Research Paper on Hate Speech Detection Model

Monika Sharma

School of Computer Science and Engineering, Galgotias University, Greater Noida

monikasharma1899@gmail.com

Mohd. Kashaf Siddiqui

School of Computer Science and Engineering, Galgotias University, Greater Noida

siddiquikashaf786@gmail.com

Yogesh Pandey

School of Computer Science and Engineering, Galgotias University, Greater Noida

yogeshpandey403@gmail.com

Abstract: A Hate Speech Detection model is a model that identifies the hate provoking texts from the given input text. The primal and immediate objective of the said model would be to incessantly trace the social media and public forums like YouTube and Twitter for a textual content that might seem to be igniting hostility among the masses. Further on, the model would revalidate and classify the target as being provoking or not. It would also be using a keyword-based detection approach for posts.

---

1. Introduction

The use of social media platforms is increasing drastically over a few years, and with the numerous benefits of these platforms, comes the negative aspects as well. We often see a lot of hate provoking comments on such platforms, and these can sometimes be targeting on a particular section of the society, a particular community, caste, gender or even an individual. Not only these provoke hatred but can also cause a serious effect on a person's mental health. Also, if a larger section of society is involved, the online dispute on these social media platforms can also result in a huge outrage affecting the lives and property of many innocents. So, a person's hate speech can cause harm to the society on all: emotional, mental and economical levels.

To prevent these types of practices, there are various steps that are taken by the companies

that run these social media platforms: like assigning a separate team to monitor these types of comments and texts and remove them from their platform, or warn the user about it, or to take any other necessary action. But as these platforms and the audience to use it is increasing drastically, it is really a quite hard job to be able to monitor each and every text posted. Hence, if this entire process could be automated, it would then be a lot easier to monitor and take action against spreading such negativity through texts on the digital platforms.

But, Hate Speech Detection can be a very challenging task. This is because first we have to decide what should be the criteria for defining the Hate speech. Also, the speech might not be directly identifiable. For example, the speech might not have any bad word, but the sentiments of the text can be negative. Also, the perspectives of people vary. There can be texts which are considered as hate speech by some and are okay for others. And removing such texts can affect the users of that platform and can affect the company's business. Hence, trying to apply all these criteria in a model is a very challenging tasks. Also, if the decision is made by the Machine Learning Hate Detection model that works on a very complex underlying algorithm, without any human intervention, it is difficult to give reasons for why a particular text was detected as a hate speech. Also, every system has some limitations in detecting the hate speech from the text. Hence, no model can be termed as completely accurate.

---

2. Methods and Techniques used

In this section, we will discuss the Methods and Techniques used in the Hate Speech detection Model in order to identify the hate and violence inciting texts from different social media platforms.

2.1.Naïve Bayes Classifier

The Naïve Bayes Classifier is basically a probabilistic classification algorithm through which we can sort the labelled data into various classes. This classifier makes vital use of the naïve-Bayes theorem forming a core part of conditional probability.

As per our case, the naïve-Bayes classification model will take a dataset from any social media platform, which would be a document of large number of public comments that are both stemmed and lemmatizes and have undergone several other pre-processing operations. It would use 80 percent of the data for training and rest 20 percent for testing.

2.2.Bag of words for Sentiment Analysis:

Topic classification and Sentiment classification are two broadly categorized spectrums, and the bag of words model is extensively used for the purpose of solving the task of topic classification mainly.
It is also inferred that this methodology is used for the problems based upon sentiment classification, but the approach is not as efficient as it should be.

2.3.*Keyword-based approaches*:

Another methodology for identifying hate speech is using a keyword-based approach. Keyword-based approaches are fast and

straightforward to understand. To implement this method as an additional feature of our model, we would be using an API for accessing the data containing the hate speech keywords and phrases hatebase.org, for Hatebase maintains a database of derogatory terms for many groups across 95 languages. However, this model would be utilizing such keywords as black-markers so as to mark them with offensive/provoking intentions. Consequently, if the user is found making use of such words more than a predefined threshold value, his post would eventually be deleted from the feed; and if possible, the user might also get reported within a definite due time.

2.4. *Term Frequency/Inverse Document Frequency (TF-IDF):*

TF*IDF is an information retrieval technique that weighs a term's frequency (TF) and its inverse document frequency (IDF). Here, we calculate the TF and IDF values for the input data. Each word or term has its respective TF and IDF score. The product of the TF and IDF scores of a term is called the TF*IDF weight of that term, the higher the TF*IDF score (weight), the rarer the term, and vice versa.
Hence, TF-IDF method can be used to check the frequency of hate inciting words in the input text.
Currently existing works have used the method of classifying the posts available on social media by passing their term frequency-inverse document frequency (TF-IDF) values to multiple machine learning models. It has been observed that after tuning the model prior to its thorough evaluation, accuracy of about 95.6% is obtained upon the test data set.

3. Data Elicitation

The target of the Hate Speech detection model would be the words that incite hate, violence disrespect. In order to make the model know what words are to be considered under this category, we must provide the model with some exemplary data, i.e. the training data. Using this training data, our model will be trained to recognize the words that are not fit to be posted on the social media platforms. Collecting data for training a Hate Speech detection model is a challenging task. This is because the various social media platforms has strict data security policy and their data is not readily available to be used for the training purpose. Also, these data sets are not publicly available.

In this model, we have used data sets from the following sources:

3.1. Kaggle: Kaggle.com hosted a shared data set on detecting insulting comments. This dataset consists of 8,832 social media comments which are labeled as insulting or not insulting.

In order to use this dataset for training our model, we performed under-sampling on the data in order to balance it and then 80 percent of the results are used as training data and 20 percent as testing data.

4. Implementation, Experimental Setup and Results

The complete walkthrough of our approach to bringing this project into action follows the pathway as discussed below. All the details and tasks done henceforth in the following are not at all subjected to any kind of plagiarism. The chronological order of the implementation of all steps are described below:

4.1. IMPLEMENTATION

4.1.1. DATA GATHERING: We have tried to collect as much data as it would need to increase the precision of the predictions of our model. We have gathered data from Kaggle. The dataset is populated by the public tweets on Twitter and has a label attached to each one of them denoting whether they are provocative or not.

Link: a Twitter dataset for hate speech

Besides obtaining a dataset for training the model, we also required a list of all the slang words that are used on social media oftentimes. The link for the same is attached here

4.1.2. DATA PREPROCESSING: Since it is a cumbersome task to deal with a textual data as it is in a Natural Language Processing project. Hence, the first thing we need to do is to filter out the irrelevant data out of our dataset with the help of certain techniques.

The methods that we have made use of are explained as:

- **Tokenization** - It is a stage of pre-processing of data in which the textual content is divided or chopped into smaller constituent components, which are known as Tokens. These tokens might be in the form of numbers, characters, special symbols (such as '@' for twitter handles, emojis like :), ;), :0, :| and many more like these)

  Example: "*I am going out in the rain to get sick lol".*

  This statement can be broken down into simpler tokens which would be as:

  ["I", "am", "going" "out", "in", "the", "rain", "to", "get", "sick", "lol"]

  To perform the task of Tokenization upon our dataset, we have made use of the

Punkt, and *nltk.tokenize* modules from Natural Language ToolKit.

The punkt sentence tokenizer makes use of unsupervised algorithms at its core in order to construct a model for abbreviation words, along with the words that start sentences to derive the boundaries between different words within a sentence.

```
#using the punkt module upon the tweets
tokenized_pos_tweets = twitter_samples.tokenized("positive_tweets.json")
tokenized_neg_tweets = twitter_samples.tokenized('negative_tweets.json')
```

- **Stemming and Lemmatization -** Both of these operations are performed to get the original or root form of a word (which we obtained after performing Tokenization). A document can use multiple forms of a single word in the same sentence at different places, thus making it extremely difficult for the predicting model to get the results accurately.
  The primary goal of Stemming and Lemmatization operations is to obtain the root form of the words by reducing their inflectional forms.

  Example: *am, are, is ----> can be converted into "be"*

  like, liking, likes, liked ---> can be converted into "like"

  play, playmate, playful, playing --->can be converted into "play"

  Thus, in this way these different forms of words are being reduced into their common/ root forms by with the help of above mentioned processes.

  Now to apply these techniques to our dataset, we have used two modules. One of which aims at Stemming and the other one handles Lemmatization of the text.

For stemming, we used *"porterstemmer"* module

```
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize
ps = PorterStemmer()
```

- **Removal of Stop Words -** Stop words are the common words that do not play much role in the analysis of a text and hence, they can be completely ignored thereby reducing the size of the dataset and hence the number of tokens to process. Example of such words include *is, am ,a, an, the, are, for* etc.

```
from nltk.corpus import stopwords
nltk.download('stopwords')
from nltk.tokenize import word_tokenize

text = "Nick likes to play football, however he is not too fond of tennis."

text_tokens = word_tokenize(text)

tokens_without_sw = [word for word in text_tokens if not word in
stopwords.words()]

print(tokens_without_sw)
```

OUTPUT: ['Nick', 'likes', 'play', 'football', ',', 'however', 'fond', 'tennis', '.']

4.1.3. ***<u>APPLYING BAG OF WORDS (BoW):</u>*** The above operations are done for pre-processing for the data so that it could be fed to the predicting algorithm. After all

the above mentioned methods are done we now implement the Bag of Words (BoW) model.

A bag-of-words is a representation of text that describes the occurrence of words within a document. The occurrence of words is represented in a numerical feature. It is a way of extracting features from the text for use in modelling, such as with machine learning algorithms. The approach is very simple and flexible and can be used for extracting features from documents. But there is some complexity on two cases i.e., one is on designing the vocabulary of known words and the other is on scoring the presence of known words.

Explanation of bow model:

We have a text corpus as containing multiple observations/documents as:

S.No.	**Text**	**Class**
01	They are violent	0
02	This is good	1
03	Its dusty	0

We want to build an NLP model which would be able to predict the classes to which these or external documents belong.

- We construct a vocabulary which is basically a set of words that we already know the meaning of.
- We take a new input, and count the number of words present in the same by comparing it with the vocabulary.
- Vocab = [They, are, violent, this, is, good, its, dusty]
- Then, we draw the feature matrix as follows:

S.No	**They**	**Are**	**Violent**	**This**	**Is**	**Good**	**It**	**dusty**
01	1	1	1	0	0	0	0	0
02	0	0	0	1	1	1	0	0
03	0	0	0	0	1	0	1	1

And the target vector is [0]

[1]

[0]

- Now, when we encounter a text, we compare the words present in that with the feature vector and thus determine which word would finally be able to categorize the entire text as being into +ve or –ve class.

4.1.4. ***TF-IDF:*** Term Frequency-Inverse Document Frequency (TF-IDF) is a method that is primarily used in keyword-based approach to perform classification tasks.

To get the frequency of occurrence of a keyword in a Tweet, we calculate the Term's/word's frequency and it's Inverse frequency. The product of TF and IDF score is then analysed. The higher the value of this score, the lower the occurrence of the term in the concerned tweet.

We use this formula to calculate the overall weight of occurrence of a word, and then determine the significance of that word in categorizing the tweet.

$$W_{t,d} = TF_{t,d} \log (N/DF_t)$$

4.1.5. ***NAIVE-BAYES CLASSIFICATION ALGORITHM:*** Naive Bayes algorithm is a supervised machine learning algorithm that is used in classification tasks, and uses the concept of conditional probability. The detailed working of the same is as below:

Naive bayes algorithm:

i. A training data set of 'D' consisting of documents that are classified to +ve and –ve classes.

ii. Positive (01) = No. of objects in (+) / total no.
Negative (00) = No. of objects in (-) / total no.

iii. Find total words (significant words) that are present in both classes.
n_p = words in (+) class
n_n = words in (-) class

iv. Finding the conditional probability of keyword occurrence in a concerned class:
P(word1/(+) class) = n_i / P(01)
P(word1/(-) class) = n_i / N(00)

v. Now, a new document is classified on the basis of the conditional probability for classes P(01) and N(00).

Resulting sentiment analysis:

- Assume a document 'a'.
- Set of classes C = { c_1, c_2, c_3 }
- A set of 'm' documents that belong to a specific class out of these known ones.

Now, training the model by applying algorithm:
For a document 'a', class c_1:

$P(c_1/a) = [P(a/c_1) * P(c_1)] / P(a)$
$= P (t_1 . t_2 . t_3 / c_1)$

And, $P(c_1)$ is the total probability of the c_1 class.

Finally to conclude, the implementation of all the above methods used in completing our project are linked here

---

4.2.Comparative analysis

- ***Comparative analysis of feature extraction techniques:*** The experiments that are conducted with an aim of text manipulation and obtaining inferences, the datasets that are available on the internet for such tasks are mostly unstructured and unfiltered, and hence they require a tedious amount of preprocessing.
  Extracting information from text helps in analyzing the text data for numerous applications and finding out critical features. Some of the main preprocessing and feature extraction techniques along with their comparative analysis are explained as:

❖ *Bag of Words (BoW) Model:* The Bag of Words model is one of the simplest yet effective feature extraction methods. The bag of words is the most common and the simplest among all the other feature extraction methods; it forms a word

Presence feature set from all the words of an instance. This model is known as the “Bag” because the method doesn't care about how many times a word occurs or the order of the words, all what matters is whether the word is present in a list of words.

- ➔ For feature extraction purposes most of the models make use of the Word2Vec approach which was originally developed by Google.
- ➔ This approach is advantageous and is great at the same time in finding the context of a specific word by looking up that word in a dictionary.
- ➔ Although this method provides accuracy upto a certain level, but the speed of processing the dataset is significantly slower and follows a lazy approach.

However, the Bag of Words model is a far more widely used, effective and immediate alternative to the Word2Vec approach. The only caveat this model provides is that the context of a word is not provided by any dictionary or external database (for referencing), and due to this reason the complexity of this model is far more optimal as compared to the Word2Vec technique.

Though the BoW becomes less fulfilling when it comes to considering the words that appear most number of times in a corpus. The words having the highest frequency become dominant in the numerical feature vector. This may lead to poor performance of the model as a whole. To avoid this situation, we make use of TF-IDF technique alongside the BoW model so that the frequency of the words is rescaled by considering how frequently the words occur in all the documents. Due to this, the scores for frequent words are also frequent among all the documents are reduced.

Sr. No.	Feature Extraction Method	Classification Algorithm	Accuracy	Log Loss
1	Bag of Words	Logistic Regression	48%	1.65
	Bag of Words	Random Forest Classifier	50%	1.44
2	TF-IDF	Logistic Regression	46%	1.50
	TF-IDF	Random Forest Classifier	51%	1.35
3	Word2Vec	Logistic Regression	53%	1.32
	Word2Vec	Random Forest Classifier	57%	1.21

Table 1. Accuracy and Log Loss comparison for different methods

- ***Comparative analysis of the classification model:***

❖ *Naive-Bayes Classifier:* This is the main classification algorithm that we have made use of for segregating the texts into different classes. Approximately half of the methods for text analysis make use of Support Vector Machines (SVMs) and Decision Tree ensemble classifiers.

❖ SVM algorithm represents the text document as a vector where the dimension is the number of distinct keywords. If the document size is large then the dimensions are enormous of the hyperspace in text classification which causes high computational cost. The feature extraction and reduction can be used to reduce the dimensionality.

❖ On the other hand, Naive Bayes is a fast and yet gives a better accuracy as compared to Logistic Regression and KNN models (except for the SVMs), so it forms a base-line in text classification.

❖ Naive Bayes models the distribution of the documents in each class using a probabilistic model with independence assumptions about the distributions of different terms.

- To further enhance the efficiency of the classifier, the dataset is further filtered out and the redundant documents are removed before the data is fed into the model. Doing this decreased the correlation between the features and hence the makes it easier for the model to identify/classify the documents.

---

5. Conclusion

As hate speech is still a societal problem, the want for automated hate speech detection structures will become greater apparent. We offered the modern-day methods for this venture in addition to a brand new gadget that achieves affordable accuracy. We additionally proposed a brand new technique that could outperform present structures at this venture, with the brought advantage of advanced interpretability. Given all of the demanding situations that remain, there's a want for greater studies in this problem, which includes each technical and sensible matters.

---

6. Acknowledgements

We thank Mr. Sudeept Kumar Gupta and Mr. Karthick R for reviewing this paper and for their helpful feedback on the work.

---

7. References

To help ourselves with a kick-start for this project, there are certain resources and white papers which we have taken our references from. Those along with their links are mentioned below:

7.1. Hate speech detection: Challenges and solutions:
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6701757/

7.2. Hate Speech Detection: A Solved Problem? The Challenging Case of Long Tail on Twitter:
https://arxiv.org/pdf/1803.03662.pdf

7.3. Effective hate-speech detection in Twitter data using recurrent neural networks:
https://www.researchgate.net/publication/326517890_Effective_hate-speech_detection_in_Twitter_data_using_recurrent_neural_networks

7.4. Hate Speech Detection: Challenges and Solutions
https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0221152