# DETECTION OF HATE SPEECH USING DEEP LEARNING

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY
IN
**COMPUTER SCIENCE AND ENGINEERING**

Submitted by

**SHIVAM SHARMA**
**2K21/CSE/20**

Under the Supervision Of

**PROF. SHAILENDER KUMAR**


**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

MAY 2023

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

## CANDIDATE'S DECLARATION

I, (Shivam Sharma), Roll No: 2K21/CSE/20 student of MTech (Computer Science & Engineering), hereby declare that the Project Dissertation titled "DETECTION OF HATE SPEECH USING DEEP LEARNING" which is submitted by me to the Department of Computer Science and Engineering, Delhi Technological University (formerly DCE), Delhi in partial fulfillment for the award of the degree Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship, or other similar title or recognition.

Place: Delhi
Date:

(Shivam Sharma)
2K21/CSE/20

**DEPARTMENT OF**
**COMPUTER SCIENCE & ENGINEERING**

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-11042

## CERTIFICATE

I hereby certify that the Project Dissertation titled "DETECTION OF HATE SPEECH USING DEEP LEARNING" which is submitted by Shivam Sharma, Roll No: 2K21/CSE/20. DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree Master of Technology, is a record of the project work carried out by the student under my supervision. To the best of my knowledge, this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi
Date:

**PROF. SHAILENDER KUMAR**
**SUPERVISOR**

## ACKNOWLEDGEMENT

I am very grateful to **PROF. SHAILENDER KUMAR** ( Department of Computer Science and Engineering) and all faculty members of the Department of Computer Science and Engineering at DELHI TECHNOLOGICAL UNIVERSITY for their huge support and assistance in the completion of the project.

I would also like to acknowledge the much crucial role of the university in contributing to us with well-equipped laboratories, infrastructure, testing facilities, and positive space which motivates me to do work without any stumbling.
And I am also thankful to the laboratory staff, seniors who guided me and further to the colleagues who helped me in clarifying certain issues.

Shivam Sharma

(2K21/CSE/20)

# ABSTRACT

One of the most significant difficulties in recent decades has been identifying hate speech. These last few years have been contributed to the research based on identifying hate speech on social networking platforms. It appears to be a divisive mode of communication that expresses a hate ideology through misunderstandings. The protected characteristics that are the target of hate speech include gender, religion, race, and disability. For some it might be just fun but for others ,it can become the reason for depression, anxiety, etc. Therefore, it is crucial to keep an eye on user posts and filter out any that may be related to hate speech before it spreads. The number of tweets sent and received on Twitter, however, is over 600 every second and over 500 million each day. This type of incoming traffic makes manual detection of hate speech near to impossible. Hence, a model is proposed to detect hate speech using transformers, deep convolutional networks, and long short-term memory.

The proposed model is given raw tweets as input and these tweets are passed through the transformer layers followed by deep convolutional networks. The output is fed to the LSTM network. The final output results in whether the tweet is hateful or not when passed through the SoftMax layer. The proposed model has been trained and tested on 5 state-of-the datasets and compared with the previous models too.

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| **MHS** | Measuring Hate Speech |
| **TSA** | Twitter Sentiment Analysis |
| **RNN** | Recurrent Neural Network |
| **BERT** | Bidirectional Encoder Representations from Transformers |
| **LSTM** | Long Short Term Memory |
| **SVM** | Support Vector Machine |
| **GloVE** | Global Vectors for Word Representation |
| **CNN** | Convolutional Neural Network |
| **MeL** | Machine Learning |
| **DeL** | Deep Learning |
| **DCNN** | Deep Convolutional Neural Network |
| **TCL** | Transformer CNN LSTM |
| **BiCHAT** | BiLSTM with deep CNN and Hierarchical Attention-based network |
| **SOTA** | State of the art |

# CHAPTER-1

# INTRODUCTION

## 1.1DEEP LEARNING

Deep Learning is a subset of machine learning . When a neural network works on more that three layers it becomes a deep learning model. The main idea to design this model is to study the functionality of human brain so that people can learn how a huge amount of data in our mind is learned and stored. Neural network designed with single layer can output results but if some hidden layers added can contribute to optimized result and betterment of accuracy.

Deep Learning underpins many ArI programmes and services, upgrading the functionality by doing mental and physical work done which is earlier done by humans. Both established and fast technology, like assistants, TV remotes which uses voice functionality[3], and detecting fraud in credit card, are powered by deep learning.

These complex Neural Network uses datas, their weights, including their biases, artificially designed neural networks, described as Deep Learning neural networks. It is a way to clone the functionality of brain of the human. These all together contribute to correct identification and better entity characterization in the data.

## 1.2 HATE SPEECH

The last few decades have seen changes in consumers' communication habits due to advancements in computing technology, particularly in online social networks (OSNs).

Platforms for social networking online, like Facebook, Twitter, Weibo, and WhatsApp, are widely used and ingrained in society. Users share their opinions and discuss current events on these platforms[2], then they circulate them around their OSN-based virtual networks of family and friends. Users typically use more than one OSN platform and currently use one or more of them. The big user base's interactions generate a tonne of data, which we may mine to glean insightful information.

On OSNs[2], users discuss a variety of topics, including travel and the outdoors. Other popular topics include politics, democracy, economics, fashion, and science and technology. One of the most popular OSNs is Twitter, a microblogging network site in which users can interact with other users; express their opinions but within a limit . Its users typically include ones who want to connect themselves with the public like artists, politicians etc. These well-known individuals use it to tweet about events and professional developments, causing platform users to react with a range of emotions and viewpoints. These OSNs have also drawn antisocial and narrow-minded individuals, who typically reply to tweets with derogatory and hostile remarks. In general, the antagonists and antisocial elements target a certain neighborhood, race, gender, or sociopolitical group. In addition to having negative effects, hateful and abusive content can occasionally cause issues with depression and anxiety in the community or individual being targeted. Even OSNs do not give their nods to the definition of hate speech that is generally acknowledged by various scholars and institutes. Different researchers have different meanings of hate speech[11]: some interpret it as abusive , some as offensive, etc. In short, hate speech is defined as offensive and hostile material directed at particular groups with racial, religious, sexual, or other discrimination as their basis. Similarly, Twitter has its own way of defining hate speech. It states that any tweet that encourages violence or instills feeling of hatred against one another based on religion, age , sex, disability or any other kind of discrimination will be considered as hate speech. Because if hate speech is not controlled on OSN platforms can start riots, which leads to social unrest in the real world. For instance, hate speech and anti-social propaganda in relation to the recent Hindu-Muslim clashes inundated OSNs. During the COVID-19 epidemic, anti-Asian and sinophobic

propaganda became pervasive on OSN platforms, accusing Asians of being to blame for the pandemic. To deal with such content, OSN service providers have policies and procedures in place. But as of now, there isn't a practical, widely-accepted solution. For instance, Twitter occasionally removes tweets and comments as well as suspends users who violate its policies.[8] (HCovBi-Caps)

Researchers have been interested in the aforementioned problem for a while, and as a result, various ArIn sub-domains such as MCL[7], and Deep Learning[5] have been suggested in different research. Existing models, however, fall short of the necessary requirements because there are still a lot of HS-related posts circulating around on Twitter. This inspired us to create a model that would stop hate speech-related material from posting.

The latest scholars have considered models based on convolutional neural networks (CNN) for various problems. These problems include spam detection[3], fake news detection; models that make automatic answers to a question .This project also addresses the hate speech identification issue by using a transfer-based learning technique along with (D-CNN), and LSTM in accordance with them. This design of the model performs various operations on the given dataset which in return extract semantic relations in the sentence.

Following are the workings of this project:

- Creation of a model using transformers, deep convolutional neural network (D-CNN), and LSTM layers (TCL) for hate speech detection.
- The proposed model takes raw tweets as input; hence the overhead of feature extraction is minimal.
- The proposed TCL model is a binary classification model; it classifies whether the tweet is hateful or not.

# CHAPTER 2

# LITERATURE SURVEY

In a popular approach, the author created a dataset that has around 16000 hate speech tweets. [32]The model uses n-gram features for the training of the classifier . It is a classification on hate and not-hate tweets. This paper achieves 73.89% as their F1-score. It also considers other features with n-gram such as gender and location; gender with n-gram performed the best.

Hate speech does not have many categories; it should have more categories such as offensive, hatred etc. This is done by Davidson et al. [31]. He tried to find how different hate speech categories can be. He uses POS tagging based on n-gram finding differences between hate, offensive, and abusive. [14][12][11] discuss the process of detecting hate speech . It talks about how detecting hate is a challenging task. It uses neural networks for classification.[4] works on the study of various machine learning algorithms and their application . [16]Paul, Chayan & Bora, Pronami. aims to create a model based on Bi-LSTM and LSTM. It does this by making the data set balanced so that the accuracy can be better.[20] defines various criteria on which a corpus can be analyzed. It uses features based on languages with the n-gram technique for classification. [28] researches about how COVID-19 impacted the life of Asian Americans. Adding more to it, it talks about the increase in hate speech and online crimes toward Asian Americans during the pandemic.

Another model is known as BiCHAT[9], which combines various algorithms such CNN and BiLSTM in an Hierarchical way along with attention layer for tweet representation learning towards hate speech detection. An attention-aware deep convolutional layer is added after the BERT layer in the model the author constructed, which uses tweets as its input. The convolutionally encoded representation is subsequently processed by an attention sensitive BiLSTM network. The tweet is then classified as hostile or common by the model using a SoftMax layer. Three benchmark datasets acquired from Twitter were used to train and evaluate the proposed model,

which outruns the SOTA . There are more models besides BiCHAT, such as those created by [18]. They introduced a zero-shot learning-based method for detecting hatred in cross-lingual text.

In a different approach, [12], the authors first present a transfer learning method for detection of hateful data based on the BERT language model, which has been pre-trained, and then they analyze their created method using two datasets that are available publicly and that have been annotated for racist, sexist, or other ways to define as offensive content on Twitter. They use bias alleviation techniques so that our already trained BERT model designed for hate speech identification can have less effect in terms of bias. To do this, they first re-tuned model based on BERT having new re-weighted data using well established regularization approach to again weigh inputs, therefore reducing the influence of strongly correlated training sets' n-grams with on labelled class. Another model, DeepHate[19], is a cutting-edge deep learning framework that integrates several text representations, including word embeddings, feelings, and topical data, to identify hate speech on social media sites. They do in-depth tests and assess DeepHate using three sizable real-world datasets that are available to the public.

Additionally, researchers are looking into hate speech from a variety of perspectives, including how to analyze social groups that are the targets of hate speech (Mossie and Wang, 2020), performing the evaluation analysis hate speech detection models over wide range of datasets, and the analysis of hate categories (Fortuna et al., 2021).

# CHAPTER 3

# DATASET DESCRIPTION

All experiments and study have been developed over the dataset that had been taken from proceedings of the AAAI conference. The proceedings of the AAAI Conference were first started in 1980 from Standford University, California USA.

The proceeding is sponsored by the Association for the Advancement of Artificial Intelligence. It has provided open access to the entire scientific community resulting in enhancing the research methodology.

### 3.1 DAVIDSON DATASET

Collection of dataset is done from the t-Davidson repository [7] which includes lexicon of hate speech containing different types of sentences and words, these words are taken from the internet based on their usage by various people and further, these lexica are used as a comparison standard against the tweets. These tweets are generated by the API: Twitter API. It generated 85.5 million tweets accounting for 33,458 Twitter users. Out of these tweets, 25,000 tweets are extracted. It is done by CrowdFlower (CF) staff manually. Then these tweets are divided into three categories: hateful, offensive, and neither.

Total: 24783

| **Dataset** | **Class Categories** | | |
|---|---|---|---|
| Davidson | Hate | Offensive | NonHateful |
| | 1430 | 19,190 | 4163 |

Table.3.1. Details of the Davidson dataset

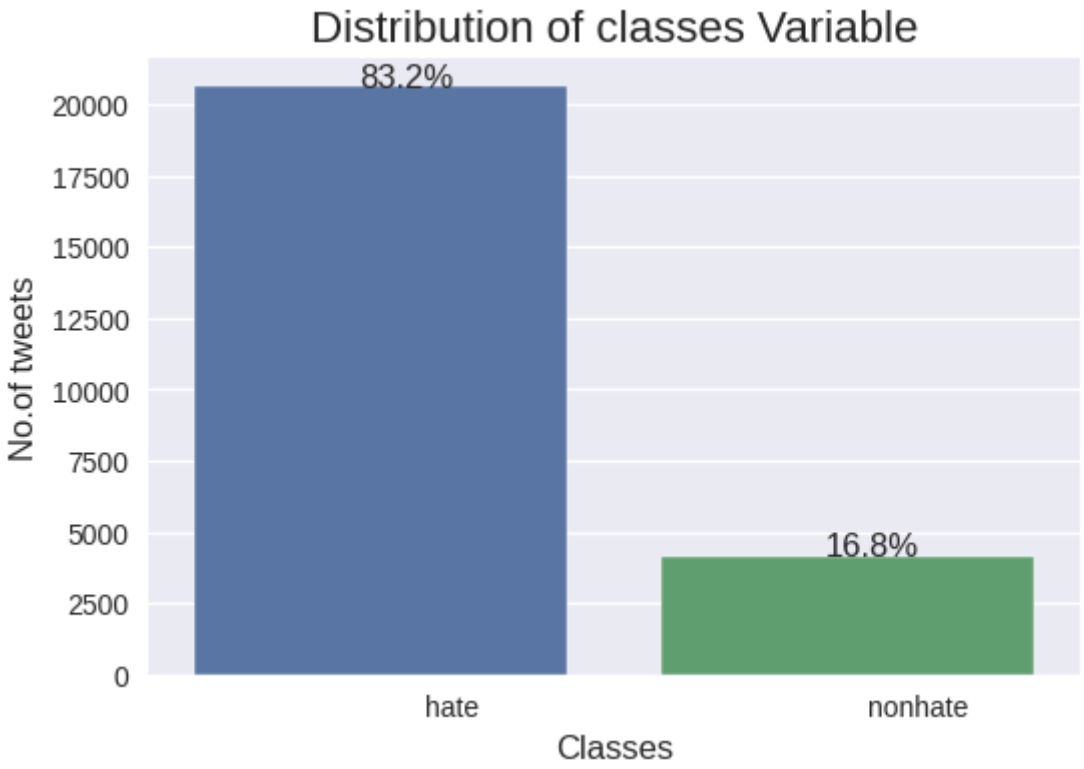



Figure 3.1. Davidson dataset distribution

### 3.2. HATE-CHECK DATASET

Dataset has been collected from the hate-check repository [29]. It is a collection of functional test cases to test hate speech detection capability. This data has been gathered by reviewing various previous research and interviewing various people.

This above dataset has been to cross-validate the proposed model. These tweets are mainly divided into two categories: hateful and non-hateful.

Total: 3091

| **Dataset** | **Class Categories** | |
|---|---|---|
| Hate-Check | Hateful | NonHateful |
| | 2659 | 1242 |

Table 3.2. Details of Hate-Check dataset

### 3.3. TWITTER SENTIMENT ANALYSIS DATASET

This data is provided by Analytics Vidya. This dataset is made by crawling tweets. This dataset classifies data on the basis of hateful and non-hateful.

Total: 31962

| **Dataset** | **Class Categories** | |
|---|---|---|
| TSA | Hateful | NonHateful |
| | 2242 | 29720 |

Table 3.3. Details of TSA dataset

### 3.4. TOXICR DATASET

Dataset has been collected from the WSU-SEAL[20] repository. This dataset has been published in ACM Transaction on Software Engineering and Methodology. This dataset mainly classifies the dataset as toxic and non-toxic which has been relabeled as hateful and non-hateful respectively.

Total: 19671

| **Dataset** | **Class Categories** | |
|---|---|---|
| ToxiCR | Hateful | NonHateful |
| | 3757 | 15894 |

Table 3.4. Details of ToxiCR dataset

### 3.5. MEASURING HATE-SPEECH DATASET

The dataset has been taken from the UC Berkeley data lab repository. This dataset contains a wide range of labels namely hateful, non-hateful, sentiment, etc. This dataset has been sourced from Twitter, Reddit, and YouTube from 50,000 users. Our main use of this dataset is to classify whether data is hateful or not.

Total: 135556

| Dataset | Class Categories | |
|---|---|---|
| MHS | Hateful | NonHateful |
| | 89535 | 46021 |

Table 3.5. Details of MHS dataset

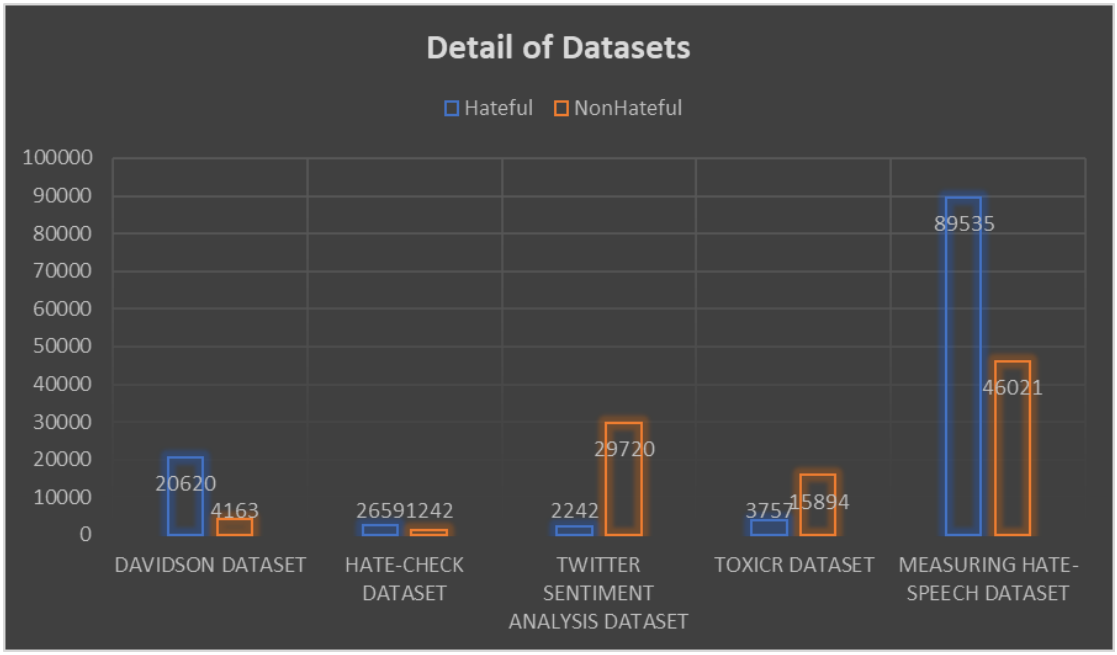



Figure 3.2. Details of all the datasets

# CHAPTER 4

# DATA PREPROCESSING

This chapter provides a brief introduction to data processing techniques that are performed before deploying the dataset to classification.

The adopted techniques in this work are as listed below:

## 4.1.REMOVAL OF STOPWORDS

Stopwords often consist of words that are frequently used. The majority of search engine software is configured to disregard these terms. These include phrases like "the," "a," "an," "in," "on," "was," "were," "it," etc. These are some words that are frequently used and can be found in both hateful and non-hateful tweets.

```
   ...: print(stops)
{'these', 'theirs', 'in', 'herself', 'ourselves', 'no',
"wouldn't", 'by', 'd', 'can', "won't", 'wouldn', 'itself',
'being', "that'll", 'won', 'been', 'not', 'where', 'shan',
"you're", 'himself', 'was', 'during', "haven't", 'out', 'if',
'yourself', 'were', 'it', 'his', 'there', 'll', 'of', 'own',
'against', 'o', 'yourselves', 'yours', 'on', "didn't", 'an',
'didn', "doesn't", 'off', 'be', 'too', 'for', 'only', 'have',
't', "it's", 'weren', 'ours', 'than', 'up', 'between', 'here',
'hasn', 'over', "you'd", 'its', 'am', 'will', 'm', 'me',
'again', "weren't", 'what', 'each', 'needn', 'into', "you've",
'nor', 'their', 'don', 'about', 'those', 'them', 'at', 'just',
'we', 'that', 'while', 'any', 'isn', "wasn't", 've', 'with',
'so', 'are', 'they', "couldn't", 'under', 'who', 'had',
'some', 'him', 'should', 'few', 'how', 'she', 'other',
"shouldn't", "isn't", 'has', 'my', 'a', "mightn't", 'but',
'mustn', 'as', 'shouldn', 'ma', 'is', "shan't", 'from',
"hasn't", 'same', 'most', 'ain', "you'll", 'now', "aren't",
'or', 'after', 'themselves', 's', 'haven', 'i', "needn't",
"she's", "mustn't", 'below', 'your', 'why', 'to', 'mightn',
'did', 'very', 'doesn', 'further', 'having', 'such', 'y',
'he', 'her', 'because', 'you', 'hadn', 'our', 'then', "don't",
'when', 'wasn', 'until', 're', "hadn't", 'before', 'aren',
'hers', 'do', 'and', 'doing', 'couldn', 'does', 'all', 'whom',
'once', 'this', 'above', 'through', 'both', 'down', 'the',
"should've", 'myself', 'more', 'which'}
```

Figure 4.1. Stopwords list

Due to their inclusion in both groups, these could not be of much assistance in the classification of our data. Actually, these words are not that crucial for categorizing a message as hateful or non-hateful, or they do not convey much information that could be used for classification, so we remove all such words from our data frame to save computation time and space and also to avoid the curse of dimensionality. Additionally, it eliminates superfluous symbols like '#' and '@'.

### 4.2.GLoVE

GloVe[8] is an unsupervised learning approach. It is designed by Stanford University. It is an updated version of Bag Of Words. GLoVe converts texts into its vector representation.

It functions by building a matrix of co-occurrences. The frequency with which certain words appear together in the text is taken into consideration by a matrix called a co-occurrence matrix. It fills its value to zero if there is no chance of them coming together. It reads the entire corpus at once, filling the matrix with the necessary data. GloVe saves money by doing this because it only needs to go through the corpus once. As the matrix fills, the subsequent iterations train significantly more quickly.

GloVe generates a word-to-word co-occurrence matrix which becomes the base for the semantic relations between words. This ratio might infer some meaning and can serve as a critical factor for developing the model. A vector is taken out of the pre-trained embedding for each token. Now, if a word is absent in GloVe, the default vector is returned and labeled as “a word unknown”.

### 4.3. TRANSFORMERS

The Transformer[26] is an advanced model that makes use of attention to speed up the rate at which various other models may be taught. It surpasses the Google Neural Machine Translation model in several tasks. The greatest gain, though, comes from its capacity to be parallelized. Actually, Google Cloud suggests using The Transformer as a benchmark when using their Cloud TPU product.

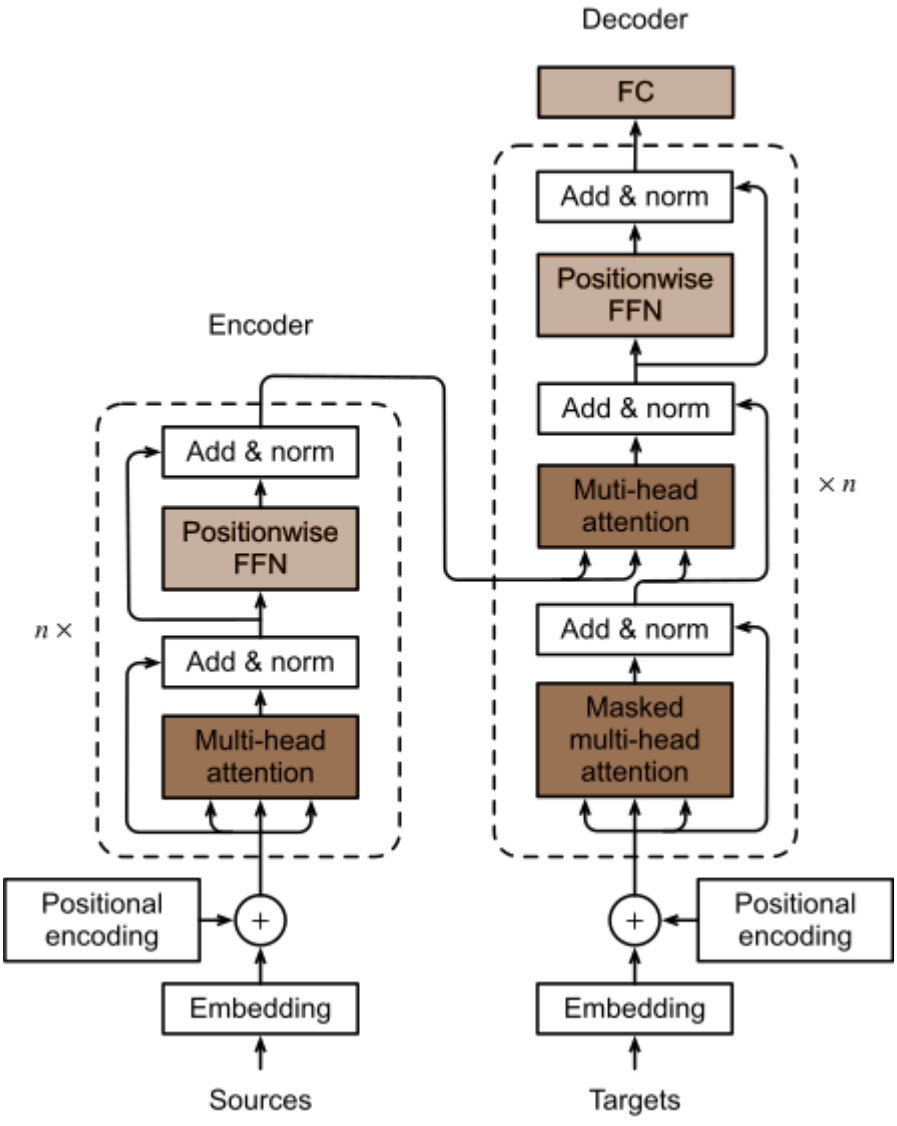



Figure 4.2. Working of Transformer

The input tokens are transformed into vectors in Transformers, and we then add information related to position (positional encoding) to account for the order of the

tokens during the concurrent processing of the model. The attention model refers to these. BERT is one of the attention models we used.

**BERT**.

The science of language understanding has evolved greatly thanks to the ground-breaking natural language processing (NLP) model known as BERT[26]. Word embedding models that are more common, word2vec and GloVe, for example, encode words separately from their context. By capturing bidirectional context, which enables it to comprehend words in relation to their surrounding words in a phrase, BRT addresses this problem.

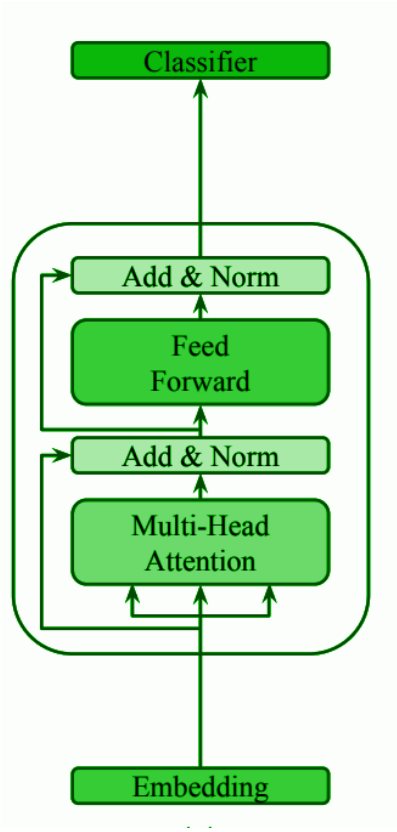



Figure 4.3. Working of BERT

Transformers[], a class of neural networks that makes use of self-attentional mechanisms to identify dependencies between words in a phrase, constitute the foundation of the BERT architecture. The model is made up of an encoder that processes

incoming text through a number of transformer blocks. The presence of numerous self-attention layers and feed-forward neural networks in each transformer block enables BERT to efficiently capture long-range dependencies and contextual information.

BERT goes through pretraining on vast amounts of unlabeled text data, like Wikipedia articles. Masked language modeling (MLM) and predicting what sentences are going to come next are some targets of BERT. A portion of the input words in MLM are masked randomly, and the model learns to classify the masking words using the context the surrounding words offer. On the other hand, NSP entails determining whether two sentences in the original text appear in order. BERT may learn detailed representations of words and their contextual links thanks to this pretraining procedure.

Following pretraining, BERT is adjusted for particular downstream NLP tasks. BERT is adjusted during fine-tuning to task-specific structures and goals. The output of BERT, for instance, is fed into additional classification layers for text categorization. Tasks involving named entity recognition include teaching BERT to recognize and categorize named entities in text. The method of fine-tuning enables BERT to use its pre-trained representations and adapt them to various tasks, leading to enhanced performance and less dependency on sizable, labeled datasets.

BERT's capacity to capture contextual word representations by taking into account both left and right context is one of its main features. BERT's bidirectional nature enables it to comprehend the complex meaning of words based on their surrounding context, in contrast to earlier models that rely on fixed-size word embeddings. Many NLP tasks, such as analysis of sentiments, text categorization, named entity identification, machine translation, and answering to the question asked, have significantly improved as a result of contextual understanding.

# CHAPTER 5

# CLASSIFICATION

Our study's primary goal is to determine whether a specific tweet qualifies as hate speech or not. Therefore, we develop a categorization model. Our deep learning-based model is created.

## 5.1 DEEP LEARNING:

We actually attempt to imitate how a person thinks or how a human brain functions in deep learning. The human body has several receptors that pick up information from the environment. For example, our eyes perceive through sight, our noses sense through smell, our tongues sense through taste, our ears pick up on environmental soundwaves, and our hands sense the world through touch. These are some receptors that take in signals from their environment in order to obtain or retrieve information about their immediate surroundings.

### CONVOLUTION NEURAL NETWORK

Convolutional Neural Networks (CNNs) [34] have successfully implemented and used in a wide range of fields, which includes fields like computer vision. They have also shown promise in deep learning applications requiring text classification. CNNs are highly suited for analyzing the sequential structure of text data because they are particularly useful at finding local patterns and capturing hierarchical representations. An overview of deep learning's use of CNNs for text classification will be given in this paragraph.

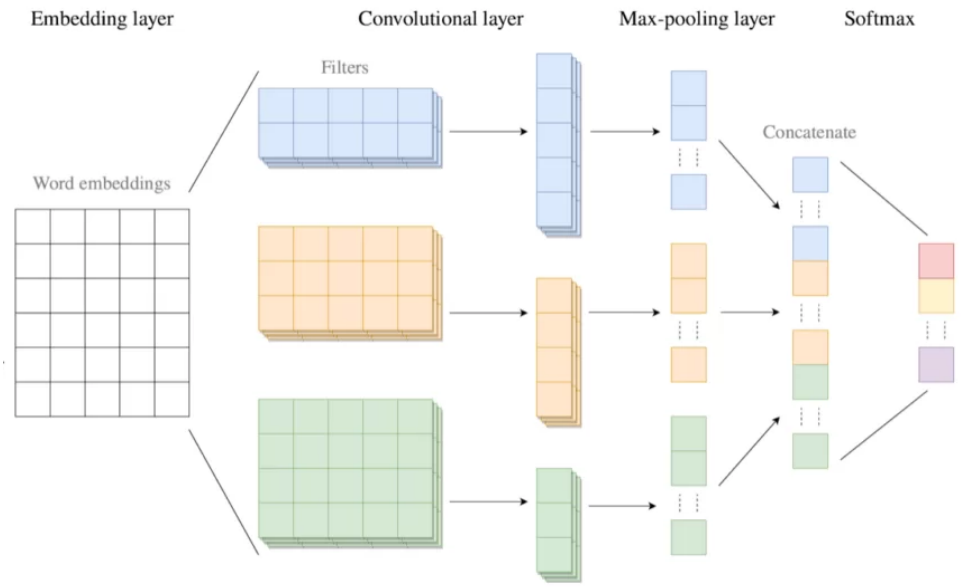



Figure 5.1. Working of Convolution Neural Network[12]

The input for text classification is frequently a list of words or characters. A vector is used to represent each word or letter, frequently utilizing word embeddings like Word2Vec or GloVe. The CNN architecture then runs the input sequence through a number of convolutional layers.

The input sequence is subjected to a series of learnable filters, commonly referred to as kernels, in the convolutional layer. The filters slide through the sequence while multiplying the relevant input items by their associated filter weights. This process creates feature maps that identify specific n-grams or local patterns in the text.

Downsampling feature maps and capturing the most important features often involve pooling layers, such as max pooling[] or average pooling. The dimensionality of the data is decreased through pooling while vital information is kept.

To capture more complicated patterns and advanced info along with multiple convo and layers that are used for pooling can be stacked. The network can learn representations at various levels of abstraction thanks to this hierarchical architecture, which enables it to find pertinent patterns in the text.

The collected features are then transmitted via fully connected layers for classification following the convolutional and pooling layers. Using activation functions like softmax, these layers combine the retrieved information and map them to the appropriate number of classes.

Through backpropagation and gradient descent, the network learns the best filter weights and biases during training, minimizing classification errors by optimizing a given loss function (such as cross-entropy).

There are other CNN model types, including R-CNN and Faster R-CNN. There are various CNN models that can be used depending on the situation. Similar to how words can be grouped based on their weights using CNN.

In a number of applications, such as sentiment analysis, subject categorization, and document classification, CNNs for text classification have had noteworthy success. In text classification, CNNs are advantageous because they can recognize local patterns and hierarchical representations, which are essential for comprehending the sequential character of text. In addition, CNNs are more easily parallelized than other sequence-based models like recurrent neural networks (RNNs) and are computationally efficient, allowing for quicker training and inference.

**LSTM**

[33][36]Similar to the manner in which the human body tries to process each signal it receives by accepting it and sending it to the brain for analysis, there are various devices in machines that attempt to sense, capture, or comprehend their surroundings. These devices then send the information they have gleaned to the CPU for processing. Here, we attempt to teach our CPU-specific algorithms that are based on the idea of the brain's neuronal organization. We introduce a comparable structure in machines through the notion of "Perceptron " that will assist machines to function similarly to the human brain, much as the brain has many thousands of neurons combined in a network that aids it in making decisions whenever there is any sense from the outside environment. In light of this concept, LSTM deep

Artificial (ReNN) is the foundation of the deep learning architecture known as long short-term memory [17][18]. In contrast to conv feedforward neural networks, LSTM has a feedback connection. It can handle both discrete data streams, like audio or video, as well as individual data points, like images. LSTM can be applied to tasks like anomaly detection, speech recognition, and linked handwriting detection. The graphical representation of a recurrent neural network is shown in Figure 5.1.

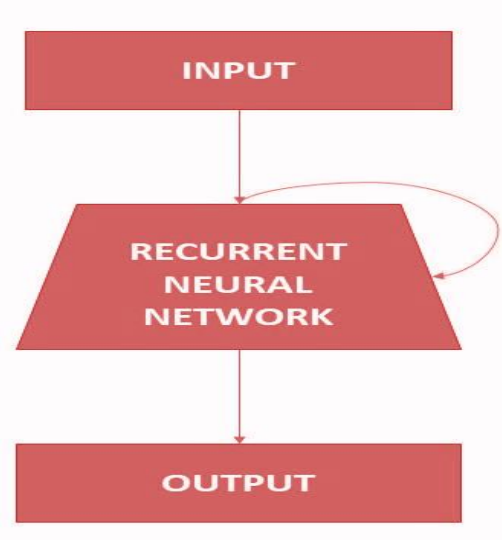



Figure 5.2. Recurrent Neural Network

An unfolded RNN would look similar to the one given in Figure 5.2.

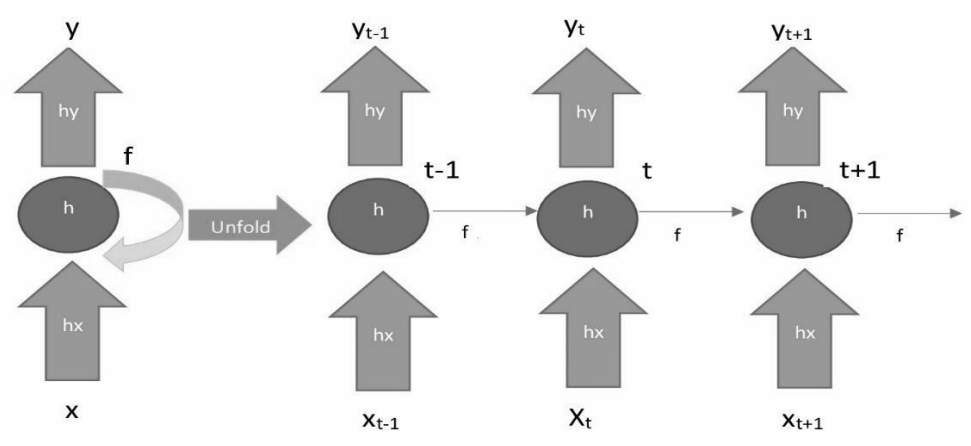



Figure 5.3. Unfolded RNN

The vanishing gradient issue that arises in RNN is addressed by LSTM, a synthetic RNN. Based on the input context, LSTM attempts to remember and forget things. It is carried out by numerous gates that serve a variety of functions. Four gates make up the core of an LSTM, and they are as follows:-

- Input Gate
- Output Gate
- Input Modulation Gate
- Forget Gate

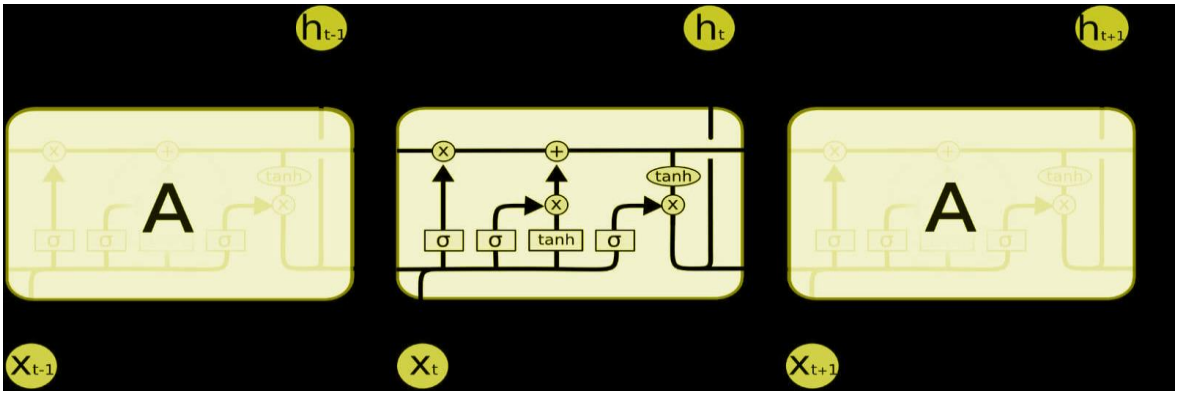


Figure 5.4. LSTM Network [19]

The visual depiction of the LSTM model is shown in Figure 5.3. Here is a brief explanation of how the processing is carried out. The first stage in an LSTM is to remove some words from the cell's state. This is accomplished by the "Forget gate layer," which is depicted in Figure 5.4. (i) and has a mathematical definition in Equation 5.1. It concentrates on the inputs $h_{t-1}$ and $x_t$ and outputs 0 or 1 depending on the cell's current state, or $C_{t-1}$. The numbers 0 and 1 represent "completely get rid of this" and "completely keep this," respectively.

$$f_t = \sigma(W_t.[h_{t-1}, x_t] + b_t) \quad (5.1)$$

The input gate, whose purpose is to preserve information in the cell's state, is shown in Fig. L (ii). It had basically been done in two parts: an "input gate layer" and a sigmoid layer. The mathematical term is given in Equations (8) and (9). We have a "tan h" function that creates a vector, $C_t$, that is extended to the state, for additional values that will come later.

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \quad (5.2)$$

$$C_t = \ tanh(W_c * [h_{t-1}, x_t] + b_c) \quad (5.3)$$

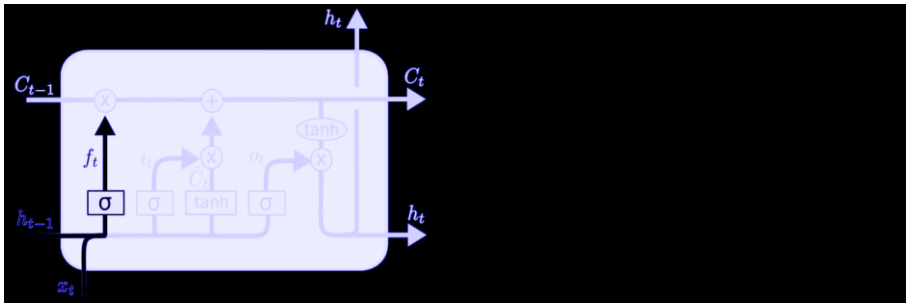
To finish the input determining gate, we now add the former state of the cell, represented by $C_{t-1}$, to the current state of the cell, represented by $C_t$. Its term in mathematics is provided by equation (10). also depicted in Fig. 5.4 (iii). The next step is to multiply the previous state by foot before adding it to the Ct. Therefore, the decision to update each value of a cell's state depends on its new values.

$$C_t = \ f_t * C_{t-1} + i_t * C_t \quad (5.4)$$

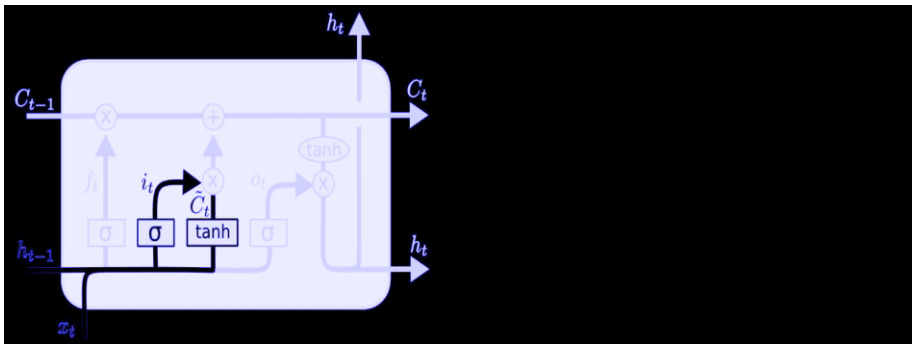
We must now draw a conclusion about the output, which will depend on the condition of the cell. Equations (11) and (12) provide the mathematical calculation for the output gate, and Fig. L (iv) provides a visual representation of the calculation. As a result, we first implement a sigmoid layer that determines the portion of the cell that produces the output, after which we feed the cell's state through the "tan h" function and multiply it with the output produced by the sigmoid gate.

$$O_t = \ \sigma(W_o * [h_{t-1}, x_t] + b_0) \quad (5.5)$$

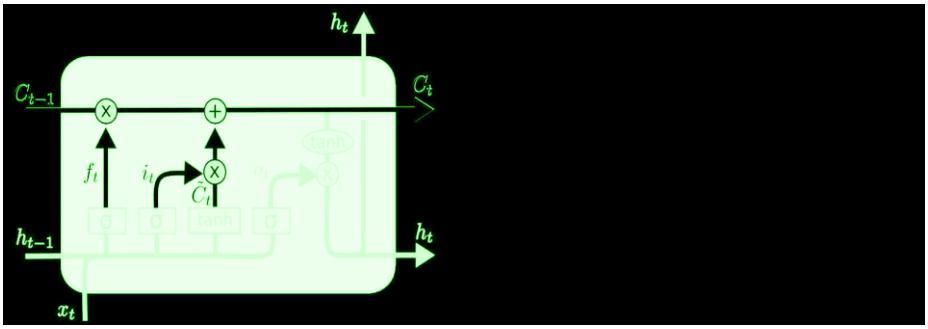
$$h_t = \ O_t \, tanh(C_t) \quad (5.6)$$



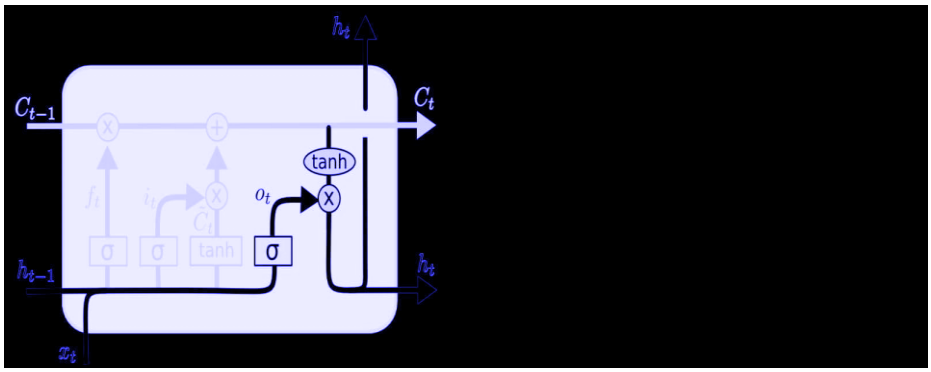

(i) Forget Layer




(ii) Input Gate




(iii) Input Deciding Gate




(iv) Output Gate

Figure 5.5. Gates in LSTM network[19]

Some LSTM restrictions:

- Training takes too much time.
- Implementing dropouts is challenging since it is sensitive to various weight initializations.

LSTM is used for numerous applications and a variety of extensions and modifications are common despite its constraints.

## 5.2. MACHINE LEARNING

Many machine learning techniques have been used widely for solving classification problems. Some of them are listed below.

### SUPPORT VECTOR MACHINE

[4] Support Vector Machine or you can say SVM is one of the famous algorithms. This algorithm is based on a supervised learning approach . It comes in handy for both classification and regression problems.

The SVM algorithm aims at creating is also known as a decision boundary. In this way, it can separate multi-dimensional space into different sets of classes. This will help it to put points into classes according to the requirements.

The best-created decision boundary among all is named hyperplane.

It selects vectors or extreme points. This vector in the future will help it to create hyperplanes.

Basically, two types of SVM can be defined

Linear SVM

When the data given can be separable linearly, this type of SVM is used . Linear separable data is the data whose classification can be done in a single line.

Non-Linear SVM

When the data provided can’t be separated by a single line , non-linear SVM comes into the picture. This type of data is known as non-linear data.

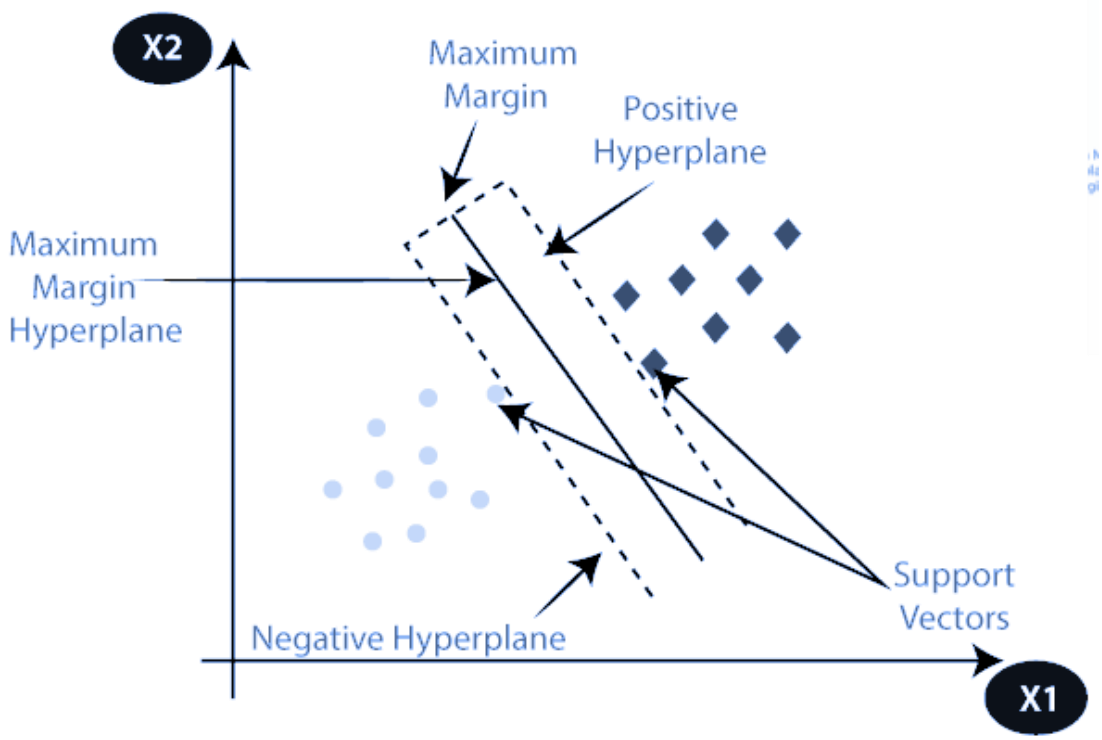



Figure 5.6. Support Vector Machine[10]

Limitations:

- Does not give satisfactory results on large dataset.
- As the noise in the data set increases, performance of SVM decreases. If the features are more than the samples in training perform degrades.
- No proof in terms of probability for the classification since it puts data up and down the hyperplane.

**RANDOM FOREST**

[35] Random Forest is based on decision tree . This means that various decision trees are connected to each other. Each decision tree is assigned a subset of the dataset to work upon. When each decision tree produces their outputs, the average of all the outputs of these decision trees becomes the output of random forest classifier.

It is one among the famous algorithm of machine learning. Like SVM, it is also a supervised learning model and can be used for both regression and classification.

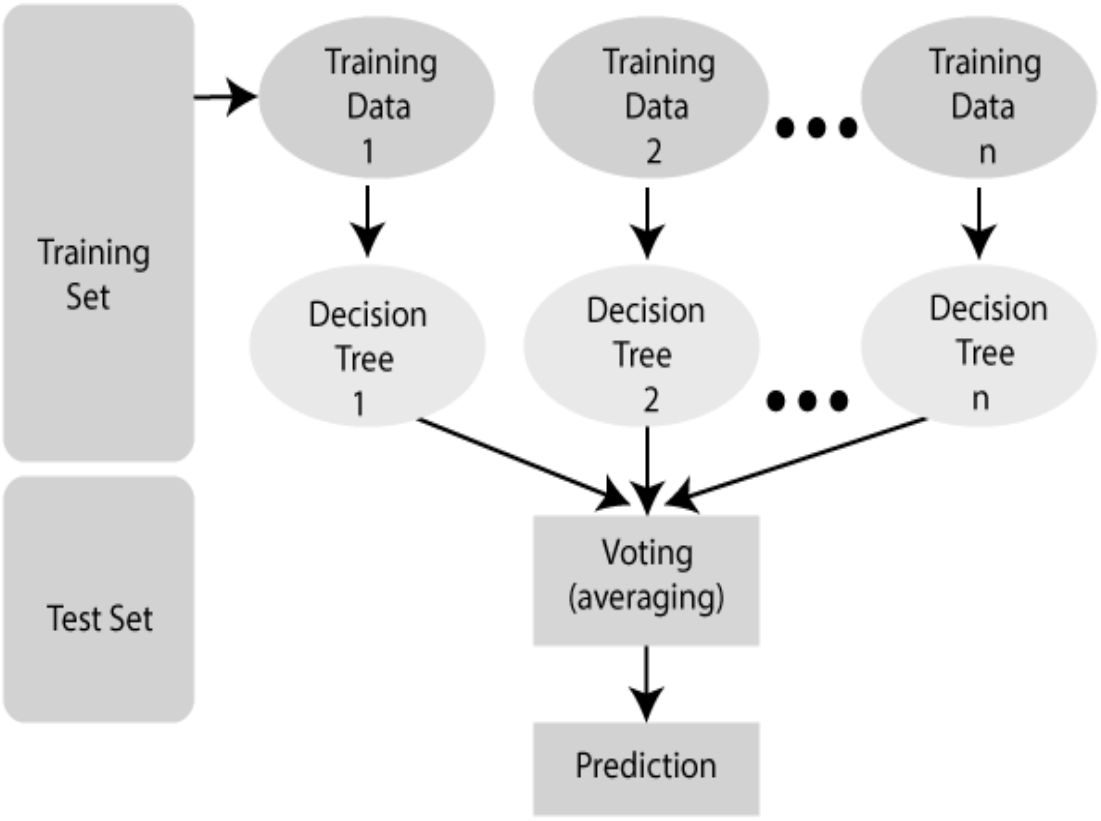



Figure 5.7. Random Forest [35]

The method it uses to work is known as ensemble which states that a complex problem can be solved if we have multiple classifiers for it. As various classifiers are combined, it also upgrades the computation of the model.

The biggest advantage of this random forest algorithm is that it does depend on one decision tree rather it provides input to each decision tree and take their output as prediction . Then it does the polling and outputs the classification based on the majority votes .

Advantage of Random Forest are as follows:

- High Accuracy
- Efficient irrespective of size of dataset
- Stable accuracy even on missing dataset.
- Training time is less as compared to other algorithms.
- Solves both classification and regression problems.

# CHAPTER 6

# PROPOSED MODEL

In this work a novel ensemble algorithm had been provided for algorithms. This algorithm is based on three levels of filtering features. The first layer is based on self-attention and is based on the self-attention mechanism (BERT) whose output is fed as an input to a deep learning model which is designed on the basis of CNN. The output produced by the second layer is the input to the third layer which is based on LSTM. The proposed algorithm effectively reduces computational time and enhances the accuracy of the model.

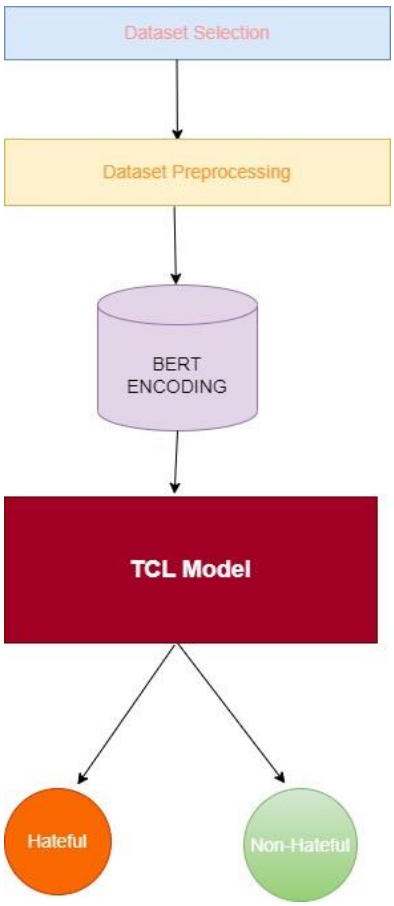



Figure6.1. Flow chart of the proposed algorithm.

The Proposed model  is scalable in a way that if the data is large, it will still perform better than another previously made model. The created model surpasses the results of various previous machine learning and deep learning models.

The flow of the model step-by-step is as follows:

1. Dataset Collection

Davidson's repository's dataset has been used[7]. Hate, offensive, and neither are the three different labels used to categorize the dataset. We combined hate and offensive since they are closer to one another than either class and because the data was so severely unbalanced. Then, the labels for these classes are changed to indicate whether they are hateful or not.

2. Data preprocessing

Stop words are eliminated in this step because they don't help with classification. They merely lengthen the computing process.

Data Splitting: The dataset is divided into a test dataset and a training dataset in the proportions of one to nine.

Apart from it, the following actions are carried out:

TensorFlow Dataset objects can be created to make working with data in TensorFlow more effective and convenient. Using them makes it simple to perform operations and transformations on the data while building and testing models. Tensor slices are produced using a technique for turning the input tensors into a dataset. To generate a dataset with each element being a pair of text and label, the text values and label values are supplied as tensors in this instance.

Shuffling: Data shuffles are necessary to bring randomization into the training examples' order. This is especially helpful for avoiding bias that can result from having the data in a certain order. The model benefits from learning from a variety of instances in each training batch thanks to shuffling.

Batching: Organizing several examples into batches is a practice known as batching. It is done to make it possible for training and evaluation to process data

effectively in parallel. The training process can be considerably sped up by batching the data and running computations on a number of samples at once.

Drop_remainder: If the final batch has less elements than the designated batch size, the Drop_remainder parameter defines whether it should be dropped. If the total number of components is not divisible by 32, you can allow the last batch to have fewer than 32 elements by setting it to False. By doing this, even if the final batch is less than the designated batch size, it is ensured that all data instances are included.

To prepare the data for feeding into a machine learning model for training and evaluation, we created TensorFlow Dataset objects, shuffled the data, and batched the samples.

3. BERT Layer

Text encoding employs the BERT model. BERT is a potent language representation model that can be customized for different natural language processing (NLP) tasks and has been pre-trained on a sizable corpus of text data.

It can be classified into two parts:

- Preprocessing: It prepares the supplied text for use by processing it.
- Encoder: The BERT encoder takes into account the full input sequence to construct contextualized word representations. Each word's significance in the context of the complete phrase is captured by the contextualized representations. In order to improve the representations at each layer and capture more detailed contextual information, it makes use of data from earlier layers.

4. TCL MODEL:

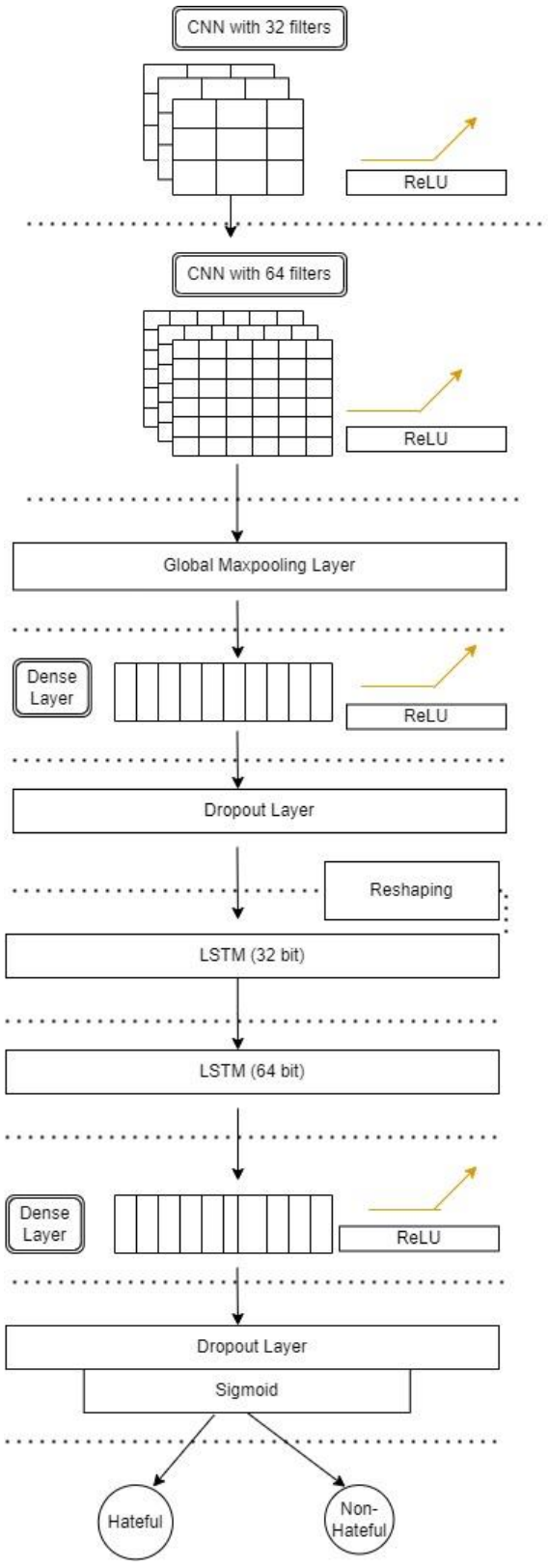



Figure 6.2. Detailed TCL Model

To perform the binary classification challenge, the TCL Model was developed. The above figure provides more details.

1. Input Layer

A tensor of strings is used as the model's input for text data. These are the inputs produced during the preparation of the data.

2. Preprocessing Layer:

Using a pre-trained transformer model called BERT, the aforementioned tensor strings that are provided as input is processed. As a result, the input data can be tokenized, which converts sentences into tokens and paragraphs into tokens. In addition, this layer performs the necessary preprocessing steps.

3. Encoder Layer:

The input for the BERT encoding layer is the output from the layer above. Contextualization, or context encoding, is carried out by this layer on top of the incoming data, and embeddings are produced.

4. CNN Layers.

Two 1D CNN layers are present. Using a 1-dimensional CNN layer is appropriate because the data being used is text-based.

32 filters and a kernel size of two make up the first CNN layer, while 64 filters and a kernel size of two make up the second CNN layer. Normal features are captured by the first layer, while more complicated values and interactions are captured by the model's second CNN layer, which has a filter of size 64.

Each filter in the Conv1D layer does a dot product with the matching window in the input and applies a sliding window of size 2 over the input sequence. This procedure aids in the sequence's local pattern and feature interaction capturing.

ReLU (Rectified Linear Unit) activation function: This function is used after both CNN layers. Since ReLU introduces non-linearity and aids the model in learning intricate correlations between the input and the output, it is a frequent choice for activation functions in CNNs.

A tensor containing the learned features retrieved from the input sequence is the result of the second CNN layer. The number of output channels in the final tensor depends on the number of filters in the CNN layers. In this instance, the first CNN layer generates a tensor with 32 channels, and the second CNN layer raises that number to 64.

5.Global Max Pooling Layer:

This layer implements a global max pooling operation along the input tensor's time dimension. As a result, a fixed-length vector representation is produced by taking the highest value possible for each channel (feature) in the input tensor. Global max pooling decreases the spatial dimensions of the tensor and aids in extracting the key features from trained feature maps.

6.Dense Layer:

Each neuron in the dense layer is linked to every other neuron in the preceding layer, making it a fully connected layer. After the global max pooling layer in this model, a dense layer with 256 units and a ReLU activation function is introduced. The input characteristics are subjected to a linear transformation in the dense layer before being activated. It aids in the discovery of intricate, non-linear correlations in the data.

7.Dropout Layer:

This layer is intended to regularize the model and decrease overfitting. It assists in preventing the model from becoming overly dependent on particular features or co-adapting to the training data by randomly setting a portion of the input units to 0 during training. In this model, the dense layer is followed by a dropout rate of 0.1 (10%).

8.Reshape Layer:

The tensor is reshaped into the desired shape by this layer. It is applied in this model to

alter the dropout layer's output. The shape is defined as (-1, network_layer.shape[-1]), which indicates that the second dimension is set to the number of units in the input tensor's last dimension and that the first dimension is automatically determined based on the shape of the input tensor. The purpose of this reshaping is often to get the data ready for additional processing, including feeding it into recurrent layers.

9. LSTM Layers:

Following the convolutional layers in the provided model, LSTM layers are used to process the input sequence. In this model, there are two defined LSTM layers:

LSTM layer one: 32 units make up the first LSTM layer, which is configured to return sequences. It utilizes the output of the convolutional layers from the preceding layers. The output shape of the preceding layers determines the input shape for this layer. This layer's goal is to identify dependencies and sequential patterns in the input data.

It enables the following layers to access the output of each time step by returning sequences. This layer outputs a series of LSTM hidden states, each of which represents the input's learned representation at a specific time step.

The second LSTM layer is made up of 64 units and is not configured to return sequences. It accepts as input the output sequence from the top LSTM layer. The previous LSTM layer's sequential information is further processed by this layer, which results in a single output that contains a summary of the complete sequence. This layer concentrates on capturing higher-level patterns and dependencies across the full input sequence by not returning sequences.

10.Dense Layer:

After the LSTM layers, a second completely connected Dense layer with 256 units and the ReLU activation function is used.

11. Dropout Layer:

There is an additional dropout layer added with a dropout rate of 0.1.

12.Classification Layer:

For binary classification, a Dense layer having only one unit and sigmoid as an activation function is utilized. It generates the likelihood that the input falls into the positive class.

The model architecture combines LSTMs' capacity for sequential modelling with CNNs' advantages in identifying local patterns. Convolutional operations are used to collect sequential information, LSTM layers are used to capture learned representations from the BERT encoder, and ultimately a binary classification prediction is made.

# CHAPTER 7

# EXPERIMENT AND RESULTS

In this part, the proposed model's description, experimental parameter setting, evaluation metrics, and experimental findings are all evaluated over five benchmark datasets. Additionally, we contrast the suggested model with a number of standard SOTA and baseline techniques.

## 1. SOFTWARE USED

| Software/Libraries | Version | Usage |
|---|---|---|
| Google colab | 1 | Platform for implementing the model |
| Pandas | 1.5 | For extracting and assessing the dataset. |
| Numpy | 1.23.5 | For calculation |
| Matplotlib | 3.6.2 | For data visualization |
| Nltk | 2.0 | For data preprocessing such as stop words removal |
| Keras | 3.7.0 | For creation of deep learning models |
| Sklearn | 0.24 | To provide evaluation metrics |
| Tensorflow | 2.10.1 | Implementing deep learning models. |

Table 7.1. Software Implementation

Operating System: Windows 10 (Version: 20H2)

## 2. HYPERPARAMETERS USED

| Hyperparameters | Assigned |
|---|---|
| Embed. Dimension | 300 |
| Length of the sequence (maximum) | 30 |
| CNN Kernel Size | 2 |
| No. of Filters in CNN | 256 |
| No. of neurons in LSTM | 256 |
| Dropout | 0.3 |
| Size of the Batch | 32 |
| Optimization Algorithm | Adam |

Table 7.2. List of Hyperparameters

## 3. PERFORMANCE METRICS

i) Metric Formulas.

Following metrics are used to evaluate model performance.

Precision (P) - It is the ratio of actually positive predicted class to the ratio of positive class predicted by model.

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \quad (1)$$

Recall (R)- It is the ratio of actually positive predicted class to the ratio of actually positive class in the data.

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \quad (2)$$

F1 score - It is the score produced by dividing twice the value of product of precision and recall with the summation of value of precision and recall

$$\text{F1} - \text{Score} = \frac{2*\text{Precsion}*\text{Recall}}{(\text{Precision}+\text{Recall})} \quad (3)$$

Both the macro F1 score and weighted average F1 score is considered in this classification implementation for the overall performance. It is because the data is highly imbalanced.

The Macro F1 score and weighted F1 score are the main evaluation metrics for the classification . This is because of the fact that data is unbalanced, and it might not give relevance to the respective class.

Macro F1 score- Summation of F1 scores divided by a number of classes.

$$\text{Macro-F1}=\frac{1}{Q}\sum_{j=1}^{Q}\frac{2*p_j*r_j}{p_j+r_j} \quad (4)$$

Where Q is the no. of classes and pj and rj are precision and recall of jth class respectively.

Weighted F1 score- Summation of the product of the F1 scores divided by a total number of classes.

$$\text{Weighted F1} = \frac{1}{N}\sum_{i=1}^{i=N} F_{1i}*N_i \quad (5)$$

Where Ni is the weight of class i and F1i is the F1-score of class i .

## 4. EXPERIMENTS AND RESULTS

Series of experiments has been done. Various parameters have been compared with other models. The proposed model surpasses all other models providing best results.

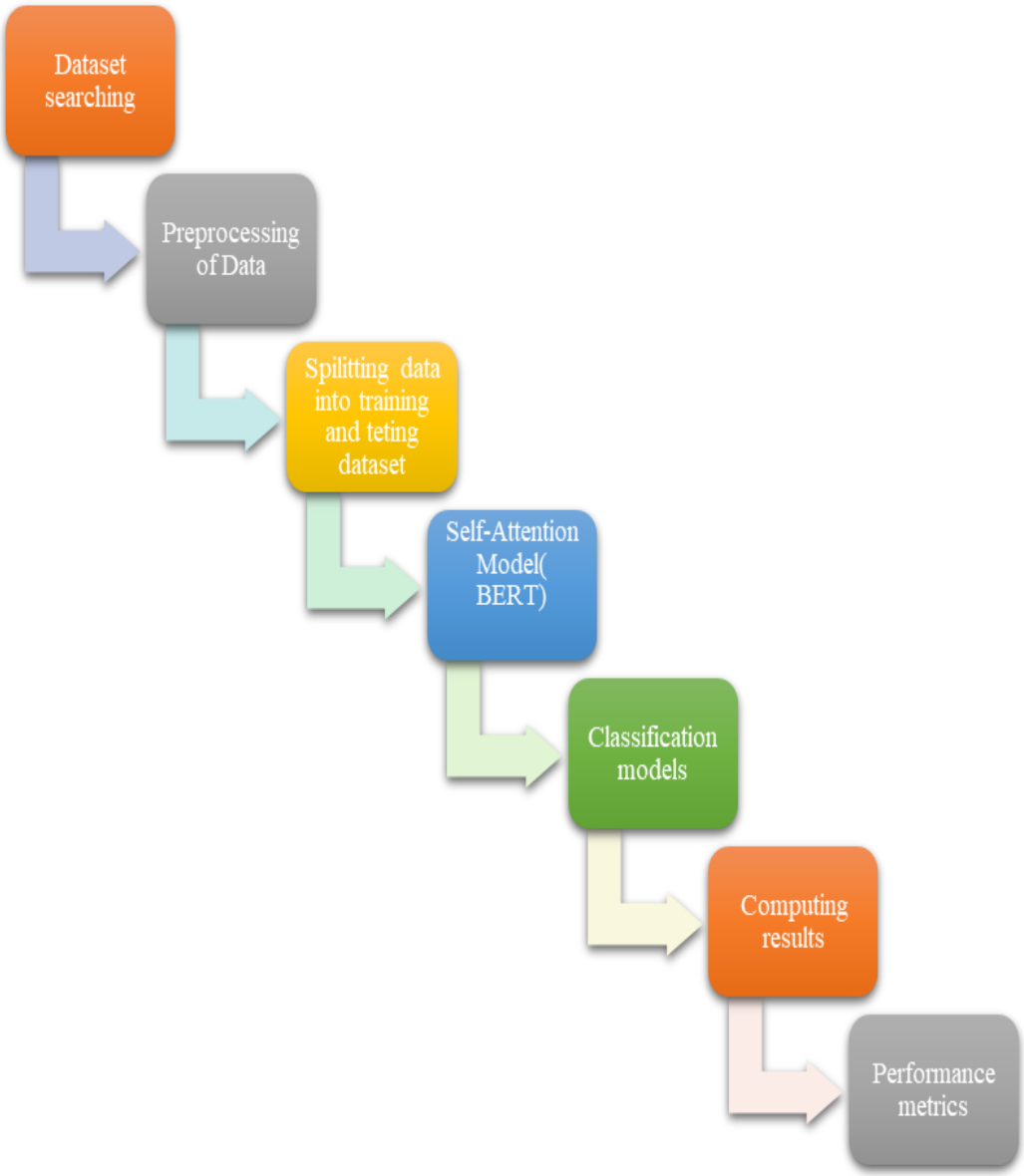



Figure 7.1. Flow of proposed work

Given flow above describes the working of proposed work.

Different datasets have been run through the TCL model and the ratio of testing data to training data is 2:8.

| | **Percentage** |
|---|---|
| **Testing Dataset** | 20% |
| **Training Dataset** | 80% |

Table 7.3. Dataset distribution statistics

**RESULTS**

1. Evaluating TCL model on Davidson Dataset

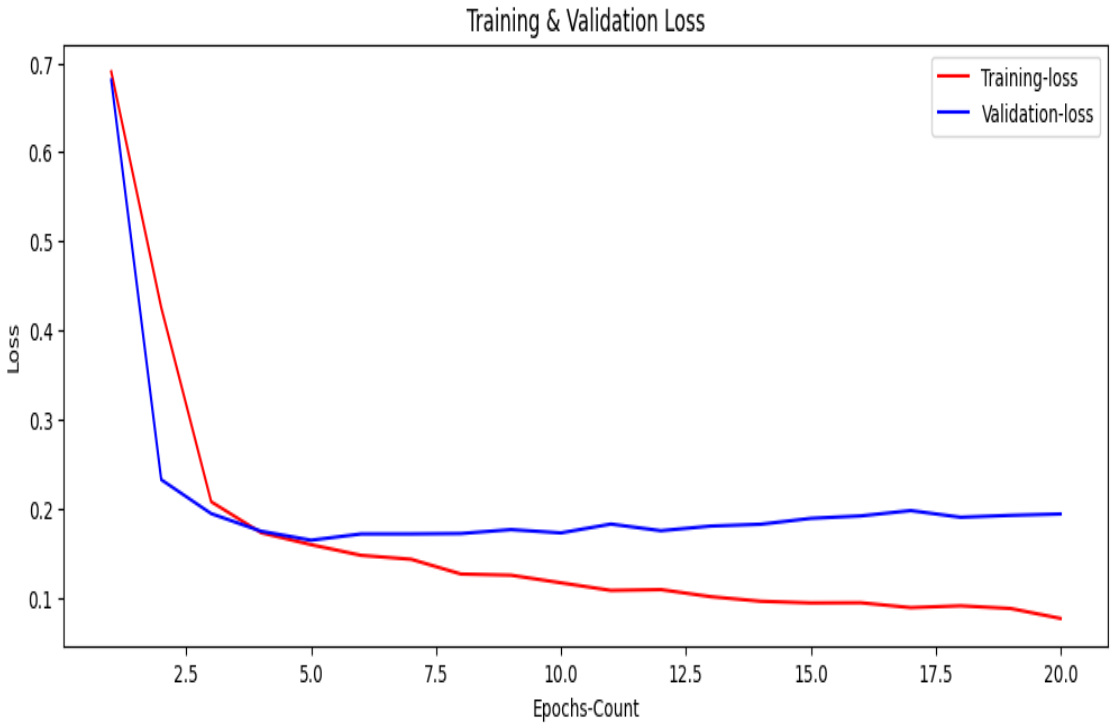



Figure 7.2. Training and Validation Loss on Davidson Dataset

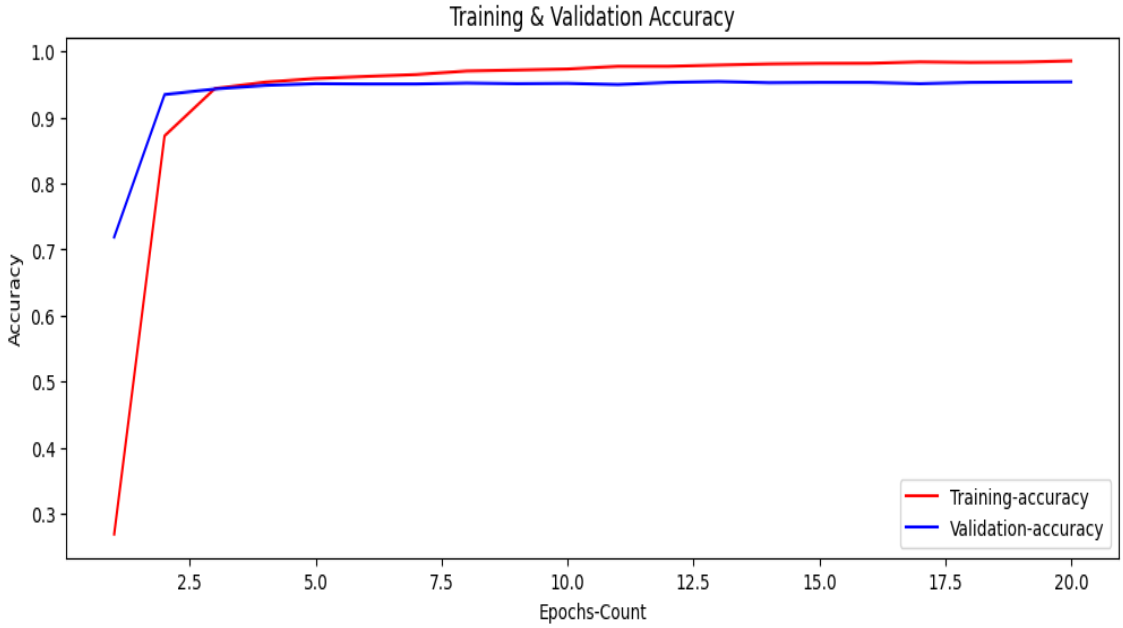



Figure 7.3. Training and Validation Accuracy on Davidson Dataset

As seen from the above graphs, with increasing epochs the validation loss decreases and validation increases signifying the model is predicting correctly.

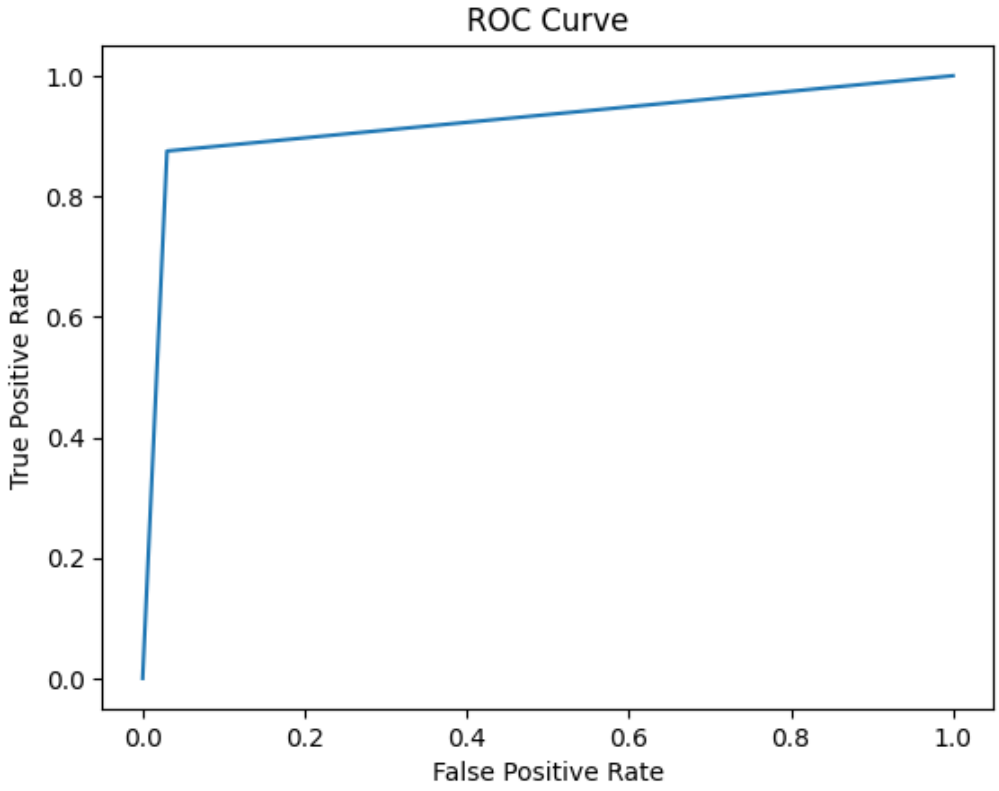



Figure 7.4. ROC Curve of TCL model on Davidson dataset

The ROC-AUC score is 0.92 resulting in outstanding predicting capability.

2. Evaluating TCL model on different dataset

| Dataset | Weighted | | | Macro | | |
|---|---|---|---|---|---|---|
| | **Prc** | **Rcl** | **F1S** | **Prc** | **Rcl** | **F1S** |
| Davidson | 0.95 | 0.95 | 0.95 | 0.91 | 0.92 | 0.92 |
| Hate-check | 0.93 | 0.95 | 0.94 | 0.92 | 0.93 | 0.91 |
| TSA | 0.95 | 0.94 | 0.95 | 0.93 | 0.90 | 0.91 |
| MHS | 0.89 | 0.90 | 0.90 | 0.87 | 0.88 | 0.88 |
| ToxiCR | 0.94 | 0.95 | 0.94 | 0.92 | 0.90 | 0.91 |

Table 7.4. TCL model results on different dataset

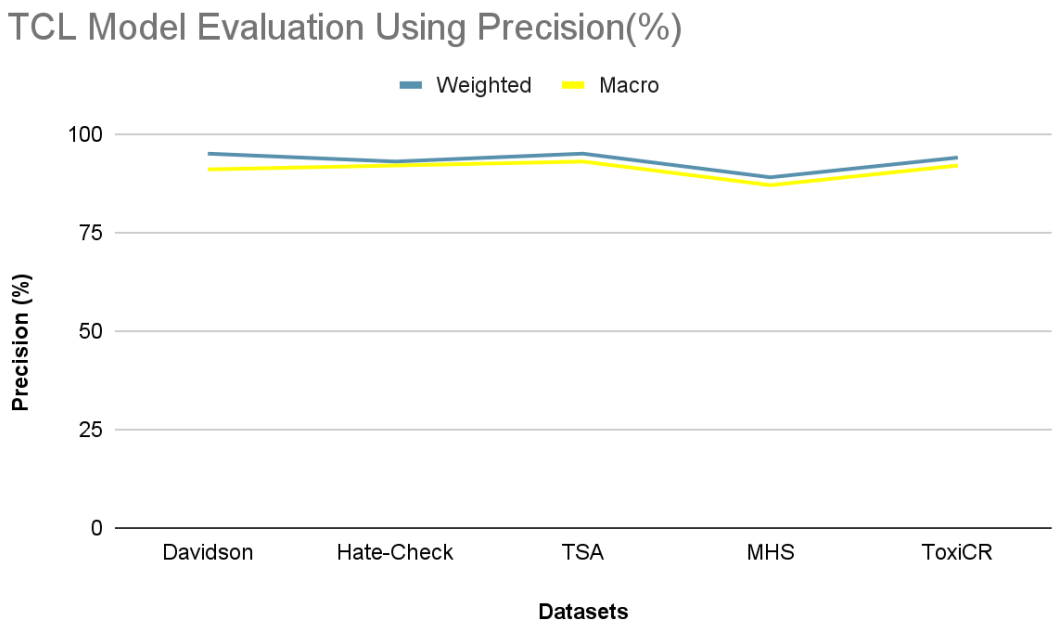



Figure 7.5. Comparison of precision(%) of different datasets

As from the above graph, we can see that each dataset is showing precision above 90% except MHS dataset.

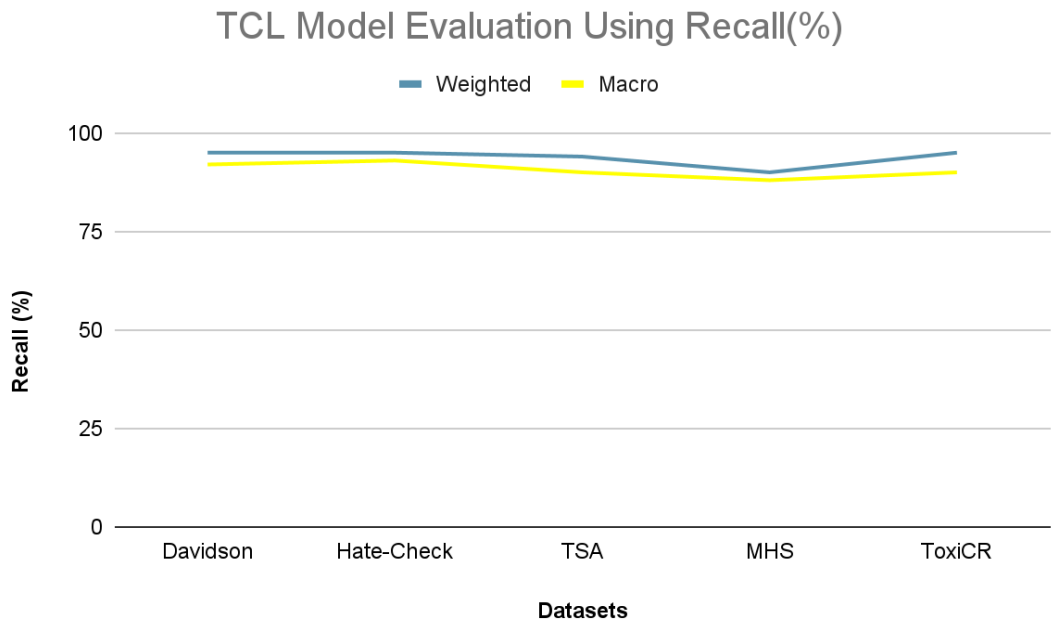



Figure 7.6. Comparison of Recall(%) of different datasets

More the value of Recall near 1 or 100% more is the predicting capability of the created model to predict the hate speech data correctly which can be seen in this case.

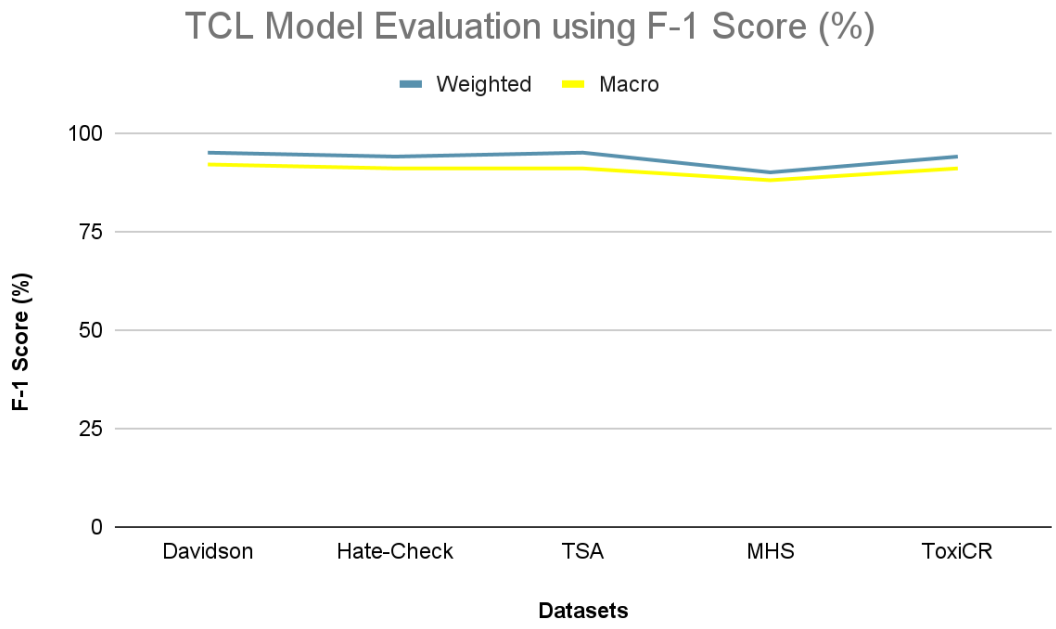



Figure 7.7. Comparison of F1-Score(%) of different datasets

F1-Score tells us how no. times model predicted the values correctly across the entire dataset. As we can see F1-Score is also above 90% giving the satisfactory results,

3. Comparison of SOTA models against TCL

Various other SOTA models has been compared against TCL using Davidson Dataset.

| Models | Training Accuracy | Validation Accuracy |
|---|---|---|
| SVM | 0.83 | 0.80 |
| LSTM | 0.88 | 0.85 |
| BiCHAT | 0.87 | 0.89 |
| HCovBi-Caps | 0.87 | 0.80 |
| D-CNN | 0.87 | 0.88 |
| **TCL** | **0.97** | **0.95** |

Table 7.5. Training and Validation Accuracy of different models.

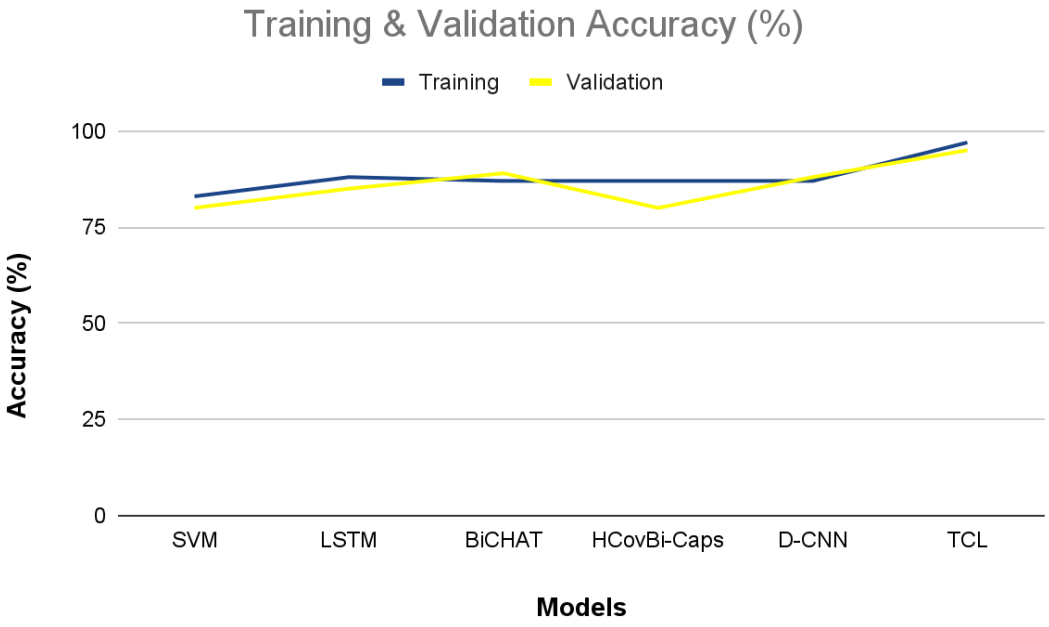



Figure 7.8. Comparison of TCL against various SOTA models.

From the above graph, it is clear that our model is surpassing other SOTA models.

4. TCL model evaluation on balanced and imbalanced datasets.

Model is evaluated on Davidson dataset in two cases: with and without data balancing.

Table 7.6. Comparison between balanced and imbalanced datasets.

| **Dataset** | **With weight balancing** | | | | | |
|---|---|---|---|---|---|---|
| | **Weighted** | | | **Macro** | | |
| | **Prc** | **Rcl** | **F1S** | **Prc** | **Rcl** | **F1S** |
| Davidson Dataset | **0.95** | **0.95** | **0.95** | **0.91** | **0.92** | **0.92** |
| | **Without weight balancing** | | | | | |
| | 0.83 | 0.81 | 0.81 | 0.78 | 0.74 | 0.76 |

Table 7.6. Comparison between balanced and imbalanced datasets.

From the above it is clear that TCL performed better on balanced dataset than an imbalanced dataset.

5. Cross-Domain Experiment

TCL model is also evaluated in the cross-domain experiment.

Below are the details of the datasets used.

Figure 4: Cross-domain experiment description

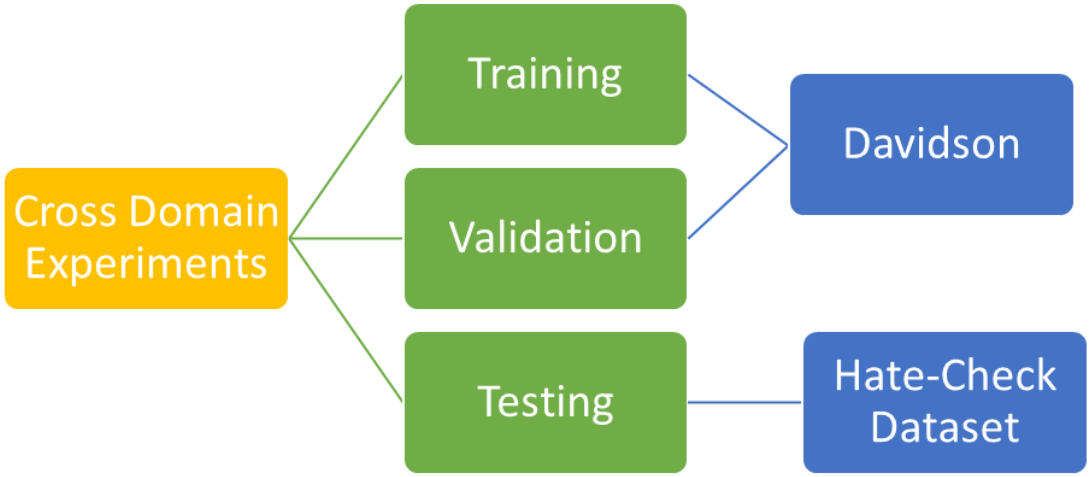



Figure 7.9. Details of Cross-Domain Experiment

| | **Dataset** |
|---|---|
| **Testing Dataset** | Hate-Check Dataset |
| **Training Dataset** | Davidson Dataset |

Table 7.7. Training and Testing Dataset statistics

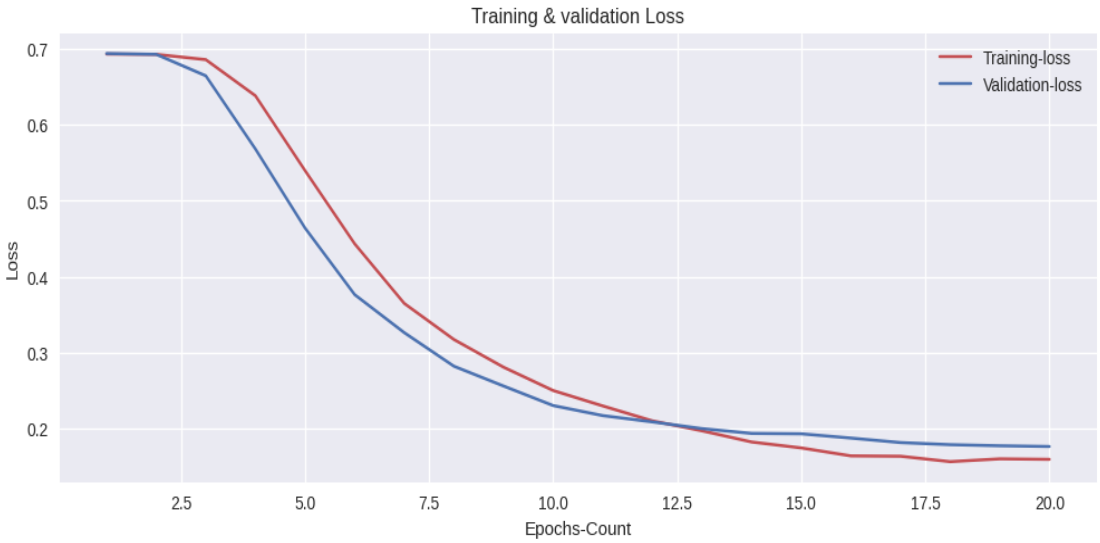



Figure 7.10. Comparative Analysis of Training and Validation Loss of Cross-Domain Experiment


Figure 7.11. Comparative Analysis of Training and Validation Accuracy of Cross-Domain Experiment

| Weighted | | | Macro | | |
|---|---|---|---|---|---|
| **Prc** | **Rcl** | **F1S** | **Prc** | **Rcl** | **F1S** |
| 0.95 | 0.95 | 0.95 | 0.97 | 0.94 | 0.95 |

Table 7.8. Weighted and Macro Results of Cross-Domain Experiment

From the above table and graphs, it can be seen that our model cross-domain experiment is giving excellent results.

From the results obtained, it can be claimed that the proposed work has given more acceptable results than compared with another SOTA model.

TCL Model, when trained and tested on the same dataset or when trained and tested on different datasets performs better than other models.

When compared on precision, recall, and F1-score it is giving the excellent results. As we know, if F1-score and recall are rising in the direction nearer 1 then the model is working fine. In each experiment of ours, the TCL model is proving the above point.

When evaluated and compared against different SOTA models, it is proving to be one of the best models giving a validation accuracy of 95%.

# CONCLUSION

As with this invention of new technologies that are being spreading worldwide the large amount of data that is generated has be to classified against the rogue data is generated every day in the server or network. Though the machine learning and deep learning models are there that are available to classify the data but the prediction of these models is not at par with the dataset. The work proposed in this study is to create a powerful model which can save computational cost and increase prediction power. The model proposed has given acceptable results over the five different datasets. It also outperforms different SOTA models giving an accuracy of 95%.

The proposed model serves as an example of multilevel filtering that could be employed in real world for better classification of such unsolicited tweets from the network.

# FUTURE SCOPE

The advancement of technology is bringing new techniques and research periodically. These are inducing a large amount of data in the environment that needs some more manageable and strong automated way to throw out such high volume of undesired data. Here we have taken datasets that are highly imbalanced and inadequate. In the future, we would be creating a dataset that is balanced and relevant. Also, there is a requirement for datasets that are closer to this era as new words trend every day which can become a new source of hate. Some language independent models and cross domain and platform models are also the prerequisite of the future security models.

# REFERENCES

[1] Jaydeb Sarker, Asif Kamal Turzo, Ming Dong, and Amiangshu Bosu. 2023. Automated Identification of Toxic Code Reviews Using ToxiCR. ACM Trans. Softw. Eng. Methodol. Just Accepted (February 2023). https://doi.org/10.1145/3583562

[2] Konrad Rudnicki, Heidi Vandebosch, Pierre Voué & Karolien Poels (2023) Systematic review of determinants and consequences of bystander interventions in online hate and cyberbullying among adults, Behaviour & Information Technology, 42:5, 527-544, DOI: 10.1080/0144929X.2022.2027013.

[3] Tariq Abdullah and Ahmed Ahmet. 2022. Deep Learning in Sentiment Analysis: Recent Architectures. ACM Comput. Surv. 55, 8, Article 159 (August 2023), 37 pages. https://doi.org/10.1145/3548772

[4] P. P. Jemima, B. R. Majumder, B. K. Ghosh, and F. Hoda, "Hate Speech Detection using Machine Learning," 2022 7th International Conference on Communication and Electronics Systems (ICCES), 2022, pp. 1274-1277, doi: 10.1109/ICCES54183.2022.9835776

[5] Malik, Jitendra Singh, Guansong Pang, and Anton van den Hengel. "Deep learning for hate speech detection: a comparative study." arXiv preprint arXiv:2202.09517 (2022).

[6] Nabil Badri, Ferihane Kboubi, and Anja Habacha Chaibi. 2022. Combining FastText and Glove Word Embedding for Offensive and Hate speech Text Detection. Procedia Comput. Sci. 207, C (2022), 769–778. https://doi.org/10.1016/j.procs.2022.09.132

[7] A. Tiwari and A. Agrawal, "Comparative Analysis of Different Machine Learning Methods for Hate Speech Recognition in Twitter Text Data," 2022 Third International Conference on Intelligent Computing Instrumentation and Control Technologies (ICICICT), 2022, pp. 1016-1020, doi: 10.1109/ICICICT54557.2022.9917752.

[8] S. Khan et al., "HCovBi-Caps: Hate Speech Detection Using Convolutional and Bi-Directional Gated Recurrent Unit With Capsule Network," in IEEE Access, vol. 10, pp. 7881-7894, 2022, doi: 10.1109/ACCESS.2022.3143799.

[9] Khan, Shakir & Fazil, Mohd & Sejwal, Vineet & Alshara, Mohammed & Alotaibi, Reemiah & Kamal, Ashraf & Baig, Abdulrauf. (2022). BiCHAT: BiLSTM with deep CNN and hierarchical attention for hate speech detection. Journal of King Saud University - Computer and Information Sciences. 34. 10.1016/j.jksuci.2022.05.006.

[10] Kennedy CJ, Bacon G, Sahn A, von Vacano C. Constructing interval variables via faceted rasch measurement and multitask deep learning: a hate speech application. arXiv preprint arXiv:2009.10277. 2020 Sep 22.

[11] A. B. Pawar, P. Gawali, M. Gite, M. A. Jawale and P. William, "Challenges for Hate Speech Recognition System: Approach based on Solution," 2022 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), 2022, pp. 699-704, doi: 10.1109/ICSCDS53736.2022.9760739.

[12] Qasim, Rukhma & Bangyal, Waqas & Alqarni, Mohammed & Almazroi, Abdulwahab. (2022). A Fine-Tuned BERT-Based Transfer Learning Approach for Text Classification. Journal of Healthcare Engineering. 2022. 1-17. 10.1155/2022/3498123.

[13] B. P. Putra, B. Irawan, C. Setianingsih, A. Rahmadani, F. Imanda and I. Z. Fawwas, "Hate Speech Detection using Convolutional Neural Network Algorithm Based on Image," 2021 International Seminar on Machine Learning, Optimization, and Data Science (ISMODE), 2022, pp. 207-212, doi: 10.1109/ISMODE53584.2022.9742810.

[14] P. Shah and A. Patel, "A Comprehensive Study and Detailed Review On Hate Speech Classification : A Systematic Analysis," 2021 International Conference on Advances in Computing, Communication, and Control (ICAC3), 2021, pp. 1-6, doi: 10.1109/ICAC353642.2021.9697110.

[15] P. K. Jain, R. Pamula, and G. Srivastava, ‘‘A systematic literature review on machine learning applications for consumer sentiment analysis using online reviews,’’ Comput. Sci. Rev., vol. 41, no. 1, 2021, Art. no. 104926.

[16] Paul, Chayan & Bora, Pronami. Detecting Hate Speech Using Deep Learning Techniques. International Journal of Advanced Computer Science and Applications. 12. 10.14569/IJACSA.2021.0120278. (2021)

[17] Farooqi, Z. M., Ghosh, S., & Shah, R. R. (2021). Leveraging Transformers for Hate Speech Detection in Conversational Code-Mixed Tweets. arXiv preprint arXiv:2112.09986.

[18] Pamungkas EW, Basile V, Patti V. A joint learning approach with knowledge injection for zero-shot cross-lingual hate speech detection. Information Processing & Management. 2021 Jul 1;58(4):102544.

[19] Fazil, M., Sah, A.K., Abulaish, M., 2021. Deepsbd: A deep neural network model with attention mechanism for socialbot detection. IEEE Trans. Inf. Forensics Secur. 16 (8), 4211–4223.

[20]Fortuna, P., Soler-Company, J., Wanner, L., 2021. How well do hate speech, toxicity, abusive and offensive language classification models generalize across datasets? Inf. Process. Manage. 58 (3), 1–17.

[21] P. William, P. Kumar, G. S. Chhabra and K. Vengatesan, "Task Allocation in Distributed Agile Software Development using Machine Learning Approach," 2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON), 2021, pp. 168-172, doi: 10.1109/CENTCON52345.2021.9688114.

[22] Tontodimamma, A., E. Nissi, A. Sarra, and L. Fontanella. 2021. "Thirty Years of Research into Hate Speech: Topics of Interest and Their Evolution." Scientometrics 126: 157– 179.

[23] Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8, 53 (2021). https://doi.org/10.1186/s40537-021-00444-8

[24] Kovács, G., Alonso, P. & Saini, R. Challenges of Hate Speech Detection in Social Media. SN COMPUT. SCI. 2, 95 (2021). https://doi.org/10.1007/s42979-021-00457-3.

[25] V. -I. Ilie, C. -O. Truică, E. -S. Apostol and A. Paschke, "Context-Aware Misinformation Detection: A Benchmark of Deep Learning Architectures Using Word Embeddings," in IEEE Access, vol. 9, pp. 162122-162146, 2021, doi: 10.1109/ACCESS.2021.3132502.

[26] Mozafari, Marzieh, Reza Farahbakhsh and Noël Crespi. "Hate speech detection and racial bias mitigation in social media based on BERT model." PLoS ONE 15 (2020): n. pag.

[27] Z. Waseem and D. Hovy, ''Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter,'' in Proc. NAACL-HLT, Sacramento, CA, USA, 2020, pp. 88–93.

[28] A. R. Gover, S. B. Harper, and L. Langton, ''Anti-Asian hate crime during the COVID-19 pandemic: Exploring the reproduction of inequality,'' Amer. J. Criminal Justice, vol. 45, no. 7, pp. 647–667, 2020.

[29] Röttger P, Vidgen B, Nguyen D, Waseem Z, Margetts H, Pierrehumbert JB. HateCheck: Functional tests for hate speech detection models. arXiv preprint arXiv:2012.15606. 2020 Dec 31.

[30] P. K. Roy, A. K. Tripathy, T. K. Das and X. -Z. Gao, "A Framework for Hate Speech Detection Using Deep Convolutional Neural Network," in IEEE Access, vol. 8, pp. 204951-204962, 2020, doi: 10.1109/ACCESS.2020.3037073.

[31] T. Davidson, D. Warmsley, M. Macy, and I. Weber, ''Automated hate speech detection and the problem of offensive language,'' in Proc. 11th Int. AAAI Conf. Web Social Media, Montréal, QC, Canada, May 2017, pp. 512–515.

[32] P. Badjatiya, S. Gupta, M. Gupta and V. Varma ”Deep Learning for Hate Speech Detection in Tweets” of the 26th International Conference on World Wide Web Companion -WWW ’17 Companion Processing - 2017

[33] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2016). LSTM: A search space odyssey. IEEE transactions on neural networks and learning systems, 28(10), 2222-2232.

[34]“AndreaPerlato,” CNN and Softmax - Andrea Perlato. https://www.andreaperlato.com/aipost/cnn-and-softmax/ (accessed Nov. 28, 2022).

[35]“Random Forest Algorithm,” www.javatpoint.com. https:// www.javatpoint.com /machine-learning-random-forest-algorithm (accessed Nov. 28, 2022)

[36] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735.

# LIST OF PUBLICATIONS

1. Shivam Sharma, Prof. Shailender Kumar " Hate Speech Detection Using Deep Learning", Accepted at the " **3rd International Conference on Advanced Computing Technologies and Applications-2023 (ICACTA)** "

Indexed by Scopus

Paper Id: 245

2. Shivam Sharma, Prof. Shailender Kumar " Hate Speech Detection Using Transformers ", Accepted at " **5th International Conference on Advances in Computing, Communication Control and Networking- ICAC3N.** "

Indexed by Scopus

Paper Id: 643

Gmail **Shivam Sharma <shivamhc005@gmail.com>**

# Acceptance of Paper 245 for ICACTA 2023

2 messages

**Microsoft CMT** <email@msr-cmt.org> Mon, May 22, 2023 at 2:18 PM
Reply-To: Kiran Bhowmick <kiran.bhowmick@djsce.ac.in>
To: Shivam Sharma <shivamhc005@gmail.com>
Cc: kiran.bhowmick@djsce.ac.in

Dear Shivam Sharma
CONGRATULATIONS!

Your manuscript with PaperID: 245 and title "Hate Speech Detection Using Transformers" submitted to 3rd International Conference on Advanced Computing Technologies and Applications-2023 (ICACTA) has been accepted.

Please complete the registration and manuscript revision process as given below:
1. Registration:
You are requested to proceed for the registration to the conference ICACTA 2023 https://djicacta.in/index.html.
The registration details can be found at https://djicacta.in/reg-details.html
Please fill the google form https://forms.gle/k7uR7iVZroDSv6UF8 to complete the registration process after payment of fees
Please note that at least one of the authors should register and present the paper in ICACTA 2023 to be considered for inclusion in IEEE Xplore.
Registration fees payment receipt will be provided after payment confirmation of the same.
Please do not repeat registration process if done already.

2. Manuscript revision:
Please read the reviewers' suggestions/comments under reviews column of your CMT account.

To track the required corrections, please insert at the beginning of your revised article a detailed response to the reviewers' suggestions/comments. Make sure you address each suggestions/comments thoroughly and methodically. You should also highlight the updated text in the revised manuscript.
Upload your revised manuscript without author information (double blind) using filename Revised_paperID.pdf on your ICACTA CMT account.
In case of any query contact us on icacta2023@djsce.edu.in

Regards,
Technical Committee
ICACTA 2023



**Shivam Sharma** <shivamhc005@gmail.com> Wed, May 24, 2023 at 5:58 PM

# Payment Complete

SENT TO

DJ **DJICACTA**
XXXX-9491

AMOUNT
**₹ 7,000.00**

BRANCH: JUHU VILE PARLE

IFSC: CBIN0281621

REMARKS:

SENT FROM

**XXXX-0638**
SA

## Payment Details

SUCCESS

MODE: Instant Pay
IMPS | CHARGES: ₹5

RECEIPT NO: BAAMORMH4234

RRN: 314911252481

DATE: 29/05/2023

Gmail **Shivam Sharma <shivamhc005@gmail.com>**

# Acceptance Notification 5th IEEE ICAC3N-23 & Registration: Paper ID 643

2 messages

**Microsoft CMT** <email@msr-cmt.org> Sun, May 21, 2023 at 9:44 PM
Reply-To: Vishnu Sharma <vishnu.sharma@galgotiacollege.edu>
To: Shivam Sharma <shivamhc005@gmail.com>

Dear Shivam Sharma,
Delhi Technological University

Greetings from ICAC3N-23 ...!!!

Congratulations....!!!!!

On behalf of the 5th ICAC3N-23 Program Committee, we are delighted to inform you that the submission of "Paper ID- 643 " titled " Hate Speech Detection Using Deep Learning " has been accepted for presentation and further publication with IEEE at the ICAC3N- 23 subject to incorporate the reviewers and editors comments in your final paper. All accepted papers will be submitted to IEEE for inclusion into conference proceedings to be published on IEEE Xplore Digital Library.

For early registration benefit please complete your registration by clicking on the following Link: https://forms.gle/8e6RzNbho7CphnYN7 on or before 31 May 2023.

Registration fee details are available @ https://icac3n.in/register.
https://drive.google.com/file/d/1RGu6i4eGUI5B07zOfgRmDRfyjOhOhiC6/view?usp=share_link

You can also pay the registration fee by the UPI. (UPI id - icac3n@ybl ) or follow the link below for QR code:
https://drive.google.com/file/d/1Ry-sF0apvy_0zUjM8lNW02lZgWAsE0YD/view?usp=sharing

You must incorporate following comments in your final paper submitted at the time of registration for consideration of publication with IEEE:

Reviewers Comments:
Title is suitable for publication
Literature review is adequate
Result are well explained
Paper is well written.

Editor Note:
1. All figures and equations in the paper must be clear. Equation and tables must be typed and should not be images.
2. Final camera ready copy must be strictly in IEEE format available on conference website www.icac3n.in.
3. Transfer of E-copyright to IEEE and Presenting paper in conference is compulsory for publication of paper in IEEE.
4. If plagiarism is found at any stage in your accepted paper, the registration will be cancelled and paper will be rejected and the authors will be responsible for any consequences. Plagiarism must be less then 20% (checked through Turnitin). However the author will be given fair and sufficient chance to reduce plagiarism.
5. Change in paper title, name of authors or affiliation of authors will not be allowed after registration of papers.
6. Violation of any of the above point may lead to rejection of your paper at any stage of publication.
7. Registration fee once paid will be non refundable.

If you have any query regarding registration process or face any problem in making online payment, you can Contact @ 8168268768 (Call) / 9467482983 (Whatsapp/UPI) or write us at icac3n23@gmail.com.

Regards:
Organizing committee

Gmail

Shivam Sharma <shivamhc005@gmail.com>

# Registration Confirmation, 5th IEEE ICAC3N-23, Paper ID 643

1 message

**Microsoft CMT** <email@msr-cmt.org> Sun, May 28, 2023 at 6:09 PM
Reply-To: Vishnu Sharma <vishnu.sharma@galgotiacollege.edu>
To: Shivam Sharma <shivamhc005@gmail.com>

Dear Shivam Sharma,
Delhi Technological University

Greetings from ICAC3N-23 ...!!! Thanks for Completing your registration...!!

Paper ID- "643 "
Paper Title- " Hate Speech Detection Using Deep Learning "

This email is to confirm that you have successfully completed your registration for your accepted paper at ICAC3N-2023. We have received your registration and payment details. Further, your submitted documents will be checked minutely and if any action will be required at your end you will be informed separately via email.

For further updated regarding conference please keep visiting conference website www.icac3n.in or for any query please write us at icac3n23@gmail.com.

Regards:
Organizing committee
ICAC3N – 2023

Note:
1. Transfer of E-copyright to IEEE and Presenting paper in conference is compulsory for publication of paper in IEEE. ( For this you will be informed separately via email well before conference)
2. If plagiarism is found at any stage in your accepted paper, the registration will be cancelled and paper will be rejected and the authors will be responsible for any consequences. Plagiarism must be less then 20% (checked through Turnitin). However, author will be given sufficient and fair chance to reduce the plagiarism.
3. Change in paper title, name of authors or affiliation of authors is not allowed now.
4. Violation of any of the above point may lead to cancellation of registration.
5. Registration fee once paid is non-refundable.

PAPER NAME

**DetectionOfHateSpeechUsingDeepLearning.pdf**

WORD COUNT

**9905 Words**

CHARACTER COUNT

**52198 Characters**

PAGE COUNT

**61 Pages**

FILE SIZE

**1.3MB**

SUBMISSION DATE

**May 25, 2023 12:28 PM GMT+5:30**

REPORT DATE

**May 25, 2023 12:29 PM GMT+5:30**

## 12% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 7% Internet database
- 4% Publications database
- Crossref database
- Crossref Posted Content database
- 10% Submitted Works database

## Excluded from Similarity Report

- Bibliographic material
- Quoted material
- Cited material
- Small Matches (Less then 8 words)