A Project Report

on

HATE SPEECH DETECTION – A Comparative Study

Submitted in partial fulfillment of the

requirement for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

Under The Supervision of
Mrs. Sonia Kukreja
Assistant Professor
Department of Computer Science and Engineering

Submitted By

19SCSE1180068 – Jagrit Popli

19SCSE1180012 – Karnica Vineet Katiyar

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA, INDIA

DECEMBER - 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the project, entitled "Hate Speech Detection – A Comparative Study" in partial fulfillment of the requirements for the award of the BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of JULY-2021 to DECEMBER-2021, under the supervision of Mrs. Sonia Kukreja, Assistant Professor, Department of Computer Science and Engineering of School of Computing Science and Engineering , Galgotias University, Greater Noida .

The matter presented in the project has not been submitted by me/us for the award of any other degree of this or any other places.

19SCSE1180068 Jagrit Popli

19SCSE1180012 Karnica Vineet Katiyar

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor

(Mrs. Sonia Kukreja, Assistant Professor)

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of 19SCSE1180068 Jagrit Popli 19SCSE1180012 Karnica Vineet Katiyar has been held on _____ and his/her work is recommended for the award of BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING.

Signature of Examiner(s) Signature of Supervisor(s)

Signature of Project Coordinator Signature of Dean

Date:

Place:

ACKNOWLEDGEMENT

We would like to express my sincere gratitude to several people for supporting me throughout my project . First, We wish to express my sincere gratitude to my Project guide , Mrs. SONIA KUKREJA (Assistant Professor), for his enthusiasm, patience, insightful comments, helpful information, practical advice and unceasing ideas that have helped me tremendously at all times . Without his support and guidance, this project would not have been possible. We also wish to express my sincere thanks to GALGOTIAS UNIVERSITY for accepting us into the graduate program. In addition, we are thankful to our parents whose help at every stage has made this even easier. We are grateful to each other's wealthy contribution without which this project was unimaginable .

Thanks for all your encouragement!

ABSTRACT

The increasing use of social media and information sharing has given major benefits to humanity. With the enhancement in technology , social media showed immense growth but with the emerging trends there also comes the misuse of the freedom of speech , i.e. spread of hate content . These online abominations lead to real life consequences, such as escalation of fear and hate throughout communities . It is high time to realize and find viable solutions in order to control the same .

It is high time to develop solution for the existing problem . The proposed idea is to invent a tool for hate speech detection whose purpose is to pick out the abusive language and slangs determining the context of the speech on different datasets. In order to develop the same we are going to make use of the python based natural language processing(NLP) machine learning techniques to accomplish various tasks by training them through data .To remove the unwanted content text pre-processing technique is applied where we remove punctuation, stopwords, stemming and removal of urls, the processed data is passed for feature extraction. Here we are going to include sentimental analysis to differentiate the content into polarities and thus finally detecting the hate content.

The results clearly show that differentiating hate speech and offensive language is a challenging task. It also indicates the benefits of using the proposed features, and provides a valuable resource for detecting the problem of toxic language on twitter. Although a detailed analysis of the features as well as errors could lead to more robust feature extraction methods and also help us in solving the existing challenges in this field.

TABLE OF CONTENTS

List of Table

List of Figures

Acronyms

TF - IDF	Term Frequency-Inverse Document Frequency
SVM	Support Vector Machine
FE	Feature Extraction
LR	Logistic Regression
NB	Naïve Bayes
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

CHAPTER-1

INTRODUCTION

Hate crimes are unfortunately nothing new in society. However, social media and other means of online communication have begun playing a larger role in hate crimes. For instance, suspects in several recent hate-related terror attacks had an extensive social media history of hate-related posts, suggesting that social media contributes to their radicalization. In some cases, social media can play an even more direct role; video footage from the suspect of the 2019 terror attack in Christchurch, New Zealand, was broadcast live on Facebook. Vast online communication forums, including social media, enable users to express themselves freely, at times, anonymously. While the ability to freely express oneself is a human right that should be cherished, inducing and spreading hate towards another group is an abuse of this liberty.

Detecting hate speech is a challenging task, however. First, there are disagreements in how hate speech should be defined. This means that some content can be considered hate speech to some and not to others, based on their respective definitions. We start by covering competing definitions, focusing on the different aspects that contribute to hate speech.

The proposed solutions employ machine learning techniques to classify text as hate speech. One limitation of these approaches is that the decisions they make can be opaque and difficult for humans to interpret why the decision was made. This is a practical concern because systems that automatically censor a person's speech likely need a manual appeal process. To address this problem, we propose a new hate speech classification approach that allows for a better understanding of the decisions and show that it can even outperform existing approaches on some datasets. Some of the existing approaches use external sources, such as a hate speech lexicon, in their systems. This can be effective, but it requires maintaining these sources and keeping them up to date which is a problem in itself. Here, our approach does not rely on external resources and achieves reasonable accuracy.

1.1 FORMULATION OF PROBLEM

The term ‘hate speech’ was formally defined as ‘any communication that disparages a person or a group on the basis of some characteristics (to be referred to as types of hate or hate classes) such as race, colour, ethnicity, gender, sexual orientation, nationality, religion, or other characteristics’. Building effective counter measures for online hate speech requires as the first step, identifying and tracking hate speech online. For years, social media companies such as Twitter, Facebook, and YouTube have been investing hundreds of millions of euros every year on this task but are still being criticised for not doing enough. This is largely because such efforts are primarily based on manual moderation to identify and delete offensive materials. The process is labour intensive, time consuming, and not sustainable or scalable in reality.

A large number of research has been conducted in recent years to develop automatic methods for hate speech detection in the social media domain. These typically employ semantic content analysis techniques built on Natural Language Processing (NLP) and Machine Learning (ML) methods, both of which are core pillars of the Semantic Web research. The task typically involves classifying textual content into non-hate or hateful, in which case it may also identify the types of the hate speech. Although current methods have reported promising results, we notice that their evaluations are largely biased towards detecting content that is non-hate, as opposed to detecting and classifying real hateful content.

1.2 PROPOSED IDEA

It is high time to develop solution for the existing problem . The proposed idea is to invent a tool for hate speech detection whose purpose is to pick out the abusive language and slangs determining the context of the speech on different datasets. In order to develop the same we are going to make use of the python based natural language processing(NLP) machine learning techniques to accomplish various tasks by training them through data .To remove the unwanted content text pre-processing technique is applied where we remove punctuation, stopwords, stemming and removal of urls, the processed data is passed for feature extraction. Here we are going to include sentimental analysis to differentiate the content into polarities and thus finally detecting the hate content.

‘

1.3 ARCHITECTURAL DIAGRAM FOR PROPOSED SYSTEM

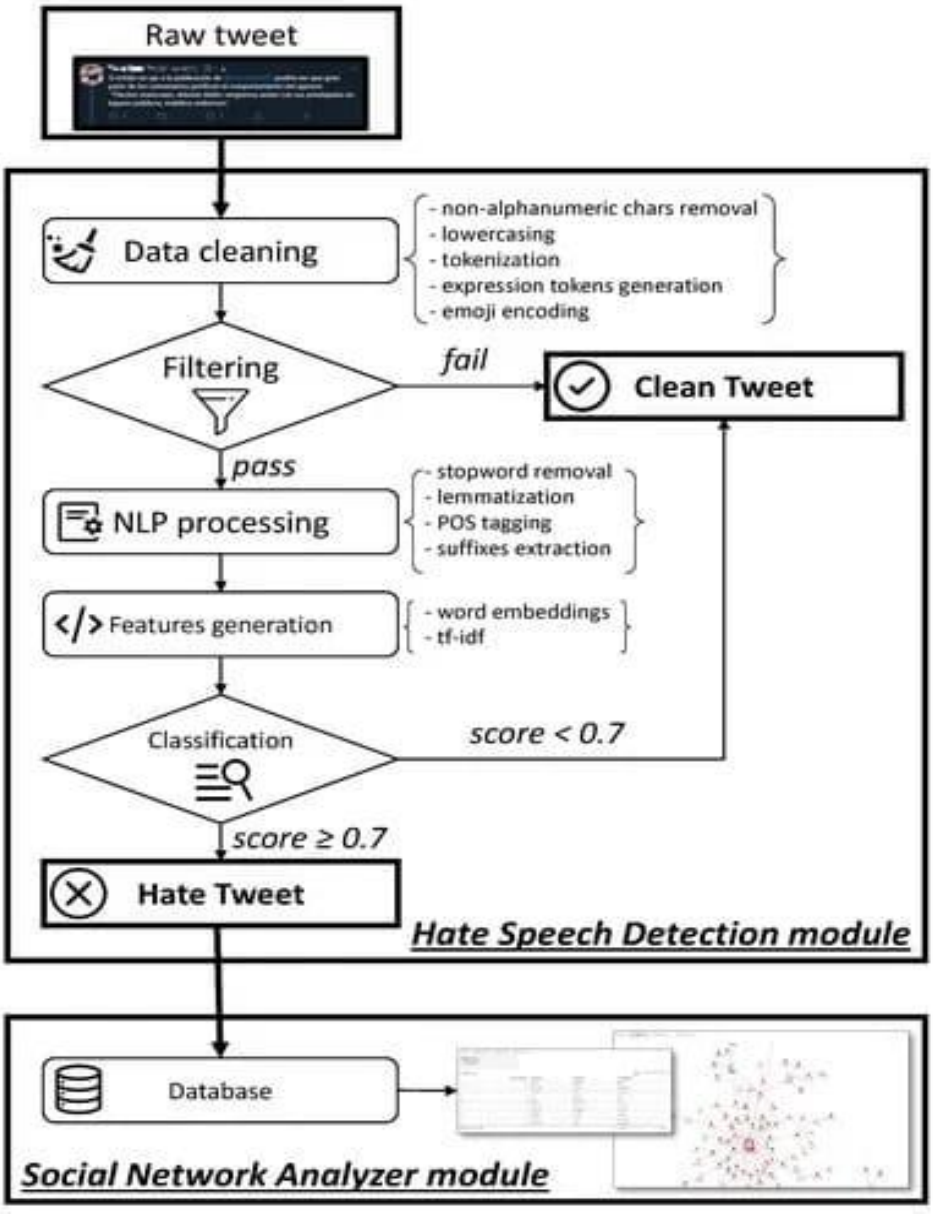

FIG 1 ARCHITECTURE DIAGRAM

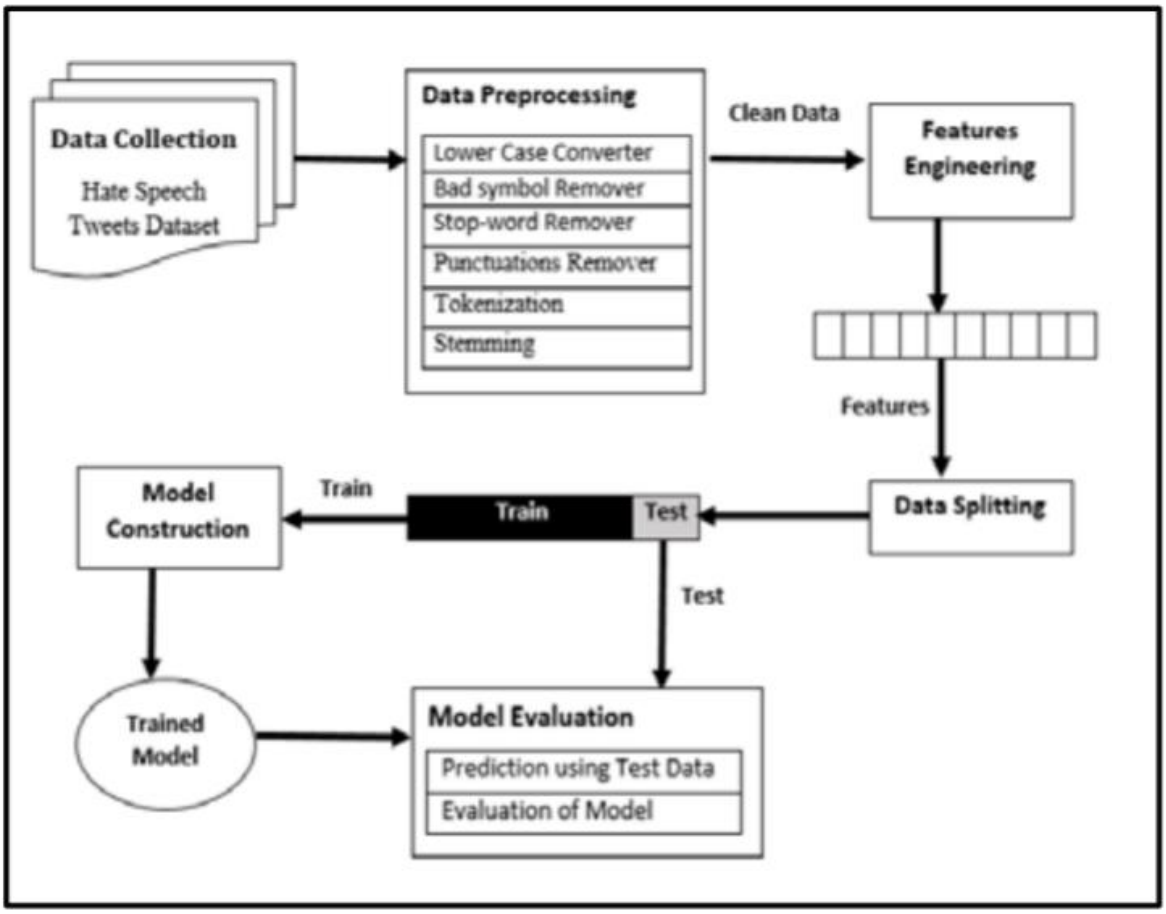

FIG 2 SYSTEM OVERVIEW

CHAPTER 2
LITERATURE SURVEY

Title: Automated Hate Speech Detection and the Problem of Offensive Language

Authors:
Thomas Davidson (Cornell University)
Dana Warmsley (Cornell University)
Michael Macy(Cornell University)
Ingmar Weber(Hamad Bin Khalifa University)

Year of Publications:

2017-05-03

ABSTRACT:
A key challenge for automatic hate-speech detection on social media is the separation of hate speech from other instances of offensive language. Lexical detection methods tend to have low precision because they classify all messages containing particular terms as hate speech and previous work using supervised learning has failed to distinguish between the two categories. We used a crowd-sourced hate speech lexicon
to collect tweets containing hate speech keywords. We use crowd-sourcing to label a sample of these tweets into three categories: those containing hate speech, only offensive language, and those with neither. We train a multi-class classifier to distinguish between these different categories. Close analysis of the predictions and the errors shows when we can reliably separate hate speech from other offensive language and when this differentiation is more difficult. We find that racist and homophobic tweets are more likely to be classified as hate speech but that sexist tweets are generally classified as offensive. Tweets without explicit hate keywords are also more difficult to classify.

CONCLUSION:

If we conflate hate speech and offensive language then we erroneously consider many people to be hate speakers (errors in the lower triangle of Figure 1) and fail differentiate between commonplace offensive language and serious hate speech (errors in the upper triangle of Figure 1). Given the legal and moral implications of hate speech it is important that we are able to accurately distinguish

between the two. Lexical methods are effective ways to identify potentially offensive terms but are inaccurate at identifying hate speech; only a small percentage of tweets flagged by the Hatebase lexicon were considered hate speech by human coders.4 While automated classification methods can achieve relatively high accuracy at differentiating between these different classes, close analysis of the results shows that the presence or absence of particular offensive or hateful terms can both help and hinder accurate classification.

Title: Deep Learning Models for Multilingual Hate Speech Detection

Authors: Sai Saketh Aluru , Binny Mathew , Punyajoy Saha , and Animesh Mukherjee

Year of Publications: 14 Apr 2020

ABSTRACT:
Hate speech detection is a challenging problem with most of the datasets available in only one language: English. In this paper, we conduct a large scale analysis of multilingual hate speech in 9 languages from 16 different sources. We observe that in low resource setting, simple models such as LASER embedding with logistic regression performs the best, while in high resource setting BERT based models perform better. In case of zero-shot classification, languages such as Italian and Portuguese achieve good results. Our proposed framework could be used as an efficient solution for low-resource languages. These models could also act as good baselines for future multilingual hate speech detection tasks. We have made our code and experimental settings public for other researchers.

CONCLUSION:
In this paper, we perform the first large scale analysis of multilingual hate speech. Using 16 datasets from 9 languages, we use deep learning models to develop classifiers for multilingual hate speech classification. We perform many experiments under various conditions – low and high resource, monolingual and multilingual settings – for a variety of languages. Overall we see that for low resource, LASER + LR is more effective while for high resource BERT models are more effective. We finally suggest a catalogue which we believe will be beneficial for future research in multilingual hate speech detection.

Title: HateCheck: Functional Tests for Hate Speech Detection Models

Authors:
Paul Rottger , Bertram Vidgen , Dong Nguyen, Zeerak Waseem , Helen Margetts, and Janet B. Pierrehumbert

Year of Publications:
2021

ABSTRACT:
Detecting online hate is a difficult task that even state-of-the-art models struggle with. Typically, hate speech detection models are evaluated by measuring their performance on held-out test data using metrics such as accuracy and F1 score. However, this approach makes it difficult to identify specific model weak points. It also risks overestimating generalisable model performance due to increasingly well-evidenced systematic gaps and biases in hate speech datasets. To enable more targeted diagnostic insights, we introduce HATECHECK, a suite of functional tests for hate speech detection models. We specify 29 model functionalities motivated by a review of previous research and a series of interviews with civil society stakeholders. We craft test cases for each functionality and validate their quality through a structured annotation process. To illustrate HATECHECK's utility, we test near-state-of-the-art transformer models as well as two popular commercial models, revealing critical model weaknesses.

CONCLUSION:
In this article, we introduced HATECHECK, a suite of functional tests for hate speech detection models. We motivated the selection of functional tests through interviews with civil society stakeholders and a review of previous hate speech research, which grounds our approach in both practical and academic applications of hate speech detection models. We designed the functional tests to offer contrasts between hateful and non-hateful content that are challenging to detection models, which enables more accurate evaluation of their true functionalities. For each functional test, we crafted sets of targeted test cases with clear gold standard labels, which we validated through a structured annotation process. We demonstrated the utility of HATECHECK as a diagnostic tool by testing near-state-of-the-art transformer models as well as two commercial models for hate speech detection. HATECHECK showed critical weaknesses for all models.

Specifically, models appeared overly sensitive to particular keywords and phrases, as evidenced by poor performance on tests for reclaimed slurs, counter speech and negated hate. The transformer models also exhibited strong biases in target coverage. Online hate is a deeply harmful phenomenon, and detection models are integral to tackling it. Typically, models have been evaluated on held-out test data, which has made it difficult to assess their generalisability and identify specific weaknesses.

CHAPTER 3
WORKING OF PROJECT

3.1 IMPORTING LIBRARIES:

```
import pandas as panda
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem.porter import *
import string
import nltk
from textstat.textstat import *
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import confusion_matrix
import seaborn
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.svm import LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
import numpy as np
from nltk.sentiment.vader import SentimentIntensityAnalyzer as VS
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
%matplotlib inline
```

3.2 DATA COLLECTION

In this research study, we collected publicly available hate speech tweets dataset. This dataset is compiled and labeled by CrowdFlower. In this dataset, the tweets are labeled into three distinct classes, namely, hate speech, not offensive, and offensive but not hate speech.

Our project analyzed a dataset CSV file from Kaggle containing 10,670 tweets. The dataset was heavily skewed with 93% of tweets or 10,670 tweets containing non-hate labeled Twitter data and 7% or 476 tweets containing hate-labeled Twitter data.

	Unnamed: 0	count	hate_speech	offensive_language	neither	class	tweet
0	0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't...
1	1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...
2	2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...
3	3	3	0	2	1	1	!!!!!!!! RT @C_G_Anderson: @viva_based she lo...
4	4	6	0	6	0	1	!!!!!!!!!!! RT @ShenikaRoberts: The shit you...
...	...	...	...	...	...	...	...
10666	10946	3	0	3	0	1	I like good weed, I like bad bitches that can ...
10667	10947	3	0	3	0	1	I like herbal tea bitches no matter if anyone ...
10668	10948	3	0	3	0	1	I like how cays mf ass favorited my tweet but ...
10669	10949	3	2	1	0	0	I like how niggas try an come at me and im lik...
10670	10950	3	0	3	0	1	I like it when you call me McSully I like when ba

10671 rows × 7 columns

3.3 TEXT PREPROCESSING

Several research studies have explained that using text preprocessing makes better classification results. So, in our dataset, we applied different preprocessing-techniques to filter noisy and non-informative features from the tweets. In preprocessing, we changed the tweets into lower case. Also, we removed all the URLs, usernames, white spaces, hashtags, punctuations and stop-words using pattern matching techniques from the collected tweets. Besides this, we have also performed tokenization and stemming from preprocessed tweets. The tokenization, converts each single tweet into tokens or words, then the porter stemmer converts words to their root forms, such as offended to offend using porter stemmer.

CODE:

```
# Adding text-length as a field in the dataset
dataset['text length'] = dataset['tweet'].apply(len)
print(dataset.head())
```

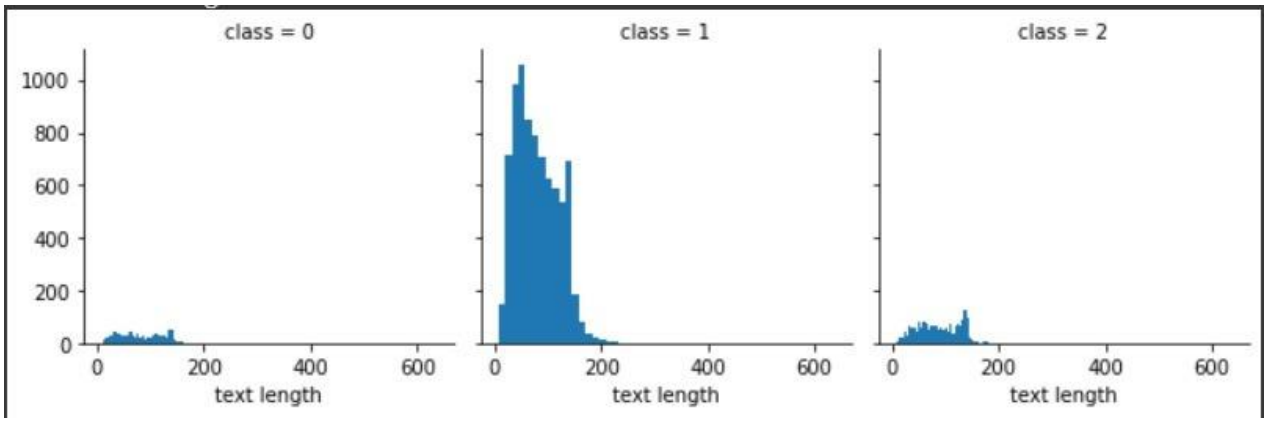

FIG-3 Text Length of a tweet

```
# collecting only the tweets from the csv file into a variable name tweet
tweet=dataset.tweet
```

Preprocesing of tweets
1. Removal of punctuation and capitlization
2. Tokenizing
3. Removal of stopwords
4. Stemming

```
stopwords = nltk.corpus.stopwords.words("english")

#extending the stopwords to include other words used in twitter such as retweet(rt) etc.
other_exclusions = ["#ff", "ff", "rt"]
stopwords.extend(other_exclusions)
stemmer = PorterStemmer()

def preprocess(tweet):

    # removal of extra spaces
    regex_pat = re.compile(r'\s+')
    tweet_space = tweet.str.replace(regex_pat, ' ')

    # removal of @name[mention]
    regex_pat = re.compile(r'@[\w\-]+')
    tweet_name = tweet_space.str.replace(regex_pat, '')

    # removal of links[https://abc.com]
    giant_url_regex =  re.compile('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|'
            '[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+')
    tweets = tweet_name.str.replace(giant_url_regex, '')

    # removal of punctuations and numbers
    punc_remove = tweets.str.replace("[^a-zA-Z]", " ")
    # remove whitespace with a single space
    newtweet=punc_remove.str.replace(r'\s+', ' ')
    # remove leading and trailing whitespace
    newtweet=newtweet.str.replace(r'^\s+|\s+?$','')
    # replace normal numbers with numbr
    newtweet=newtweet.str.replace(r'\d+(\.\d+)?','numbr')
    # removal of capitalization
    tweet_lower = newtweet.str.lower()

    # tokenizing
    tokenized_tweet = tweet_lower.apply(lambda x: x.split())
```

```
    # removal of stopwords
    tokenized_tweet=  tokenized_tweet.apply(lambda x: [item for item in x if item not in stopwords])
    # stemming of the tweets
    tokenized_tweet = tokenized_tweet.apply(lambda x: [stemmer.stem(i) for i in x])

    for i in range(len(tokenized_tweet)):
        tokenized_tweet[i] = ' '.join(tokenized_tweet[i])
        tweets_p= tokenized_tweet

    return tweets_p

processed_tweets = preprocess(tweet)

dataset['processed_tweets'] = processed_tweets
print(dataset[["tweet","processed_tweets"]].head(10))
```

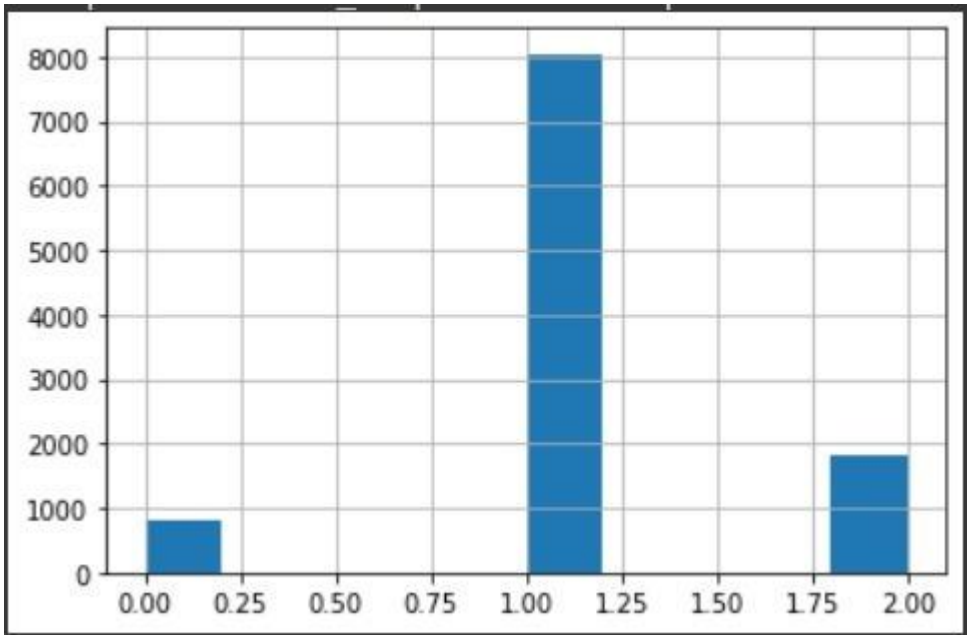

Fig-4 Histogram of Processed Tweet

3.4 FEATURE ENGINEERING

The ML algorithms cannot understand the classification rules from the raw text. These algorithms need numerical features to understand classification rules. Hence, in text classification one of the key steps is feature engineering. This step is used for extracting the key features from raw text and representing the extracted features in numerical form.

	tweet	processed_tweets
0	!!! RT @mayasolovely: As a woman you shouldn't...	woman complain clean hous amp man alway take t...
1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...	boy dat cold tyga dwn bad cuffin dat hoe st place
2	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...	dawg ever fuck bitch start cri confus shit
3	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...	look like tranni
4	!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...	shit hear might true might faker bitch told ya
5	!!!!!!!!!!!!!!!!!!"@T_Madison_x: The shit just...	shit blow claim faith somebodi still fuck hoe
6	!!!!!!"@__BrighterDays: I can not just sit up ...	sit hate anoth bitch got much shit go
7	!!!!“@selfiequeenbri: cause I'm tired of...	caus tire big bitch come us skinni girl
8	" & you might not get ya bitch back & ...	amp might get ya bitch back amp that
9	" @rhythmixx_ :hobbies include: fighting Maria...	hobbi includ fight mariam bitch

FIG-5 Visualization of most commonly used hate words

3.5 DATA SPLITTING

It is the class-wise distribution of the overall dataset as well as data set after splitting (i.e. Training set and Test set). We have used the 80-20 ratio to split the preprocessed data (i.e. 80% for Training Data and 20% for Test Data). The training data is used to train the classification model to learn classification rules. Moreover, the test data is further used to evaluate the classification model.

3.6 MACHINE LEARNING MODELS

According to “no free lunch theorem” , there is no any single classifier which best performs on all kinds of datasets. Therefore, it is recommended to apply several different classifiers on a master feature vector to observe which one reaches to the better results. Hence, we selected four different classifiers .

- **Naïve Bayes** : It's a probabilistic based classification algorithm, which uses the "Bayes theorem" to predict the class. It works on conditional independence among features.

```
X_train_tfidf, X_test_tfidf, y_train, y_test = train_test_split(X.toarray(), y, random_state=42, test_size=0.2)
nb=GaussianNB()
nb.fit(X_train_tfidf,y_train)
y_preds = nb.predict(X_test_tfidf)
acc2=accuracy_score(y_test,y_preds)
report = classification_report( y_test, y_preds )
print(report)
print("Naive Bayes, Accuracy Score:",acc2)
```

```
              precision    recall  f1-score   support

           0       0.14      0.53      0.23       157
           1       0.86      0.63      0.73      1599
           2       0.51      0.53      0.52       379

    accuracy                           0.60      2135
   macro avg       0.51      0.56      0.49      2135
weighted avg       0.75      0.60      0.65      2135

Naive Bayes, Accuracy Score: 0.6032786885245902
```

- **Logistic Regression** : It is a predictive analysis. It uses a sigmoid function to explain the relationship between one independent variable and one or more independent variables.

```
X = panda.DataFrame(modelling_features_two)
y = dataset['class'].astype(int)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, test_size=0.2)

model = LogisticRegression().fit(X_train,y_train)
y_preds = model.predict(X_test)
report = classification_report( y_test, y_preds )
print(report)
```

```
acc=accuracy_score(y_test,y_preds)
print("Logistic Regression, Accuracy Score:" , acc)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       157
           1       0.78      0.96      0.86      1599
           2       0.56      0.26      0.35       379

    accuracy                           0.76      2135
   macro avg       0.45      0.40      0.40      2135
weighted avg       0.68      0.76      0.71      2135

Logistic Regression, Accuracy Score: 0.7620608899297424
```

- **Support Vector Machine** : : It's a supervised classification algorithm which constructs an optimal hyperplane by learning from training data which separates the categories while classifying new data.

```
X = panda.DataFrame(modelling_features_three)
y = dataset['class'].astype(int)
X_train_features, X_test_features, y_train, y_test = train_test_split(X, y, random_state=0, test_size=0.2)
support =LinearSVC(random_state=20)
support.fit(X_train_features,y_train)
y_preds = support.predict(X_test_features)
acc3=accuracy_score(y_test,y_preds)
report = classification_report( y_test, y_preds )
print(report)
print("SVM, Accuracy Score:" ,acc3 )
```

```
              precision    recall  f1-score   support

           0       0.42      0.25      0.31       164
           1       0.89      0.93      0.91      1597
           2       0.82      0.82      0.82       374

    accuracy                           0.86      2135
   macro avg       0.71      0.67      0.68      2135
weighted avg       0.85      0.86      0.85      2135

SVM, Accuracy Score: 0.8608899297423888
```

- **Random Forest :** It's a type of ensemble classifier consisting of many decision trees. It classifies an instance based on voting decision of each decision trees class predictions.

```
X = panda.DataFrame(modelling_features_three)
y = dataset['class'].astype(int)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, test_size=0.2)
rf=RandomForestClassifier()
rf.fit(X_train,y_train)
y_preds = rf.predict(X_test)
acc1=accuracy_score(y_test,y_preds)
report = classification_report( y_test, y_preds )
print(report)
print("Random Forest, Accuracy Score:",acc1)
```

```
              precision    recall  f1-score   support

           0       0.41      0.08      0.14       157
           1       0.79      0.92      0.85      1599
           2       0.45      0.30      0.36       379

    accuracy                           0.75      2135
   macro avg       0.55      0.43      0.45      2135
weighted avg       0.70      0.75      0.71      2135

Random Forest, Accuracy Score: 0.746135831381733
```

3.7 Classifier Evaluation

In this step, the constructed classifier predicts the class of unlabeled text (i.e. "hate speech, offensive but not hate speech, neither hate speech nor offensive speech") using test set. The classifier performance is evaluated by calculating true negatives (TN), false positives (FP), false negatives (FN) and true positives (TP). Different performance metrics are used to assess the performance of the constructed classifier. Some common performance measures in text categorization are discussed briefly below-:

1. Precision: Precision is also known as the positive predicted value. It is the proportion of predictive positives which are actually positive. $Precision = TP/(TP+FP)$
2. Recall: It is the proportion of actual positives which are predicted positive.

$Recall = TP\ (TP+FN)$

3. F-Measure: It is the harmonic mean of precision and recall. The standard F-measure (F1) gives equal importance to precision and recall.

   $F - measure = 2 \times (precision \times recall\ )/\ (precision+recall\ )$

4. Accuracy: It is the number of correctly classified instances (true positives and true negatives).

   $Accuracy = (TP+TN)\ TP+FP+TN+F$N

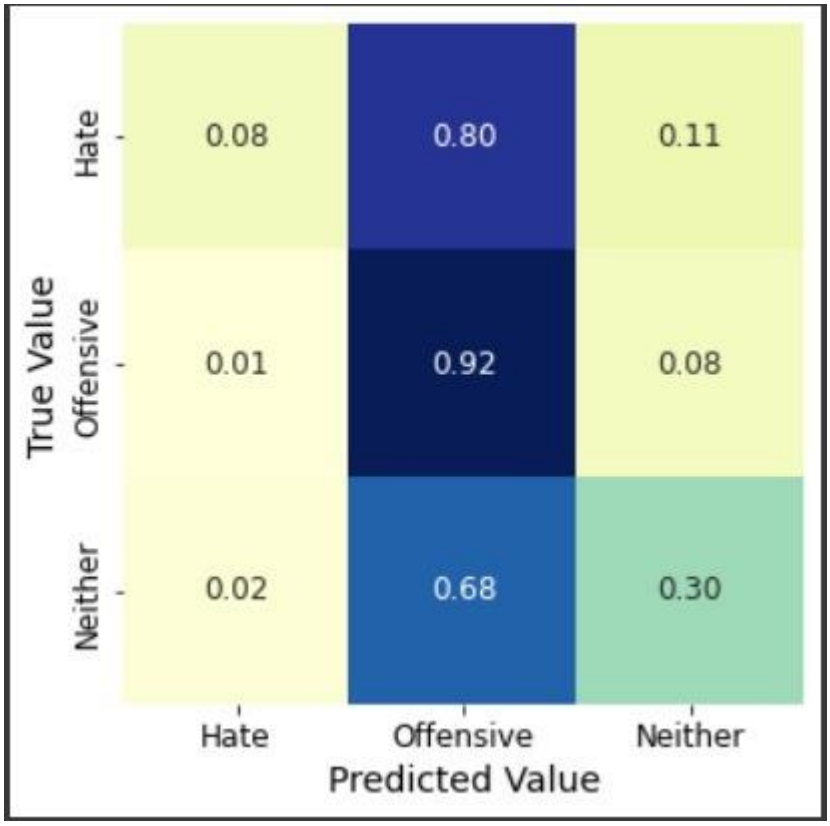

FIG 6 CONFUSION MATRIX

CHAPTER 4
RESULTS AND DISCUSSIONS

4.1 RESULTS:

The logistic regression algorithm works consistently well with all feature sets except for F7 as precision, recall and subsequently the f1-score for "hate" label results in zero . Random Forest classifier works pretty well when it comes to F1 and also shows a significant performance in all other feature sets but its performance is hugely impacted when tf-idf scores are not included in the feature. The overall performance of the Naïve Bayes classifier is found to be less significant for the purpose of classifying tweets into hate, offensive or neither labels but it performs significantly better with feature set of F7 compared to other feature sets . SVM classifier also seems to be consistent throughout all feature sets except for F4 and F7 as shown in . From the above graphs we analyse that the most important feature was found to be F1 i.e. the tf-idf scores which helps in better classification of hate speech. The sentiment scores also prove to be an important feature for the differing of hate speech and offensive language .Doc2vec columns are not found to be very significant in classification purpose as it makes very less difference when it's removed from the feature set. On comparing all the graphs above Random Forest is clearly the winner.

4.1.1 CALCULATED ACCURACIES :

CLASSIFIER	ACCURACY
SVM	86.08
NAÏVE BAYES	60.32
RANDOM FORREST	74.61
LOGISTIC REGRESSION	76.20

4.2 DISCUSSIONS:

In the experimental work, we have evaluated four classifiers over two different feature engineering techniques, giving different analyses over hate speech dataset containing three classes. Our experimental results showed that the SVM algorithm with the combination of bigram with TFIDF FE techniques showed the best results. The theoretical analysis is discussed in subsequent sections.

A. Feature Engineering :

The selection of feature engineering is important in text classification. In this study, we compared two distinct feature extraction techniques namely, Bigram with TFIDF, and doc2vec. The experimental results exhibited that from these three techniques, bigram with TFIDF outperformed. Conversely, the Doc2vec showed lower results. The possible reason for the outperformance of bigram and TFIDF is that bigram maintains the sequence of words compared to doc2vec. Moreover, several studies showed that the TFIDF representation technique is better than the binary and term frequency representation. In our experimental results, Doc2Vec also showed lower performance. This might be because it performs low in case of very short length documents and the tweets which we used in our dataset often having 280 character length.

B. **Machine Learning Classifier:**

Several studies proved that no single ML algorithm performed better on all kinds of data. Therefore, the comparison of various ML algorithms is required to discover which one is best performing on the given dataset. Hence, on our dataset, we used four different ML algorithms as discussed in ML Models. The experimental results proved that SVM achieved the best performance possibly because SVM uses threshold functions to separate the data, not the number of features based on margin. This shows that SVM is independent upon the presence of the number of features in the data. In addition, SVM has the capability to best perform on non-linear data apart from the linear data because of its kernel functions. The results obtained with RF and LR classifiers are a little lower than SVM results but are somewhat higher than the results of NB. The low performance of RF might be due to the unavailability of informative features which leads to incorrect predictions. It is possible that the performance of LR might be lower because its decision surface is linear in nature and cannot handle nonlinear data adequately. The lowest performance was obtained amongst the NB classifiers. The NB classifier works on conditional independence among features. Thus, the performance of the NB classifier is negatively affected as the conditional dependence becomes more complicated due to the increase in the number of features.

C. **Classwise Performance:**

A we have three classes name “hate speech”, “offensive but not hate speech” and “neither hate speech nor offensive speech”. The results show that all features and classifiers performed well for two classes (i.e. offensive but not hate speech, and neither hate speech

nor offensive speech). Our experimental results showed that the combinations performed lowest for class hate speech. The class “Hate Speech” has the lowest training instances as compared to other classes, but the major reason for misclassification of class “Hate Speech” might be overlapping of different bigram words with higher frequency in other classes than hate speech class. For example, bigrams like “lame nigga, white trash, bitch made” are more frequently appearing in class “Offensive but not Hate Speech” as compared to class “Hate Speech”. Hence, it might be possible that the classifier learned weak learning rules.

CHAPTER 5
CONCLUSIONS AND FUTURE SCOPE

5.1 CONCLUSION:

This study employed automated text classification techniques to detect hate speech messages. Moreover, this study compared three feature engineering techniques and eight ML algorithms to classify hate speech messages. The experimental results exhibited that the bigram features, when represented through TFIDF, showed better performance as compared to Doc2Vec features engineering techniques. Moreover, SVM and RF algorithms showed better results compared to LR, NB. The lowest performance was observed in KNN. The outcomes from this research study hold practical importance because this will be used as a baseline study to compare upcoming researches within different automatic text classification methods for automatic hate speech detection. Furthermore, this study also holds a scientific value because this study presents experimental results in form of more than one scientific measures used for automatic text classification. Our work has two important limitations. First, the proposed ML model is inefficient in terms of real-time predictions accuracy for the data. Finally, it only classifies the hate speech message in three different classes and is not capable enough to identify the severity of the message. Hence, in the future, the objective is to improve the proposed ML model which can be used to predict the severity of the hate speech message as well. Moreover, to improve the proposed model's classification performance two approaches will be used. First, the lexiconbased techniques will be explored and assessed by comparing with other current state-of-the-art results. Secondly, more data instances will be collected, to be used for learning the classification rules efficiently.

5.2 FUTURE SCOPE:

Future scope includes removal of hate speech after its detection . We will also analyze the gaps in the existing solutions and fulfill them to present the most feasible solutions.

Firstly, we will explore other branches of methods that aim at compensating the lack of training data in supervised learning tasks. Methods such as transfer learning could be potentially promising. Secondly, the presence of abstract concepts such as 'sexism', 'racism' or even 'hate' in general is very difficult to detect if solely based on linguistic content. Therefore, we see the need to go beyond pure text classification and explore possibilities to model and integrate features about users, social groups, mutual communication and even background knowledge (e.g., concepts expressed from tweets) encoded in existing semantic knowledge base.

REFERENCES

[1] Hern, A., Facebook, YouTube, Twitter, and Microsoft sign the EU hate speech code. The Guardian, 2016. 31.

[2] Rosa, J., and Y. Bonilla, Deprovincializing Trump, decolonizing diversity, and unsettling anthropology. American Ethnologist, 2017. 44(2): p. 201-208.

[3] Travis, A., Anti-Muslim hate crime surges after Manchester and London Bridge attacks. The Guardian, 2017.

[4] MacAvaney, S., et al., Hate speech detection: Challenges and solutions. PloS one, 2019. 14(8): p. e0221152.

[5] Fortuna, P. and S. Nunes, A survey on automatic detection of hate speech in text. ACM Computing Surveys (CSUR), 2018. 51(4): p. 85.

[6] Mujtaba, G., et al., Prediction of cause of death from forensic autopsy reports using text classification techniques: A comparative study. Journal of forensic and legal medicine, 2018. 57: p. 41-50.

[7] Cavnar, W.B. and J.M. Trenkle. N-gram-based text categorization. in Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval. 1994. Citeseer.

[8] Ramos, J. Using tf-idf to determine word relevance in document queries. in Proceedings of the first instructional conference on machine learning. 2003. Piscataway, NJ.

[9] Mikolov, T., et al. Distributed representations of words and phrases and their compositionality. in Advances in neural information processing systems. 2013.

[10] Le, Q. and T. Mikolov. Distributed representations of sentences and documents. in International conference on machine learning. 2014.

[11] Kotsiantis, S.B., I.D. Zaharakis, and P.E. Pintelas, Machine learning: a review of classification and combining techniques. Artificial Intelligence Review, 2006. 26(3): p. 159-190.

[12] Lewis, D.D. Naive (Bayes) at forty: The independence assumption in information retrieval. in European conference on machine learning. 1998. Springer.

[13] Xu, B., et al., An Improved Random Forest Classifier for Text Categorization. JCP, 2012. 7(12): p. 2913-2920.