

Hybrid Deep Learning model for Hate Speech Detection in Tweets (BERT + CNN + LSTM)

Submitted by,

Name	Roll No.	Registration No.
Diya Neogi	11700120131	201170100120004
Ashif Mondal	11700119010	038895
Arpon Roy	11700119024	031262
Bidisha Saha	11700119021	038523

UNDER THE SUPERVISION OF

Mrs. Monika Singh

Assistant Professor of Computer Science & Engineering,
RCC Institute of Information Technology

PROJECT REPORT SUBMITTED IN FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING
RCC INSTITUTE OF INFORMATION TECHNOLOGY



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
RCC INSTITUTE OF INFORMATION TECHNOLOGY

[Affiliated to Maulana Abul Kalam Azad University of Technology]

CANALSOUTH ROAD, BELIAGHATA,

KOLKATA-700015

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
RCC INSTITUTE OF INFORMATION TECHNOLOGY



TO WHOM IT MAY CONCERN

I hereby recommend that the Project entitled “**Hybrid Deep Learning model for Hate Speech Detection in Tweets (BERT + CNN + LSTM)**” prepared under my supervision by Diya Neogi (Roll No.: 11700120131), Ashif Mondal (Roll No.: 11700119010), Arpon Roy (Roll No.: 11700119024), Bidisha Saha (Roll No.: 11700119021), of B. Tech (8th Semester), may be accepted in fulfillment for the degree of **Bachelor Of Technology in Computer Science & Engineering** under Maulana Abul Kalam Azad University of Technology (MAKAUT).

.....

Project Supervisor

Department of Computer Science and Engineering

RCC Institute of Information Technology

Countersigned:

.....

Head

Department of Computer Sc. & Engg.,

RCC Institute of Information Technology

Kolkata – 700015



CERTIFICATE OF APPROVAL

The foregoing Project is hereby accepted as a credible study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the project only for the purpose for which it is submitted.

Name of the Examiner

Signature with Date

1.

.....

2.

.....

3.

.....

4.

.....

ACKNOWLEDGEMENT

We acknowledge our overwhelming gratitude & immense respect to our revered guide, **Mrs. Monika Singh** (Assistant Professor of Computer Science & Engineering, **RCC Institute of Information Technology**) under whose scholarly guideline, constant encouragement & untiring patience; we have proud privilege to accomplish this entire project work. We feel enriched with the knowledge & sense of responsible approach we inherited from our guide & shall remain a treasure in our life.

Date:

Diya Neogi
Reg. No.: 201170100120004 OF 2020-21
Roll No.: 11700120131

Date:

Ashif Mondal
Reg. No.: 038895 OF 2019-20
Roll No.: 11700119010

Date:

Arpon Roy
Reg. No.: 031262 OF 2019-20
Roll No.: 11700119024

Date:

Bidisha Saha
Reg. No.: 038523 OF 2019-20
Roll No.: 11700119021

ABSTRACT

With the massive availability of social networks to everyone and with the increase in social interactions on online social networks, we can observe a massive number of hateful activities in social networks. This scenario has influenced the researchers to deal with this problem programmatically as manual detection of such activities is not scalable. Now a days deep Learning based methods are used for such hate speech detection on Twitter, this task is critical for applications like controversial event extraction, building AI chatterbots, content recommendation, and sentiment analysis. In our work we would like to develop an AI chatterbox which will help us differentiate between messages that express explicit hate and messages that are just offensive. This report presents a study on hate speech detection in Twitter using a hybrid model of BERT (Bidirectional Encoder Representations from Transformers), CNN (Convolutional Neural Networks), and LSTM (Long Short-Term Memory). The findings of this study contribute to the development of effective techniques for hate speech detection in social media platforms, particularly on Twitter. The hybrid architecture combining BERT, CNN, and LSTM models shows promising results and can be further enhanced and applied to real-world scenarios to mitigate the harmful effects of hate speech and promote a safer online environment.

CONTENTS

Page No

CERTIFICATE OF APPROVAL	I
ACKNOWLEDGEMENT	II
ABSTRACT	III
CONTENTS	IV
LIST OF ABBREVIATIONS	1
LIST OF FIGURES	2
LIST OF TABLES	3

CHAPTER-1

1.1 INTRODUCTION.....	4
1.2 LITERATURE REVIEW.....	6

CHAPTER-2

2.1 WORKFLOW.....	9
2.2 METHODOLOGIES OF IMPLEMENTATION.....	9
2.3 SOFTWARE & HARDWARE REQUIREMENTS.....	12

CHAPTER-3

3.1 TEST CASES & SYSTEM VALIDATION.....	13
3.2 OBSERVED OUTPUT.....	13
3.3 PERFORMANCE ANALYSIS	15

CHAPTER-4

4.1 CHALLENGES.....	17
4.2 CONCLUSION.....	17
4.3 FUTURE SCOPE.....	17

REFERENCES.....	18
-----------------	----

LIST OF ABBREVIATIONS

Abbreviation	Full Form / Meaning
DNN	Deep Neural Network
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory
Bi-LSTM	Bidirectional Long Short-Term Memory
BERT	Bidirectional Encoder Representations from Transformers
GRU	Gated Recurrent Unit
TF-IDF	Term Frequency - Inverse Document Frequency
BoWV	Bag of Word Vectors
GloVe	Global Vectors for word representation

LIST OF FIGURES

Fig. No.	Description	Page No.
Figure 1	Percent that agree “People should be able to make statements that are offensive to minority groups publicly” (2015)	4
Figure 2	Amount of hate speech used in Twitter between October 22 and October 28, 2022	5
Figure 3	Workflow diagram of model	9
Figure 4	Group-by view of train validation and test data of each category	10
Figure 5	A schematic depiction of the BERT model	11
Figure 6	Model layers	16
Figure 7	Accuracy and other parameters	16
Figure 8	Testing model with hate speech	19
Figure 9	Testing model with offensive speech	19
Figure 10	Testing model with normal speech	19
Figure 11	Result of all testing	19
Figure 12	Graph of training accuracy and validation accuracy vs no of epochs	20
Figure 13	Confusion matrix	20

LIST OF TABLES

Table. No.	Description	Page No.
Table 1	Literature survey	8
Table 2	Comparison of accuracy with our different models	21

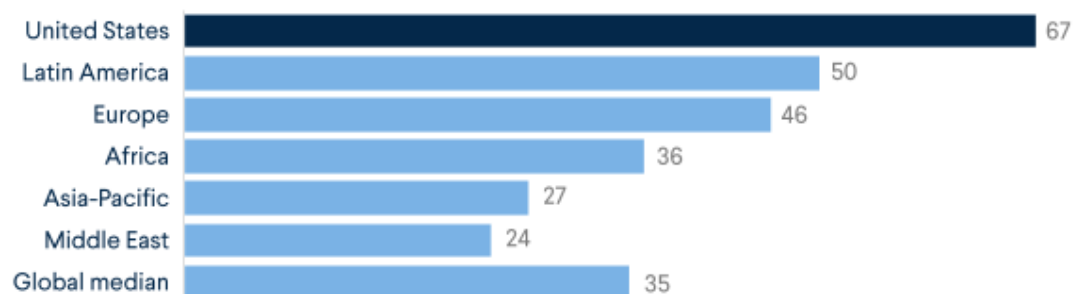
CHAPTER-1

1.1 Introduction

Hate crimes are unfortunately nothing new in society. Hate speech ^[1] is one of the serious issues we see on social media platforms like Facebook and Twitter, mostly from people with political views. Now a days the usage of hate speech in social media is increasing massively. On Twitter hate speech are those which contains violence or abuse speech towards a person or group based on religion, gender, organization, country etc. Social media platforms such as twitter need to detect hate speech and prevent it from going viral or ban it at the right time.

Social media has experienced incredible growth over the last decade, both in its scale and importance as a form of communication. Vast online communication forums, including social media, enable users to express themselves freely, at times, anonymously. Depending on the forum, such posts can be visible to many millions of people.

Percent that agree “People should be able to make statements that are offensive to minority groups publicly” (2015)



Note: Displays the median among countries included in the survey.

Figure 1: Percent that agree “People should be able to make statements that are offensive to minority groups publicly” (2015)

Source: Council on Foreign Relations ^[2]

Automated methods for detecting hate-speech are necessary in some circumstances. By automating its detection, the spread of hateful content can be reduced.

According to New research from the Joetta Di Bella and Fred C. Sautter III Center for Strategic Communication at Montclair State University shows that in the hours following Elon Musk’s acquisition of Twitter, the use of hate speech terms

increased immediately on the social media platform.

Looking at Twitter data between October 22 and October 28, using the Tweetbinder analytics program, the study examined a range of vulgar and hostile terms for individuals based on race, religion, ethnicity and orientation. The seven-day average of Tweets using the studied hate terms prior to Musk’s acquisition was never higher than 84 times per hour. However, on October 28 from midnight to noon (immediately following Musk’s acquisition), the studied hate speech was Tweeted some 4,778 times.

Elon Musk has promised to reduce restrictions on the platform and “free the bird.” From these results, this directive represents an obvious danger to young people using the platform.

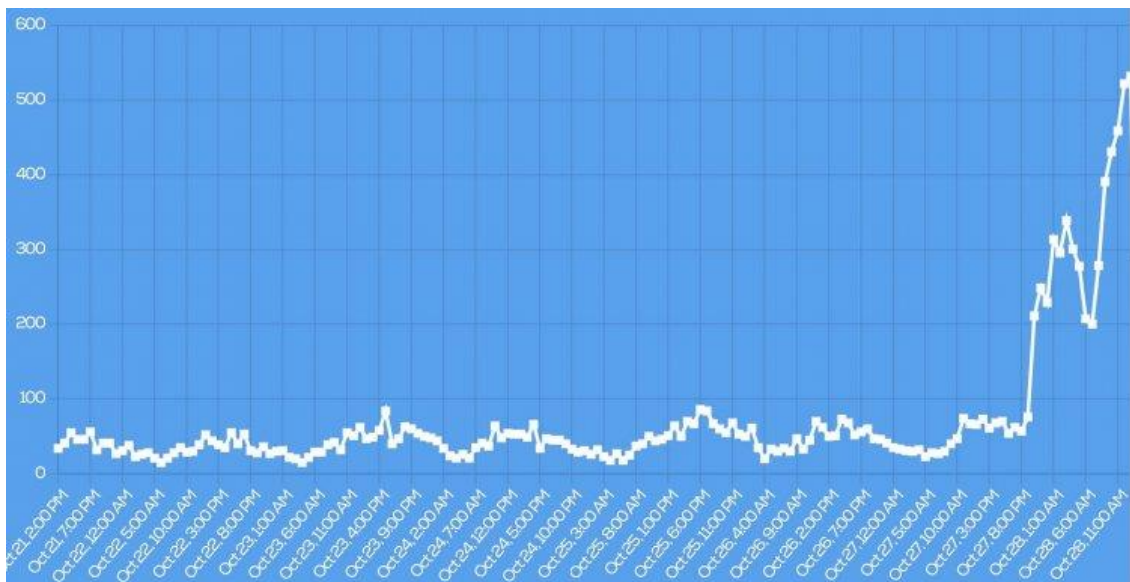


Figure 2: Amount of hate speech used in Twitter between October 22 and October 28, 2022

Source: Montclair State University ^[3]

In this project we classified the tweets as hate, offensive or neither. This project is quite difficult because the tweets can have different form of hatred, different target but representing the same meaning. In such kind of topics deep learning methods has the most accuracy over large number of complex problems. In this project we used deep learning architectures such as BERT ^[4], Convolutional Neural Network (CNN) ^[5], Long Short-term Memory Network (LSTM) ^[6].

1.2 Literature Review

After going through some research paper on this topic we got basic ideas of the word classifiers such as Char n grams, TF-IDF, BoWV and the deep learning algorithm such as Convolutional Neural Network (CNN) ^[5], Gated Recurrent Unit (GRU), FastText, Long Short-Term Memory Network (LSTM), Bi-Directional Long Short-Term Memory Networks (Bi-LSTM) ^[7] etc.

A text classifier labels unstructured texts into predefined text categories. Instead of users having to review and analyse vast amounts of information to understand the context, text classification helps derive relevant insight.

The word classifiers used here are char n grams, TF-IDF, BoWV.

Compared to word n-grams, char n grams only capture the identity of a word and its possible neighbours, character n-grams are additionally capable of detecting the morphological makeup of a word. TF-IDF transforms text into a meaningful representation of numbers which is used to fit machine algorithm for prediction. BoWV uses the average of the word (GloVe) embeddings to represent a sentence. We experimented with multiple classifiers for both the TF-IDF and the BoWV approaches.

Neural Networks replicate the way humans learn, inspired by how the neurons in our brains fire, only much simpler. The most common Neural Networks consist of three network layers: (i) An input layer, (ii) A hidden layer (this is the most important layer where feature extraction takes place, and adjustments are made to train faster and function better), and (iii) An output layer. Each sheet contains neurons called “nodes,” performing various operations.

Convolutional neural network (CNN) ^[5] is a feed-forward neural network that is generally used to analyze text by processing data with grid-like topology. Convolutional Neural Network provides accurate result on text classification such as sentiment analysis. A convolution neural network has multiple hidden layers that help in extracting information from a text. FastText is also a feed-forward neural network but it uses backpropagation to update the word vectors. Backpropagation sends error information from the network's last layer to all the weights within the network. Basically, it fine tunes the weights of a neural network based on the previous epoch's error rate. It improves the model by

reducing the error. Gated Recurrent Unit (GRU) is a recurrent neural network. It has a memory unit and two gates such as update gate and reset gate. Update gate determines how much of the past knowledge needs to be passed along into the future. Reset gate determines how much of the past knowledge to forget. Like recurrent neural network, LSTM uses internal memory to process the sequence of inputs. Recurrent Neural Networks were created to solve the sequential input data time-series problem. RNN has an internal memory that stores the information from previous samples' computations. The goal of AI in this case is to create a system that can understand human spoken natural languages, such as natural language modeling, word embedding, and machine translation. LSTM has a forget gate, input gate and output gate. Input gate determines the extent of information be written onto the Internal Cell State. Output gate determines what output to generate from the current Internal Cell State. And LSTM can save the data in its memory block until the forget gate triggers. Unlike standard LSTM, the input flows in both directions, and it's capable of utilizing information from both sides which makes a Bi-LSTM ^[7] different from the regular LSTM. The main reason is that every component of an input sequence has information from both the past and present. For this reason, Bi-LSTM ^[7] can produce a more meaningful output, combining LSTM layers from both directions.

By Ji Ho et al's paper over a dataset by Waseem and Hovy dataset, 2016 ^[12] Hybrid CNN Classifier (wordCNN + charCNN) gives F1 score of 0.827. On the paper by Zhang et al, 2018 ^[14] we can see over 24k dataset CNN + GRU gives 0.919 F1 score. By Ziqi Zhang, David Robinson, Jonathan Tepper ^[13] CNN + LSTM, emb-ggl2 gives 0.919 F1 score on 24k dataset. According to the research paper "Deep Learning for Hate Speech Detection in Tweets" published on 2020 by Pinkesh Badjatiya, Shashank Gupta, Manish Gupta and Vasudeva Varma ^[9] ^[10] we came to know that over a dataset of 16k tweets CNN + Random Embedding + GBDT gives F1 score of 86.4%, FastText + Random Embedding + GBDT gives F1 score of 88.6%, LSTM + Random Embedding + GBDT gives F1 score of 93%. On the basis of another research paper "Detecting Hate Speech using Deep Learning Techniques" published on 2021 by Chayan Paul and Pronami Bora ^[8] ^[11] we saw that on a dataset of 16k data LSTM gives F1 score of 97.85% where Bi-LSTM gives F1 score of 97.81%.

Reference	Dataset Size	Method	Result			
			Accuracy	Precision	Recall	F1
Ji Ho et al, Waseem and Hovy dataset, 2016	25k	Hybrid CNN	0.827	0.827	0.827	0.827
Ziqi Zhang, David Robinson, Jonathan Tepper, 2019	24k	CNN + LSTM, emb-ggl2	0.919	0.919	0.919	0.919
Zhang et al, 2018	24k	CNN + GRU	0.94	0.94	0.94	0.94
“Deep Learning for Hate Speech Detection in Tweets” published on 2020 by Pinkesh Badjatiya, Shashank Gupta, Manish Gupta and Vasudeva Varma	16k	CNN + Random Embedding + GBDT	0.864	0.864	0.864	0.864
		FastText + Random Embedding + GBDT	0.886	0.886	0.887	0.886
		LSTM + Random Embedding + GBDT	0.930	0.930	0.930	0.930
“Detecting Hate Speech using Deep Learning Techniques” published on 2021 by Chayan Paul and Pronami Bora	16k	LSTM	0.9785	0.9598	0.9986	0.9785
		Bi-LSTM	0.9781	0.9582	0.9990	0.9781

Table 1: Literature survey

CHAPTER 2

2.1 WORKFLOW

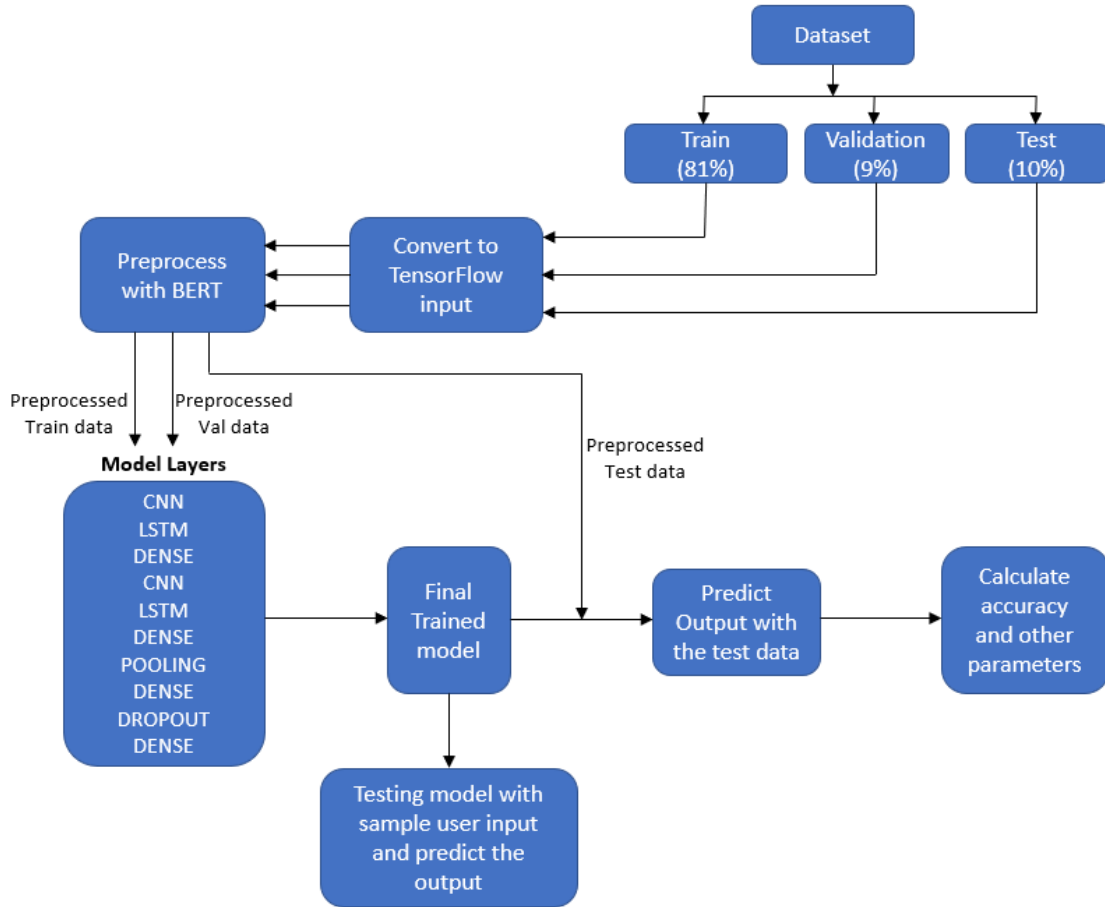


Figure 3: Workflow diagram of model

2.2 METHODOLOGIES OF IMPLEMENTATION

Preparing datasets:

We used a merged dataset for our model.

- (i) Hate speech offensive tweets by Davidson et al. ^[15] The data is compiled using a hate speech lexicon and all the instances are from Twitter. A minimum of 3 coders labelled tweets into classes Hate speech, Offensive and Neither. The final sample consisted of 24794 examples.
- (ii) Hate and not hate by Basile et al., 2019. ^[16] It has 12970 hate and non-hate speech from Twitter.

- (iii) Offensive and not offensive by Zampieri et al., 2019. ^[17] It has 14100 offensive and normal speech from Twitter.

We merged these three datasets for our model. It has 34896 text speeches, and divided into three categories:

Label 0: hate speech

Label 1: offensive speech

Label 2: neither hate nor offensive

We used only text column and label column for our model.

Creating train validation and test sets:

We divided our dataset in such a way that 81% data belongs to train data, 9% data belongs to validation data and 10% data belongs to test data. After that the datasets are converted into tensor batch dataset.

			text
category	label	data_type	
hate_speech	0	test	690
		train	5591
		val	621
neither	2	test	416
		train	3372
		val	375
offensive_language	1	test	2384
		train	19302
		val	2145

Figure 4: Group-by view of train validation and test data of each category

Build TensorFlow input:

The tf.data API enables you to build complex input pipelines from simple, reusable pieces. For example, the pipeline for an image model might aggregate data from files in a distributed file system, apply random perturbations to each image, and merge randomly selected images into a batch for training. The pipeline for a text model might involve extracting symbols from raw text data, converting them to embedding identifiers with a lookup table, and batching together sequences of different lengths. The tf.data API makes it possible to handle large amounts of data, read from different data formats, and perform complex transformations.

The tf.data API introduces a tf.data.Dataset abstraction that represents a sequence of elements, in which each element consists of one or more components. For example, in an image pipeline, an element might be a single training example, with a pair of tensor components representing the image and its label.

There are two distinct ways to create a dataset: A data source constructs a Dataset from data stored in memory or in one or more files. A data transformation

constructs a dataset from one or more `tf.data.Dataset` objects.

Pre-processing with BERT:

Bidirectional Encoder Representations from Transformers (BERT) is a popular deep learning model that is used for numerous different languages understanding tasks. BERT shares the same architecture as a transformer encoder, and is extensively pre-trained on raw, unlabeled textual data using a self-supervised learning objective, before being fine-tuned to solve downstream tasks (e.g., question answering, sentence classification, named entity recognition, etc.). At the time of its proposal, BERT obtained a new state-of-the-art on eleven different language understanding tasks, prompting a nearly-instant rise to fame that has lasted ever since.

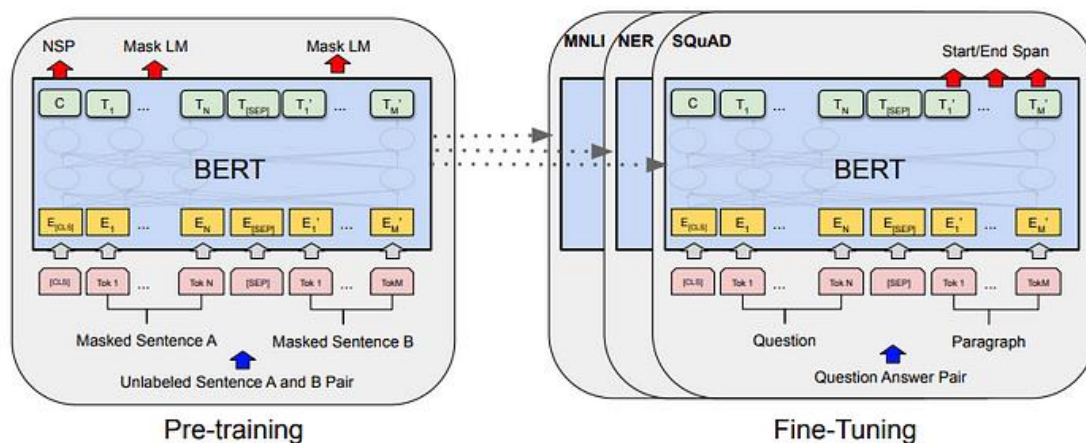


Figure 5: A schematic depiction of the BERT model

We've chosen "bert_en_uncased_L-12_H-768_A-12" for pre-process our data. This belongs to small-bert package. It uses L=12 hidden layers (i.e., Transformer blocks), a hidden size of H=768, and A=12 attention heads. Text inputs need to be transformed to numeric token ids and arranged in several Tensors before being input to BERT. TensorFlow Hub provides a matching pre-processing model for each of the BERT models, which implements this transformation using TF ops from the `TF.text` library. Hence, it is not necessary to run pure Python code outside the TensorFlow model to pre-process text.

Building model layers, training model:

We tested BERT + CNN, BERT + LSTM, BERT + Bi-LSTM and BoWV +

LSTM in our previous work. But nowadays it's a trend that hybrid models are one step ahead than non-hybrid models in terms of performance and accuracy. So, we decided to make a hybrid model which performs better than those previous models. We've chosen a hybrid approach of BERT + CNN + LSTM to build our models architecture. In this report we proposed to classify the tweets using a hybrid model of BERT, Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM).

A. CNN (Convolutional Neural Network): It is a category of Neural Networks that have proven very effective in areas such as image recognition and classification. Because of these successes, many researchers try to apply them to other problems, like NLP. A CNN's architecture is analogous to the connectivity pattern of the human brain. Just like the brain consists of billions of neurons, CNNs also have neurons arranged in a specific way. In fact, a CNN's neurons are arranged like the brain's frontal lobe, the area responsible for processing visual stimuli. This arrangement ensures that the entire visual field is covered, thus avoiding the piecemeal image processing problem of traditional neural networks, which must be fed images in reduced-resolution pieces. Compared to the older networks, a CNN delivers better performance with image inputs, and with speech or audio signal inputs. Most computations happen in the convolutional layer, which is the core building block of a CNN. A second convolutional layer can follow the initial convolutional layer. The process of convolution involves a kernel or filter inside this layer moving across the receptive fields of the image or text, checking if a feature is present in the image or text. Like the convolutional layer, the pooling layer also sweeps a kernel or filter across the input image or text. But unlike the convolutional layer, the pooling layer reduces the number of parameters in the input and results in some information loss. On the positive side, this layer reduces complexity and improves the efficiency of the CNN. The Fully connected layer is where image classification happens in the CNN based on the features extracted in the previous layers. Here, fully connected means that all the inputs or nodes from one layer are connected to every activation unit or node of the next layer. To use a CNN for hate speech detection in Twitter, we need to preprocess the text data and convert it into a suitable format for the CNN model. We already did the preprocessing earlier, now the sequence outputs of BERT preprocessed data are sent to the CNN layer.

B. Long Short-Term Memories (LSTM): LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behaviour, not something they struggle to learn! It is a type of recurrent neural network (RNN) architecture that is widely used in natural language processing (NLP) tasks, including hate speech detection. Unlike traditional RNNs, LSTM networks are designed to overcome the vanishing gradient problem, which occurs when gradients diminish exponentially as they propagate through time steps in a sequence. LSTMs are well-suited for processing sequential data and capturing long-term dependencies. They achieve this by introducing a memory cell, which allows the network to selectively retain or forget information over time. The memory cell is composed of three main components: an input gate, a forget gate, and an output gate. The input gate determines how much new information should be stored in the memory cell based on the current input and the previous hidden state. The forget gate controls the extent to which the previous memory cell should be forgotten. The output gate determines the amount of information to be outputted from the memory cell to the next time step. The ability of LSTM to retain and selectively update information in the memory cell makes it effective in capturing long-range dependencies in sequences. This is particularly useful in NLP tasks where the meaning of a word or phrase can depend on words that appear far away in the text. In hate speech detection, LSTM models can be trained on labelled data to learn patterns and linguistic cues associated with hate speech. The model takes in a sequence of words or tokens as input, and the LSTM layers process the sequence, updating the hidden state and memory cell at each time step. The final hidden state is then passed through a fully connected layer followed by a SoftMax activation function to obtain the predicted hate speech label. LSTMs have been widely used in NLP tasks due to their ability to capture long-term dependencies, handle sequential data, and effectively model linguistic patterns. However, they may struggle with capturing certain contextual nuances and complex interactions within the text. This is where hybrid models, such as combining LSTM with other architectures like CNN or BERT, can provide enhanced performance by leveraging the strengths of different models and capturing both local and global information in the text.

C. Dense layer: A dense layer, also known as a fully connected layer, is a fundamental component in neural network architectures. It is called "dense" because each neuron in the dense layer is connected to every neuron in the

previous layer. The dense layer transforms the input data by performing a linear operation followed by a non-linear activation function. In a dense layer, each neuron is connected to every neuron in the previous layer. The output of a dense layer is computed by taking a weighted sum of the inputs, where each input is multiplied by its corresponding weight. The weighted sum is then passed through an activation function to introduce non-linearity into the network.

D. Pooling layer: In deep learning, a pooling layer is a type of layer that is often used after convolutional layers in convolutional neural networks (CNNs). The pooling layer helps to reduce the spatial dimensions of the input while retaining the most important features. The pooling operation replaces a group of pixels or activations in the input with a summary statistic, such as the maximum value (max pooling) or the average value (average pooling). By reducing the spatial dimensions of the input, the pooling layer helps to decrease the computational complexity of the network and reduce the number of parameters. It enables the network to focus on the most relevant features while discarding less informative details. Pooling helps to create translation invariance, which means that the network can recognize patterns regardless of their position in the input. This is particularly useful in tasks such as image classification, where the location of a specific feature may vary. The pooling operation extracts the most salient features from the input. By summarizing a group of activations into a single value, the pooling layer helps to capture the dominant features and discard irrelevant variations within the local neighbourhood. The pooling layer in deep learning plays a critical role in reducing the spatial dimensions of the input, extracting important features, and enabling translation invariance. It helps to make deep learning models more computationally efficient, robust, and capable of capturing meaningful patterns in the data.

E. Dropout Layer: Dropout is a regularization technique commonly used in deep learning models to prevent overfitting. Overfitting occurs when a model becomes too complex and starts to memorize the training data instead of learning general patterns that can be applied to new, unseen data. Dropout helps to mitigate overfitting by introducing randomness and reducing the interdependencies between the neurons in a neural network. During training, dropout randomly sets a fraction of the input units (neurons) to zero at each update, effectively "dropping out" those units from the network. This means that the dropped-out units do not contribute to the forward pass and backpropagation during a particular training

iteration. The fraction of units to be dropped out is a hyperparameter that needs to be specified. By randomly dropping out units, dropout prevents neurons from relying too much on the presence of specific input features or co-adapting with other neurons. This encourages the network to learn more robust and generalizable representations. Dropout acts as a form of ensemble learning, where multiple subnetworks are trained simultaneously, each with a different subset of units active. At test time, when making predictions on new data, dropout is typically turned off or modified to use all units. The weights of the network are scaled to reflect the expected contribution of each unit during training. This scaling is done to ensure that the expected output of the network remains the same during training and testing.

We used mainly tensorflow and keras to build our model input and layers. TensorFlow offers multiple levels of abstraction so we can choose the right one for our needs. Building and training models by using the high-level Keras API makes deep learning and machine learning easy.

Our hybrid model consists of CNN, LSTM, dense, pooling and dropout layer. Output shape of these layers are optimized so that it is compatible with model's next layer.

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
text (InputLayer)	[(None,)]	0	[]
preprocessing (KerasLayer)	{'input_word_ids': (None, 128), 'input_mask': (None, 128), 'input_type_ids': (None, 128)}	0	['text[0][0]']
BERT_encoder (KerasLayer)	{'pooled_output': (None, 512), 'sequence_output': (None, 128, 512), 'default': (None, 512), 'encoder_outputs': [(None, 128, 512), (None, 128, 512), (None, 128, 512), (None, 128, 512)]}	28763649	['preprocessing[0][0]', 'preprocessing[0][1]', 'preprocessing[0][2]']
conv1d (Conv1D)	(None, 127, 32)	32800	['BERT_encoder[0][6]']
lstm (LSTM)	(None, 127, 64)	24832	['conv1d[0][0]']
dense (Dense)	(None, 127, 64)	4160	['lstm[0][0]']
conv1d_1 (Conv1D)	(None, 126, 128)	16512	['dense[0][0]']
lstm_1 (LSTM)	(None, 126, 256)	394240	['conv1d_1[0][0]']
dense_1 (Dense)	(None, 126, 256)	65792	['lstm_1[0][0]']
global_max_pooling1d (GlobalMaxPooling1D)	(None, 256)	0	['dense_1[0][0]']
dense_2 (Dense)	(None, 512)	131584	['global_max_pooling1d[0][0]']
dropout (Dropout)	(None, 512)	0	['dense_2[0][0]']
classifier (Dense)	(None, 3)	1539	['dropout[0][0]']
Total params: 29,435,108 Trainable params: 29,435,107 Non-trainable params: 1			

Figure 6: Model layers

We trained our model with 80 epochs and each epoch has 884 steps. Each epoch is trained with train and validation data. Finally, the model is tested with the test data and accuracy, loss and other parameters are determined. Our model has given 90.54% accuracy over the test data.

Accuracy: 0.9054441452026367 F1 Score: 0.8904957175254822 Precision: 0.8048020601272583 Recall: 1.0007003545761108

Figure 7: Accuracy and other parameters

2.3 SOFTWARE & HARDWARE REQUIREMENTS

Software Requirements:

- Google Colab
- Microsoft Edge / Google Chrome
- Google Drive
- Microsoft Excel

Hardware Requirements:

Local Machine:

- Windows 10 / Windows 11

Google Colab environment:

- System RAM (16GB)
- NVIDIA T4 Tensor Core GPUs (12GB)
- SSD (120GB)

CHAPTER 3

3.1 TEST CASES & SYSTEM VALIDATION

The dataset used has 35K Tweets along with Sentiment Labels. The dataset has two columns, the first column has the tweets and the second column indicates their Sentimental Label.

The model returns array of three elements. If the maximum of the array present in index 0 then it'll be detected as hate speech, if it is in index 1 then it'll be offensive speech and if it is in position 2 then it is neutral.

Language used:

- Python

Packages used:

- Tensorflow ^[18]
- Keras ^[19]
- Sklearn ^[20]
- Pandas ^[21]
- Numpy ^[22]
- Matplotlib ^[23]
- Streamlit ^[24]
- Localtunnel ^[25]

Layers Used:

- Pre-processing layer of BERT (Keras Layer)
- CNN (Convolutional Neural Network) Layer
- LSTM Layer
- Dense Layer ^[26]
- Pooling Layer ^[27]
- Dropout Layer ^[28]

3.2 OBSERVED OUTPUT

Testing model with sample user input:

We've created a webapp for testing the user inputs and predicting the results. Webapp is created with the help of streamlit and converted to a public link via localtunnel. After each execution the user inputs and predicted result is stored in a 'result.csv' file.

Tweet Detector

Please enter a tweet:

Dammit, this country was so great before these dirty immigrants showed up.

Check

Hate speech

Figure 8: Testing model with hate speech

Tweet Detector

Please enter a tweet:

As a loser you shouldn't be here.

Check

Offensive speech

Figure 9: Testing model with offensive speech

Tweet Detector


Please enter a tweet:

This is such a beautiful piece of art.

Check

Neutral

Figure 10: Testing model with normal speech

1 to 3 of 3 entries Filter 

text	type
Dammit, this country was so great before these dirty immigrants showed up.	Hate speech
As a loser you shouldn't be here.	Offensive
This is such a beautiful piece of art.	Neutral

Show 10 per page

Figure 11: Result of all testing

3.3 PERFORMANCE ANALYSIS

Model Performance:

To determine the model's performance training loss, validation loss and training accuracy and validation accuracy is plotted against each epoch using matplotlib and confusion matrix is also determined.

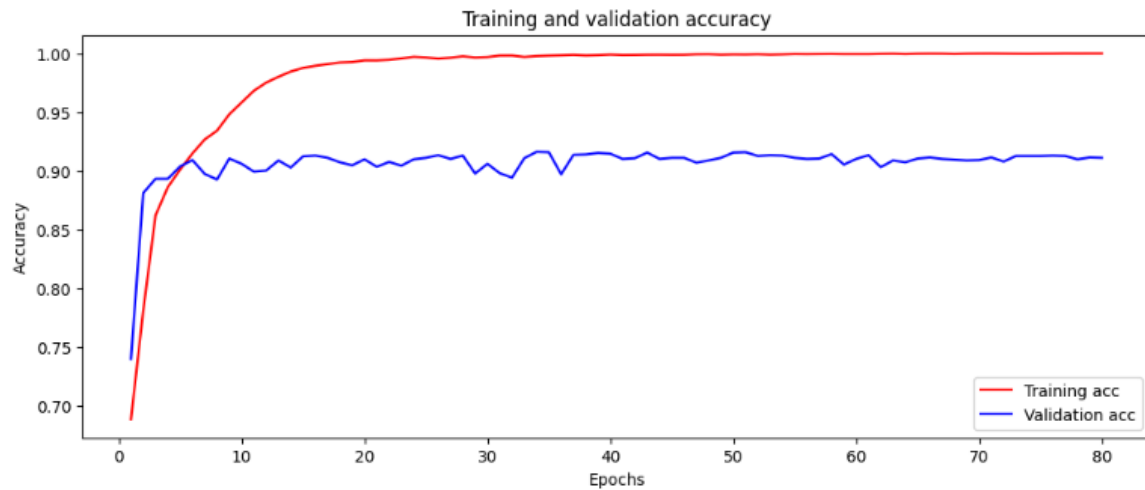


Figure 12: Graph of training accuracy and validation accuracy vs no of epochs

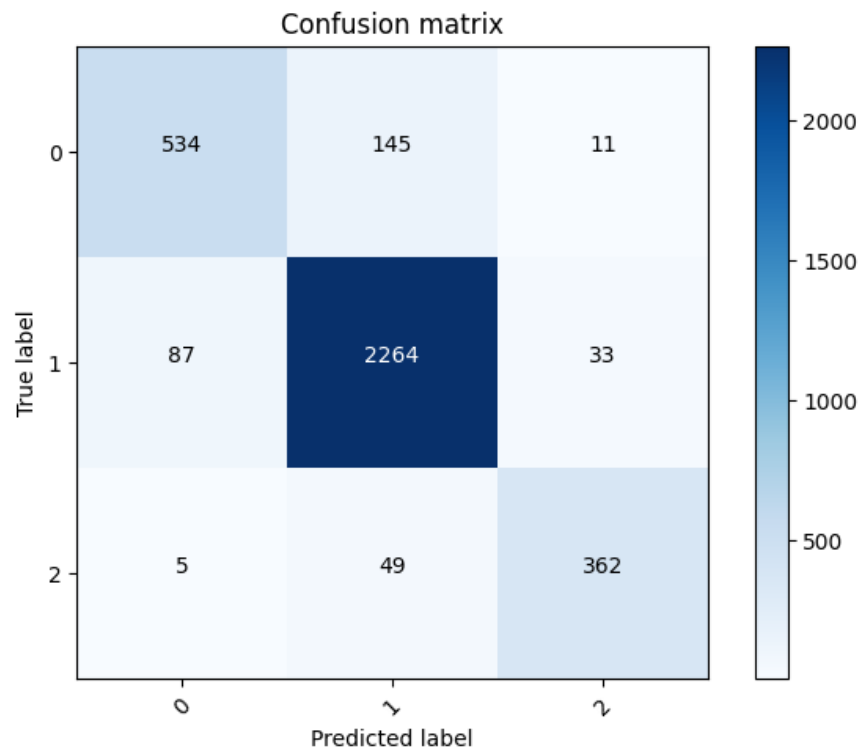


Figure 13: Confusion matrix

Comparative analysis of BERT + CNN + LSTM for hate speech detection:

Model	Accuracy
BoWV + LSTM	0.7981
BERT LSTM	0.9032
BERT CNN	0.9027
BERT Bi-LSTM	0.8921
BERT CNN BiLSTM	0.8985
BERT CNN LSTM	0.9054

Table 2: Comparison of accuracy with our different models

It appears that the BERT-based models outperform the BoWV + LSTM model in terms of accuracy. Both BERT + LSTM, BERT + CNN and BERT + CNN + LSTM achieve higher accuracies than the BoWV + LSTM model, with accuracies ranging from 0.9027 to 0.9054. The BERT + CNN model also performs well, with an accuracy of 0.9027.

The addition of convolutional layers (CNN) to the BERT-based models, such as BERT + CNN + Bi-LSTM and BERT + Bi-LSTM, does not seem to significantly improve accuracy compared to the BERT + CNN + LSTM model. However, the accuracies remain relatively high, ranging from 0.8921 to 0.8985.

Overall, the BERT-based models, particularly those involving BERT + LSTM + CNN, demonstrate better performance in hate speech detection compared other models. BERT's ability to capture contextual information and semantic relationships in text contributes to its effectiveness in this task.

CHAPTER 4

4.1 CHALLENGES

During this project we faced a lot of issues. Initially we tried using Jupyter Notebook but there is library importing error such as TensorFlow official library, tensorflow-text module and adamw optimizer. In Google Colab there is a limited GPU quota for non cloab pro and our GPU quota finished a several times before even finish all the epochs. Also, we faced some errors during the creation of model function and adjusting the model layers. The shape of the model layers should be maintained so that it is compatible with model's next layer.

4.2 CONCLUSION

We introduced a method for automatically classifying hate speech on twitter using hybrid approach of deep neural networks (DNN) such as LSTM, CNN, BERT and we investigated the application of deep neural network architectures for the task of hate speech detection. According to the result of accuracy, precision, recall and F1 score BERT-CNN-LSTM hybrid model, BERT-CNN and BERT-LSTM model performs better than other models. However, the models give almost the same score. The score is very similar to draw comparison between these models. Utilizing a hybrid model for hate speech detection in Twitter can offer several advantages. By combining different models such as BERT, BiLSTM, and CNN, we can leverage their respective strengths and improve the overall performance of the hate speech detection system. The hybrid model can capture a wide range of features, including local context, semantic relationships, and patterns, which are essential for effectively identifying hate speech in tweets. The ensemble learning techniques employed in the hybrid model can enhance the accuracy and robustness of the predictions. By aggregating the outputs of multiple models through methods like majority voting or weighted averaging, the hybrid model can mitigate biases and errors, leading to more reliable hate speech detection. Furthermore, the hybrid model is adaptable and flexible, allowing for the incorporation of new models or techniques as they become available. This adaptability enables the system to continuously improve its performance over time as new insights and advancements in hate speech detection emerge. Overall, the hybrid model approach provides a powerful and comprehensive solution for

hate speech detection in Twitter, addressing the limitations of individual models and offering a more accurate and nuanced understanding of hate speech in social media context.

4.3 FUTURE SCOPE

This study can be further extended for other social media platforms such as Facebook, WhatsApp, Snapchat, Line etc. and it's interesting to see how the model performs over the new datasets. It will be interesting to see how these models perform on new data set. We also created webapp and stored the result in databases. In future it can be used on production basis. Same approaches can be applied for multilingual languages such as Hindi, Bengali, German, Spanish, Arabic, Chinese or Japanese. Also, we can add some algorithm for the emoji, picture, GIF, stickers which can properly identify the emoji and convert into text such as happiness, sorrow, ignorance etc. or analyze the meaning of picture, GIF or stickers if it is used in good motives or making fun of a person or some community; another case is that if someone uses hate word as a meme or joke, the model can correctly identify if it's a joke or hate word.

REFERENCES

1. [Deep Learning for Hate Speech Detection: A Large-scale Empirical Evaluation | by Guansong Pang | Towards Data Science](#)
2. [Hate Speech on Social Media: Global Comparisons | Council on Foreign Relations \(cfr.org\)](#)
3. [Study Finds Hate Speech On Twitter Increased Quickly After Elon Musk Takeover – Press Room - Montclair State University](#)
4. [BERT Explained: State of the art language model for NLP | by Rani Horev | Towards Data Science](#)
5. [A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | by Sumit Saha | Towards Data Science](#)
6. [An Overview on Long Short Term Memory \(LSTM\) - Analytics Vidhya](#)
7. [Bi-LSTM. What is a neural network? Just like our... | by Raghav Aggarwal | Medium](#)
8. [Detecting Hate Speech using Deep Learning Techniques \(thesai.org\)](#)
9. [\(PDF\) Deep Learning for Hate Speech Detection in Tweets \(researchgate.net\)](#)
10. Pinkesh Badjatiya, Shashank Gupta, Manish Gupta and Vasudeva Varma, "Deep Learning

for Hate Speech Detection in Tweets”, 2020.

11. Chayan Paul and Pronami Bora, “Detecting Hate Speech using Deep Learning Techniques”, 2021.
12. Ji Ho et al, Waseem and Hovy, 2016.
13. Ziqi Zhang, David Robinson, Jonathan Tepper, “Hate Speech Detection: A Solved Problem? The Challenging Case of Long Tail on Twitter”, 2018.
14. Zhang et al, 2018
15. [hate-speech-and-offensive-language/abeled_data.csv at master · t-davidson/hate-speech-and-offensive-language · GitHub](#)
16. [tweeteval/datasets/hate at main · cardiffnlp/tweeteval · GitHub](#)
17. [tweeteval/datasets/offensive at main · cardiffnlp/tweeteval · GitHub](#)
18. [TensorFlow](#)
19. [Keras: Deep Learning for humans](#)
20. [scikit-learn: machine learning in Python — scikit-learn 1.2.2 documentation](#)
21. [pandas - Python Data Analysis Library \(pydata.org\)](#)
22. [NumPy](#)
23. [Matplotlib — Visualization with Python](#)
24. [Streamlit • A faster way to build and share data apps](#)
25. [Localtunnel ~ Expose yourself to the world \(theboroer.github.io\)](#)
26. [A Complete Understanding of Dense Layers in Neural Networks \(analyticsindiamag.com\)](#)
27. [Pooling Layer - an overview | ScienceDirect Topics](#)
28. [Dropout layer \(keras.io\)](#)