Hate Speech Detection

Pranav Dayanand Pawar
IIT Madras
Mentor - Mr. Lokesh Kumar Kriplani

Index -

---

1. Problem Statement

a. With the massive increase in social interactions on online social networks, there has also been an increase in hateful activities that exploit such infrastructure.

b. On Twitter, hateful tweets are those that contain abusive speech targeting individuals (cyber-bullying, a politician, a celebrity, a product) or particular groups (a country, LGBT, religion, gender, an organization, etc.).

c. Detecting such hateful speech is important for analyzing the public sentiment of a group of users towards another group, and for discouraging associated wrongful activities. It is also useful to filter tweets before content recommendation, or learning AI chatbots from tweets.

d. The manual way of filtering out hateful texts is not scalable, motivating us to identify automated ways. Baseline work has been done in this domain, but it mainly focuses on tweets, although we will propose a framework to consider the **maximum area of social media for our detection purpose**.

2. Data Collection -

a. 3 datasets used for analysis -

i. Crowdflower - Readily available as comma-separated files, with uniquely labeled tweets as hate speech, offensive speech and neither. Although we further consider both hate speech and offensive speech labels together as hate speech. This was done

by carefully examining sample tweets from both the categories, and it was confirmed both categories represent the same set of sentiment, that is hate speech.

ii. ZeerakW - Two datasets containing tweet IDs with annotated labels. Both the datasets were scraped from twitter API using Tweepy. Tweets were labeled as sexist, racist and neither. Hence similar to the previous dataset, here we considered sexist and racist tweets together as hate speeches.

3. Data Analysis and Cleaning -

a. To reduce the target bias all three datasets were merged at the end, finally contributing to around ~41k unique tweets. Out of the total available tweets, 24k are hate speeches, while 17k are neutral speeches.

b. Basic text preprocessing such as lowercasing, stopwords removal, lemmatization, stemming was performed on the text data, we used NLTK and TextBlob to execute these operations.

c. It's necessary to represent the text data in numeric format to feed in the ML model, so we experimented with various methods of text to numeric transformations.

4. Text to Numeric Transformation- Initially we will discuss a few baseline methods and then will discuss the proposed approach. In all these methods, and a numeric embedding/ vector is generated for a tweet and is used as its feature representation with a classifier.

a. **TF-IDF Vectorizer** - Being one of the most conventional methods to transform words into numbers. It accounts for the term frequency `(# time word occurs in document / # words in the document)` and the inverse document frequency `log( # total documents in the dataset/ # documents containing the word 'X')`. This measure is the multiplication of TF and IDF. Sklearn TF-IDF Vectorizer was used which generates multidimensional vector for the tweet.

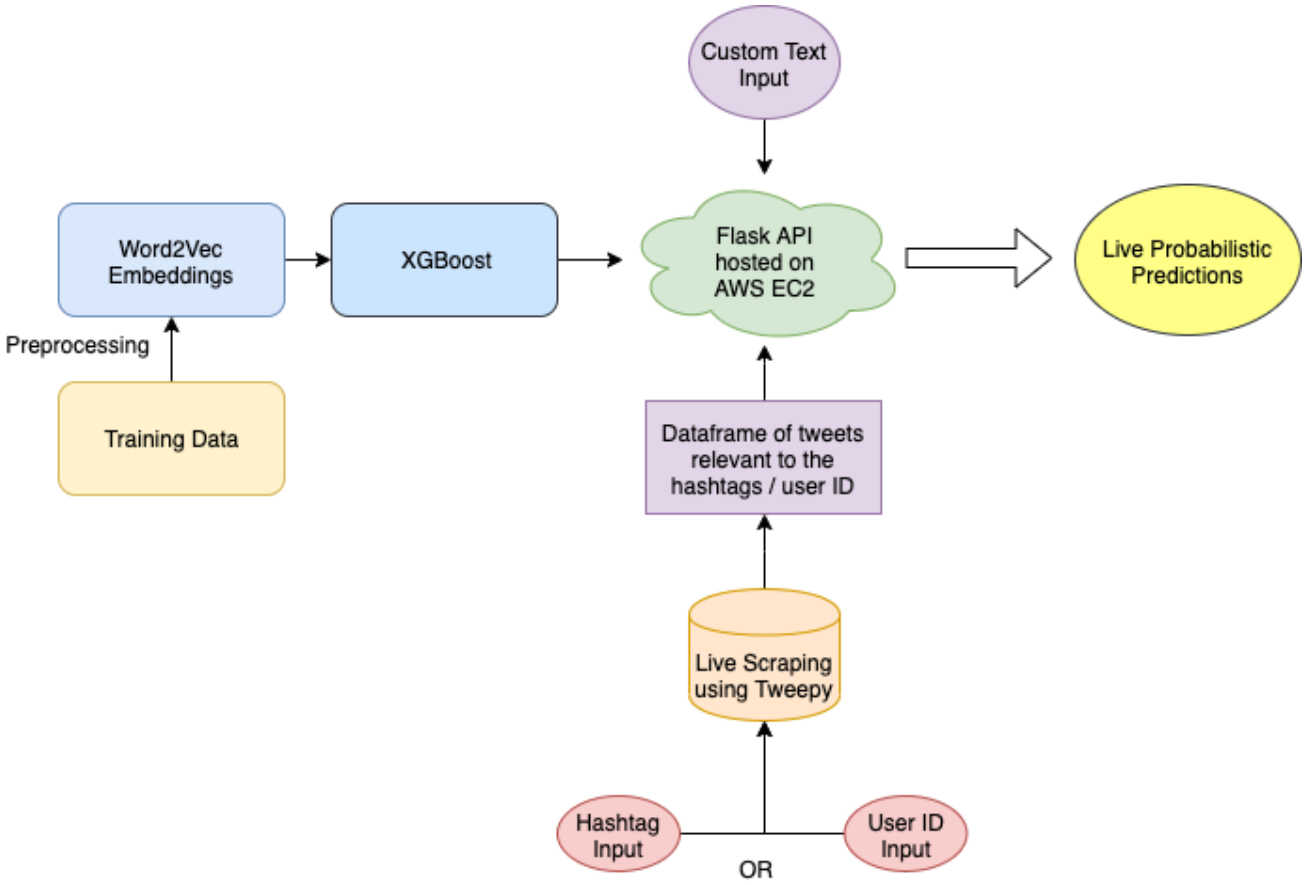
b. **Word2Vec** - One of the most popular techniques which transform a word into a multi-dimensional vector based on the meaning of the word. We used the Continuous Bag Of Words (CBOW) algorithm for embedding formation using a shallow neural network.

i. For example is a tweet has 10 words, we use word2vec to transform each word into a multidimensional vector and then taking an average of all the 10 vectors gives us an embedding for the complete tweet.

ii. Here we have used google news pretrained embeddings, these embeddings are generated by training on part of Google News dataset (about 100 billion words) which gives a 300-Dimensional representation of a word.

iii. We have also made use of the transfer learning technique in Word2Vec which is explained further in this document.

c. **Doc2Vec** - This is an advanced version of word2vec, where instead of learning embeddings for a single word we learn to create embeddings for the whole document, considering the words in it, and the context of the document.



5. Proposed methods -

a. Firstly we used Word2Vec embeddings, Google-News pre-trained word embeddings were used to transform every word in our data corpus.
b. This transforms words into 300-Dimensional vector embedding, although there was one flaw in using this method.
c. The embeddings were trained on the Google News dataset, which does not really contain many hateful words, as it's obvious that news agencies tend to be neutral and to use clean language. So the model built using these embeddings was giving mediocre results, because hateful words were not getting correct representation in vector space.
d. Hence we decided to use the Transfer Learning approach here. Basically, training new embeddings on top of the current google pre-trained embeddings architecture.
e. Basically using functions in the Gensim library, we merge our data vocabulary with the google pretrained model vocabulary and then using the CBOW algorithm it learns embeddings for all the words in the domain, as well as the generic words which were already present in the pretrained model.
f. This method was used in the final method and it indeed improved the accuracy by a significant margin.

6. Model Building -

a. We experimented with Logistic Regression, Gradient Boosting Trees (XGBoost, LightGBM, GBDT) and Convolutional Neural Networks with various embedding methods explained above

b. Experimental results are as follows - Accuracy and F1-Score are being calculated on the validation dataset which was sampled randomly with 20% data out of the whole dataset.

S. No.	Experiment	Accuracy	F1-Score
1	TF-IDF + Logistic Regression	0.8554	0.8784
2	TF-IDF + XGboost	0.8567	0.8815
3	Word2Vec Pretrained + Logistic Regression	0.8536	0.8820
4	Word2Vec Pretrained + XGBoost	0.8599	0.8885
5	Word2Vec (Domain) + Logistic Regression	0.8371	0.8694
6	Word2Vec (Domain) + XGBoost	0.8591	0.8886
7	Word2Vec (Domain+Pretrained) + Logistic Reg	0.8672	0.8934
8	**Word2Vec (Domain+Pretrained) + XGBoost**	**0.8772**	**0.9021**
9	Word2Vec (Domain+Pretrained) + LightGBM	0.8740	0.8983
10	Word2Vec (Domain+Pretrained) + GBDT	0.8716	0.8968
11	Doc2Vec + Logistic Regression	0.6469	0.6363
12	Word2Vec (Domain + Pretrained) + CNN CNN (optimizer = 'adam', loss = 'categorical_crossentropy', batch_size=100, epoch = 10)	0.8947	0.8755

c. It's clear from the above results that the XGBoost model with word2vec embeddings trained using transfer learning approaches (i.e. domain + pretrained embeddings) tend to perform the best, yielding an F1-Score of 0.9021, and accuracy of 0.8772.

7. Advanced Experiments - (SOTA - State Of The Art)

a. We further planned to use the current state of the art algorithm in NLP, **Bidirectional Encoding Representation from Transformer a.k.a. BERT**. It's the most recent breakthrough in NLP tasks, which has power of performing 12 NLP tasks. BERT was published by researchers at Google AI Language in 2018.

b. Hence for our binary classification task, we experimented with basic BERT architecture and the results were astonishing.

c. The accuracy on the same validation set went up to 0.917, and the F1 Score increased to

0.933. This was a 4% improvement in accuracy, while a 3% improvement in F1-Score, by training 3 epochs of the network.

d. Please note that for training the BERT model, we did not remove stopwords from the data because they are helpful in considering the context. While in Word2Vec + XGBoost case, including stopwords gave bad performance, as these words are just noise for this model.
e. After testing a few examples with BERT, we observed that BERT considers the context very carefully even if it's a very small reference.
f. We tested some racist and offensive examples, few such examples are: (**Predicted hate probability for the text is given as % in the table)**
g.

Tweet	XGBoost	BERT
u women always want equality until it comes to paying bills	60.01%	97.99%
we need a city of terrorists to save kashmir	13.50%	65.31%
All muslims are **not** terrorists	71.59%	23.42%

h.

8. Out of Time Validation - (With XGBoost model)

a. To validate our results on out of distribution datasets, we decided to test our model on Jigsaw Toxic Comment classification data. Link.
b. The dataset contains Wikipedia comments labeled as Toxic, Offensive, Identity hate, Threat, Obscene, Hate.
c. The task in this competition is multiclass classification but we can consider all the above labels as hate speeches.
d. Data size is ~160k out of which 16225 hate speeches, and the rest ~143k are neutral texts.
e. The model accuracy over this data is around 90%
f. The confusion matrix is as follows -

Actual \ Predicted	0 (Not hate speech)	1 (Hate Speech)
0	133256 (True Negatives)	7057 (False Positives)
1	10090 (False Negatives)	9168 (True Positives)

g. While diagnosing the bad predictions we observed that out of 160k, 14k are getting misclassified and most of them are very unclear and bad texts in terms of grammar.
h. Probable reasons behind such results are, it's the data from a different website(Wikipedia

page comments), these are all comments instead of tweets, so the use of slang is different than Twitter, also a lot of lingos and errors unlike the tweets which are clean with pretty much good grammar as twitter is a more widely public platform where people are concerned about their public image. While the Wikipedia comment section can be accessed by any anonymous identity.

9. API Development and Hosting

a. Our project aim is to create an open-source platform for businesses/ agencies/ governments to run sentiment analysis on their public social media handles (mainly twitter).
b. We also aim to provide a service where any custom text input can be tested out with its hate origin probability.
c. For achieving all of this we decided to build a simple API using Flask which provides all of these services faster and more efficiently.
d. Standard HTML, CSS practices were used for website creation, while we deployed the python code and ML model using Flask.
e. Main features in API -
    i. Custom Text Input testing
        1. Given a text input, we can generate the probability of hate speech with an accuracy of 93.33% (using BERT model)
    ii. Hashtag analysis
        1. Given a valid hashtag, API scrapes the latest n tweets for that hashtag and performs an evaluation on it using our deployed model. Finally generates a sorted list of tweets according to their hate probability.
        2. Here the input is a hashtag, no. of tweets, and date since you want to perform the evaluation upon
    iii. User analysis
        1. Given a valid twitter user ID, API scrapes the latest n tweets on the user's timeline and similar to the previous case generates a table of a sorted list of tweets according to their hate probability.
        2. Here the input is only the hashtag and no. of tweets to scrape
    iv. The API is hosted on AWS EC2 Virtual Machine and is publically accessible at http://hatespeechbert.ml:5000/
    v. (*It is advisable to check the probability of predictions rather than the predicted label, as the probability threshold can vary with respect to the context of the use case)*
    vi. Please note that while using a feature, make sure to erase text in other feature's white boxes.

10. Limitations

a. Time lag in processing real time requests on API due to model complexity
b. Text containing unpopular lingos and abbreviations are not in the familiar vocabulary of the model, so frequent occurrence of such words limit the model capability to consider its meaning and context of use.

11. Future Scope

a. Our experiments with BERT architecture for hate speech detection are unique and have not been practiced in the industry as well as in academia as of now. So we wish to take this work further for a publication in an NLP conference or journal.
b. The platform can be used by Cyber Police for quick analysis of social media platforms, especially around days of popular events such as Elections, Festivals, Cricket Matches.
c. Our platform can be used by businesses across the world to generate descriptive analytics insights on their service/ product launch reactions over social media platform. This could be helpful for deciding the marketing strategies for the product/ service promotion.