TOXIC COMMENT CLASSIFICATION

A Project Report in partial fulfillment of the degree

Bachelor of Technology

in

Computer Science & Engineering/ Electrical & Electronics Engineering

By

19K41A04C4	B. Varshith Reddy
19K41A0580	P. Gyaneshwara Rao
19K41A0575	Md Rahail Pasha
19K41A04C6	Rajesh Chaganti

Under the Guidance of
Mr. D. Ramesh

Assistant Professor, Department of CSE

Submitted to

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
S R ENGINEERING COLLEGE(A), ANANTHASAGAR, WARANGAL
(Affiliated to JNTUH, Accredited by NBA)

Nov-2022

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the Project Report entitled "Toxic comment classification" is a record of bonafide work carried out by the student(s) B. Varshith Reddy, P.Gyaneshwara Rao, Md Rahail Pasha, Rajesh Chaganti bearing Roll No(s) 19K41A04C4, 19K41A0580,19K41A0575, 19K41A04C6 during the academic year 2021-2022 in partial fulfillment of the award of the degree of ***Bachelor of Technology*** in **Computer Science/Electronics and communication Engineering** by the Jawaharlal Nehru Technological University, Hyderabad.

Supervisor **Head of the Department**

External Examiner

Table of Contents

S.NO	Content	Page No
1	Abstract	4
2	Introduction	5
3	Literature Review	7
4	Flow Chart	10
5	Data Set	11
6	Data Pre Processing	14
7	Methodology	15
8	Results	21
9	References	25

1. ABSTRACT

Online forums and social media platforms have provided individuals with the means to put forward their thoughts and freely express their opinion on various issues and incidents. In some cases, these online comments contain explicit language which may hurt the readers. Comments containing explicit language can be classified into myriad categories such as Toxic, Severe Toxic, Obscene, Threat, Insult, and Identity Hate. The threat of abuse and harassment means that many people stop expressing themselves and give up on seeking different opinions.

So, We proposed this project for identifying the toxicity in the comment. In this Project we are going to Classify the Comment Depending on the Toxicity of the Comment. In this project, we want to create a model that predicts to classify comments into different categories. Comments in social media are often abusive and insulting. Organizations often want to ensure that conversations don’t get too negative.

Hence, we are suggesting a solution for classifying toxic comments in several categories using NLP methods. We are going to use Text Vectorizer for conversion of sentences to vectors and we will use BiDirectional-Lstm model for training our Model.

2. INTRODUCTION

The origin of text classification was far back to the early '60s, but machine learning techniques were effectively realistic in the '90s (Kajla, Hooda, & Saini, 2020). For over a decade, social media and social networking have been growing in geometric progression. Today, all around the world people are expressing themselves with their opinions and also discuss among others via the media. In such a setup, it is quite observable that discussions may arise due to differences in opinion. But often these discussions take a dirty side and may result in combats over the social media platforms, these might result in offensive language termed as toxic comments that may be used from one side (Chakrabarty, 2012). Machine learning has unwrapped numerous doors for researchers in text analysis. Text classification is one of them which means a task of classifying text into different predefined classifications (Kajla et al, 2020); (Mozafari, Farahbakhsh,& Crespi, 2019. LSTMs were introduced by (Schmidhuber, & Hochreiter, 1997) to alleviate the disappearing gradient problem (Guggilla, Miller, & Gurevych, 2016). Generated hateful and toxic content by a portion of users in social media is a rising phenomenon that inspired researchers to devote considerable efforts to the challenging direction of hateful content identification. We not only need an effective automatic hate speech detection model based on advanced machine learning and natural language processing, but also an adequately large amount of annotated data to train a model (Mozafari, Farahbakhsh, & Crespi, 2019). Nowadays the Internet has become the leading platform to represent our skills. Several websites allow people to use their platform to display their skills through articles, videos, and other information in different formats. Most of the websites provide a facility for commenting on any uploaded information and there is the possibility that people can use abominable language in their comments (Kajla et al, 2020). Toxic comment classification has become a dynamic research field with many recently proposed methods van (Aken, Risch, Krestel, & Löser, 2018). These toxic comments may be threatening, obscene, insulting, or identity-based hatred. Thus, these pose the threat of abuse and harassment online. Consequently, certain individuals stop giving their views or give up seeking different opinions which results in the unhealthy and unfair discussion. As a result, different platforms find it very difficult to facilitate fair conversation and are often forced to either limit user comments or get disbanded by shutting down user comments completely. The Conversation AI team, a study group founded by Jigsaw and Google has been working on techniques for providing a healthy setting for communication.

The identification and tagging of offensive content have been heavily explored with different classical Natural Language Processing (NLP) and Machine Learning techniques. However, considering the constraints associated with the natural language such as word spell variations and typos, contextual ambiguity, and semantic variations, the supervised machine learning technique turns out to be the most suitable technique for the task. The vast amount of data being generated from the digital platform, daily, usually aids in favor of the supervised machine learning technique. Though offensive text classification naturally tends to be in the arena of the traditional NLP method owing to the heavy reliance on the text, the traditional NLP approach fails to meet the enormous details associated with the texting/commenting styles of different users, has not proven to be scalable. It has been established that the bi-LSTM (bi directional Long Short-Term Memory) have been the state-of-the-art sequence modelling neural networks.

3.Literature Survey

In [1] Based on the identified problems in the area of distinguishing toxic comments, the solutions that are assessed in the current study involved three primary components. First, a baseline classification solution using a Naive Bayes [NB] approach was created. Subsequently; NB, as a powerful and widely recognized classifier, provided a firm benchmark for our additional work. Following this, we developed a classification solution framed around an LSTM/RNN algorithm. Since challenges with computational expense of a finely tuned LSTM/RNN solution was expected, the solution was scaled with an Elastic Compute Cloud[EC2] component. The AWS Sage maker platform provided access to GPU capability which enables the full capability of Deep Neural Networks, such as an LSTM/RNN model

In [2] Much work on the toxic comments detection been carried out regarding different data sources. For example, Prabowo and colleagues evaluated Naive Bayes (NB),Support Vector Machine (SVM), and Random Forest Decision Tree (RFDT) algorithms for detecting hate speech and abusive language on Indonesian Twitter [34]. The experimental results demonstrated an accuracy of 68.43% for the hierarchical approach with word uni-gram features and the SVM model. In the paper [15], Foundational. proposed a deep GRU-based neural network with pre-trained GloVe embeddings for toxic texts classification. The developed model achieved high performance across five abusive texts datasets, with the AUC value to ranged from 92% to 98%.

In [3] Haralabopoulosetal (2020) worked on Ensemble Deep Learning for Multilabel Binary Classification of comments. Ensemble learning combines the single-model outputs to improve predictions and generalization. The researchers noted that Ensemble learning improves upon three key aspects of learning, statistics, computation, and representation Ensemble methods reduce the risk of data miss representation, by combining multiple models. The researchers reduce the risk of employing a single model trained with biased data, while most learning algorithms search locally for solutions which in turn confines the optimal solution, ensemble methods can execute random seed searches with variable start points with less computational resources. Wikipedia dataset is used. The approach weighted ensemble outperformed the baseline stacked ensemble in 75% of cases by 1.5% to 5.

In [4] Sentiment classification regarding toxicity has been intensively researched in the past few years, largely in the context of social media data where researchers have applied various machine learning systems to try and tackle the problem of toxicity as well as the related, more well-known, task of sentiment analysis. Comment abuse classification research initially began with Yin etal's application of combining TF-IDF with sentiment/contextual features. They compared the performance of this model with a simple TF-IDF model and reported a 6% increase in Fl score of the classifier on chat style datasets (Kongregate, MySpace)[l]. We discuss further related works specific to our approaches below.

In [5] Recent work based on this particular topic is done by the group who created the dataset used in the Kaggle challenge [1]. They used binary identification of toxic comments without fine-grained classification where they applied simple n-gram NLP method and suggested future work on complex methods like LSTM. Since then, there have been number of different attempts by incorporating CNN,RNN with word level embedding. With the release of BERT by google in 2018,this particular pre trained model has also been used to take on this problem set. For this project, the approach taken was to observe how the performance could be improved by the application of attention mechanism in bi-directional LSTM.

In [6] Veral studies have formerly investigated hate speech using neural network techniques; Badjatiya et al., used extensive experiments with multiple deep learning architectures to learn semantic word embedding to handle toxic comments identification [13]. In another study, sentiment analysis model of YouTube video comments, using a deep neural network was proposed that leaded to 70-80%accuracy [14]. Furthermore, Farag, El-Seoud [18]reported that extensive numbers of literature have shown that supervised learning techniques have been the most frequently used methods for cyber-bullying detection. Nevertheless, other non-supervised techniques and methods have recognized to be operative on cyber-bullying recognition. Also, Karlekar and Bansal[19] reported an increased number of personal sexual harassment and abuse that are shared and posted online. In this study, authors presented the task of automatically categorizing and analyzing various forms of sexual harassment, based on stories shared on the online forum SafeCity and used labeling levels of groping, ogling, and commenting; their results indicated that single-label CNN-RNN model achieves an accuracy of 86.5.

S.No	PAPER NAME	PUBLISHED YEAR	MODELS USED	BEST MODEL ACCURACY
1	Machine learning methods for toxic comment classification : a systematic review	2020	Logistic Regression	92.46%
2	Toxic Comment Classification	2019	OneVsRest classifier	97.53%
3	Detecting and classifying Toxic Comments	2021	MLP,LSTM,CNN,SVM	82.03%
4	Toxic comment classification in Russian.	2020	Multilingual Universal Sentence Encoder(USE), M-BERT	92.04%
5	Identifying Aggression and toxicity in comments using capsule network	2022	LSTM using max-pooling	89.46%
6	Prediction of toxicity – generating news using machine learning.	2020	SVM, naïve bayes, logistic regression, GBDT	74.2%
7	Toxic comment classification	2020	LSTM/RNN , naïve bayes	81%
8	Online hate classifier for multiple social media platforms	2020	Naïve bayes,svm, XGBoost	94.3%
9	Toxic comment classification using neural networks and machine learning	2018	LSTM,CNN,RNN	90.8%
10	Toxic Comment Detection in online Discussions	2020	LSTM with gradient boosted decision trees	93%

Table-1:Literature survey on various journal papers

3-FLOWCHART

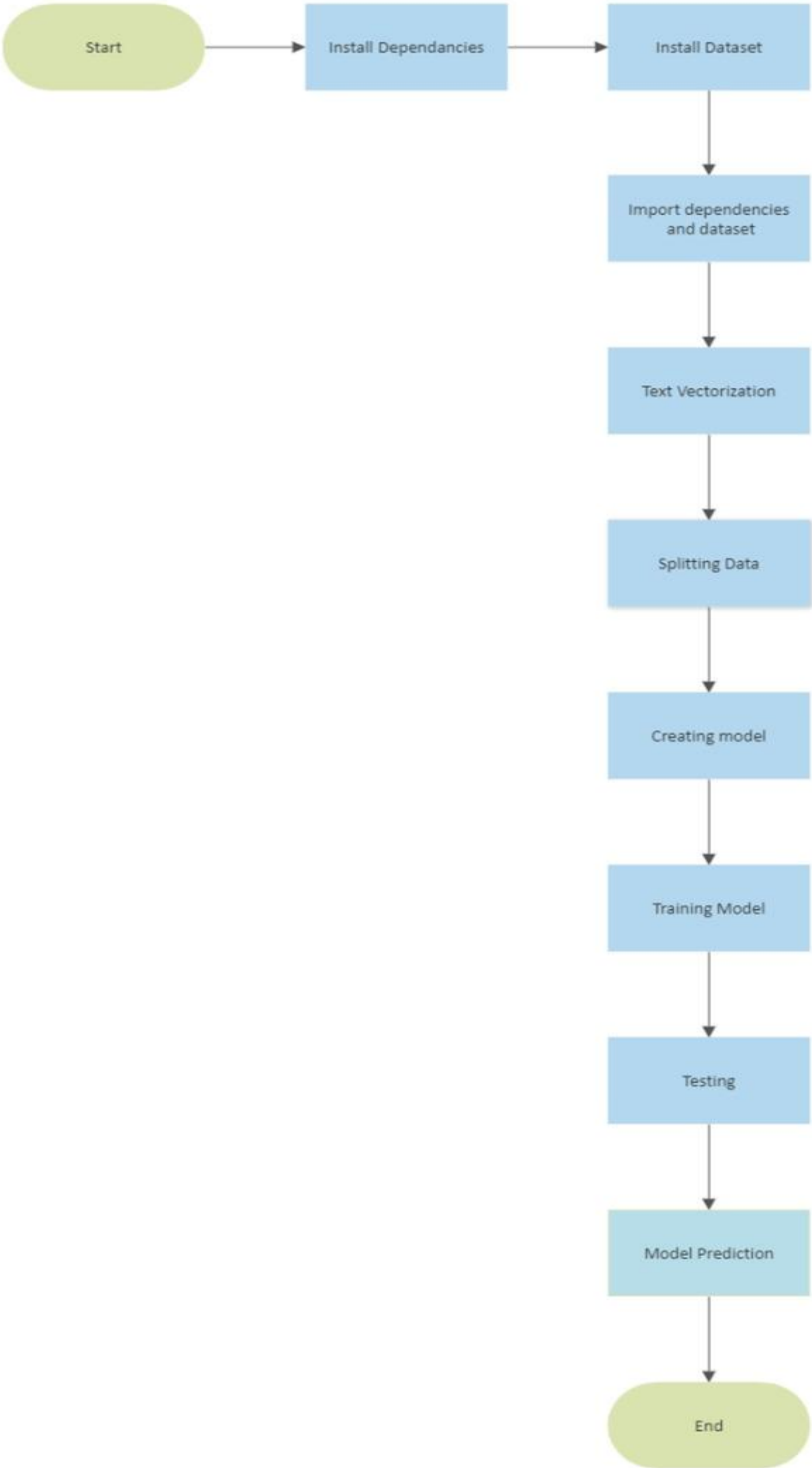


Figure 3.1 - Flow chart of the technique**.**

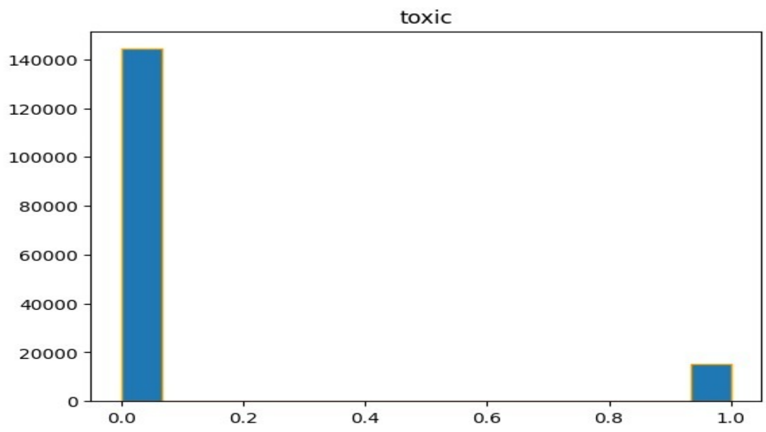
4.DATASET:

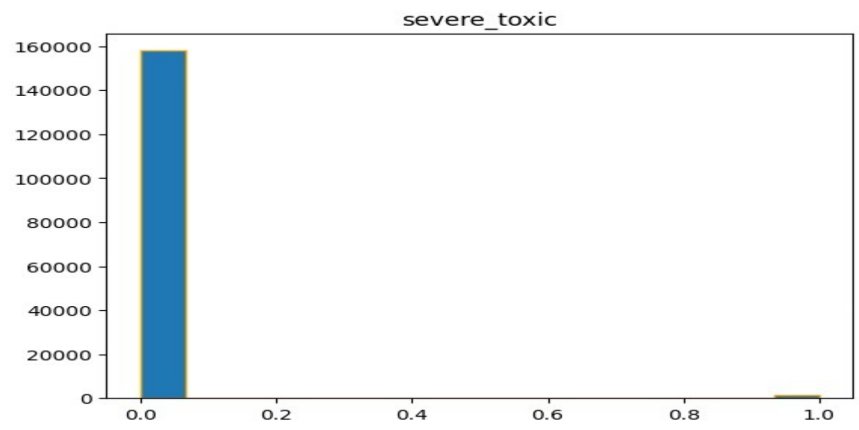
We have downloaded the data set from a open source website called Kaggel. The data set name is called as jigsaw-toxic-comment-classification-challenge. It consists of a file known as train.csv file. This file consists of 150000 of rows and 6 columns. The columns are : 'toxic', 'severe_toxic', 'obscene', 'threat', 'insult','identity_hate'.

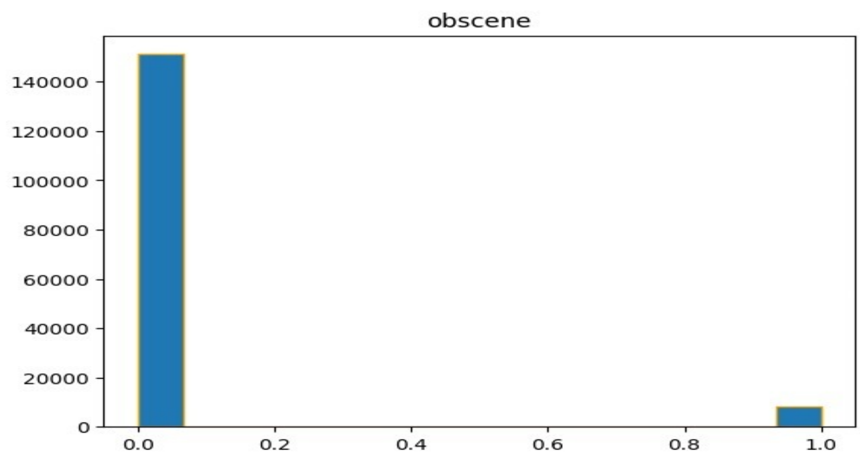
Figure-4.1: Visualizing the attributes of the dataset

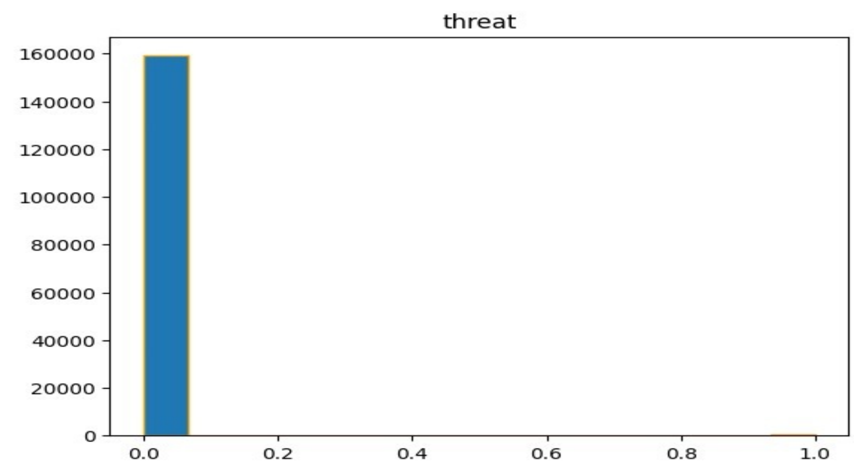
	A	B	C	D	E	F	G	H
1	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
2	0000997932d777bf	Explanation	0	0	0	0	0	0
3	000103f0d9cfb60f	D'aww! He matches t	0	0	0	0	0	0
4	000113f07ec002fd	Hey man, I'm really no	0	0	0	0	0	0
5	0001b41b1c6bb37e	"	0	0	0	0	0	0
6	0001d958c54c6e35	You, sir, are my hero.	0	0	0	0	0	0
7	00025465d4725e87	"	0	0	0	0	0	0
8	0002bcb3da6cb337	COCKSUCKER BEFORE	1	1	1	0	1	0
9	00031b1e95af7921	Your vandalism to the	0	0	0	0	0	0
10	00037261f536c51d	Sorry if the word 'non	0	0	0	0	0	0
11	00040093b2687caa	alignment on this subj	0	0	0	0	0	0
12	0005300084f90edc	"	0	0	0	0	0	0
13	00054a5e18b50dd4	bbq	0	0	0	0	0	0
14	0005c987bdfc9d4b	Hey... what is it..	1	0	0	0	0	0
15	0006f16e4e9f292e	Before you start	0	0	0	0	0	0
16	00070ef96486d6f9	Oh, and the girl above	0	0	0	0	0	0
17	00078f8ce7eb276d	"	0	0	0	0	0	0
18	0007e25b2121310b	Bye!	1	0	0	0	0	0
19	000897889268bc93	REDIRECT Talk:Voyda	0	0	0	0	0	0
20	0009801bd85e5806	The Mitsurugi point m	0	0	0	0	0	0
21	0009eaea3325de8c	Don't mean to	0	0	0	0	0	0
22	000b08c464718505	"	0	0	0	0	0	0
23	000bfd0867774845	"	0	0	0	0	0	0
24	000c0dfd995809fa	"	0	0	0	0	0	0
25	000c6a3f0cd3ba8e	"	0	0	0	0	0	0
26	000cfee90f50d471	"	0	0	0	0	0	0

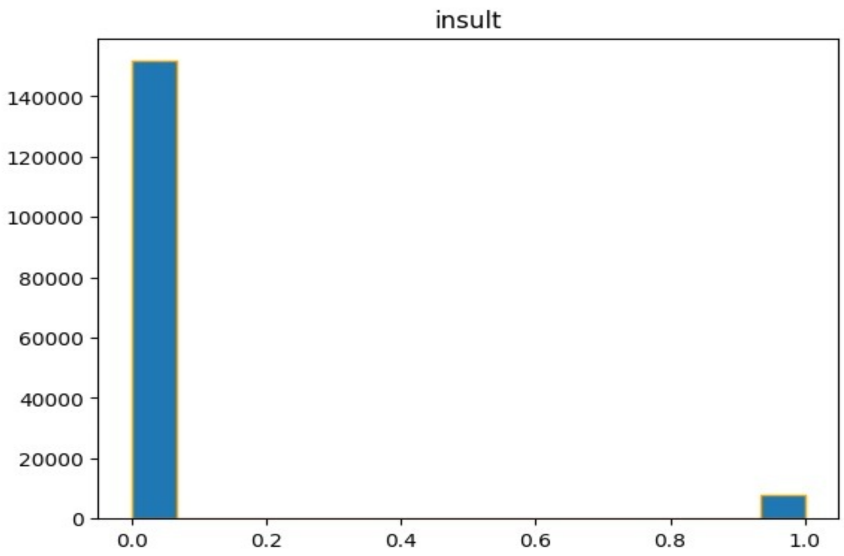
DATA VISUALIZATIONS:

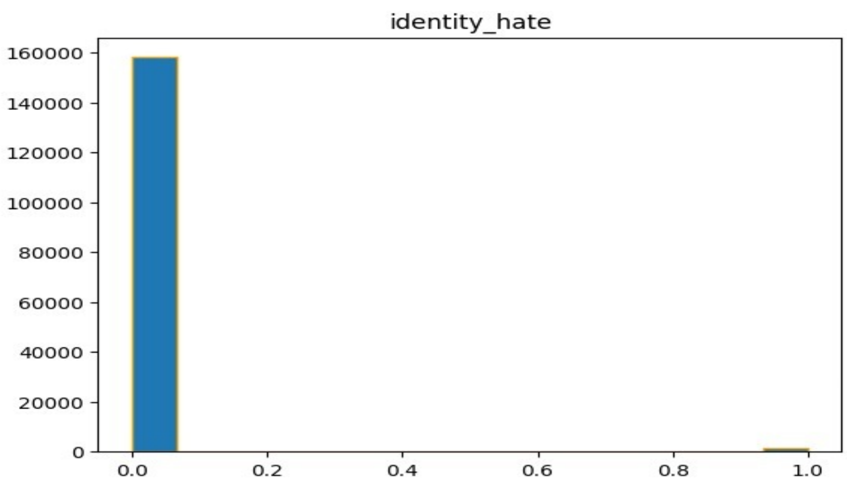



severe_toxic
160000
140000
120000
100000
80000
60000
40000
20000
0
0.0
0.2
0.4
0.6
0.8
1.0


obscene
140000
120000
100000
80000
60000
40000
20000
0
0.0
0.2
0.4
0.6
0.8
1.0


threat
160000
140000
120000
100000
80000
60000
40000
20000
0
0.0
0.2
0.4
0.6
0.8
1.0


insult
140000
120000
100000
80000
60000
40000
20000
0
0.0
0.2
0.4
0.6
0.8
1.0


identity_hate
160000
140000
120000
100000
80000
60000
40000
20000
0
0.0
0.2
0.4
0.6
0.8
1.0

5. DATA PREPROCESSING:

After Importing the data set in our code we are converting the sentences to vectors by using Text Vectorizer.
We will be assigning Comment Sentences to X, toxicity of the comment to y.
Then we will be declaring the vectorizer like below :
vectorizer = TextVectorization(max_tokens=MAX_FEATURES,output_sequence_length=1800,output_mode='int')

The sequence length is the size of vector after converting the sentence using text vectorizer.

The process of text tokenization:

The processing of each example contains the following steps:

1. Standardize each example (usually lowercasing + punctuation stripping)
2. Split each example into substrings (usually words)
3. Recombine substrings into tokens (usually n-grams)
4. Index tokens (associate a unique int value with each token)
5. Transform each example using this index, either into a vector of int’s or a dense float vector.

6.METHODOLOGY:

The model that we created is based on the concept that we want to make predictions based on different features of the dataset and to check the results based on them. The features that seemed to perform better giving a conceptual reason for that are "threat", "insult", "obsense",”toxic”.. given that we want to predict the "LABEL" feature.

So our model consists of some functions that are shown under the following architecture :
Our model is sequential bi-lstm model and in that the max features are used 200000.That are number of words int the vocab. Then we have add max-features with the embeddings(output of text tokenizer) with the help of thar we have extracted the features and given it to the fully connected layer for the out put prediction.

Next we have used some drop out layers(0.4)to the model Bi-LSTM. Then we have used dropout layers and we have used RELU activation function as input activation function. then after some dense layers and drop out layers the output activation function is also “RELU” activation function and our model is ready to predict.

6.1 BI-LSTM(BI-DIRECTIONAL LONG SHORT TERM MEMORY)

Bidirectional long-short term memory(bi-lstm) is the process of making any neural network to have the sequence information in both directions backwards (future to past) or forward(past to future).

In bidirectional, our input flows in two directions, making a bi-lstm different from the regular LSTM. With the regular LSTM, we can make input flow in one direction, either backwards or forward. However, in bi-directional, we can make the input flow in both directions to preserve the future and the past information.

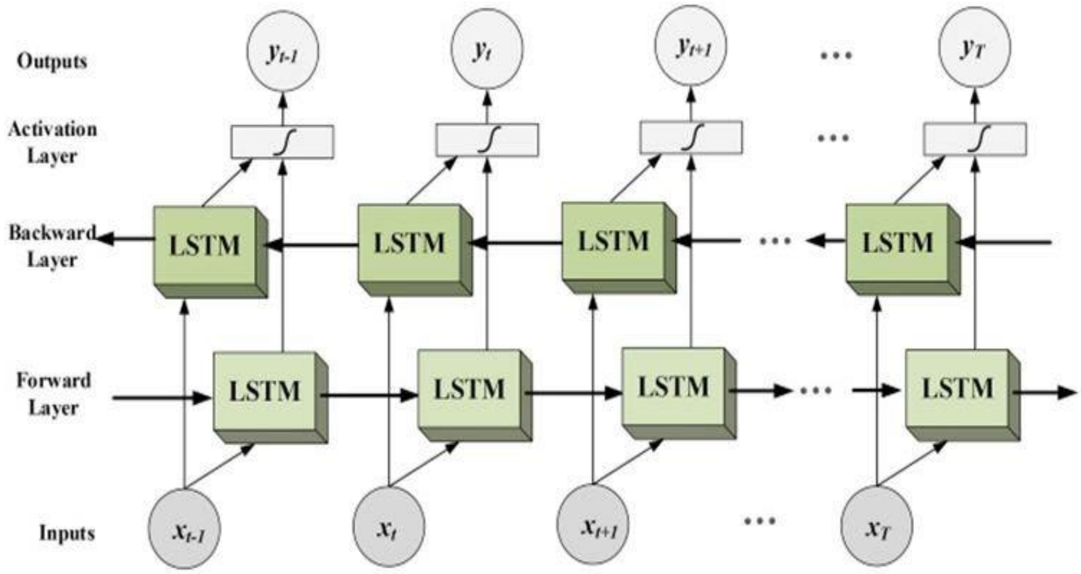


Figure-6.1 :BI LSTM Model

In the diagram, we can see the flow of information from backward and forward layers. BI-LSTM is usually employed where the sequence to sequence tasks are needed. This kind of network can be used in text classification, speech recognition and forecasting models.

This type of architecture has many advantages in real-world problems, especially in NLP. The main reason is that every component of an input sequence has information from both the past and present. For this reason, BiLSTM can produce a more meaningful output, combining LSTM layers from both directions.

LSTMs have three types of gates they are:

Input gates, forget gates, and output gates which controls the flow of information. The hidden layer output of LSTM includes the hidden state and the memory cell. Only the hidden state is passed into the output layer. The memory cell is entirely internal.

6.2 TEXT TOKENIZER:

The processing of each example contains the following steps:

1. Standardize each example (usually lowercasing + punctuation stripping)
2. Split each example into substrings (usually words)
3. Recombine substrings into tokens (usually n-grams)

4. Index tokens (associate a unique int value with each token)
5. Transform each example using this index, either into a vector of int's or a dense float Vector.

```
vectorized_text

<tf.Tensor: shape=(159571, 1800), dtype=int64, numpy=
array([[  643,    76,     2, ...,     0,     0,     0],
       [    1,    54,  2506, ...,     0,     0,     0],
       [  425,   440,    70, ...,     0,     0,     0],
       ...,
       [32141,  7329,   383, ...,     0,     0,     0],
       [    5,    12,   533, ...,     0,     0,     0],
       [    5,     8,   130, ...,     0,     0,     0]], dtype=int64)>

vectorized_text[1]

<tf.Tensor: shape=(1800,), dtype=int64, numpy=array([   1,   54, 2506, ...,    0,    0,    0], dtype=int64)>
```

Fig 6.2 : Showing the Showing the out put vectorized text

6.3 ACTIVATTION FUNCTIONS :

We have selected adam optimizer for our model because,The results of the Adam optimizer are generally better than every other optimization algorithms, have faster computation time, and require fewer parameters for tuning. So, we have selected the Adam optimizer.

6.3.1 Adam Optimizer :

Adam may be thought of as a cross between *RMSprop and Stochastic Gradient Descent with momentum*. It scales the learning rate using squared gradients, similar to RMSprop, and it makes use of momentum by utilizing a moving average of the gradient rather than the gradient itself, similar to SGD with momentum.

Adam is an adaptive learning rate approach that calculates individual learning rates for various parameters. Adam employs estimations of the first and second moments of the gradient to change the learning rate for each weight of the neural network, thus the name adaptive moment estimation.N-th moment of a random variable is defined as the expected value of that variable to the power of n.

More formally:

$$m_n = E[X^n]$$

m — moment, X -random variable.

It might be tough to grasp that concept for the first time, so if you don't get it completely, keep reading; you'll eventually understand how algorithms function. Because it is normally assessed on a tiny random batch of data, the gradient of the cost function of a neural network can be regarded as a random variable. The first is mean, while the second is uncentered variance (i.e., we don't remove the mean when calculating variance). We'll look at how we utilize these values later; for now, we must decide how to obtain them. Adam calculates exponentially moving averages based on the gradient of a current mini-batch to estimate the moments.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

Moving averages of gradient and squared gradient.

Where m and v are moving averages, g is the gradient on the current mini-batch, and betas are the algorithm's newly introduced hyper-parameters. They both have excellent default values of 0.9 and 0.999. These settings are rarely changed. The first iteration of moving averages starts with zeros in the vectors. Let's look at the predicted values of our moving averages to see how they relate to the instant described in the first equation. We want m and v to have the following attribute since they are estimates of first and second moments:

$$E[m_t] = E[g_t]$$
$$E[v_t] = E[g_t^2]$$

The estimators' anticipated values should match the parameter we're trying to estimate; in our instance, the parameter is also the expected value. If these properties were true, then we would have unbiased estimators. (See Ian Goodfellow's Deep Learning book, Chapter 5 on machine learning foundations, for further information on statistical features of alternative estimators.) We can now observe that this is not the case for our moving averages. The estimators are skewed towards zero since the averages are started with zeros. Let us demonstrate this for m. (the proof for v would be analogous). To demonstrate this, we must apply the formula to the very first gradient. Let's try unrolling a number of m values to see what pattern we'll use:

$$m_0 = 0$$
$$m_1 = \beta_1 m_0 + (1-\beta_1) g_1 = (1-\beta_1) g_1$$
$$m_2 = \beta_1 m_1 + (1-\beta_1) g_2 = \beta_1 (1-\beta_1) g_1 + (1-\beta_1) g_2$$
$$m_3 = \beta_1 m_2 + (1-\beta_1) g_3 = \beta_1^2 (1-\beta_1) g_1 + \beta_1 (1-\beta_1) g_2 + (1-\beta_1) g_3$$

As you can see, the higher the value of m, the less the first values of gradients contribute to the total value, as they are multiplied by smaller and smaller beta. We may revise the calculation for our moving average by capturing this pattern. Now let's look at the predicted value of m to see how it corresponds to the genuine first instance so that we can account for the difference:

$$E[m_t] = E[(1-\beta_1) \sum_{i=1}^{t} \beta_1^{t-i} g_i]$$
$$= E[g_i](1-\beta_1) \sum_{i=1}^{t} \beta_1^{t-i} + \zeta$$
$$= E[g_i](1-\beta_1^t) + \zeta$$

Bias correction for the first momentum estimator

To enlarge m in the first row, we utilize our modified moving average algorithm. Then we use g[t] to estimate g[i]. We may now remove it from the sum because it is no longer dependent on i. The mistake C appears in the formula as a result of the approximation. We just apply the formula for the sum of a finite geometric series in the last line. We should take two things away from that equation.

1. Our estimate is biased. This isn't only true for Adam; it's also true for algorithms that use moving averages (SGD with momentum, RMSprop, etc.).

2. It won't have much of an effect unless you start the training at the beginning, because the value beta to the power of t is rapidly approaching zero. Now we must update the estimator such that the expected value matches the desired value. Bias correction is the name given to this step. Our estimator's final formulae will be as follows:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

Bias corrected estimators for the first and second moments

The only thing left to do is to use those moving averages to scale learning rate individually for each parameter. The way it’s done in Adam is very simple, to perform weight update we do the following:

$$w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Where w is model weights, eta (look like the letter n) is the step size (it can depend on iteration).
And that’s it, that’s the update rule for Adam. For some people it can be easier to understand such
concepts in code, so here’s possible implementation of Adam in python:

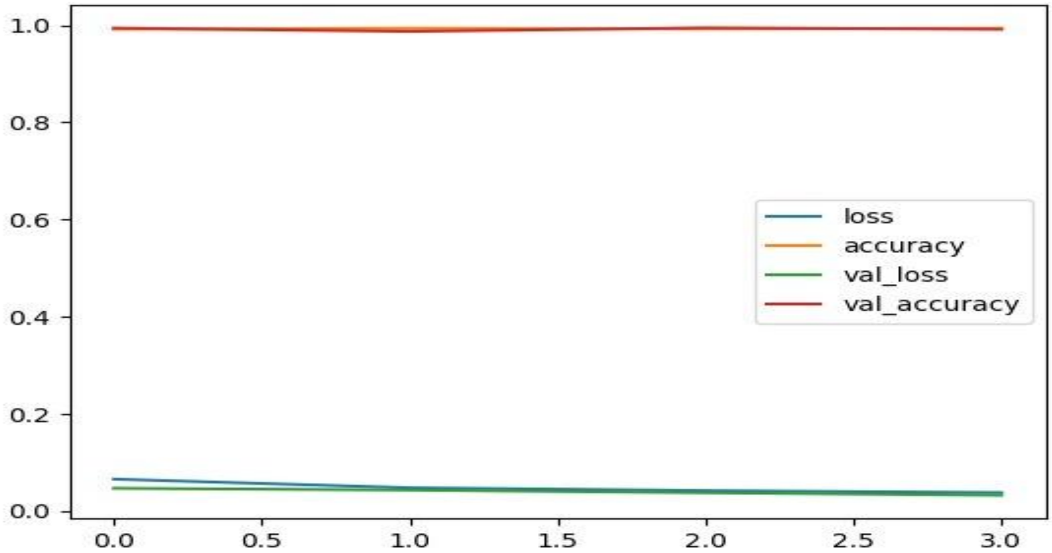
```
for t in range(num_iterations):
g = compute_gradient(x, y)
m = beta_1 * m + (1 - beta_1) * g
v = beta_2 * v + (1 - beta_2) * np.power(g, 2)
m_hat = m / (1 - np.power(beta_1, t))
v_hat = v / (1 - np.power(beta_2, t))
w = w - step_size * m_hat / (np.sqrt(v_hat) + epsilon)
```

Since, Adam uses both RMS Prop and Momentum we will be getting better accuracy than *Stochastic Gradient Descent with momentum*.

7.RESULTS

The results are shown Below :

Graphical representation of loss,accuracy,val_loss,val_accuracy :



The Below figure shows the declaration of our model :

```
model.summary()

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

 bidirectional (Bidirectiona  (None, 64)               16640
 l)

 dense (Dense)               (None, 128)               8320

 dense_1 (Dense)             (None, 256)               33024

 dense_2 (Dense)             (None, 128)               32896

 dense_3 (Dense)             (None, 6)                 774

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

Figure 7.1: The summary of our model.

So, After declaration of the model we need to train the model.We are training our model on four epochs.

The Below figure shows the training of our model on the data set :

```
history = model.fit(train, epochs=4, validation_data=val)

Epoch 1/4
6981/6981 [==============================] - 5119s 732ms/step - loss: 0.0650 - accuracy: 0.9919 - val_loss: 0.0462 - val_accuracy: 0.9941
Epoch 2/4
6981/6981 [==============================] - 4399s 630ms/step - loss: 0.0471 - accuracy: 0.9938 - val_loss: 0.0424 - val_accuracy: 0.9873
Epoch 3/4
6981/6981 [==============================] - 4394s 629ms/step - loss: 0.0413 - accuracy: 0.9923 - val_loss: 0.0369 - val_accuracy: 0.9946
Epoch 4/4
6981/6981 [==============================] - 4532s 649ms/step - loss: 0.0371 - accuracy: 0.9937 - val_loss: 0.0318 - val_accuracy: 0.9917
```

Figure-7.2: output of model with respect to accuracy and epocs.

Hence after training our model we had saved our model in a .h5 file. Here we are importing that model and testing the model on text.

```
[4]  model = tf.keras.models.load_model('toxicity.h5')
     2.9s

[26] input_text = vectorizer('Hey!! Fucking hate you')
     0.6s

[27] res = model.predict(np.array([input_text]))
     1.3s
...  1/1 [==============================] - 1s 1s/step

[28] clms
     0.4s
...  ['toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate']

[29] (res > 0.5).astype(int)
     0.4s
...  array([[1, 0, 1, 0, 1, 0]])
```

Figure 7.3: Final result analysis of the model

8.CONCLUSION

This work aimed at improving the accuracy of comment classification by using the Epoch approach on bi directional long short term memory (LSTM).

In the first section, the introduction of the whole work is presented.

Section two presents related works of literature.

The third section provides the methodologies of the proposed system.

In the fourth section, the experimental design is described, while the result and discussion of the system are presented in the fifth section.

By using tokenizer we can convert the given text into vector form and using the bi-LSTM model we trained the text in tokenized form with the model in 4 epochs and got an accuracy of 98%.

9.REFERENCES

[1]. Wulczyn, E., Thain, N., & Dixon, L. (2017) Ex Machina: Personal Attacks Seen at Scale.

[2]. Pennington, J., Socher, R., & Manning, C. (2014) GloVe: Global Vectors for Word Representa-
tion. Pro-ceedings of the 2014 Conference on Empirical Methods in Natural Language Processing.

[3].https://www.tensorflow.org/tutorials/text/nmt_with_attention.

[4].Mark Hsueh, Kumar Yogeeswaran, and Sanna Malinen. Leave your comment be-
low: Can biased online comments influence our own prejudicial attitudes and behaviors? Human communication research, 41(4):557–576, 2015.

[5]. Maria Koutamanis, Helen GM Vossen, and Patti M Valkenburg. Adolescents'comments in social media: Why do adolescents receive negative feedback who is most at risk? Computers in Human Behavior, 53:486–494, 2015.

[6].https://www.tensorflow.org/api_docs/python/tf/keras/layers/TextVectorization