A Project Report

on

Violent Speech Detection On Videos Using NLP

Submitted in partial fulfillment of the

requirement for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
Dr.Ganga Sharma
Associate Professor

Submitted By

18SCSE1140042 - Ayush Mehrotra
18SCSE1140059 - Siddhant Chaudhary

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
DECEMBER-2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the project, entitled **"Violent Speech Detection On Videos Using NLP"** in partial fulfillment of the requirements for the award of the Bachelor of Technology submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of September 2021 to December 2021, under the supervision of Dr.Ganga Sharma (Associate Professor), 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.

Ayush Mehrotra, 18SCSE1140042
Siddhant Chaudhary, 18SCSE1140059

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

Dr. Ganga Sharma

Associate Professor

CERTIFICATE

The Final Project Viva-Voce examination of Ayush Mehrotra(18SCSE1140042), Siddhant Chaudhary (18SCSE1140059) has been held on _________________ and his work is recommended for the award of B.Tech.

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

Signature of Project Coordinator **Signature of Dean**

Date: December, 2021

Place: Greater Noida

Abstract

The rapid growth of Internet users led to unwanted cyber issues, including cyberbullying, hate speech, and many more. This paper deals with the reviewing of different techniques used to detect hate speech by many scholars and researchers.Hate speech occurs when an individual or a group of individuals attack or use derogatory or discriminatory words towards a group of people based on characteristics such as origin, sexuality, ethnicity, religious background, socioeconomic status, race, gender, and other factors. When such action takes place on social networking sites, blogs, creative material, and other forms of online media, it is referred to as Online Hate Speech [1]. Hate speech appears to be an inflammatory kind of interaction process that uses misconceptions to express a hate ideology Hate speech focuses on various protected aspects, including gender, religion, race, and disability [2]. Owing to hate speech, sometimes unwanted crimes are going to happen as someone or a group of people get disheartened. Hence, it is essential to monitor users' posts and filter the hate speech related post before it is spread. However, Twitter receives more than six hundred tweets per second and about 500 million tweets per day. Manually filtering any information from such a huge incoming traffic is almost impossible. Concerning this aspect, many techniques have been published using different aspects of machine learning and deep learning. Several attempts have been made to classify hate speech using machine learning, this is targeted to the use of primitive NLP feature engineering techniques

Table of Contents

CHAPTER-1

Introduction

Social media giants such as Facebook, Twitter, Instagram, Youtube that are curbing hate speech are pushed to deal with the questions of infringing on rights to speak and post what they want. For an instance, the native English speakers are frequent user of words like b*tch and h*e in day to day language. It is noticeable nowadays that even the lyrics of songs consists of slurs such as f*g and n*gga which is frequently used in online communication.

Insufficient data is one of the major issues to automate hate speech for detection of different languages. However, there is sufficient dataset for the English language to train and test the detection. Irregular datasets and annotation, making it tougher for regulated hate speech detection.

Unavailability of in-depth annotation and classification, that is, in most cases the classes targeted are solely hate and non-hate although the crucial target is addressing different classes (e.g. racism, online bullying, body shaming, criticism). When it comes to hate speech detection, twitter is the most relied social media as it contains enormous linguistic diversity in the content. Most of the publicly available hate speech annotated data in English are from twitter. The dataset which is used in

this paper is a publicly available hate speech dataset on CrowdFlower which has

been used previously in Davidson and Warmsley (2018) that consists of 25K tweets

in English. This publicly available dataset consisting of 25k tweets are categorized

into three classes with class labels.9

While the process of detection and classification of hate speech using NLP in social media, it is Important have to have noise free and clean data in order to get accurate results using machine learning techniques/algorithms. During processing of twitter dataset, the tweets are mostly attached with useless or unknown strings, used with informality.

Specifically, tweets also have a different formatting starting with authors usernames, URLs, hashtags, which need to be removed/no-use or parsed.

Tweets are cleaned by removing extra spaces, tags, links, punctuations,numbers, time date, locations. Then cleaned tweets are lowercase and tokenized. Next, the stop words are removed. Lastly, the stemming is performed using the stemmer.

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. For instance, The American Bar Association asserts that in the United States, hate speech is legal and protected by the First Amendment, although not if it directly calls for violence As such, many online forums such as Facebook, YouTube, and Twitter consider hate speech harmful, and have policies to remove hate speech content Due to societal concerns and how widespread hate speech is becoming on the Internet , there is strong motivation to study automatic detection of hate speech. By automating its detection, the spread of hateful content can be reduced. 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. We are by no means, nor can we be, comprehensive as new definitions appear regularly. Our aim is simply to illustrate variances highlighting difficulties that arise from such.

Competing definitions provide challenges for evaluation of hate speech detection systems; existing datasets differ in their definition of hate speech, leading to datasets that are not only from different sources, but also capture different information. This can make it difficult to directly access which aspects of hate speech to identify. We discuss the various datasets available to train and measure the performance of hate speech detection systems in the next section. Nuance and subtleties in language provide further challenges in automatic hate speech identification, again depending on the definition.

CHAPTER-2 Literature Review

Any statement that disparages a person or a group based on a trait such as race, colour, ethnicity, gender, sexual orientation, nationality, religion, or another attribute is characterised as violent Speech. As a result of the massive increase of user-generated web content, particularly on social media networks, the number of violent Speech is continually expanding. Interest in identifying online violent Speech, particularly the automation of this process, has rapidly increased in recent years, as has the societal impact of the phenomenon. Natural language processing, focusing specifically on this topic, is essential since simple word filters are insufficient: What exactly is it?The examples in this paper are offered to show how serious the problem of violent Speech is. They are based on real-world data and do not reflect the authors' own views.

Aspects such as an utterance's domain, discourse context, and context, which includes co-occurring media e.g.mobile gallery media, digital media, songs downloaded, might all influence the content of a violent Speech message.

Joscha et. al, in their paper conceived and thought about different methods like Bag of words models, n-grams for utilizing semantic data to work on the exhibition of opinion investigation. The prior approaches didn't think about the semantic relationship between sentences or archives parts. Research by A. Hogenboom et al. neither looked at the systemic variations nor gave a technique to combine exposure units in the greatest way. They intended to further develop the opinion examination by utilizing Rhetoric Structure Theory (RST) as it gives a progressive portrayal at the report level. They proposed a mix of the matrix search and weighting to discover the normal scores of opinion from the Rhetorical Structure Theory (RST) tree. They encoded the twofold information into the arbitrary timberland by utilizing highlight designing as it enormously decreased the intricacy of the unique RST tree. They presumed that AI raised decent precision and gave a high F1 score of 71.9%.

Amir Hossein Yazdavar et al. in this paper gave a novel comprehension of the feeling examination issue containing numerated information in drug audits. They broke down sentences which contained quantitative terms to arrange them into stubborn or non-obstinate and furthermore to recognize the extremity communicated by utilizing the fluffy set hypothesis. The improvement of the fluffy information base was finished by talking to a few specialists from different clinical focuses. Although the quantity of investigations has been done in this field these don't consider the mathematical (quantitative) information contained in the audits while perceiving the feeling extremity. Likewise, the preparation information utilized has a high area reliance and thus can't be utilized in various spaces. They inferred that their proposed technique of information designing dependent on fluffy sets was a lot less difficult, productive and has high precision of more than 72% F1 esteem.

Ahmad Kamal in his paper planned an assessment mining system that works with objectivity or subjectivity examination, including extraction and audit synopsis and so forth. He utilized a regulated AI approach for subjectivity and objectivity order of audits. The different procedures utilized by him were Naive Bayes, Decision Tree, Multilayer Perceptron and Bagging. He likewise further developed mining execution by forestalling unimportant extraction and commotion as in Kamal's paper.

Humera Shaziya et al. in this paper characterized film audits for feeling examination utilizing WEKA Tool. They upgraded the prior work done in feeling order which dissects assessments which express either good or negative opinion. In this paper, they likewise thought to be the way that audits that have suppositions from more than one individual and a solitary survey might communicate both the positive and negative feeling. They directed their test on WEKA and presumed that Naive Bayes performs obviously superior to SVM for film surveys just as text. Gullible Bayes has a precision of 85.1%.

Akshay Amolik et. al. in his paper made the dataset utilizing twitter posts of film audits and related tweets about those motion pictures. Sentence level opinion investigation is performed on these tweets. It is done in three stages. Initially, preprocessing is finished. Then, at that point, the Feature vector is made utilizing significant highlights. At long last, by utilizing various classifiers like Naive Bayes, Support vector machine, Ensemble classifier, k-implies and Artificial Neural Networks, tweets were arranged into positive, negative and unbiased classes. The outcomes show that we get 75 % precision structure SVM.

The above-mentioned issue has attracted researchers over the past few years and has therefore proposed models using machine learning and deep learning techniques [8]–[14].

However, existing models do not meet the required needs, as many HS related tweets are still available on Twitter and floating across the network. This prompted us to develop a model that would capture the maximum number of HS related posts. The CNN model has been successfully used by current researchers to address various issues related to the text domain, including sentiment analysis, question answering, document classification, sentence clas-sification [15]–[17], spam filtering and others . By following them, this research also uses a deep convolutional neural network (DCNN) to address the hate speech detection issue. DCNN is capable of capturing the semantics of the sentence by performing the convolution operations over the tweets. We also tested other deep neural network-based models such as Long Short-Term Memory (LSTM), and Convolutional-LSTM (C-LSTM) network for the same and found the DCNN model is a better choice for this research.

Dataset Used

The dataset used in our experiments is a combination of the following two datasets:

a) 25k Twitter dataset[14]: This dataset comprises 24,802 labelled tweets which were randomly sampled out of 85.4 million tweets and were labelled into hate, no hate and neither. We make this dataset binary by considering offensive and neither as non-hate.

b) Hate speech and personal attack dataset in English social media This a dataset available on zenodo.org. It is a binary dataset collected for the European project for countering hate speech.

Feature Engineering

Feature engineering is the most important part of Machine learning to make any dataset usable for training any machine learning or deep learning model. There exist various types of features like semantic, lexicographic, sentiment-based and word embeddings. It is very important to decide what features we use to optimize the accuracy of our models. The following are the feature engineering techniques used :-

a) TF-IDF:

Term Frequency-Inverse Document Frequency (TF-IDF) is a technique used to give weighted importance of a word or a phrase in a document or a corpus. We used it to create feature vectors using the most common n-grams in our dataset.[17]

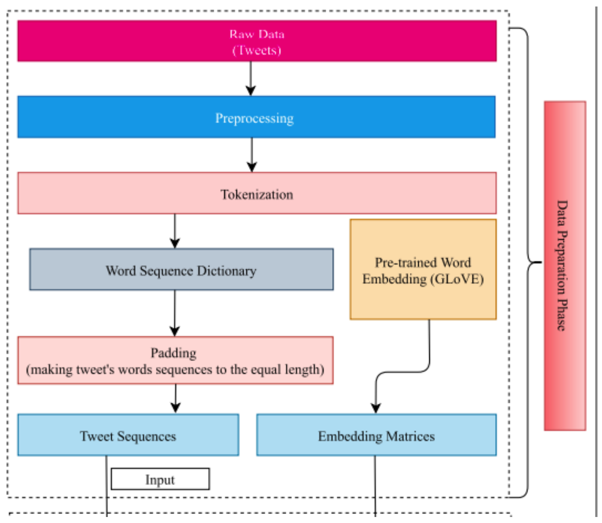
b) Bag of Words (BoW):

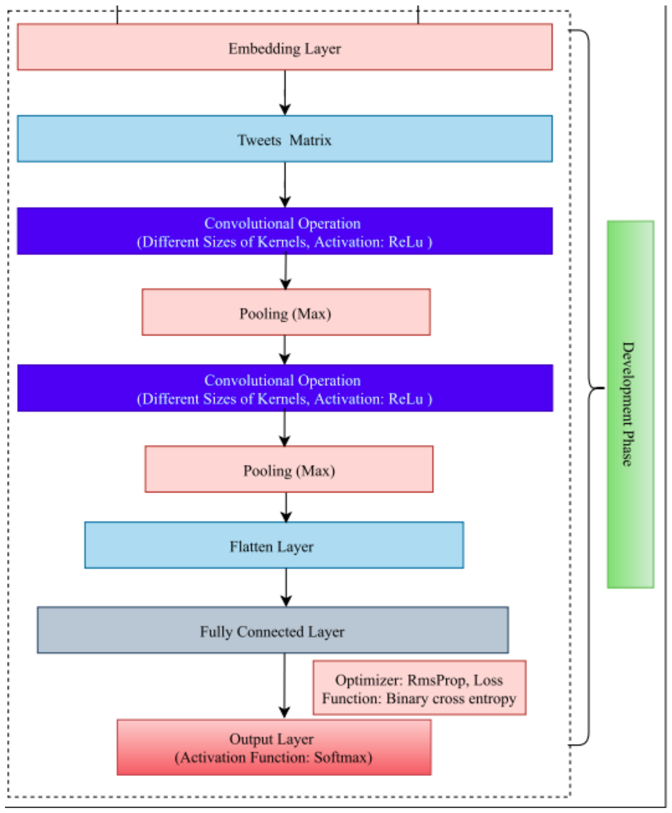
BoW is a technique used to numerically express a document in our corpus using the number of times a specific word or n-gram occurs.[16]

c) Word2vec Embeddings:

Word2vec [8] is an algorithm that uses a neural network to learn word embeddings. Its goal is to estimate each word's position in a multi dimension vector space based on the similarity of different words.

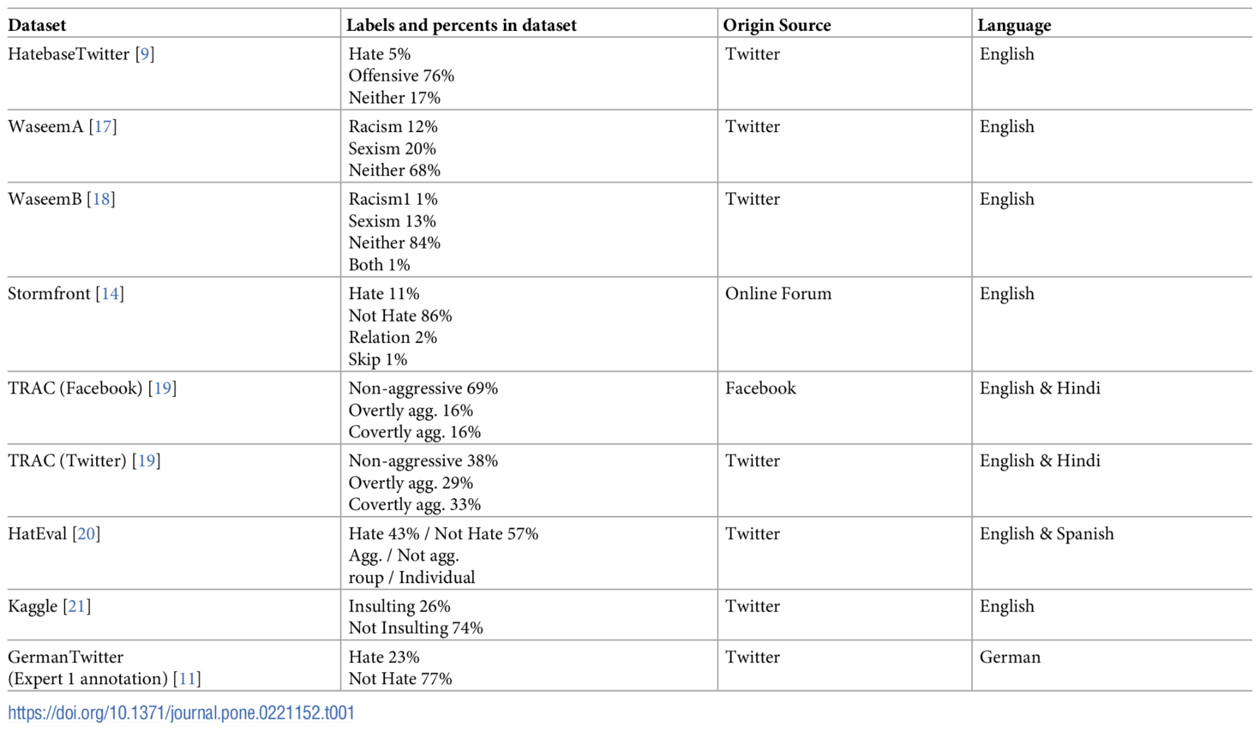
Functionality/Working and Code




Embedding Layer
Tweets Matrix
Convolutional Operation
(Different Sizes of Kernels, Activation: ReLu)
Pooling (Max)
Convolutional Operation
(Different Sizes of Kernels, Activation: ReLu)
Pooling (Max)
Flatten Layer
Fully Connected Layer
Optimizer: RmsProp, Loss Function: Binary cross entropy
Output Layer
(Activation Function: Softmax)
Development Phase

Libraries Required

```
from pydub import AudioSegment
import glob
import moviepy.editor
import os
import math
import speech_recognition as sr
from pydub import AudioSegment
from pydub.silence import split_on_silence
import numpy as np
import pandas as pd
import nltk
from nltk.stem import WordNetLemmatizer
import string
import gensim
from gensim.models import Word2Vec
import joblib
from sklearn.feature_extraction.text import TfidfVectorizer
```



Dataset	Labels and percents in dataset	Origin Source	Language
HatebaseTwitter [9]	Hate 5% Offensive 76% Neither 17%	Twitter	English
WaseemA [17]	Racism 12% Sexism 20% Neither 68%	Twitter	English
WaseemB [18]	Racism1 1% Sexism 13% Neither 84% Both 1%	Twitter	English
Stormfront [14]	Hate 11% Not Hate 86% Relation 2% Skip 1%	Online Forum	English
TRAC (Facebook) [19]	Non-aggressive 69% Overtly agg. 16% Covertly agg. 16%	Facebook	English & Hindi
TRAC (Twitter) [19]	Non-aggressive 38% Overtly agg. 29% Covertly agg. 33%	Twitter	English & Hindi
HatEval [20]	Hate 43% / Not Hate 57% Agg. / Not agg. roup / Individual	Twitter	English & Spanish
Kaggle [21]	Insulting 26% Not Insulting 74%	Twitter	English
GermanTwitter (Expert 1 annotation) [11]	Hate 23% Not Hate 77%	Twitter	German

https://doi.org/10.1371/journal.pone.0221152.t001

Table 1: Data Distribution of the Hate words

```
class SplitWavAudioMubin():
   def __init__(self, folder, filename):
       self.folder = folder
       self.filename = filename
       self.file path = folder + '\\' + filename

       self.audio = AudioSegment.from_wav(self.filepath)

   def get_duration(self):
       return self.audio.duration_seconds

   def single_split(self, from_min, to_min, split_filename):
       t1 = from_min * 60 * 1000
       t2 = to_min * 60 * 1000
       split_audio = self.audio[t1:t2]
       split_audio.export(self.folder + '\\' + split_filename,
format="wav")
```

```
def multiple_split(self, min_per_split):
    total_mins = math.ceil(self.get_duration() / 60)
    for i in range(0, total_mins, min_per_split):
        split_fn = str(i) + '_' + self.filename
        self.single_split(i, i+min_per_split, split_fn)
        print(str(i) + ' Done')
        if i == total_mins - min_per_split:
            print('All splitted successfully')
```

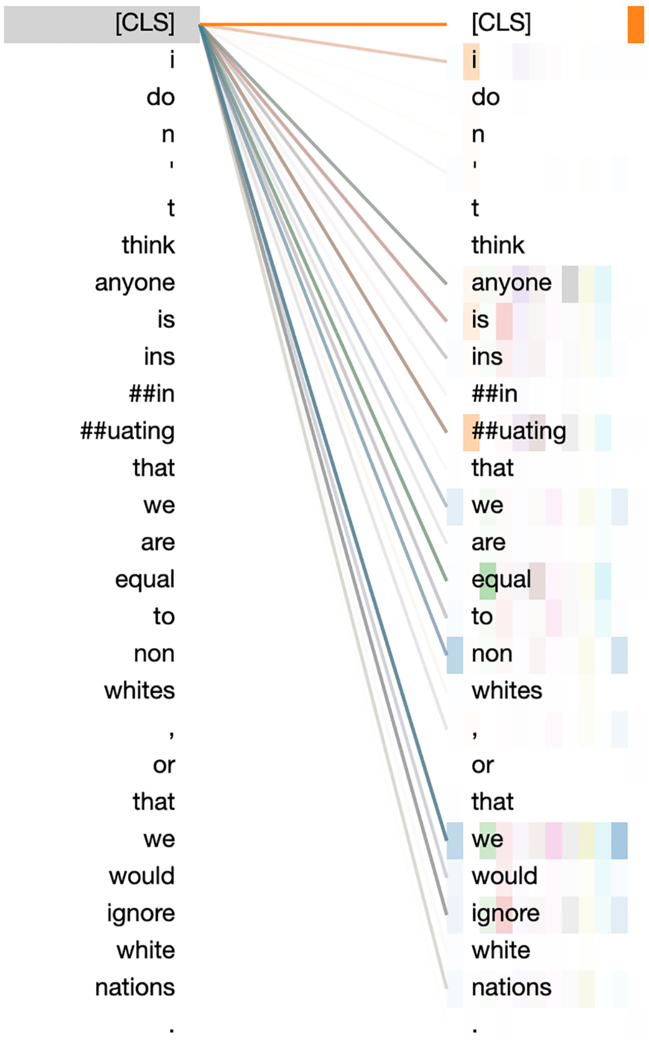


Fig 2: Marking of words

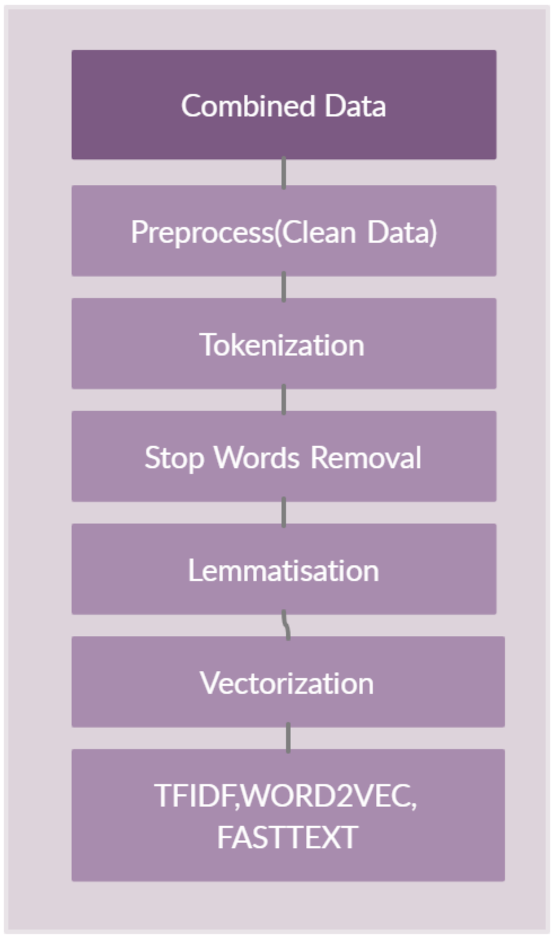
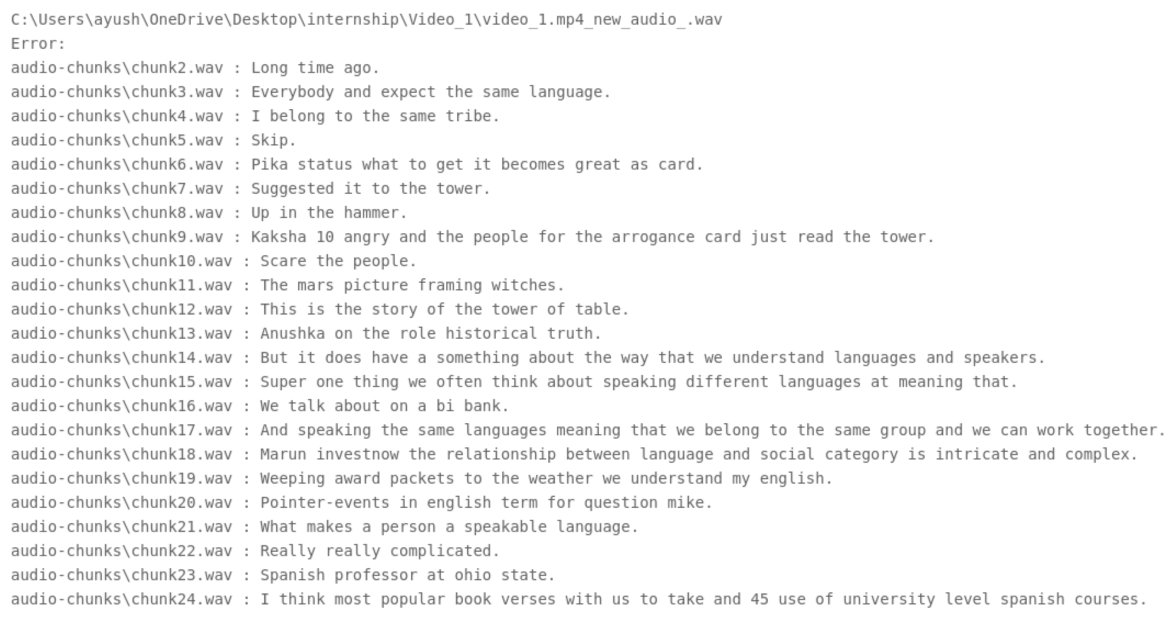


Fig3: Pipeline For Project

Checking of file extracted

```
for i in files1:
   print(i)
   get_large_audio_transcription(i)
```



```
C:\Users\ayush\OneDrive\Desktop\internship\Video_1\video_1.mp4_new_audio_.wav
Error:
audio-chunks\chunk2.wav : Long time ago.
audio-chunks\chunk3.wav : Everybody and expect the same language.
audio-chunks\chunk4.wav : I belong to the same tribe.
audio-chunks\chunk5.wav : Skip.
audio-chunks\chunk6.wav : Pika status what to get it becomes great as card.
audio-chunks\chunk7.wav : Suggested it to the tower.
audio-chunks\chunk8.wav : Up in the hammer.
audio-chunks\chunk9.wav : Kaksha 10 angry and the people for the arrogance card just read the tower.
audio-chunks\chunk10.wav : Scare the people.
audio-chunks\chunk11.wav : The mars picture framing witches.
audio-chunks\chunk12.wav : This is the story of the tower of table.
audio-chunks\chunk13.wav : Anushka on the role historical truth.
audio-chunks\chunk14.wav : But it does have a something about the way that we understand languages and speakers.
audio-chunks\chunk15.wav : Super one thing we often think about speaking different languages at meaning that.
audio-chunks\chunk16.wav : We talk about on a bi bank.
audio-chunks\chunk17.wav : And speaking the same languages meaning that we belong to the same group and we can work together.
audio-chunks\chunk18.wav : Marun investnow the relationship between language and social category is intricate and complex.
audio-chunks\chunk19.wav : Weeping award packets to the weather we understand my english.
audio-chunks\chunk20.wav : Pointer-events in english term for question mike.
audio-chunks\chunk21.wav : What makes a person a speakable language.
audio-chunks\chunk22.wav : Really really complicated.
audio-chunks\chunk23.wav : Spanish professor at ohio state.
audio-chunks\chunk24.wav : I think most popular book verses with us to take and 45 use of university level spanish courses.
```

Fig 4: Data Set After extracting audio from video

Splitting the large audio file into chunks

```
def get_large_audio_transcription(path):
    """
    Splitting the large audio file into chunks
    and apply speech recognition on each of these chunks
    """
    # open the audio file using pydub
    sound = AudioSegment.from_wav(path)
    # split audio sound where silence is 700 milliseconds or more and get
chunks
    chunks = split_on_silence(sound,
        # experiment with this value for your target audio file
        min_silence_len = 500,
        # adjust this per requirement
        silence_thresh = sound.dBFS-14,
        # keep the silence for 1 second, adjustable as well
        keep_silence=500,
    )
```

```python
folder_name = "audio-chunks"
# create a directory to store the audio chunks
if not os.path.isdir(folder_name):
    os.mkdir(folder_name)
whole_text = ""
# process each chunk
for i, audio_chunk in enumerate(chunks, start=1):
    # export audio chunk and save it in
    # the `folder_name` directory.
    chunk_filename = os.path.join(folder_name, f"chunk{i}.wav")
    audio_chunk.export(chunk_filename, format="wav")
    # recognize the chunk
    with sr.AudioFile(chunk_filename) as source:
        audio_listened = r.record(source)
        # try converting it to text
        try:
            text = r.recognize_google(audio_listened)
        except sr.UnknownValueError as e:
            print("Error:", str(e))
        else:
            text = f"{text.capitalize()}. "
            print(chunk_filename, ":", text)
            whole_text += text
            #aud1.append(whole_text)
aud1.append(whole_text)
# return the text for all chunks detected
return whole_text
```

Preprocessing Of DataSet

```
def preprocess(tweet):
  # Removing handles and hashtags
  tweet = re.sub('@[^\s]+','',str(tweet))
  tweet = re.sub(r'#', '', str(tweet))
  # Removing URLS
  tweet= re.sub('((www\.[^\s]+)|(https?://[^\s]+))','',str(tweet))
  #removing all punctuation and special character
  tweet = re.sub('[^a-zA-Z]',' ',str(tweet))
  #removing extra white space
  tweet = re.sub('[\s]+', ' ', str(tweet))
  tweet = re.sub('[\n]+', ' ', str(tweet))
  tweet=tweet.lower()
  return tweet
train_data['tweet']= train_data['tweet'].apply(lambda x:preprocess(x))
train_data.head()
```

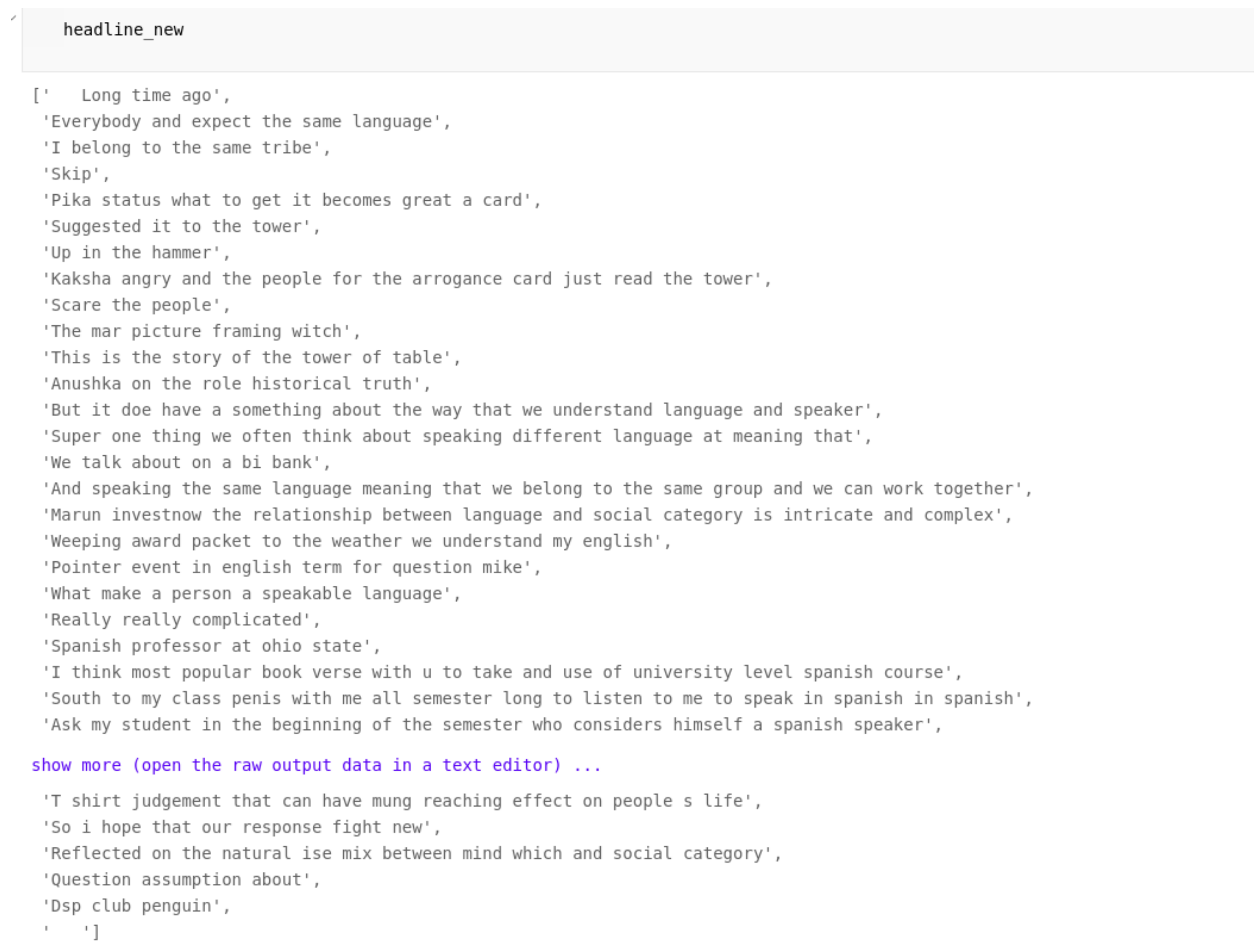
Removing of Null Values from the data

```
aud1=pd.DataFrame(aud1)
aud1.replace("[^a-zA-Z.]"," ",regex=True,inplace=True)
nan_value = float("NaN")
aud1.replace("", nan_value, inplace=True)
aud1.dropna(axis=0,how='any',thresh=None,subset=None,inplace=True)
data=aud1
#len(data)
headline=[]
for row in range(0,len(data.index)):
   headline.append(''.join(str(x) for x in data.iloc[row]))
headline=str(headline)
```

Removal of Stop Words from every Sentence

```
headline.lower()
sentence=headline.split(".")

tokenText = []
for sent in sentence:
   tok = nltk.word_tokenize(sent)
   if len(tok) > 0:
       tokenText.append(tok)
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
filtered_sentence_new = []
filtered_sentence_new = [word for word in tokenText if not word in
stopwords.words()]
```

DataSet After Extracting Audio



```
headline_new
```

```
['   Long time ago',
 'Everybody and expect the same language',
 'I belong to the same tribe',
 'Skip',
 'Pika status what to get it becomes great a card',
 'Suggested it to the tower',
 'Up in the hammer',
 'Kaksha angry and the people for the arrogance card just read the tower',
 'Scare the people',
 'The mar picture framing witch',
 'This is the story of the tower of table',
 'Anushka on the role historical truth',
 'But it doe have a something about the way that we understand language and speaker',
 'Super one thing we often think about speaking different language at meaning that',
 'We talk about on a bi bank',
 'And speaking the same language meaning that we belong to the same group and we can work together',
 'Marun investnow the relationship between language and social category is intricate and complex',
 'Weeping award packet to the weather we understand my english',
 'Pointer event in english term for question mike',
 'What make a person a speakable language',
 'Really really complicated',
 'Spanish professor at ohio state',
 'I think most popular book verse with u to take and use of university level spanish course',
 'South to my class penis with me all semester long to listen to me to speak in spanish in spanish',
 'Ask my student in the beginning of the semester who considers himself a spanish speaker',

show more (open the raw output data in a text editor) ...

 'T shirt judgement that can have mung reaching effect on people s life',
 'So i hope that our response fight new',
 'Reflected on the natural ise mix between mind which and social category',
 'Question assumption about',
 'Dsp club penguin',
 '   ']
```

Lemmatization of Every Word

```
lemmatizer = WordNetLemmatizer()
def wordlema(text):
   lem_text=" ".join([lemmatizer.lemmatize(i) for i in  text])
   return lem_text
lemtxt1 = []
for tok in filtered_sentence_new:
   lemSent = wordlema(tok)
   lemtxt1.append(lemSent)
```

Combining of Words into sentence after lemmatization

```
aud2=pd.DataFrame(lemtxt1)
aud2.replace("[^a-zA-Z.]"," ",regex=True,inplace=True)
nan_value = float("NaN")
aud2.replace("", nan_value, inplace=True)
aud2.dropna(axis=0,how='any',thresh=None,subset=None,inplace=True)
data2=aud2
#len(data)
headline_new=[]
for row in range(0,len(data2.index)):
   headline_new.append(''.join(str(x) for x in data2.iloc[row]))
```

Loading of First embedding technique "Word2Vec"

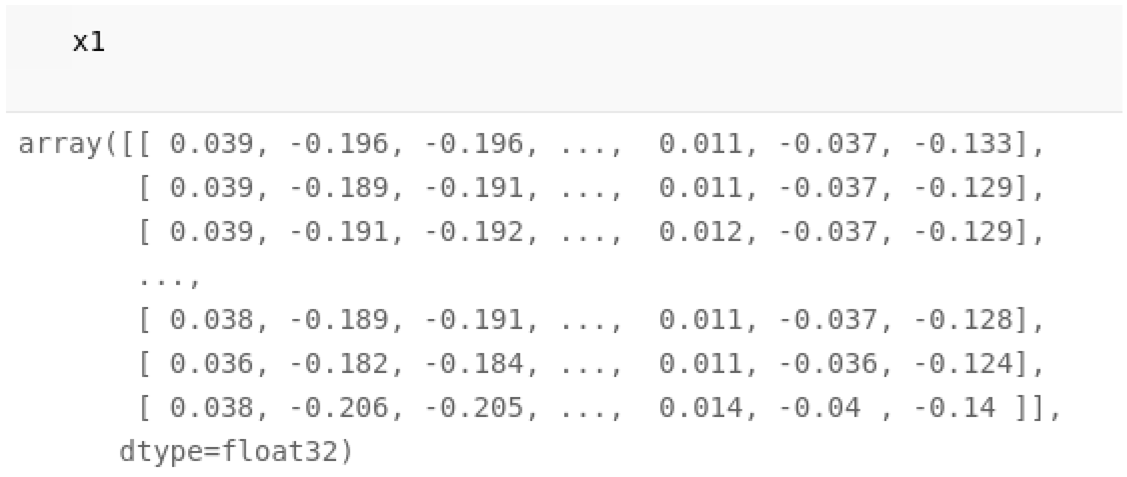
```
w2v_model=gensim.models.Word2Vec(headline_new,workers=3,size=100,min_count
=40,window=10,sample=1e-3)
```

Implementing Word2Vec

```
def word2vectranform(Z):
   zero=[]
   for i in range(0,300):
       zero.append(0)

   length=[]
   for i in range(0,len(Z)):
       #print(i)
       t=Z[i]
       k1=[]
       for j in range(0,len(t)):
           k=t[j]
```

```
            try:
                ee=model[k]
                k1.append(ee)
            except:
                ee=zero
                k1.append(ee)

        default=[]
        for l in range(0,300):
            default.append(0)

        for u in range(0, len(k1)):
            default=np.add(default,k1[u])
        try:
            default1=default/len(k1)
        except:
            default1=default

        length.append(default1)

    #print(len(length))
    return length



def make_feature_vec(words):
 #for sent in list_of_sent:
 feature_vec=np.zeros((100,),dtype='float32')
 nwords = 0
 index2word_set= set(w2v_model.wv.index2word)
 for word in words:
   if word in index2word_set:
       nwords +=1
       feature_vec=np.add(feature_vec,w2v_model[word])
 feature_vec=np.divide(feature_vec,nwords)
 feature_vec = np.around(feature_vec,3)
 return feature_vec
```

```
def get_avg_feature_vec(tweets):
   c=0
   tweet_feature_vec = np.zeros((len(tweets),100),dtype='float32')
   for tweet in tweets:
     tweet_feature_vec[c]=make_feature_vec(tweet)
     c=c+1
   return tweet_feature_vec
#tweets=[]
#for tweet in df['tweet']:
 #tweets.append(tweet)
x1= get_avg_feature_vec(headline_new)
```



```
x1
```

```
array([[ 0.039, -0.196, -0.196, ...,  0.011, -0.037, -0.133],
       [ 0.039, -0.189, -0.191, ...,  0.011, -0.037, -0.129],
       [ 0.039, -0.191, -0.192, ...,  0.012, -0.037, -0.129],
       ...,
       [ 0.038, -0.189, -0.191, ...,  0.011, -0.037, -0.128],
       [ 0.036, -0.182, -0.184, ...,  0.011, -0.036, -0.124],
       [ 0.038, -0.206, -0.205, ...,  0.014, -0.04 , -0.14 ]],
      dtype=float32)
```

Fig 5: Data After Vectorization

Training Machine on GbBoost

```
from sklearn.ensemble import GradientBoostingClassifier
clf1_gb2 =
GradientBoostingClassifier(random_state=4,n_estimators=200,learning_rate=0
.1,max_depth=5)
clf1_gb2.fit(train_x,train_y)

print(clf1_gb2.score(train_x,train_y))
print(clf1_gb2.score(test_x,test_y))


pred_gb2 = clf1_gb2.predict(test_x)
pd.DataFrame(confusion_matrix(pred_gb2,test_y))

gboost_from_joblib =
joblib.load(r'C:\Users\ayush\OneDrive\Desktop\internship\PIckel
Files\word2vecsamp.pkl')
 # Use the loaded model to make predictions
gboost_from_joblib.predict(x1)
```

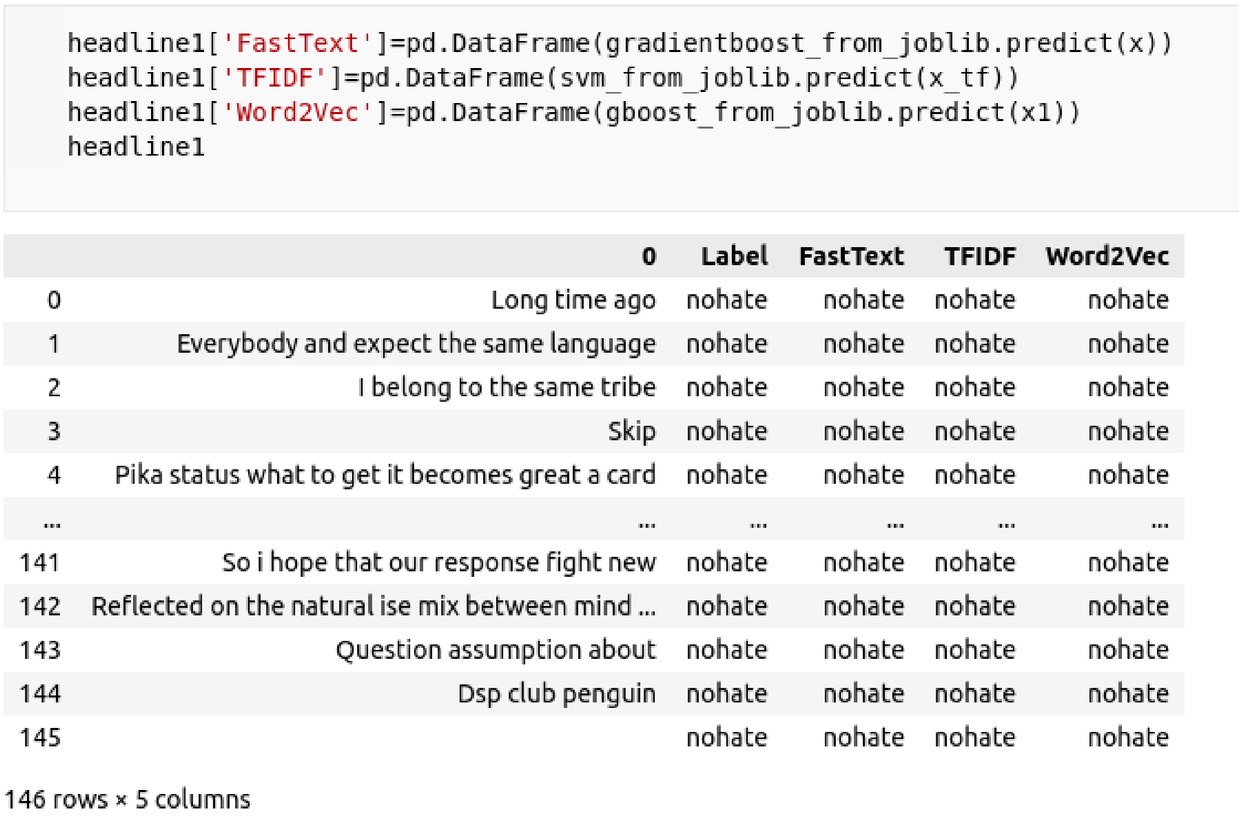
Loading Tf-Idf Vectorizer

```
tfidf = TfidfVectorizer(ngram_range=(1,3),max_features=10000)
tfidf.fit(x_tf)
x_tf=tfidf.transform(x_tf)
svm_tf1=SVC(C=100,gamma=0.01,kernel='rbf')
svm_tf1.fit(x_train,y_train)
svm_pred=svm_tf1.predict(x_test)
pd.DataFrame(confusion_matrix(svm_pred,y_test))

svm_from_joblib =
joblib.load(r'C:\Users\ayush\OneDrive\Desktop\internship\PIckel
Files\tfidfsamp.pkl')
 # Use the loaded model to make predictions
svm_from_joblib.predict(x_tf)
```

Loading of Fasttext Vectorizer

```
from gensim.models.fasttext import FastText
embedding_size = 100
window_size = 40
min_word = 5
down_sampling = 1e-2
```

```
ft_model = FastText(headline_new,
                     size=embedding_size,
                     window=window_size,
                     min_count=min_word,
                     sample=down_sampling,
                     sg=1,
                     iter=100)
def make_feature_vec(words):
 #for sent in list_of_sent:
 feature_vec=np.zeros((100,),dtype='float32')
 nwords = 0
 index2word_set= set(ft_model.wv.index2word)
 for word in words:
   if word in index2word_set:
       nwords +=1
       feature_vec=np.add(feature_vec,ft_model[word])
 feature_vec=np.divide(feature_vec,nwords)
 feature_vec = np.around(feature_vec,3)
 return feature_vec
```



```
headline1['FastText']=pd.DataFrame(gradientboost_from_joblib.predict(x))
headline1['TFIDF']=pd.DataFrame(svm_from_joblib.predict(x_tf))
headline1['Word2Vec']=pd.DataFrame(gboost_from_joblib.predict(x1))
headline1
```

	0	Label	FastText	TFIDF	Word2Vec
0	Long time ago	nohate	nohate	nohate	nohate
1	Everybody and expect the same language	nohate	nohate	nohate	nohate
2	I belong to the same tribe	nohate	nohate	nohate	nohate
3	Skip	nohate	nohate	nohate	nohate
4	Pika status what to get it becomes great a card	nohate	nohate	nohate	nohate
...	...	...	...	...	...
141	So i hope that our response fight new	nohate	nohate	nohate	nohate
142	Reflected on the natural ise mix between mind ...	nohate	nohate	nohate	nohate
143	Question assumption about	nohate	nohate	nohate	nohate
144	Dsp club penguin	nohate	nohate	nohate	nohate
145		nohate	nohate	nohate	nohate

146 rows × 5 columns

Fig 6: Predicted Data Set

Conclusion / Future Scope

This research addresses the issue of hate speech detection on Twitter using a deep convolutional neural network. Initially, the machine learning based classifiers such as LR, RF, NB, SVM, DT, GB, and KNN were used to identify the HS related tweets on Twitter with the features extracted using tf-idf technique. However, the best ML model, i.e SVM, is able to predict only 53% of HS tweets correctly on a 3:1 train-test dataset. The reason behind the low prediction of HS tweets may include the imbalanced dataset, hence the model biased towards the NHS tweets prediction as it is having the majority of instances. Deep learning based CNN, LSTM, and their combinations C-LSTM models also have similar results with the fixed partitioned dataset. The experimental outcome on both the traditional machine learning based models and deep learning based models confirmed that none of the models predicted the HS tweets with satisfactory accuracy on a fixed partitioned train-test. Finally, 10-fold cross-validation was used with the proposed CNN model and achieved the best prediction recall value of 0.88 for HS and 0.99 for NHS. The experimental results confirmed the k-fold cross-validation technique is a better choice with the imbalanced dataset. The current research addressed the HS issues with the textual data only; however, images are also widely used for the same. Hence, in the future, the researcher may include images with text or can analyse the video dataset to capture more HS related posts from Twitter. This research only used the tweets written in the English language, which can be further extended by mixing other languages such as Japanese, Hindi, Tamil, etc. The developed model achieved the recall value of 0.88, which indicates few tweets are not detected properly. In the future, a model may develop, which captures all hate speech content from the OSN. To build a general framework using deep learning models, the training dataset must have sufficient samples, in the future, the current dataset may be extended to achieve better accuracy.

REFERENCES

[1] Kia Dashtipour Scotland, United Kingdom “Multilingual Sentiment Analysis: State of the Art and Independent Comparison of Techniques”, Springer, 2016.

[2] M. Annett, G. Kondrak, “A comparison of sentiment analysis techniques: Polarizing movie Blogs”, In Canadian Conference on AI, pp. 25–35,2008.

[3] A. Mudinas, D. Zhang, M. Levene, “Combining lexicon and learning based approaches for concept-level sentiment analysis”, Proceedings of the First International Workshop on Issues of Sentiment Discovery and Opinion Mining, ACM, New York, NY, USA, Article 5, pp. 1-8, 2012.

[4] H. Wang, Yue Lu, and C. Zhai. Latent aspect rating analysis on review text data: a rating regression approach. In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 783-792. ACM, 2010.

[5] Chirag Sangani Stanford University, USA “Sentiment Analysis of App Store tweet”, 2013

[6] Giuseppe Di Fabbrizio A&T Research Labs, USA “Summarizing Online tweet Using Aspect Rating Distributions and Language Modeling”, Digital Object Identifier IEEE, 2013

[7] B. Pang and L. Lee, “Opinion mining and sentiment analysis,” Foundations and Trends in Information Retrieval 2(1-2), 2008, pp. 1–135.

[8] M. Hu and B. Liu, "Mining and summarizing customer tweet," Proceedings of the tenth ACM international conference on Knowledge discovery and data mining, Seattle, 2004, pp. 168-177.

[9] https://api.themoviedb.org/3/ - Movie tweet Input API

[10] https://www.nltk.org/ - POS Tagging and Classification

[11] https://github.com/japerk/nltk-trainer - NLTK Trainer

[12] B. Pang, L. Lee, and S. Vaithyanathan, “Thumbs up? sentiment classification using machine learning techniques,” Proceedings of the ACL-02 conference on Empirical methods in natural language processing, vol.10, 2002, pp. 79-86.

[13] Seven Rill Goethe University Frankfurt, Germany “PoliTwi- Early detection of emerging political topics on Twitter and the impact on concept-level sentiment analysis”, Elsevier, 2014

[14] Andranik Tumasjan Technical University of Munich, Germany “Predicting elections with Twitter - what 140 characters reveal about political sentiment”, 4th International AAAI Conference, 2010

[15] A. Khan, B. Baharudin, K. Khan; “Sentiment Classification from Online Customer tweet Using Lexical Contextual Sentence Structure” ICSECS 2011: 2nd International Conference on Software Engineering and Computer Systems, Springer, pp. 317-331, 2011.

[16] Vadesara A, Tanna P, Joshi H (2021) Hate speech detection: a bird’s-eye view.

[17] S. Robertson, “Understanding inverse document frequency: On theoretical arguments for IDF,” J. Doc., 2004, doi: 10.1108/00220410410560582.