School of Information Technology and Engineering
Winter Semester -2023

INFORMATION SECURITY MANAGEMENT-CSE3502

PROJECT REPORT
On

HATE SPEECH DETECTION USING WEB APP

By

KALAMEGAM V	**- 20BIT0302**
PRIYA N	**- 20BIT0305**
PRIYADHARSHINI R	**- 20BIT0307**
ABRETHA BEGAM A	**- 20BIT0309**
SUDHANKUMAR R	**- 20BIT0316**

Submitted to

Dr.Sumaiya Thaseen I
Associate Professor Grade 1
School of Information Technology and Engineering

TABLE OF CONTENTS

1.ABSTRACT

The exponential increase of social media such as Twitter and neighbourhood boards has revolutionised verbal exchange and content publishing, however is additionally an increasing number of exploited for the propagation of hate speech and the agency of hate primarily based things to do. For years, social media groups such as Twitter, Facebook, and YouTube have been combating this problem and it has been estimated that lots of hundreds of thousands of euros are invested each 12 months on counter-measures which include manpower However,they are nevertheless being criticised for no longer doing enough. This is largely because such measures contain manually reviewing on-line contents to become aware of and delete offensive materials. The procedure is labour intensive, time consuming, and no longer sustainable or scalable in reality.

Most of us begin and end our day by scrolling through social media like Instagram, Twitter, Pinterest, YouTube, WhatsApp, Facebook, Snapchat etc. And thanks to the pandemic, lockdowns were introduced due to which many people have switched to the use of social media websites and other apps to pass on their spare time. With an interest to prevent social media users from being cyber bullied, we propose an automatic hate detection system, which is one of the modules from our proposed system, along with our own web page in our project. People from all age groups use social media in their everyday lives and this number increased exponentially after the pandemic. While it benefits us in many ways it also has some demerits among which we will be focussing on cyber bullying. Though not physically, Cyber bullying can take a toll on the mental well being of a person which in turn also affects them physically. In order to prevent this we will be proposing a mechanism to detect hate comments along with a fully functional web page through which people can communicate easily. We will be using a well known Long short-term memory classification model in order to detect hate messages to which the pre-processed data will be fed. The web page proposed in the project is built using Html,Css,Javascript,Python.

2.KEYWORDS:

Cyber bullying, Hate Detection ,Feature Extraction ,Long short-term memory ,Pre-processing, Social Media ,web page.

3.INTRODUCTION:

Social Media becomes an important matter of concern when it starts affecting people's lives in a negative way. Starting the day on a good note with nothing but positive thought in our mind is proven to make an individual's day better and more efficient. But imagine waking up to a comment discriminating you or people around you based on their beliefs, appearance, lifestyle, opinions, talent etc. or receiving death threats and hatred from a place which you once thought was a place to make new friends and share your opinion.

Social media can easily be a double edged sword. Multiple studies have found a strong link between heavy social media and an increased risk for depression, anxiety, loneliness, self-harm, and even suicidal thoughts. The mental health of a person is as important as their physical health. About 10 percent of teens report being bullied on social media and many other users are subjected to offensive comments. Social media platforms such as Twitter can be hotspots for spreading hurtful rumors, lies, and abuse that can leave lasting emotional scars. But simply staying away from social media is not the solution. While it can be toxic at times, many instances prove how social media helps to connect to people, spread news, educate and sometimes it even becomes the source of an income for many. Hence making sure that social media serves the purpose for which it was created while protecting its users from cyber bullying becomes the need of the hour. Which is why we will be proposing a hate speech and comments detection mechanism in this project which will automatically detect any kind of toxic comment present in the social media.

In order to make our project more effective than the existing hate detection systems, we will mainly be focussing on building a system which will be able to detect words which include special characters, extra spaces and numbers i.e paralinguistic signals and poorly written texts as well.

4.LITERATURE REVIEW:

Name: PRIYA N(20BIT0305)

1. Using Machine Learning for Detection of Hate Speech and Offensive Code Mixed Social Media text

METHOD:

- ✓ By testing out several Classifiers, supervised machine learning is applied. The labelled datasets underwent preprocessing to have any noisy components removed. The system can learn offensive phrase patterns by extracting the proper features.
- ✓ In order to select the best performing model, the performances of various Classifiers and feature models are finally compared using standard metrics.

DATASET:

Manglish (Malayalam English) and Tanglish languages of size 4000 rows each, are shared by HASOC organizers.

PROS:

The best performing classification models developed for both languages, are applied on test datasets. The model which gives the highest accuracy result on training dataset for Malayalam language, was experiment to predict the labels of respective Test dataset.

CONS:

Only applicable to two languages Tamil and Malayalam and applied to twitter.

2. Automatic Hate Speech Detection using Machine Learning: A Comparative Study

METHOD:

- ✓ Six steps namely, data collection, data preprocessing, feature engineering, data splitting, classification model construction, and classification model evaluation.

DATASET:

Publicly available hate speech tweets dataset. This dataset is compiled and labeled by CrowdFlower.

PROS:

Bigram features when used with the support vector machine algorithm best performed with 79% off overall accuracy

CONS:

The proposed ML model is inefficient in terms of real-time predictions accuracy for the data. It only classifies the hate speech message in three different classes and is not capable enough to identify the severity of the message.

3. Detecting Hate Speech and Offensive Language on Twitter using Machine Learning: An N-gram and TF IDF based Approach

METHOD:

- ✓ Logistic regression, Naïve Bayes, Support Vector machine used to detect the offensive words.

DATASET:

Combination of 3 datasets,2 from CrowdFlower and 1 from GitHub.

PROS:

Logistic Regression performs better with the optimal n-gram range 1 to 3 for the L2 normalization of TFIDF. Upon evaluating the model on test data, achieved 95.6% accuracy.

CONS:

It was seen that the model does not account for negative words present in sentence. 4.8% of the offensive tweets were misclassified as hateful.

Name: ABRETHA BEGAM.A(20BIT0309)

4. Public Hate Speech Detection Using Machine Learning.

METHOD:

- ✓ Random Forest.

DATASET:

Real time tweets

PROS:

By using the Random Forest algorithm which is the most popular and powerful algorithm of machine learning it provided better results when compared to the results using an Support Vector Machine (SVM) algorithm

CONS:

Accuracy could be improved.

5. Offensive Language and Hate Speech Detection with Deep Learning and Transfer Learning.

METHOD:

- ✓ LSTM, Transfer Learning.

DATASET:

Used public tweet data. The data is stored as a CSV and comprised of 24,783 tweets.

PROS:

The feedback connections in LSTM enables it processes entire sequences of data without treating each point in the sequence independently which makes them particularly good at processing sequences of data such as text, speech and general time-series. Using a pre-trained BERT model that was trained on a huge dataset, as a starting point helps to avoid over-fitting.

CONS:

Few prediction errors were produced.

6. Profiling Hate Speech Spreaders on Twitter Task at PAN 2021.

METHOD:

Combination of traditional machine learning methods with novel deep learning based representations.

DATASET:

Corpus with dataset. they have collected two hundred tweets per Twitter user to make a final dataset.

PROS:

By using a combination of traditional machine learning methods such as SVM or Logistic Regression with novel deep learning-based representations such as BERT and its variations they have obtained the highest results with good accuracy.

CONS:

High number of false positives were observed

Name: PRIYADHARSHINI.R(20BIT0307)

7. Multilingual Hate Speech and Offensive Content Detection using Modified Cross entropy Loss

METHOD:

Text **classification**, Deep-learning, Transfer learning, BERT, LSTMs, CNNs and SVM

DATASET:

Offensive language Identification Dataset (imbalanced dataset).

PROS:

They used large language models which are pre-trained on large corpora for hate speech detection tasks and to evaluate predictions by different models a validation dataset was created.

CONS:

The author’s used a simple dataset.

8. Hate Speech in the Political Discourse on social media: Disparities Across Parties, Gender, and Ethnicity

METHOD:

Machine learning, regression analysis.

DATASET:

Tweets from all 541 members of the 117th U. S. Congress

PROS:

Relevant both for politicians and from a societal perspective**,** imply statistically significant differences in how politicians are treated on social media depending on their party affiliation, gender, and ethnicity.

CONS:

Impede diversity in the composition of political institutions.

9. Nipping in the Bud: Detection, Diffusion and Mitigation of Hate Speech on social media

METHOD:

Multi-modal graphics

DATASET:

Harmful meme benchmark dataset.

PROS:

Theories from sociology and physiology might help researchers and practitioners better understand the emergence and spread of hate from a socio-cultural perspective.

CONS:

It doesn't provide the solutions that are human-Centric and not data-centric.

Name: KALAMEGAM.V(20BIT0302)

10. Cyberbully Detection Using 1D-CNN and LSTM

METHOD:

1D-CNN, LSTM, BiLSTM.

DATASET:

Jigsaw LLC Dataset

PROS:

By using BiLSTM the model is able to preserve information both from both past and the future which makes it more effective.

CONS:

The model shows errors while detecting sarcasm text from real threats which deteriorate es the accuracy and precision

11. Deep Learning for Hate speech Detection: A Comparative Study.

METHOD:

A largescale empirical comparison of deep and shallow hate-speech detection methods. pre-trained models.

DATASET:

Davidson data set, Founta data set and Twitter Sentiment Analysis

PROS:

Detection accuracy, computational efficiency.

CONS:

It is difficult to collect a dataset.

12. Overview of the HASOC track at FIRE 2020: Hate Speech and Offensive Content Identification in Indo-European Languages

METHOD:

1. Identifying Hate-Offensive and Non-Hate-Offensive content
2. Identifying Hate, Profane and Offensive posts
3. Data Annotation, Tweet allocation for annotations, Inter-Coder Reliability.

DATASET:

HASOC 2019 and HASOC 2020

PROS:

The best result for English is based on a LSTM which Used Glove embeddings as input

CONS:

Performance measures for the test set in HASOC 2020 have been considerably lower than those for HASOC 2019.

Name: SUDHANKUMAR R(20BIT0316)

13. Advances in Machine Learning Algorithms for Hate Speech Detection in social media

METHOD:

Data collection, Pre-Processing, Feature Extraction, Reducing Dimensions, Classification, it compares many ML and Deep Learning algorithms

DATASET:

- ✓ Text as a dataset.
- ✓ Creating new dataset from the social medias.

PROS:

Ensemble and deep learning approaches are used, Target audience are the ones who is beginner in the field of hate speech detection, its analysis both FB and twitter

CONS:

No experiment was conducted with a given dataset because it’s newly created dataset.

14. A Framework for Hate Speech Detection Using Deep Convolutional Neural Network

METHOD:

- ✓ Uses a deep convolutional neural network (DCNN) to address the hate speech. Detection issue.
- ✓ Traditional machine learning approach ✓ Deep learning-based approach.

DATASET:

Kaggle.com It was prepared by collecting the tweets from twitter.

PROS:

- ✓ The proposed DCNN model has higher and accurate results compared to the existing classifiers.
- ✓ It uses multiple hidden layers based trainable neural network architecture to extract the hidden features from the tweets.

CONS:

- ✓ Language is limited to English only.
- ✓ Issues with some textual dataset and for image dataset also.

15. Cross-Lingual Few-Shot Hate Speech and Offensive Language Detection Using Meta Learning

METHOD:

Meta learning-based approach based by using optimization-based and metric based (MAML and proto-MAML) methods.

DATASET:

Datasets comprising 15 datasets across 8 languages for hate speech and 6 datasets across 6 languages for offensive language.

PROS:

- ✓ Hate speech detection
- ✓ Offensive language detection
- ✓ Ablation study

CONS:

Annotation schema was approximately 73%. first level of annotation schema was very high as 86%, in the second level 75%, and in the third level 60%.

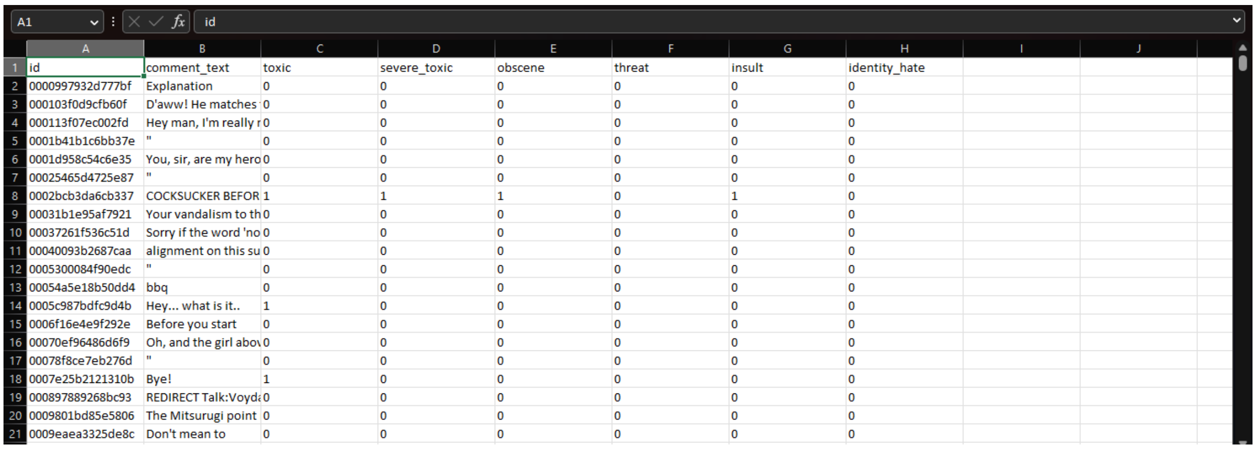
5.MATERIALS AND METHODS:

5.1. Materials:

In our project, we used two types of datasets for two types of LSTM model. In the dataset there are 1,59,571 rows are available. For the ensemble model we will be using a dataset from Github. This dataset from Github has been appended with the hate word in all forms including short forms in complete spelling, etc. by us. We will be using this dataset to compare existing machine learning classifiers.

Model-1 Data set link:
https://github.com/nicknochnack/CommentToxicity/blob/main/jigsaw-toxic-comment-classification-challenge/train.csv/train.csv

DATASET:

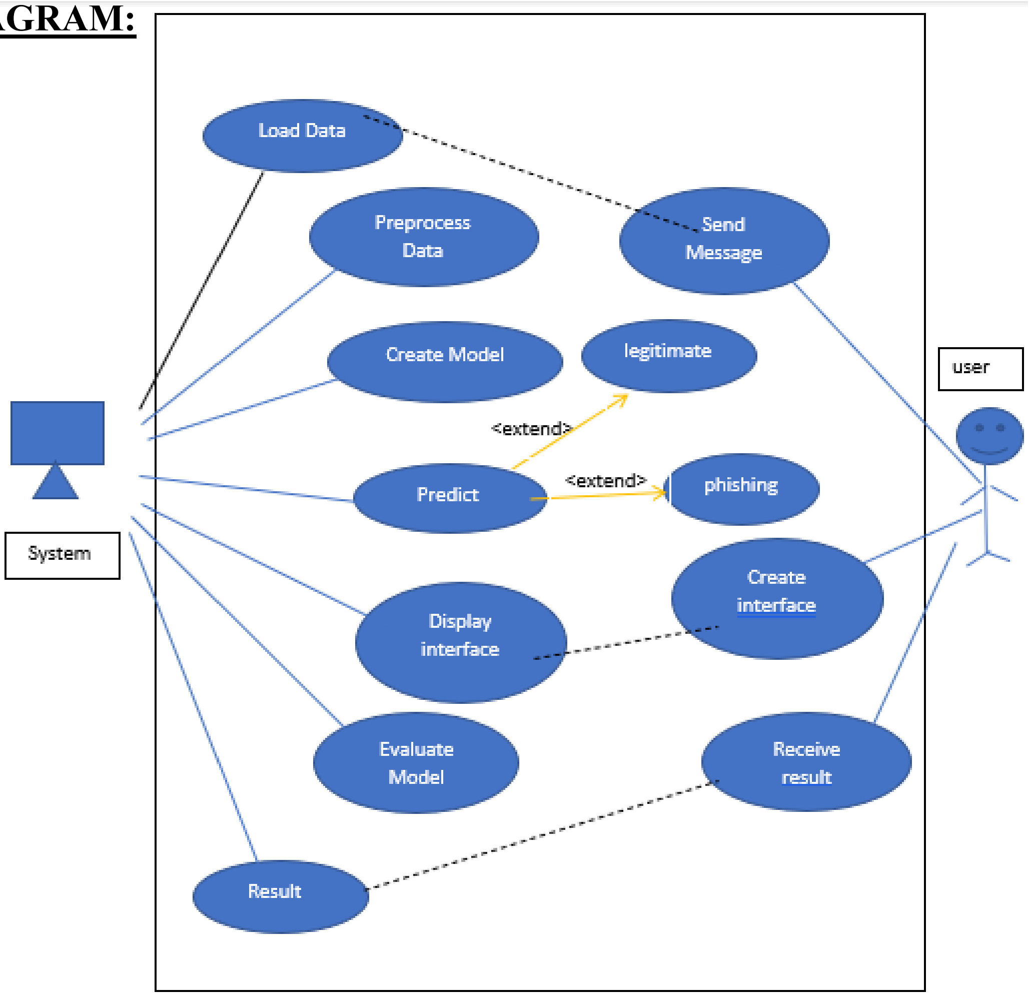


id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0000997932d777bf	Explanation	0	0	0	0	0	0
000103f0d9cfb60f	D'aww! He matches	0	0	0	0	0	0
000113f07ec002fd	Hey man, I'm really r	0	0	0	0	0	0
0001b41b1c6bb37e	"	0	0	0	0	0	0
0001d958c54c6e35	You, sir, are my hero	0	0	0	0	0	0
00025465d4725e87	"	0	0	0	0	0	0
0002bcb3da6cb337	COCKSUCKER BEFOR	1	1	1	0	1	0
00031b1e95af7921	Your vandalism to th	0	0	0	0	0	0
00037261f536c51d	Sorry if the word 'no	0	0	0	0	0	0
00040093b2687caa	alignment on this su	0	0	0	0	0	0
0005300084f90edc	"	0	0	0	0	0	0
00054a5e18b50dd4	bbq	0	0	0	0	0	0
0005c987bdfc9d4b	Hey... what is it..	1	0	0	0	0	0
0006f16e4e9f292e	Before you start	0	0	0	0	0	0
00070ef96486d6f9	Oh, and the girl abov	0	0	0	0	0	0
00078f8ce7eb276d	"	0	0	0	0	0	0
0007e25b2121310b	Bye!	1	0	0	0	0	0
000897889268bc93	REDIRECT Talk:Voyd	0	0	0	0	0	0
0009801bd85e5806	The Mitsurugi point	0	0	0	0	0	0
0009eaea3325de8c	Don't mean to	0	0	0	0	0	0

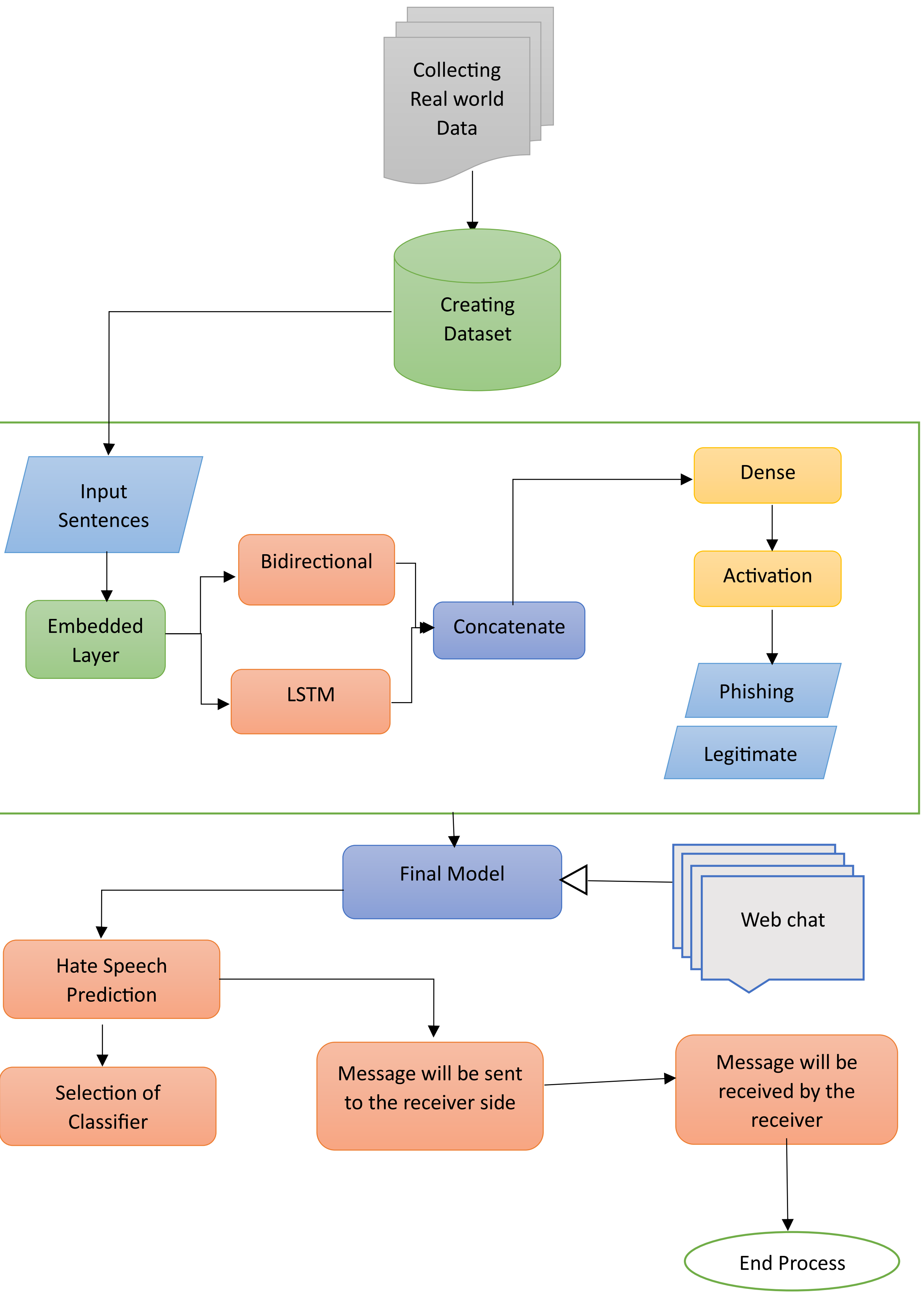
5.2. Architecture Diagram:

The architecture diagram presented below represents the whole process where the end user will be entering our website where they can enter the message which they want to send to the receiver. The message will be checked with the dataset in the backend and if the message doesn't contain any harsh comment, it will be delivered to the Receiver's side. If not, the system will alert the user saying that hate speech was found and ask the user to recheck the message which they have sent,

USE-CASE DIAGRAM:



High Level Design:



5.3.Methods:

Model-1 (BIDIRECTIONAL LSTM):

5.3.1.Preprocess:

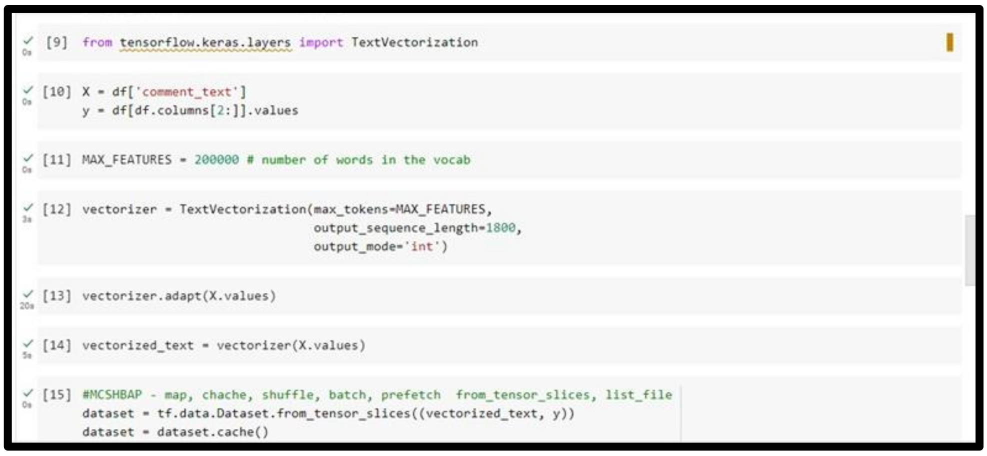
A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

The data which we collected from the dataset was first pre-processed, thus removing unwanted contents which results in the next phase of feature Extraction. We then extracted the required feature from the pre-processed data which was then fed to the LSTM classifier.

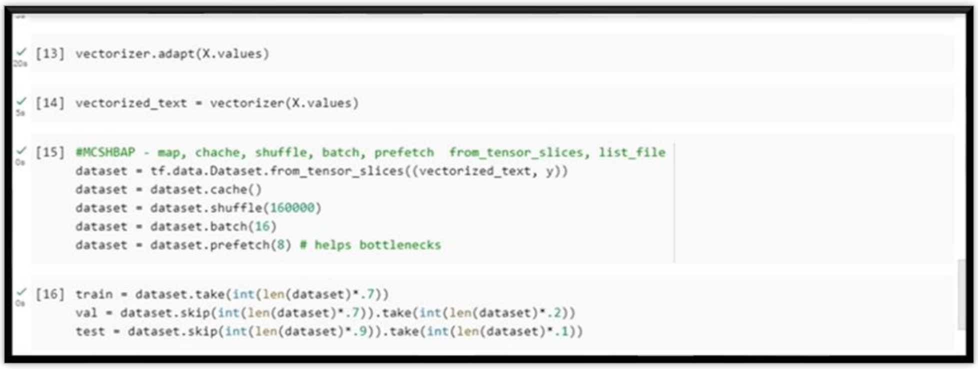
70% data – train data

20% data – validation data

10% data – test data



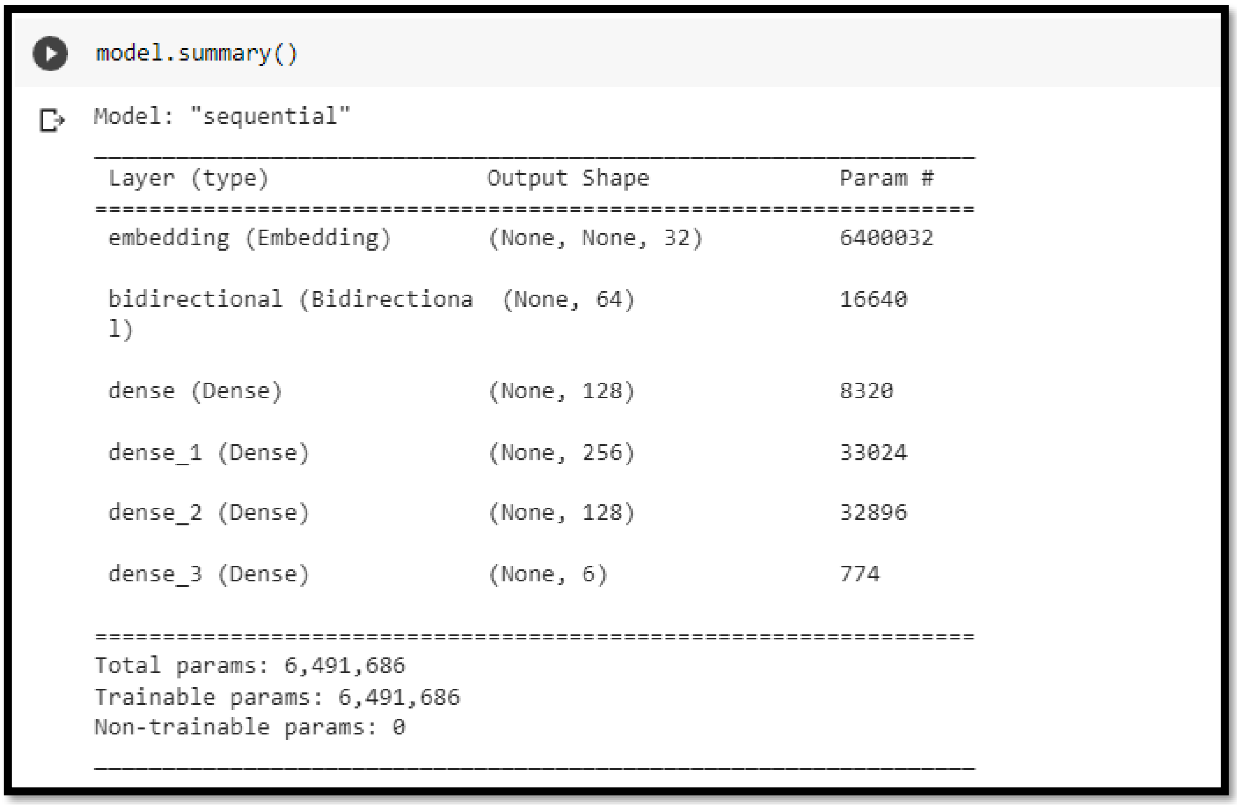
```
[9]  from tensorflow.keras.layers import TextVectorization

[10] X = df['comment_text']
     y = df[df.columns[2:]].values

[11] MAX_FEATURES = 200000 # number of words in the vocab

[12] vectorizer = TextVectorization(max_tokens=MAX_FEATURES,
                                   output_sequence_length=1800,
                                   output_mode='int')

[13] vectorizer.adapt(X.values)

[14] vectorized_text = vectorizer(X.values)

[15] #MCSHBAP - map, chache, shuffle, batch, prefetch  from_tensor_slices, list_file
     dataset = tf.data.Dataset.from_tensor_slices((vectorized_text, y))
     dataset = dataset.cache()
```



```
[13] vectorizer.adapt(X.values)

[14] vectorized_text = vectorizer(X.values)

[15] #MCSHBAP - map, chache, shuffle, batch, prefetch  from_tensor_slices, list_file
     dataset = tf.data.Dataset.from_tensor_slices((vectorized_text, y))
     dataset = dataset.cache()
     dataset = dataset.shuffle(160000)
     dataset = dataset.batch(16)
     dataset = dataset.prefetch(8) # helps bottlenecks

[16] train = dataset.take(int(len(dataset)*.7))
     val = dataset.skip(int(len(dataset)*.7)).take(int(len(dataset)*.2))
     test = dataset.skip(int(len(dataset)*.9)).take(int(len(dataset)*.1))
```

5.3.2. Create Sequential Model

We have chosen to use LSTM because it is a type of recurrent neural network but it is better than traditional recurrent neural networks in terms of memory. Having a good hold over memorizing certain patterns LSTMs perform fairly better.

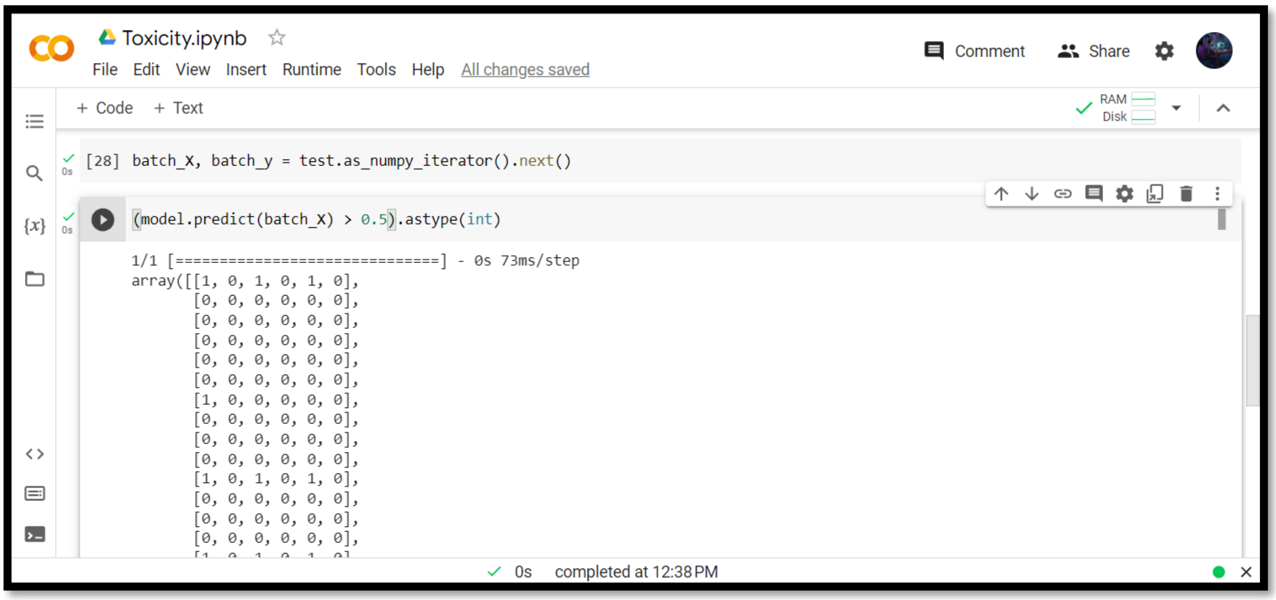


```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 32)          6400032

 bidirectional (Bidirectiona  (None, 64)               16640
 l)

 dense (Dense)               (None, 128)               8320

 dense_1 (Dense)             (None, 256)               33024

 dense_2 (Dense)             (None, 128)               32896

 dense_3 (Dense)             (None, 6)                 774

=================================================================
Total params: 6,491,686
Trainable params: 6,491,686
Non-trainable params: 0
_________________________________________________________________
```

5.3.3. Make Predictions:

Appending the input as a single array. The threshold value of our model is 0.5. Whenever the predicted result value is greater than threshold then the output will be 1 otherwise 0.



```
[28] batch_X, batch_y = test.as_numpy_iterator().next()
```

```
(model.predict(batch_X) > 0.5).astype(int)
```

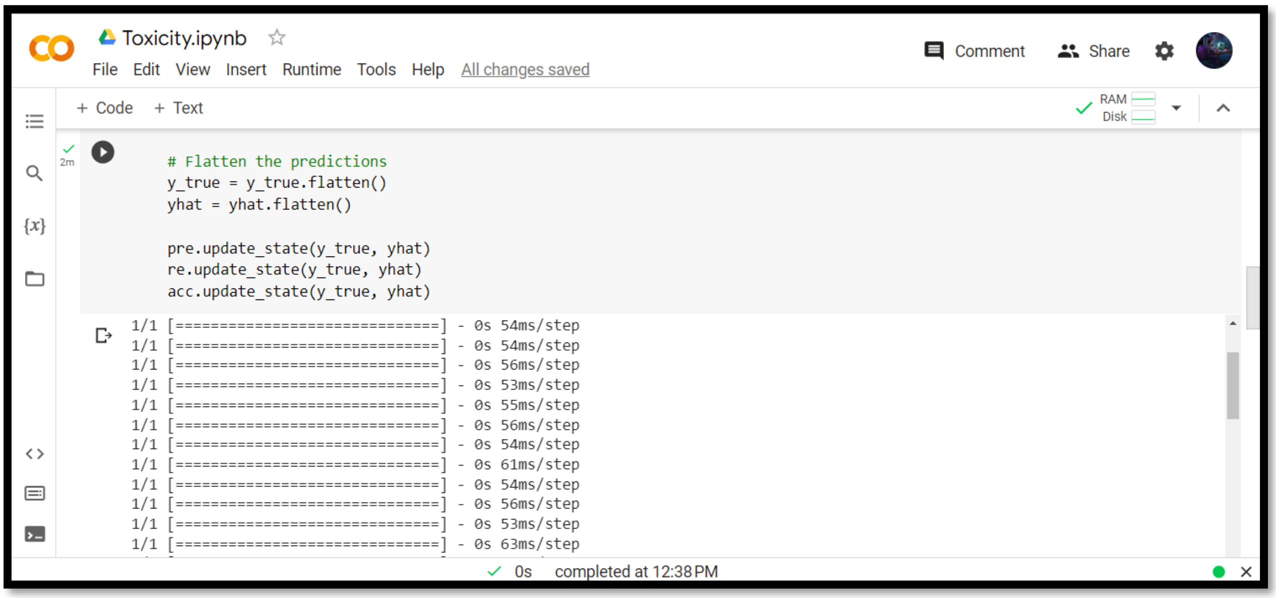
```
1/1 [==============================] - 0s 73ms/step
array([[1, 0, 1, 0, 1, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [1, 0, 1, 0, 1, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
```

5.3.4. Evaluate Model:

We have test data of 10,000 approximately

Actual output → toxic = 7,000 & normal = 3,000

Model predicted → toxic = 4,000 & normal = 1,500



```
# Flatten the predictions
y_true = y_true.flatten()
yhat = yhat.flatten()

pre.update_state(y_true, yhat)
re.update_state(y_true, yhat)
acc.update_state(y_true, yhat)
```

```
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 63ms/step
```

✓ 0s completed at 12:38 PM

Mathematical explanation:

<u>Example:</u>

We have test data of 10,000 approximately

Actual output → toxic = 7,000 & normal = 3,000

Model predicted → toxic = 4,000 & normal = 1,500

Confusion matrix:

	Model predicted	
Actual output	4,000 (TP) Toxic comment	3,000 (FN) Toxic -> normal
	1,500 (FP) Normal -> toxic	1,500 (TN) Normal comment

a) **Accuracy:**

Accuracy is a metric that generally describes how the model performs across all classes. It is useful when all classes are of equal importance. It is calculated as the ratio between the number of correct predictions to the total number of predictions.

$$Accuracy = \frac{True_{positive} + True_{negative}}{True_{positive} + True_{negative} + False_{positive} + False_{negative}}$$

$$\text{Accuracy} = \frac{4{,}000 + 1{,}500}{4{,}000 + 1{,}500 + 1{,}500 + 3{,}000}$$

$$= \frac{5{,}500}{10{,}000}$$

$$= 0.55$$

Accuracy = 55 %

b) **<u>Precision:</u>**

The precision is calculated as the ratio between the number of *Positive* samples correctly classified to the total number of samples classified as *Positive* (either correctly or incorrectly). The precision measures the model's accuracy in classifying a sample as positive.

$$Precision = \frac{True_{positive}}{True_{positive} + False_{positive}}$$

$$\text{Precision} = \frac{4{,}000}{4{,}000 + 1{,}500}$$

$$= \frac{4{,}000}{5{,}500}$$

$$= 0.7272$$

Precision = 72 %

c) **<u>Recall:</u>**

The recall is calculated as the ratio between the number of *Positive* samples correctly classified as *Positive* to the total number of *Positive* samples. The recall measures the model's ability to detect *Positive* samples. The higher the recall, the more positive samples detected.

$$Recall = \frac{True_{positive}}{True_{positive} + False_{negative}}$$

$$\text{Recall} = \frac{4{,}000}{4{,}000 + 3{,}000}$$

$$= \frac{4{,}000}{7{,}000}$$

$$= 0.5714$$

Recall = 57 %

5.4. Pseudo code:

```
    1. Initialize the net parameters.
    2. Set the max training steps.
    3. Convert the format of Input data.
    4. Input the training data.
REPEAT
    5. Calculate output Y=LSTM(X)
    6. Calculate Loss function.
    7. Back updates the net parameters
UNTIL
    8. Loss meets the termination condition, or the training steps reach the maximum
END
```

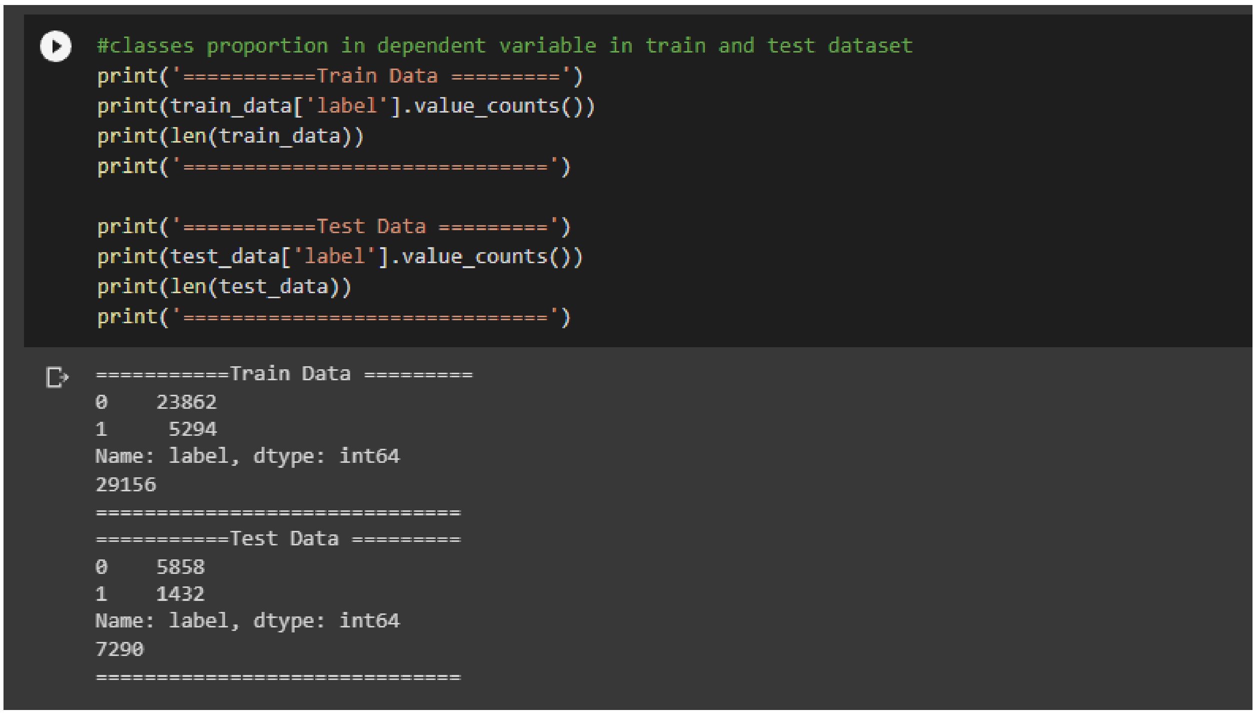
Model-1 Google-Colab link:
https://colab.research.google.com/drive/1jBXkAS2uRVewohb1I0SPD6sm_e2J3nW7?usp=sharing

6.EXPERIMENTATION AND ANALYSIS:

We have also compared many ML classification algorithms in order to get a clear idea about the accuracy offered by different algorithms and provided the results below in this report. We compared the 2 models such as LSTM and Bi-directional LSTM . We analysed both model's Precision, Recall and Accuracy. The model Which gave more Accuracy percentage we used that model to create API for our project.

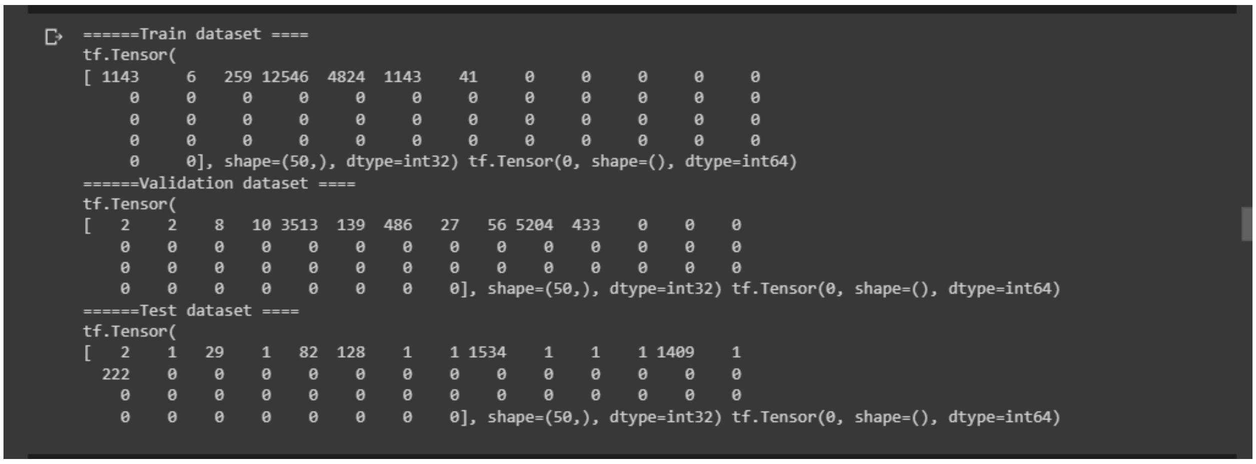
Model -2 (LSTM using Tensorflow)

Preprocess: Splitting training and testing dataset



```
#classes proportion in dependent variable in train and test dataset
print('===========Train Data =========')
print(train_data['label'].value_counts())
print(len(train_data))
print('==============================')

print('===========Test Data =========')
print(test_data['label'].value_counts())
print(len(test_data))
print('==============================')
```

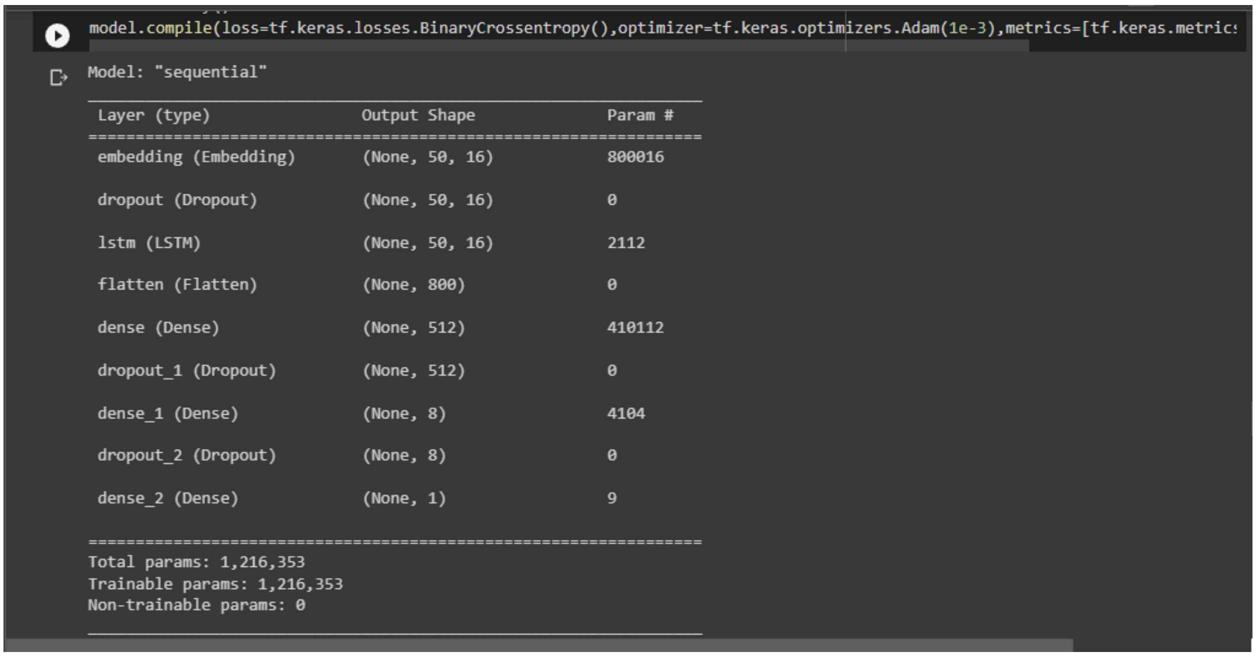
```
===========Train Data =========
0    23862
1     5294
Name: label, dtype: int64
29156
==============================
===========Test Data =========
0    5858
1    1432
Name: label, dtype: int64
7290
==============================
```



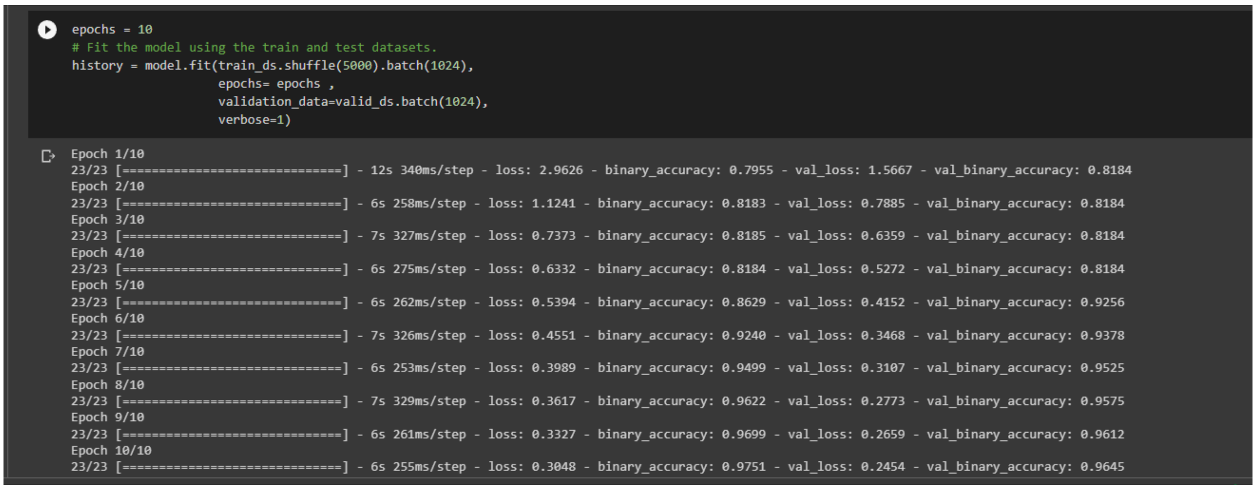
```
======Train dataset ====
tf.Tensor(
[ 1143     6   259 12546  4824  1143    41     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0], shape=(50,), dtype=int32) tf.Tensor(0, shape=(), dtype=int64)
======Validation dataset ====
tf.Tensor(
[   2    2    8   10 3513  139  486   27   56 5204  433    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0], shape=(50,), dtype=int32) tf.Tensor(0, shape=(), dtype=int64)
======Test dataset ====
tf.Tensor(
[   2    1   29    1   82  128    1    1 1534    1    1    1 1409    1
  222    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0], shape=(50,), dtype=int32) tf.Tensor(0, shape=(), dtype=int64)
```

Model-2 Data set link:
https://github.com/tahajunaid/NLP-Hate-Speech-Detection/blob/master/train.csv

Creating the model:

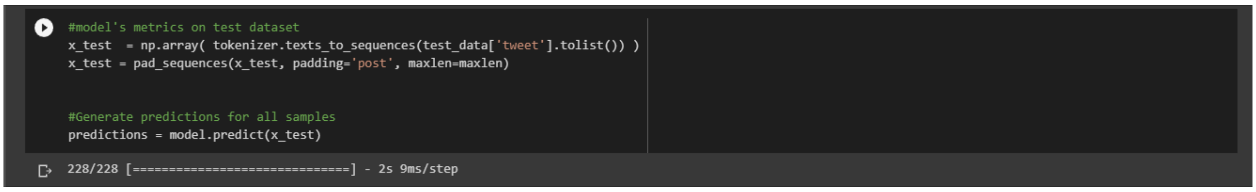


```
model.compile(loss=tf.keras.losses.BinaryCrossentropy(),optimizer=tf.keras.optimizers.Adam(1e-3),metrics=[tf.keras.metric

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 50, 16)            800016

 dropout (Dropout)           (None, 50, 16)            0

 lstm (LSTM)                 (None, 50, 16)            2112

 flatten (Flatten)           (None, 800)               0

 dense (Dense)               (None, 512)               410112

 dropout_1 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 8)                 4104

 dropout_2 (Dropout)         (None, 8)                 0

 dense_2 (Dense)             (None, 1)                 9

=================================================================
Total params: 1,216,353
Trainable params: 1,216,353
Non-trainable params: 0
_________________________________________________________________
```

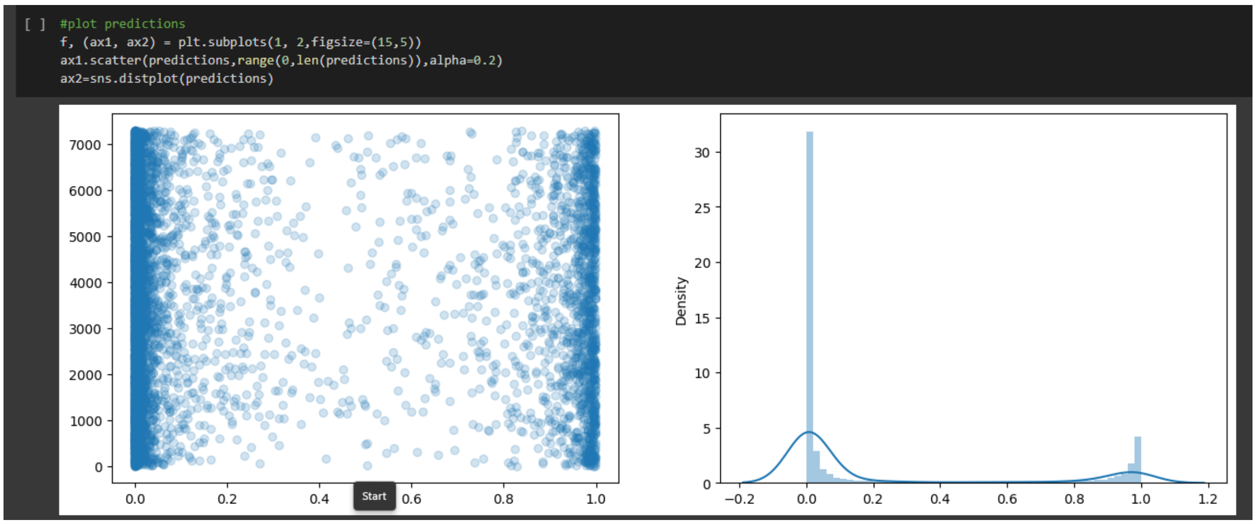


```
epochs = 10
# Fit the model using the train and test datasets.
history = model.fit(train_ds.shuffle(5000).batch(1024),
                    epochs= epochs ,
                    validation_data=valid_ds.batch(1024),
                    verbose=1)

Epoch 1/10
23/23 [==============================] - 12s 340ms/step - loss: 2.9626 - binary_accuracy: 0.7955 - val_loss: 1.5667 - val_binary_accuracy: 0.8184
Epoch 2/10
23/23 [==============================] - 6s 258ms/step - loss: 1.1241 - binary_accuracy: 0.8183 - val_loss: 0.7885 - val_binary_accuracy: 0.8184
Epoch 3/10
23/23 [==============================] - 7s 327ms/step - loss: 0.7373 - binary_accuracy: 0.8185 - val_loss: 0.6359 - val_binary_accuracy: 0.8184
Epoch 4/10
23/23 [==============================] - 6s 275ms/step - loss: 0.6332 - binary_accuracy: 0.8184 - val_loss: 0.5272 - val_binary_accuracy: 0.8184
Epoch 5/10
23/23 [==============================] - 6s 262ms/step - loss: 0.5394 - binary_accuracy: 0.8629 - val_loss: 0.4152 - val_binary_accuracy: 0.9256
Epoch 6/10
23/23 [==============================] - 7s 326ms/step - loss: 0.4551 - binary_accuracy: 0.9240 - val_loss: 0.3468 - val_binary_accuracy: 0.9378
Epoch 7/10
23/23 [==============================] - 6s 253ms/step - loss: 0.3989 - binary_accuracy: 0.9499 - val_loss: 0.3107 - val_binary_accuracy: 0.9525
Epoch 8/10
23/23 [==============================] - 7s 329ms/step - loss: 0.3617 - binary_accuracy: 0.9622 - val_loss: 0.2773 - val_binary_accuracy: 0.9575
Epoch 9/10
23/23 [==============================] - 6s 261ms/step - loss: 0.3327 - binary_accuracy: 0.9699 - val_loss: 0.2659 - val_binary_accuracy: 0.9612
Epoch 10/10
23/23 [==============================] - 6s 255ms/step - loss: 0.3048 - binary_accuracy: 0.9751 - val_loss: 0.2454 - val_binary_accuracy: 0.9645
```

Make prediction:

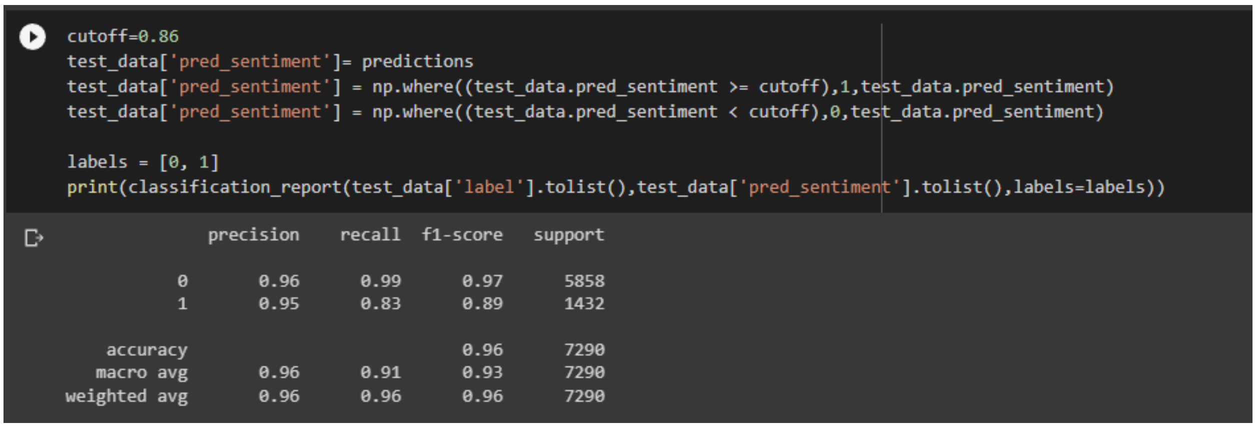


```
#model's metrics on test dataset
x_test  = np.array( tokenizer.texts_to_sequences(test_data['tweet'].tolist()) )
x_test = pad_sequences(x_test, padding='post', maxlen=maxlen)


#Generate predictions for all samples
predictions = model.predict(x_test)

228/228 [==============================] - 2s 9ms/step
```



```
[ ] #plot predictions
    f, (ax1, ax2) = plt.subplots(1, 2,figsize=(15,5))
    ax1.scatter(predictions,range(0,len(predictions)),alpha=0.2)
    ax2=sns.distplot(predictions)
```

Evaluate the model:



```
cutoff=0.86
test_data['pred_sentiment']= predictions
test_data['pred_sentiment'] = np.where((test_data.pred_sentiment >= cutoff),1,test_data.pred_sentiment)
test_data['pred_sentiment'] = np.where((test_data.pred_sentiment < cutoff),0,test_data.pred_sentiment)

labels = [0, 1]
print(classification_report(test_data['label'].tolist(),test_data['pred_sentiment'].tolist(),labels=labels))
```

```
              precision    recall  f1-score   support

           0       0.96      0.99      0.97      5858
           1       0.95      0.83      0.89      1432

    accuracy                           0.96      7290
   macro avg       0.96      0.91      0.93      7290
weighted avg       0.96      0.96      0.96      7290
```

Model-2 Google_colab link:
https://colab.research.google.com/drive/1u24AjC0HJhp_n5T9vev5xsibNGZDshat?usp=sharing

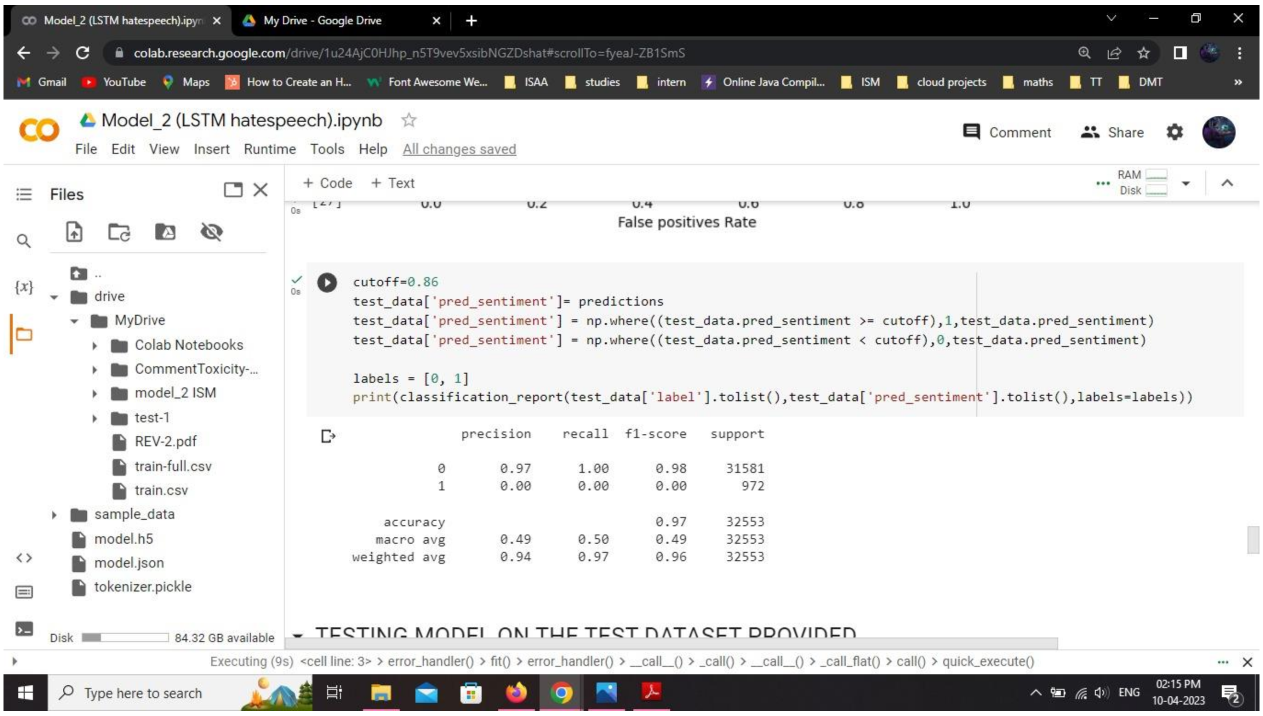
COMPARATIVE ANALYSIS:

From our comparative analysis on various Deep learning model we found the LSTM using Tensorflow to be the one with higher accuracy. Here is the accuracy percentage of both models that we used in our project.

	BIDIRECTIONAL LSTM	LSTM WITH TENSORFLOW
PRECISION	90.03%	95.5%
RECALL	92.34%	91%
ACCURACY	54.55% (for 10 epochs)	96% (for 10 epochs)

. From the table we can see that LSTM with Tensorflow gave 96% for 10 epochs. So we used this model to predict the hate speeches.

COMPARISON OF TWO MODELS WITH SAME DATASET:


Model_2 (LSTM hatespeech).ipynb
My Drive - Google Drive
colab.research.google.com/drive/1u24AjC0HJhp_n5T9vev5xsibNGZDshat#scrollTo=fyeaJ-ZB1SmS
File Edit View Insert Runtime Tools Help All changes saved
Files
drive
MyDrive
Colab Notebooks
CommentToxicity-...
model_2 ISM
test-1
REV-2.pdf
train-full.csv
train.csv
sample_data
model.h5
model.json
tokenizer.pickle
Disk 84.32 GB available
False positives Rate
TESTING MODEL ON THE TEST DATASET PROVIDED

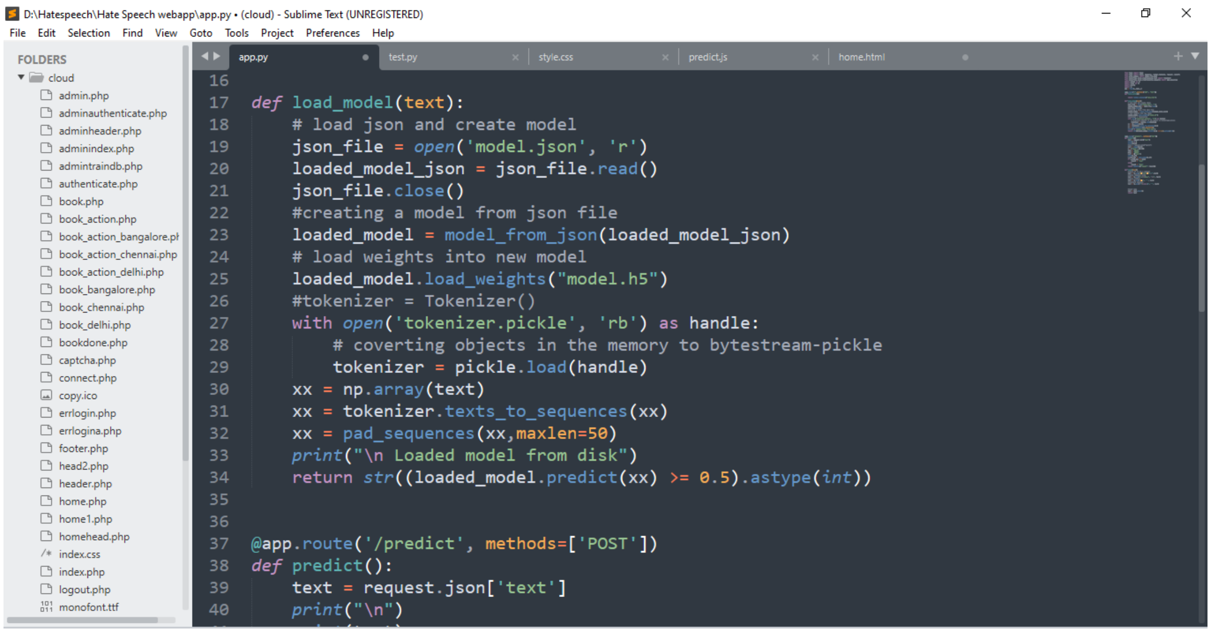
```
cutoff=0.86
test_data['pred_sentiment']= predictions
test_data['pred_sentiment'] = np.where((test_data.pred_sentiment >= cutoff),1,test_data.pred_sentiment)
test_data['pred_sentiment'] = np.where((test_data.pred_sentiment < cutoff),0,test_data.pred_sentiment)

labels = [0, 1]
print(classification_report(test_data['label'].tolist(),test_data['pred_sentiment'].tolist(),labels=labels))
```

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.98     31581
           1       0.00      0.00      0.00       972

    accuracy                           0.97     32553
   macro avg       0.49      0.50      0.49     32553
weighted avg       0.94      0.97      0.96     32553
```

Executing (9s) <cell line: 3> > error_handler() > fit() > error_handler() > __call__() > _call() > __call__() > _call_flat() > call() > quick_execute()

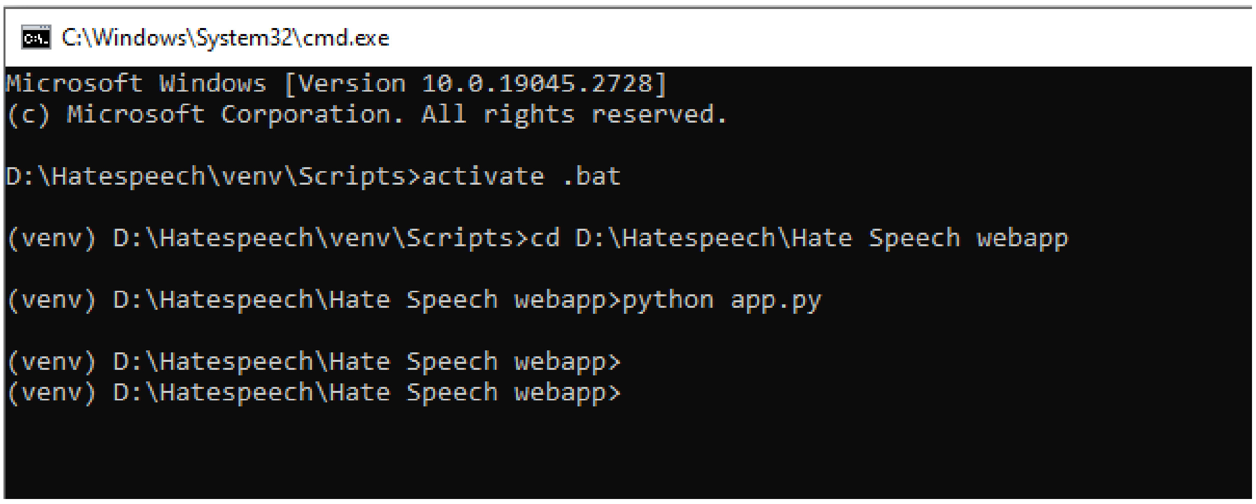
IMPLEMENTATION OF OUR MODEL:

App.py



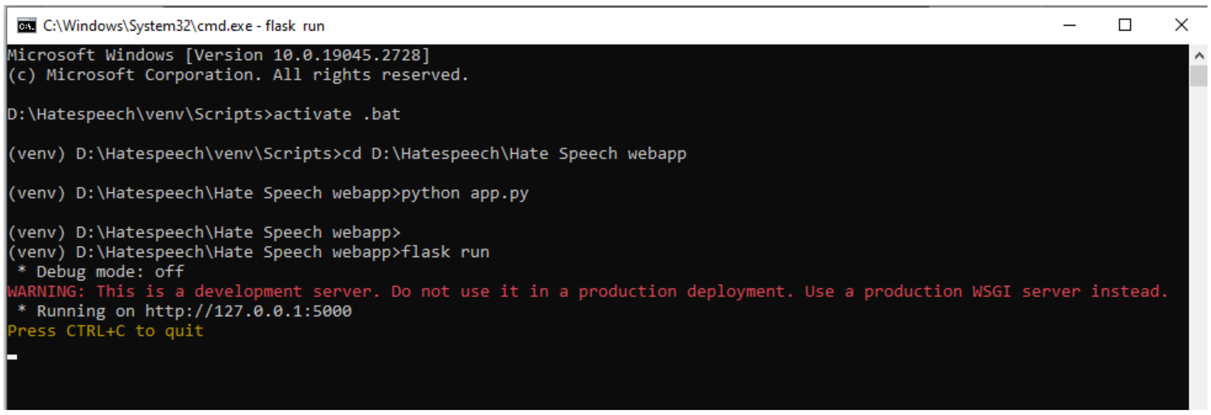
```python

def load_model(text):
    # load json and create model
    json_file = open('model.json', 'r')
    loaded_model_json = json_file.read()
    json_file.close()
    #creating a model from json file
    loaded_model = model_from_json(loaded_model_json)
    # load weights into new model
    loaded_model.load_weights("model.h5")
    #tokenizer = Tokenizer()
    with open('tokenizer.pickle', 'rb') as handle:
        # coverting objects in the memory to bytestream-pickle
        tokenizer = pickle.load(handle)
    xx = np.array(text)
    xx = tokenizer.texts_to_sequences(xx)
    xx = pad_sequences(xx,maxlen=50)
    print("\n Loaded model from disk")
    return str((loaded_model.predict(xx) >= 0.5).astype(int))


@app.route('/predict', methods=['POST'])
def predict():
    text = request.json['text']
    print("\n")
```

Running the main python file,



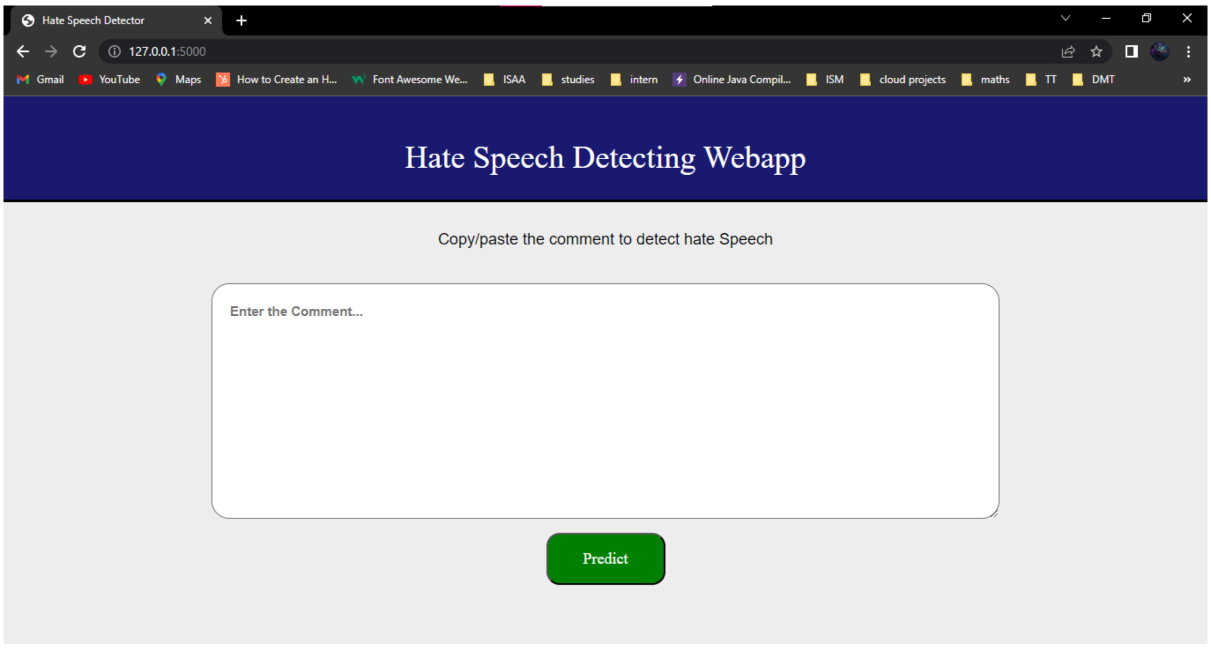
```
Microsoft Windows [Version 10.0.19045.2728]
(c) Microsoft Corporation. All rights reserved.

D:\Hatespeech\venv\Scripts>activate .bat

(venv) D:\Hatespeech\venv\Scripts>cd D:\Hatespeech\Hate Speech webapp

(venv) D:\Hatespeech\Hate Speech webapp>python app.py

(venv) D:\Hatespeech\Hate Speech webapp>
(venv) D:\Hatespeech\Hate Speech webapp>
```

Starting the Flask API,



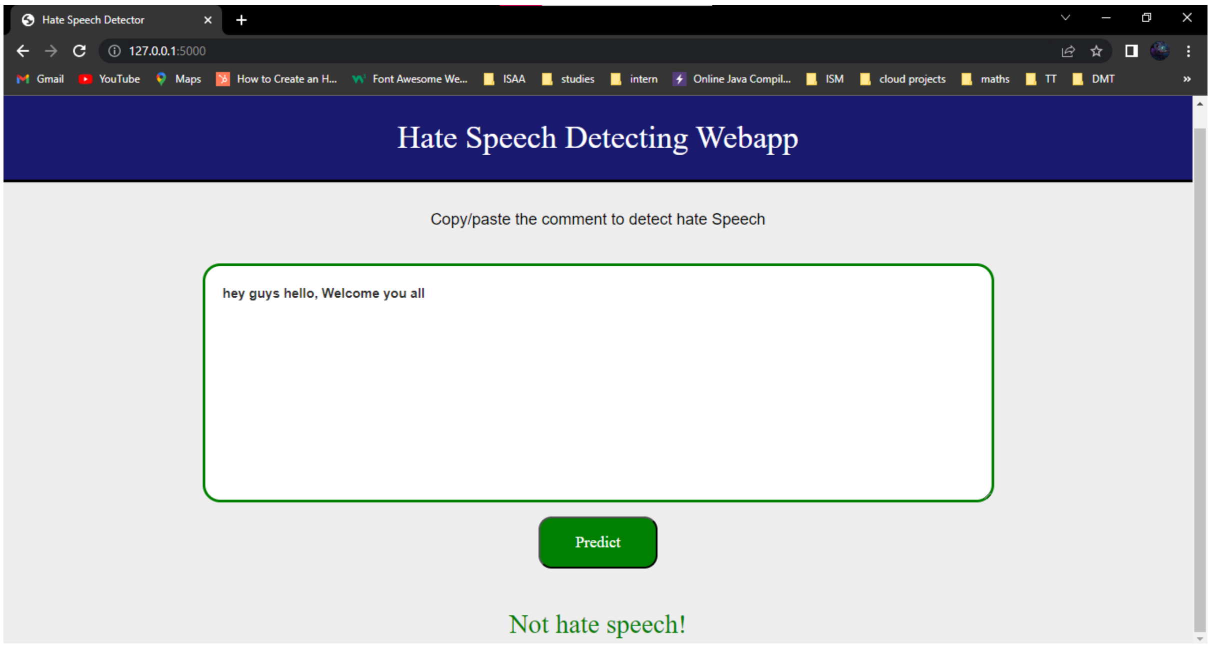
```
C:\Windows\System32\cmd.exe - flask  run
Microsoft Windows [Version 10.0.19045.2728]
(c) Microsoft Corporation. All rights reserved.

D:\Hatespeech\venv\Scripts>activate .bat

(venv) D:\Hatespeech\venv\Scripts>cd D:\Hatespeech\Hate Speech webapp

(venv) D:\Hatespeech\Hate Speech webapp>python app.py

(venv) D:\Hatespeech\Hate Speech webapp>
(venv) D:\Hatespeech\Hate Speech webapp>flask run
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

Accessing through localhost,



Results:

Normal tweets:



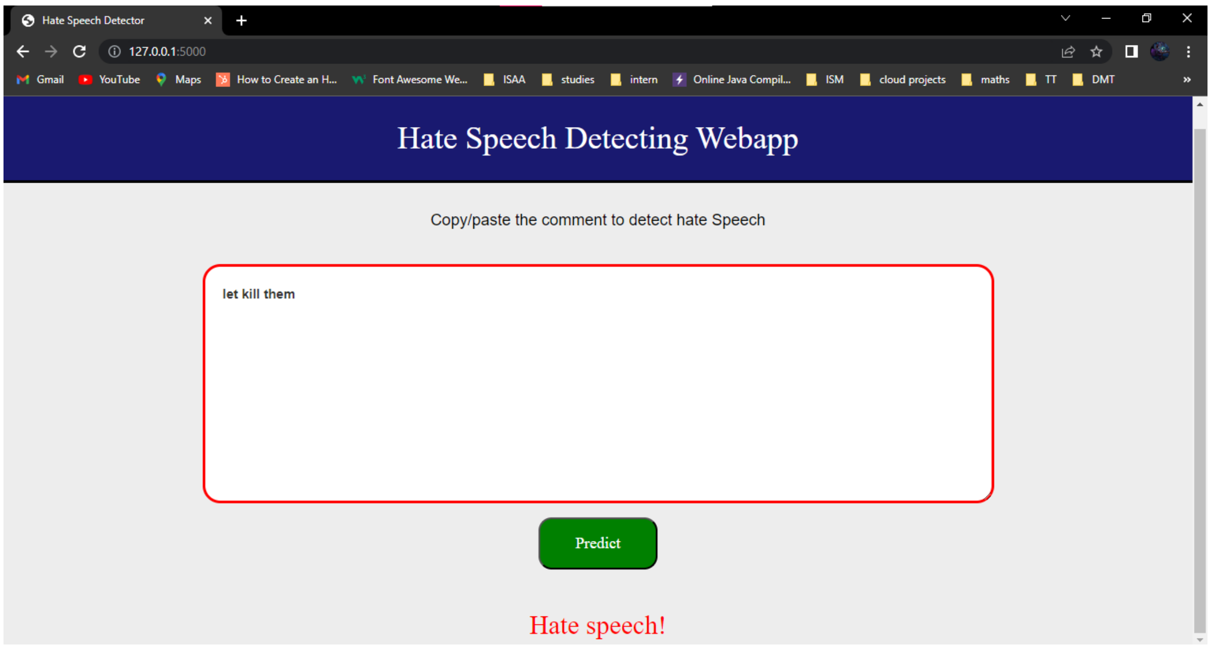
Hate Speech Detecting Webapp

Copy/paste the comment to detect hate Speech

hey guys hello, Welcome you all

Not hate speech!

Hate tweets:



Hate Speech Detecting Webapp

Copy/paste the comment to detect hate Speech

let kill them

Hate speech!

7.CONCLUSION:

In this project we have created an automatic hate speech detection mechanism which will help us to identify hate comments in chat applications, aiding us to keep cyber bullying under control. Our hate detection system using the LSTM classifier module can be implemented as a middleman between the sender and the receiver in a chat box using which people can communicate easily. Our main objective was to ensure that text messages which contain hate speech are detected while they are being written itself. This protects the receiver from the abuse and trauma he/she may face otherwise. We have chosen a supervised classification algorithm called LSTM classifier in order to classify the messages into offensive or non-offensive content. We have also compared several ML algorithms apart from the LSTM, the results of which showed that LSTM works the best for our system. We have implemented our web page using Html,css,Javascript,Python.

8.REFERENCES

[1] Pathak, V., Joshi, M., Joshi, P., Mundada, M., & Joshi, T. (2021). Kbcnmujal@ hasoc-dravidian-codemix-fire2020: Using machine learning for detection of hate speech and offensive code-mixed social media text. arXiv preprint arXiv:2102.09866.

[2] Abro, S., Sarang Shaikh, Z. A., Khan, S., Mujtaba, G., & Khand, Z. H. (2020). Automatic hate speech detection using machine learning: A comparative study. Machine Learning, 10(6).

[3] Gaydhani, A., Doma, V., Kendre, S., & Bhagwat, L. (2018). Detecting hate speech and offensive language on twitter using machine learning: An n-gram and tfidf based approach. arXiv preprint arXiv:1809.08651.

[4] Mullah, N. S., & Zainon, W. M. N. W. (2021). Advances in Machine Learning Algorithms for Hate Speech Detection in Social Media: A Review. IEEE Access.

[5] Mandl, T., Modha, S., Majumder, P., Patel, D., Dave, M., Mandlia, C., & Patel, A. (2019, December). Overview of the hasoc track at fire 2019: Hate speech and offensive content identification in indo-european languages. In Proceedings of the 11th forum for information retrieval evaluation (pp. 14-17).

[6] Jain, V., Kumar, V., Pal, V., & Vishwakarma, D. K. (2021, April). Detection of Cyberbullying on Social Media Using Machine learning. In 2021 5th International

Conference on Computing Methodologies and Communication (ICCMC) (pp. 1091-1096). IEEE.

[7] Roy, P. K., Tripathy, A. K., Das, T. K., & Gao, X. Z. (2020). A framework for hate speech detection using deep convolutional neural network. IEEE Access, 8, 204951-204962.

[8] Ketsbaia, L., Issac, B., & Chen, X. (2020, December). Detection of Hate Tweets using Machine Learning and Deep Learning. In 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) (pp. 751-758). IEEE.

[9] Wadhwani, A., Jain, P., & Sahu, S. (2021, February). Injurious Comment Detection and Removal utilizing Neural Network. In 2021 International Conference on Innovative Practices in Technology and Management (ICIPTM) (pp. 165-168). IEEE.

[10] Dubey, K., Nair, R., Khan, M. U., & Shaikh, S. (2020, December). Toxic Comment Detection using LSTM. In 2020 Third International Conference on Advances in Electronics, Computers and Communications (ICAECC) (pp. 1-8). IEEE.

[11] Kovács, G., Alonso, P., & Saini, R. (2021). Challenges of hate speech detection in social media. SN Computer Science, 2(2), 1-15.

[12] Jadhav, S. R., Rokade, A. D., Sable, A. N., & Gade, V. B. (2021). Public hate speech detection using machine learning: a review. Int J, 5(12), 72-75.

[13] Wei, B., Li, J., Gupta, A., Umair, H., Vovor, A., & Durzynski, N. (2021). Offensive Language and Hate Speech Detection with Deep Learning and Transfer Learning. arXiv preprint arXiv:2108.03305 [14] Rangel, F., Sarracén, G. L. D. L. P., Chulvi, B., Fersini, E., & Rosso, P. (2021). Profiling hate speech spreaders on twitter task at PAN 2021. In CLEF.

[15] Ghosh, S., Chaki, A., & Kudeshia, A. (2021). Cyberbully Detection Using 1D-CNN and LSTM. In Proceedings of International Conference on Communication, Circuits, and Systems (pp. 295-301). Springer, Singapore.

[16] Singh Malik, J., Pang, G., & van den Hengel, A. (2022). Deep Learning for Hate Speech Detection: A Comparative Study. arXiv e-prints, arXiv-2202.

[17] Mozafari, M., Farahbakhsh, R., & Crespi, N. (2022). Cross-Lingual Few-Shot Hate Speech and Offensive Language Detection Using Meta Learning. IEEE Access, 10, 14880-14896.

[18] Mitra, A., & Sankhala, P. (2022). Multilingual Hate Speech and Offensive Content Detection using Modified Cross-entropy Loss. arXiv preprint arXiv:2202.02635.

[19]. R.B. Pachori, P. Sircar, A novel technique to reduce cross terms in the squared magnitude of the wavelet transform and the short time Fourier transform, in IEEE International Workshop on Intelligent Signal Processing (Faro, Portugal, 2005), pp. 217–222

[20]. P. Flandrin, B. EscudiÃl’, An interpretation of the pseudo-Wigner-Ville distribution. Signal Process. 6, 27–36 (1984)

[21]. D. Ping, P. Zhao, B. Deng: Cross-terms suppression inWigner-Ville distribution based on image processing, in 2010 IEEE International Conference on Information and Automation (2010), pp. 2168–2171

[22]. P. Meena, R.R. Sharma, R.B. Pachori, Cross-term suppression in the Wigner-Ville distribution using variational mode decomposition, in 5th International Conference on Signal Processing,Computing, and Control (ISPCC-2k19) (Waknaghat, India, 2019)

[22]. R.B. Pachori, P. Sircar, A new technique to reduce cross terms in the Wigner distribution. Digital Signal Process. 17, 466–474 (2007)

[23]. N.A. Khan, I.A. Taj, M.N. Jaffri, S. Ijaz, Cross-term elimination in Wigner distribution based on 2D signal processing techniques. Signal Process. 91, 590–599 (2011)

[24]. T.A.C.M. Claasen,W.F.G. Mecklenbrauker, TheWigner distribution- A tool for time-frequency signal analysis, Part I: continuous-time signals. Philips J. Res. 35(3), 217–250 (1980)

[25]. R.B. Pachori, P. Sircar, Analysis of multicomponent nonstationary signals using Fourier-Bessel

transform and Wigner distribution, in 14th European Signal Processing Conference (2006)

[26]. R.B. Pachori, P. Sircar, Time-frequency analysis using time-order representation and Wigner

distribution, in IEEE Tencon Conference, Article no. 4766782 (2008)

[27] Solovev, K., & Pröllochs, N. (2022). Hate Speech in the Political Discourse on Social Media: Disparities Across Parties, Gender, and Ethnicity. arXiv preprint arXiv:2201.06638.

[28] Chakraborty, T., & Masud, S. (2022). Nipping in the Bud: Detection, Diffusion and Mitigation of Hate Speech on Social Media. arXiv preprint arXiv:2201.00961.

[29]. A. Upadhyay,M. Sharma, R.B. Pachori, Determination of instantaneous fundamental frequency of speech signals using variational mode decomposition. Comput. Electr. Eng. 62, 630–647(2017)

[30]. A. Upadhyay, R.B. Pachori, Instantaneous voiced/non-voiced detection in speech signals based on variational mode decomposition. J. Frankl. Inst. 352(7), 2679–2707 (2015)

Google Drive link:

https://drive.google.com/drive/folders/15szEbMtm2anGz3wAr6RpT6hedJ301i2A?usp=share_link