# An Efficient novel approach Machine learning paradigm for Detecting Hate Speech and Offensive Language on Twitter API towards N-gram and TFIDF

**[1]B. Bhaskar, [2]Alladi Sureshbabu**

[1]M.Tech (CSE), [2]Professor
Department of CSE, JNTUACE, Anantapur, A.P.

## ABSTRACT:

Toxic online content (TOC) has become a significant problem in current day's world due to uses of the internet by people of distinct culture, social, organization and industries background like Twitter, Facebook, WhatsApp, Instagram, and telegram, etc. Even now, there is lots of work going on related to single-label classification for the text analysis and to make less comparative to errors and more efficient. But in recent years, there is a shift towards the multi-label classification, which can be applicable for both text and images. But text classification is not much popular among the researchers when compared to the grading for images. So, in this work, by using the dataset which is going to be a short message, to train and develop a model which can tag multiple labels for the messages. Hate speech, and offensive language is a key challenge in automatic detection of toxic text content. In this paper, this work involves with term frequency–inverse document frequency (Tf-Idf), Random forest, Support Vector Machine (SVM) approaches for automatically classify tweets. After tuning the model giving the best results, it achieves an Efficient accuracy for evaluating test data analysis. This work also moderate and encapsulate paradigms which will communicate and working between the user and Twitter API. Instead of using the traditional techniques like Bag of words or word counter, a new technique which uses Tf-Idf is built to find the similarity, and the text is transformed into the vectors using Tf-Idf, and this is used to train the model using supervised learning technique along with the labels from the dataset. The accuracy of the model is quite good and more efficient with better results.

**Keywords:** Twitter, toxic text, Tf-Idf, machine learning

## I. INTRODUCTION

Multi-label classification is one of the most difficult and interesting technique in a classification where it generates many classes for the input. As text falls into a natural language process and the classifier cannot work on natural language, need to transform them into some other format so the classifier can understand. Many text transformation techniques are used for this purpose that is used in common they are, bow (bag of words) and word embedding, which includes a glove, word2vec. These techniques are used to transform and work with text/natural language. In this work, a multi-label(n-grams) classifier using machine learning for our short message's dataset is going to implement. Short messages are used to communicate in our daily life. Building our model using pipelining technique/ pipelines to automate the workflows/process and annotation to handle our text. The system is implemented in the following steps: Data Collection/Generation**:** The tweets related to Hate Speech and Offensive languages are retrieved using the Twitter API and Tweepy module of Python. Data Preprocessing: This step involves cleaning and simplifying the data collected by applying various preprocessing techniques such as removal of stop words, handling missing values, removal of irrelevant characters, etc. Feature Extraction: The feature extraction step identifies the features of the four classes used in this work. The feature extractor function is responsible for generating feature vectors. Feature Extraction improvement: The most important features are considered, which have similar context are manually added to the feature vector. This helps the model to be trained in a better way and classify the tweets with higher accuracy. Prediction: The model is now capable of making predictions of which class the tweet belongs to with higher accuracy than the baseline model. The tweet is given as an input to the model, which gives the label of the tweet as the output.

# II. BACKGROUND WORK

## Existing system:

Unigrams and Pragmatic approaches are used in the hate speech detection, and it becomes a major problem in current day's world due to the uses of internet by people of distinct culture, social, organization and industries background on Twitter, Facebook, WhatsApp, Instagram, and telegram, etc.

Even now, there is lots of work going on related to single-label classification for the text analysis and to make less comparative to errors and more efficient. So, this is the reason behind people facing a lot of problems of HSOL on sentiment analysis. The disadvantage of existing is Critical to find out the toxic text content problem for all perspectives.

# III. PROPOSED SYSTEM

The proposed system balanced and address the Tf-Idf, NLP, SVM, and Random Forest to the existing problem of hate speech and offensive language with all sample inputs based on sentiment analysis using Twitter API. The advantage of the system will automatically detects toxic text content and to avoid the hateful, offensive word from the tweets.

### Methodology:

The methodology of a system improves on the baseline model or paradigms by introducing a new technique to identify the similarity of the hate speech sentences and offensive languages, and it addresses the issues of the existing baseline model. The main aim of the system is to increase the accuracy and more reliable in finding the similarity of the sentences on hate speech and offensive. The present system employs the Tf-Idf a Natural Language processing technique. The Tf-Idf vectorizes the text, which is in Natural Language into a vector is used by the Machine Learning model to find the similarity of hate speech and offensive languages. The vector is generated by assigning some weights to the words by using the Tf-Idf technique, which have the productivity of two parts, is based on the frequency and the Inverse document frequency. The vector is also normalized because if the number of documents or the questions increased, the weight also increases and becomes difficult to perform operations on it by using normalization, it reduces the range and also weights in the vector. The model is now able to identify the similarity between the sentences, and it helps to increase the accuracy of the model more effectively.

## Random forest prediction Algorithm:

1. Randomly select **"k"** features from total **"m"** features.

   Where, **k << m.**

2. Among the **"k"** features, calculate the node **"d"** using the best split point.

3. Split the node into **daughter nodes** using the **best split**.

4. Repeat **1 to 3** steps until "l" number of nodes has been reached.

5. Build forest by repeating steps **1 to 4** for "n" number times to create **"n" number of trees**.

The beginning of random forest algorithm starts with randomly selecting **"k"** features out of total **"m"** features. In the algorithm, you can observe that we are randomly taking features and observations. In the next stage, it uses randomly selected **"k"** features to find the root node by using best split approach. The next stage, it will be calculating the daughter nodes using same best split approach. Will the first 3 stages until we form a tree with root node and having target as the leaf node. Finally, it repeats 1 to 4 stages to create **"n"** randomly created trees. This process randomly created trees forms the **random forest.** On top of that, it uses pipeline which automates the workflow of the entire model along with linear SVM model. The trained model is successful in generating multiple labels to the input text/ messages. The threshold value is calculated by using mean of probabilities for classes when input is given to the model.

## Dataset generated in the form of CSV format:

| sno | count | hate speech | offensive language | neither | class | tweet |
|---|---|---|---|---|---|---|
| 0 | 3 | 0 | 0 | 3 | 2 | !!! RT @mayasolovely: As a woman you shouldn't complain about cleaning up your house. & as a man you should always take the trash out... |
| 1 | 3 | 0 | 3 | 0 | 1 | !!!!! RT @mleew17: boy dats cold...tyga dwn bad for cuffin dat hoe in the 1st place!! |
| 2 | 3 | 0 | 3 | 0 | 1 | !!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby4life: You ever fuck a bitch and she start to cry? You be confused as shit |
| 3 | 3 | 0 | 2 | 1 | 1 | !!!!!!!!! RT @C_G_Anderson: @viva_based she look like a tranny |
| 4 | 6 | 0 | 6 | 0 | 1 | !!!!!!!!!!!!! RT @ShenikaRoberts: The shit you hear about me might be true or it might be faker than the bitch who told it to ya  |
| 5 | 3 | 1 | 2 | 0 | 1 | !!!!!!!!!!!!!!!!!!"@T_Madison_x: The shit just blows me..claim you so faithful and down for somebody but still fucking with hoes! 😂😂😂" |
| 6 | 3 | 0 | 3 | 0 | 1 | !!!!!!"@__BrighterDays: I can not just sit up and HATE on another bitch .. I got too much shit going on!" |

## DOWNLOADING THE NLTK

The following snippet is an example used to download the NLTK library, which is used for pre-processing the text in the SMS dataset.

```
In [5]: nltk.download('stopwords')
        nltk.download('punkt')

        [nltk_data] Downloading package stopwords to
        [nltk_data]     C:\Users\User\AppData\Roaming\nltk_data...
        [nltk_data]   Package stopwords is already up-to-date!
        [nltk_data] Downloading package punkt to
        [nltk_data]     C:\Users\User\AppData\Roaming\nltk_data...
        [nltk_data]   Package punkt is already up-to-date!

Out[5]: True
```

## DATASET BEFORE PREPROCESSING

This is the sample dataset, and its labels format we are using in this work, and it contains text along with the special characters and numbers.

| | ts | user | text | class | key |
|---|---|---|---|---|---|
| 0 | 1.503303e+09 | Balaemar | I have to pick up my car from the garage tomor... | 1 | 1503303350U035FRUCY |
| 1 | 1.503302e+09 | Ragaenys | I won't be here tomorrow, one day vacation | 2 | 1503301710U4A2FRAQ4 |
| 2 | 1.503296e+09 | Myke | Missed connection in Zurich. Will be about 5-1... | 1 | 1503296123U0MGNKETU |
| 3 | 1.503260e+09 | Drevyn | Enjoy! | 8 | 1503259722U035B8PRU |
| 4 | 1.503258e+09 | Gaelralis | I am away for 2 weeks in iceland :flag-is: | 2 | 1503258060U0HLAK1T6 |

## DATASET AFTER PRE-PROCESSING

This is the dataset format after it is pre-processed when all the special characters and numbers are removed from the dataset.

```
df_messages.head(5)
```

```
[.] Number of training samples: 1719
```

Out[7]:

| | ts | user | text | class | key |
|---|---|---|---|---|---|
| 0 | 1.503303e+09 | Balaemar | I have to pick up my car from the garage tomor... | 1 | 1503303350U035FRUCY |
| 1 | 1.503302e+09 | Ragaenys | I won't be here tomorrow, one day vacation | 2 | 1503301710U4A2FRAQ4 |
| 2 | 1.503296e+09 | Myke | Missed connection in Zurich. Will be about 5-1... | 1 | 1503296123U0MGNKETU |
| 3 | 1.503260e+09 | Drevyn | Enjoy! | 8 | 1503259722U035B8PRU |
| 4 | 1.503258e+09 | Gaelralis | I am away for 2 weeks in iceland :flag-is: | 2 | 1503258060U0HLAK1T6 |

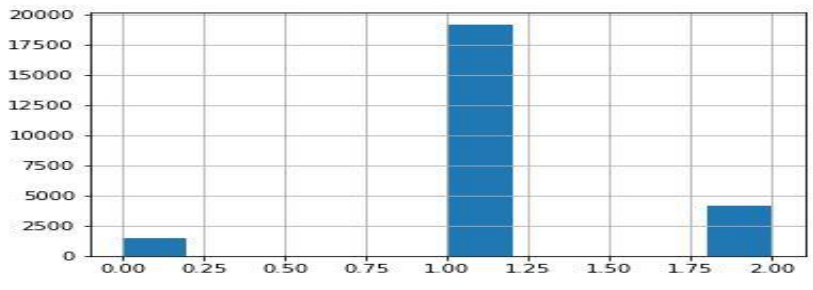
# IV. RESULTS AND ANALYSIS

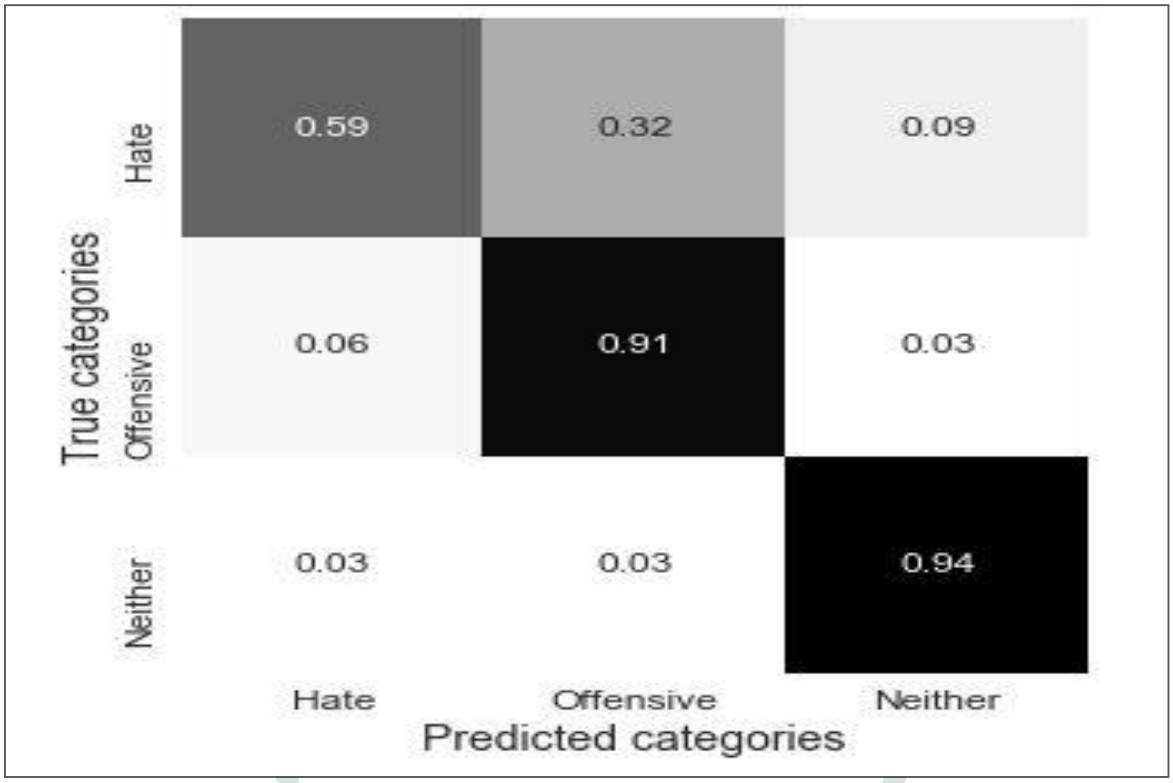
The random forest classifier model is trained only with a subset of the actual dataset. Only 18,000 short messages and their labels have been used for training due to lack of dataset. The model performs above expectations even when a subset of the whole dataset is used. The trained model is tested with messages from the testing dataset. The generated labels for the texts are as follows:

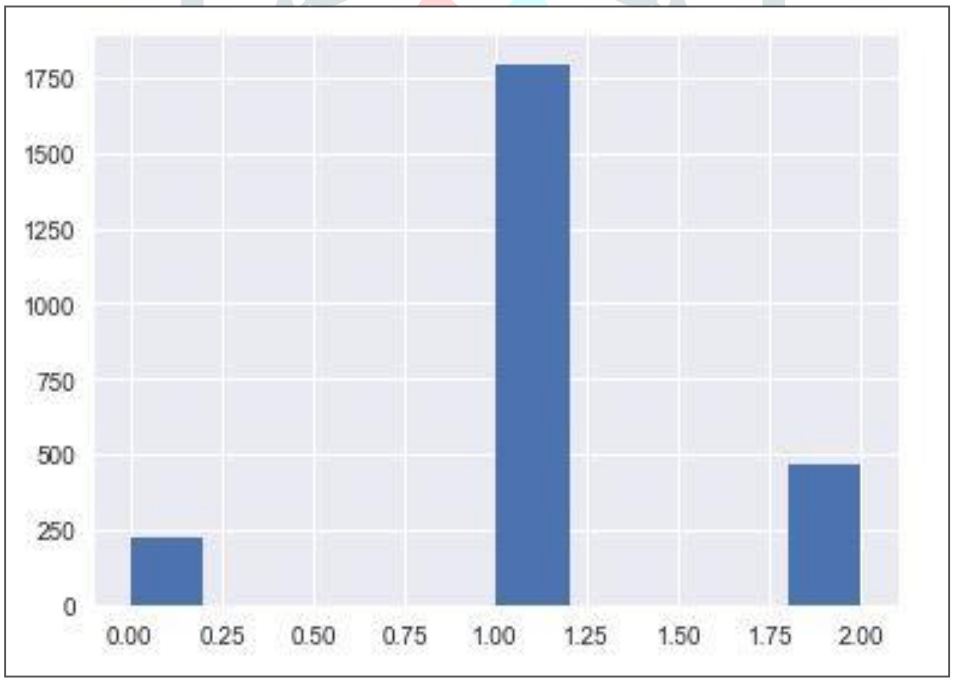
```
0 - hate speech
1 - offensive  language
2 - neither

tweet = raw tweet text

In [8]: df['class'].hist()
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0xb5b0e80ba8>
```






## V. CONCLUSION

The objective of this paper, is to contribute a solution of an efficient novel approach for prediction of hate speech and offensive language on Twitter API using ML. And n-gram features weighted with TF-IDF data exploration and determined comparative analysis, (RF)Random forest and SVM on various sets of future values and model hyperparameters. The results showed that Random Forest performs better with the optimal n-gram range from 1 to 3 for the L2 normalization of TF-IDF. The word embedding technique (Term Frequency inverse Document Frequency) has advantages over the traditional and most common bow technique (Bag of Words) which does not extract the structure of words but the frequency of words.

On top of that, it uses a pipeline which automates workflow along with the Random Forest model. The trained model is successfully generating multiple labels to the input text/ messages on HSOL and using a threshold value to find the value/ probability for the labels to be assigned to the messages. The threshold value is calculated by using mean of probabilities for classes when input is given to the model. In future work, to build a strong dictionary of HSOL paradigms that can be Moderated along with a uni-gram dictionary paradigm, to improve the efficiency of detecting hateful and offensive online texts. We have to make a quantitative and quality research study of the presence of hate speech among the different genders, age groups, and regions, etc. This work requires the users to manually add features which have a similar context to the most informative features of the provided dataset. The process would become much easier and efficient, if the addition of features can be automated without any human involvement. This would make process of training the model faster and optimal.

## REFERENCES


1. Zephoria.com, 2018. [Online]. Available: https://zephoria.com/top-15-valuable-facebook-statistics/. [Accessed: 22- Jun- 2018].
2. Twitter Usage Statistics Internet Live Stats, Internet live stats. com,2018. [Online]. Available: http://www.internetlivestats.com/twitter-statistics/. [Accessed: 22- Jun- 2018].
3. S. Hinduja and J. Patchin, "Bullying, Cyberbullying, and Suicide," Archives of Suicide Research, vol. 14, no. 3, pp. 206-221, 2010.
4. Neethu M S and Rajasree R, "Sentiment Analysis in Twitter using Machine Learning Techniques" published in Computing, Communications and Networking Technologies (ICCCNT), 2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT).
5. Agarwal, A., Xie, B., Vovsha, I., Rambow, O. and Passonneau, R. 2011, "Sentiment Analysis of Twitter Data" in LSM 2011.
6. M. Bouazizi and T. Ohtsuki, "Sentiment Analysis: from Binary to Multi-Class Classification - A Pattern-Based Approach for Multi-Class Sentiment Analysis in Twitter," in Proc. IEEE ICC, pp. 1–6,May 2016.
7. Z. Jianqiang and G. Xiaolin, "Comparison Research on Text Pre-processing Methods on Twitter Sentiment Analysis " in *IEEE Access*, vol. 5, pp. 2870-2879, 2017.
8. Mondher Bouazizi and Tomoaki Ohtsuki, "A Pattern-Based Approach for Multi-Class Sentiment Analysis in Twitter", in *IEEE Access*, vol. 5, pp. 20617-20639, 2017.