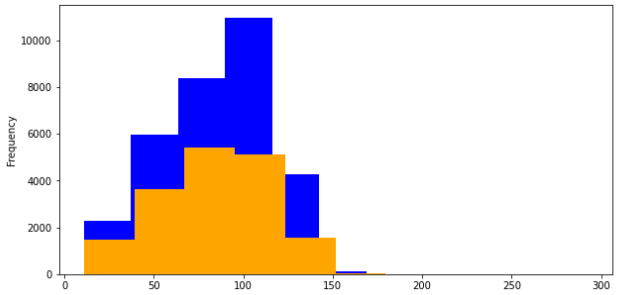


Fig. 1. Distribution of length of tweets for training and testing data.

Figure 1 shows the distribution for length of tweets for training and testing data. We can see that both the distributions are left skewed with centre around 100-120. Very few tweets have more than 120 length. Also, as the training data is bigger, we can see the area of the distribution is more than that of testing data.

To find the most occuring words in the training data, I have used CountVectorizer from sklearn library which converts the collection of words to a matrix of token counts. The output

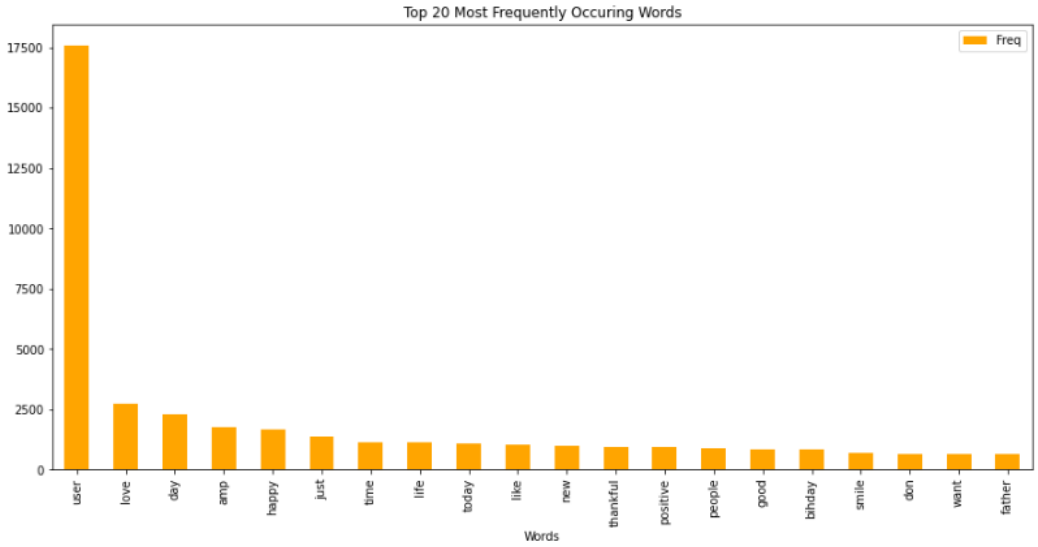


Fig. 2. Top 20 most frequently occuring words.

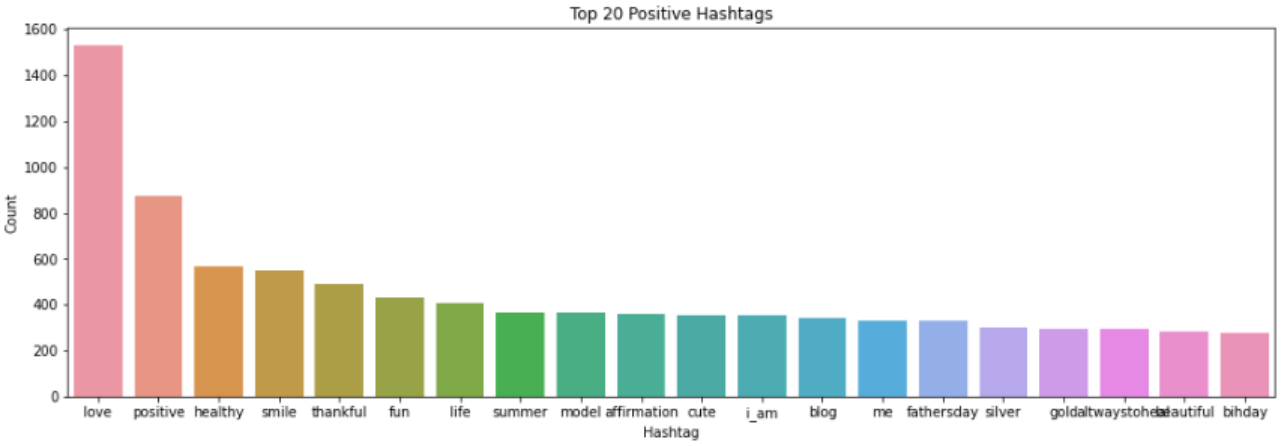


Fig. 3. Top 20 positive hashtags.

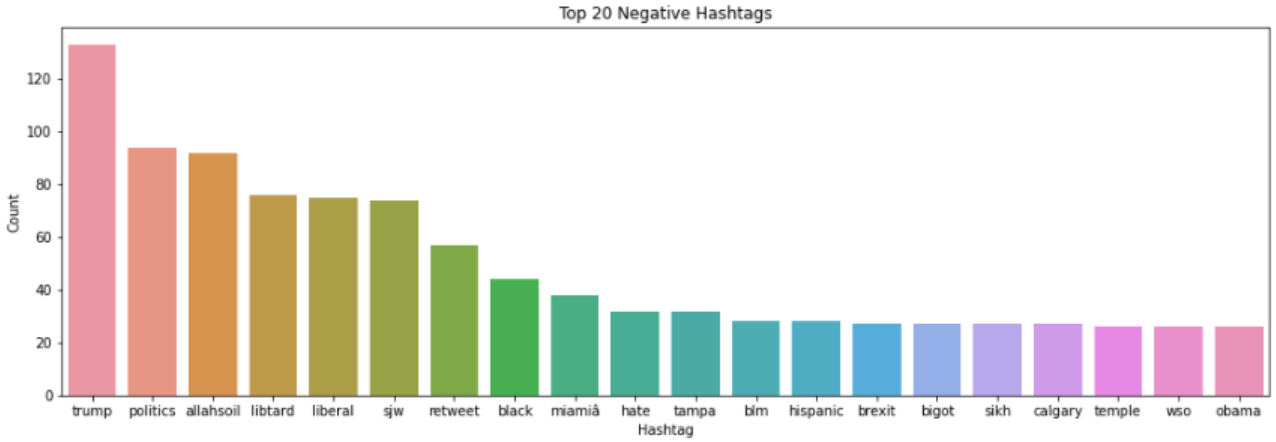


Fig. 4. Top 20 negative hashtags.

is shown in Figure 2. It can be observed that most frequently occuring word is user. This is because the tweets data has replaced the mentioned user ids with the word user. Other than user, love is the most tweeted word.

### B. Collection of Hashtags

To find the hashtags, we use re library. After obtaining list of positive and negative hashtags, I have plotted the distribution for top 20 positive and negative hashtags using FreqDist from nltk as shown in Figure 3 and Figure 4. It can be observed that love is the most used positive hashtag whereas trump is the most used negative hashtag.

### C. Tokenization and word embedding

Tokenization in our context is the process of separating every word from the tweets. It's done using split function of python. After tokenization, we need to perform word embedding to capture the meaning and context of words in the tweets. This is done by redefining the high dimensional word features to low dimensional feature vectors while preserving the context similarity of the data. For this purpose, I have used word2vec implementation from gensim library. Word2vec is an open source two layer neural-network based technique to learn word embeddings.

### D. Stopwords Removal and stemming

Stopwords are commonly used words like 'the', 'a' which are not useful for our analysis. Also, we need to perform stemming i.e. reducing inflection in the words to their root forms. For this purpose, I have used PorterStemmer from nltk.

### E. Bag of Words

To create bag of words for training and testing data, I have used CountVectorizer from sklearn. The output is converted to numpy array.

### F. Splitting the dataset and Feature Scaling

In this step, the training data is split into training and validation sets using train_test_split from sklearn. After splitting the data, we perform feature scaling using StandardScaler from sklearn. This scaling converts the data set to make it ready for feeding to the classifier.

### G. Classifier Training

As the training data set is highly imbalanced, we need a classifier which can handle the imbalanced data. For this reason, I have used the XG Boost library for the XGBClassifier which is an extremely gradient boosted decision tree. Gradient Boosting refers to ensembling of weak models to give a better model.

### H. Prediction and Evaluation Result

After the training, predictions were made on the test data set. The results were evaluated based on three metrics which are Accuracy, Confusion Matrix and F1 score. The accuracy came out to be 0.94 which is decent. But the confusion matrix(shown below) and low f1 score of 0.34 show that the model is still affected by imbalanced data. Hence, the good accuracy is coming from true positives but we have a lot of false positives and some false negatives as well.

**Prediction outcome**

| **actual value** | **p** | **n** | **total** |
|---|---|---|---|
| **p′** | True (+)ve 5941 | False (-)ve 8 | P′ |
| **n′** | False (+)ve 351 | True (-)ve 93 | N′ |
| **total** | P | N | |

## VI. Conclusion and Future Scope

This project implemented an XGBoost based classifier to perform sentiment analysis on twitter data to find out tweets with hate speech. The model performed decently however it still suffers due to imbalanced data.

In future, the classification can be improved by incorporating the following:

- Creating new features using Parts-of-Speech tagging.
- Using Bi-grams or tri-grams for bag of words.
- using pre-trained models.

## References


[1] A. Toosi, "Twitter sentiment analysis," Jan 2019. [Online]. Available: https://www.kaggle.com/datasets/arkhoshghalb/twitter-sentiment-analysis-hatred-speech

[2] "Learn machine learning, artificial intelligence, business analytics, data science, big data, data visualizations tools and techniques." [Online]. Available: https://www.analyticsvidhya.com/

[3] "Welcome to colaboratory - google research." [Online]. Available: https://research.google.com/colaboratory/


## Appendix

Data and code:
https://github.com/siddharthp20/Sentiment-Analysis