Cyberbullying Detection using Deep Learning Models

Minor Project 1

Submitted by: -

Shubh Gupta (9919103022)

Abhinav Sharma (9919103107)

Chirag Sharma (9919103108)

Shreyash Rautela (9919103110)

Under the supervision of: -

Dr. Arti Jain

Department of CSE&IT

Jaypee Institute of Information Technology, Noida

December 2021

Acknowledgement

The success and final outcome of this project, entitled “Cyberbullying Detection using Deep Learning Models”, required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of our project. All that we have done is only due to such supervision and assistance and we would not forget to thank them.

We respect and thank Dr. Arti Jain, for providing us an opportunity to do the project work in the field of deep learning and against such an important issue of cyberbullying. We are extremely thankful to her for providing such a nice support and guidance.

We are thankful to and fortunate enough to get constant encouragement, support and guidance from our respected university and the CSE department helped us in successfully completing our project work. We hope you all like our endeavour.

Group 13

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and beliefs, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma from a university or other institute of higher learning, except where due acknowledgment has been made in the text.

Place: JIIT, Noida
Date: December 01, 2021

Name: Shubh Gupta
Enrolment No.: 9919103022
Name: Abhinav Sharma
Enrolment No.: 9919103107
Name: Chirag Sharma
Enrolment No.: 9919103108
Name: Shreyash Rautela
Enrolment No.:9919103110

CERTIFICATE

This is to certify that the work titled “Cyberbullying Detection using Deep Learning Models” submitted by Name of Students of B.Tech of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of any other degree or diploma.

Digital Signature of Supervisor
Dr. Arti Jain
Assistant Professor (Senior Grade)
December 01, 2021

Table of Contents

Contents Page No.

Abstract…… 6
List of Figures…….. 7
Chapter 1: INTRODUCTION…… 8
Chapter 2: BACKGROUND STUDY…… 10
1)Deep Learning……..…… 10
Chapter 3: REQUIREMENT ANALYSIS…… 11
Chapter 4: DETAILED DESIGN…….. 12
1)Long Short Term Memory(LSTM)…….. 12
2)Bidirectional Long Short Term Memory (BLSTM)…… 13
3)Recurrent Neural Network…… 13
4)Gated recurrent Unit (GRU)…… 14
Chapter 5: IMPLEMENTATION…… 16
1)Dataset…… 16
2)Pre-Processsing……... 17
Text cleaning…… 17
Tokenization…….. 17
Stop-word removal……... 17
Pre processed dataset……... 17
Preparing Train and Test dataset……... 17
Glove word Embedding…… 18
3)Implementing LSTM Model…….. 19
4)Implementing BLSTM Model……... 20
5)Implementing GRU Model…… 21
Chapter 6: EXPERIMENTAL RESULTS AND ANALYSIS…… 22
Chapter 7: CONCLUSION AND FUTURE SCOPE……... 28
Chapter 8: REFERENCES…… 29

Abstract

Cyberbullying is worrying and disturbing on-line misconduct. It seems in varied forms and is typically during a matter format in most social networks. Intelligent systems square measure necessary for automatic detection of those incidents. a number of the recent experiments have tackled this issue with ancient machine learning models. Most of the models are applied to 1 social network at a time. the most recent analysis has seen completely different models supported deep learning algorithms build a control on the detection of cyberbullying. These detection mechanisms have resulted in economical identification of incidences whereas others have limitations of ordinary identification versions. This paper performs associate empirical analysis to work out the effectiveness and performance of deep learning algorithms in detection insults in Social comment. the subsequent three deep learning models were used for experimental results, namely: Bidirectional Long Short Term Memory (BLSTM), Gated Recurrent Units (GRU) and Long Short Term Model (LSTM). Information pre-processing steps were followed that enclosed text cleanup, tokenization, stemming, and removal of stop words. when performing arts information pre-processing, clean matter information is passed to deep learning algorithms for prediction. The results show that the BLSTM model achieved high accuracy and F1-measure scores as compared to LSTM, and GRU. Our results shown that deep learning models will be best against cyberbullying once directly compared with others and paves the means for future technologies which will be used to combat this serious on-line issue.

List of Figures

Figure	Title	Page
4.1	Long Short term Memory Architecture	13
4.2	Working Model of RNN . .....…	14
4.3	Model Layout.....……	15
6.1	LSTM Accuracy Plot .....….	22
6.2	BLSTM Accuracy Plot .....…	23
6.3	GRU Accuracy Plot .....…….....…..	24
6.4	Comparison of the accuracy of all the models……..	25
6.5	Comparison of the validation accuracy of all the models……..	25
6.6	LSTM Confusion Matrix……. .....…..	26
6.7	BLSTM Confusion Matrix .....…….....…	26
6.8	GRU Confusion Matrix…….…….....….	27

Introduction

The development of information and networking technology has created open online communication channels. Unfortunately, trolls have exploited this technology for cyber-attacks and threats. Statistics show that about 18% of Europe's children were affected either through people bullying or harassing them via the Internet and mobile communication. EU Kids Online Report of 2014 stated that nearly 20% of kids who are between the ages of 11 and 16 are vulnerable to cyberbullying. Quantitative research indicates cyber-victimization rates among adolescents ranging from 20 to 40%[9]. All of this show how important it is to find an adequate, speedy, and tested approach to solving this online pandemic.

There is a need to consider and tackle cyber-bullying from various viewpoints including automatic detection and avoidance of these accidents. There are methods already developed that can mark as instances of bullying, including the engagement of services that seek to help the victims[10]. Besides, most online channels that are widely used by adolescents have safe centres, such as YouTube Safety Centre and Twitter Safety and Protection, which offer user assistance and track communications.

A jet age transformation of cyberbullying is now in force and has become very common with students seeing it as fun on cyberspace to harass their friends and enemies. The authors in [9] discussed the metamorphosis in the domain of electronics and how the influence has become negative, creating problems for the world. Furthermore, their research follows an extension of an original study intended to evaluate by experimental procedures the nature and extent of cyberbullying. They aim to add to the fact that there can be a systemic stoppage of communication by intersecting between communication and computers. Hence, providing a backdrop of which there can be continuity of their research.

In cyberspace, cyber-bullying takes place through several mediums, and it's mainly on social media, where youth and adults' access almost all the time. A cyber-bullying research group surveyed between July and October 2016. High school students and the findings show that 34% of students had encountered cyber-bullying in their lifetime[9]. Given this, automatic and valid identification of cyber-bullying is necessary to resolve such issues. Researchers have suggested that cyber-bullying comes in various ways, such as embarrassment, stalking, coercion, exploitation, or domination of a designated victim. All types can be summarized in text format when the words are explicit or implied. Explicit 11 expressions arise by using profane words with a negative emotion, where implicit expressions may come with ironic or cynical phrases that have no foul words. There has been a lot of research on explicit speech identification. Still, a lot of work is needed to solve the implicit language that makes detecting cyber-bullying on social media a challenging job.

Nevertheless, progress has been made into the detection of cyber-bullying using approaches to machine learning (ML) and deep learning (DL). However, much of the current work needs to be more developed to provide a reliable approach that incorporates clear and indirect factors into account. Analytical research aimed at evaluating the output of DL algorithms is discussed in this paper.

The contributions of this project include:

1. Text cleansing, tokenization, stemming, Lemmatization, and removal of stop words are performed as pre-processing steps.

2. Application of three deep learning models for the experimental results, namely: Bidirectional Long Short-term Memory (BLSTM), Gated Recurrent Units (GRU) and Long Short-Term Memory (LSTM).

3. Carried out an empirical analysis to determine the effectiveness and performance of deep learning algorithms in detecting insults in Social Commentary.

4. Finally, a comparison between all the DL algorithm used. Our result shows that the BLSTM model achieved high accuracy and F1-measure scores in contrast to LSTM and GRU in detecting insults in Social Commentary.

Background Study

Deep Learning

Although neural networks have existed for several years, it was not until the last decade that they have been used competitively in dealing with real word problems. Due to its processing capabilities and the advent of fast graphics. processing units. The main arguments for the use of deep learning are its proficiency to automatically identify and extract features, therefore achieving higher accuracy and a performance. In general, the hyperparameters of classifier models are also measured automatically. In contrast to deep learning machine learning features are defined and extracted either manually or by using feature selection methods[8].

In recent years, there have been two deep learning architectures that have frequently outperformed:

Recurrent Neural Networks (RNN): RNN is described as "class of neural networks whose connections between neurons form a directed cycle, which creates feedback loops within the RNN. The main function of RNN is the processing of sequential information on the basis of the internal memory captured by the directed cycles. Unlike traditional neural networks. RNN can remember the previous computation of information and can reuse it by applying it to the next element in the sequence of inputs Convolutional Neural Networks (CNN) Typically employed within areas such as computer vision and NLP CNN is identified as a feed-forward neural network its architecture is composed of convolutional and pooling or subsampling layers These layers would then provide inputs to a fully connected classification layer. Dang. Moreno-García and De la Prieta expressed that "Convolution layers filter their inputs to extract features: the outputs of multiple filters can be combined. Pooling or subsampling layers reduce the resolution of features, which can increase the CNN's robustness to noise and distortion. Fully connected layers perform classification tasks".

A study[8] discussed how Deep Learning techniques, such as Word Embeddings in addition to Recurrent Neural Networks, have shown to have a greater potential than typical machine learning methods. Within their work they applied different Deep Leaming and machine learning techniques to predict violent incidents during psychiatric admission using clinical text. Their results identified that using Deep Learning provided an improved performance in comparison to Machine learning. Ali. El Hammid and Yousif's research introduced a developed classification sentiment analysis using deep leaming networks and compared the results of different deep learning networks. The researchers found that deep learning greatly outperformed Naive Bayes and SVM.

Requirement Analysis

Before getting into, “Why this project is needed”, let’s look into some facts and stats:

- About 37% of young people between the ages of 12 and 17 have been bullied online. 30% have had it happen more than once
- Girls are more likely than boys to be both victims and perpetrators of cyber bullying. 15% of teen girls have been the target of at least four different kinds of abusive online behaviours, compared with 6% of boys.
- About half of LGBTQ+ students experience online harassment -- a rate higher than average.
- Young people who experience cyberbullying are at a greater risk than those who don’t for both self-harm and suicidal behaviours.
- Only 1 in 10 teen victims will inform a parent or trusted adult of their abuse.

Clearly, Cyber Bullying is now becoming a crime. It’s not only hampering the minds of the young but even pushing them to take steps such as suicides. Our project cuts the bullying from the source. Trained neural networks recognize whether a sentence posted on any platform is disrespectful or not and not only this we have compared three different models that can be used to fulfil the purpose. Due to limitation of time, we were not able to deploy it on any current social platform but if more time and work given it would have been a fabulous effort. Engineering is beautiful as it uses technologies like this for human welfare, growth and conserve and we hope you like and accept our work.

Detailed Design

Long short-term memory (LSTM)

RNN, which we know as a class of artificial neural networks with connections between nodes has some setbacks due to the excess number of network layers. It was a tough task, and we had to look at a recent study[1] and discovered that the LSTM network is an answer to this challenge due to its chain structure similar to that of multiple neural network modules with RNN. Figure 4.1 represents the LSTM architecture, consisting of various gates, including 1st gate is the input gate,2nd gate is the output gate, and it also has a forget gate that was used inside the LSTM model. We have used these gates in selecting how information is accepted and rejected across the network.

When the activation function range Input from – 1 to 1 with a gate i(t) consisting of tanh, it made the current input to become x(t) with attributes h(t−1) and C(t−1). Inside forget gate f (t) there is tanh and sigmoid function which is used as an activation function. It is interesting to note here that the forget gate makes the decision of the number of information to retain when it receives information from the previous output. For example, when we have a value to be 1, it means that the data was transferred to the network. But, if it is 0, it means the data will not be allowed to pass through the network. Note that the output gate o(t) also has sigmoid as another activation function with a range of -1 to 1. It shows that at any time mark, i(t), o(t), f (t) will be computed when we apply Eqs. (1), (2), and (3), respectively.

$$i^{(t)} = \sigma(W^{i}[C^{t-1}, h^{(t-1)}, x^{(t)}] + b^{i}), \quad (1)$$

$$o^{(t)} = \sigma(W^{o}[C^{t-1}, h^{(t-1)}, x^{(t)}] + b^{i}), \quad (2)$$

$$f^{(t)} = \sigma(W^{f}[C^{t-1}, h^{(t-1)}, x^{(t)}] + b^{i}), \quad (3)$$

What we have shown is different from the usual Traditional LSTM that works on bi-direction. This research we have used two LSTMs which includes; one LSTM for upward and downward scanning and the other LSTM used is for right and left scanning. We have imputed the second LSTM as a summation of the first LSTM. Therefore, our proposed LSTM utilizes double input gates, output gates and forget gates in comparison to the traditional LSTM known. This achievement gives a better accuracy. However, our proposed model has more computational complexity and cost in terms of performance.

Bidirectional long short-term memory (BLSTM)

Bidirectional LSTM (BLSTM) model retains two separate input and forwards input states provided by two different LSTMs. The first LSTM is a regular sequence starting from the starting of the paragraph, while the second LSTM is a standard sequence, the series of inputs are fed in the opposite order. The concept behind the bi-directional network is to gather knowledge about the inputs around it. It typically knows more rapidly than a one-way approach, but it depends on the mission as shown in Fig. 3 that represents the structure of used BLSTM model. BLSTM mode consists of 200 neurons, 2nd layer has 400neurons. We have 3 dense layers, 1st dense layer has 128neurons, 2nd dense layer has 64 neurons, and 3rd dense layer has 32 neurons, respectively. We also used 3 dropout layers to avoid over-fitting.

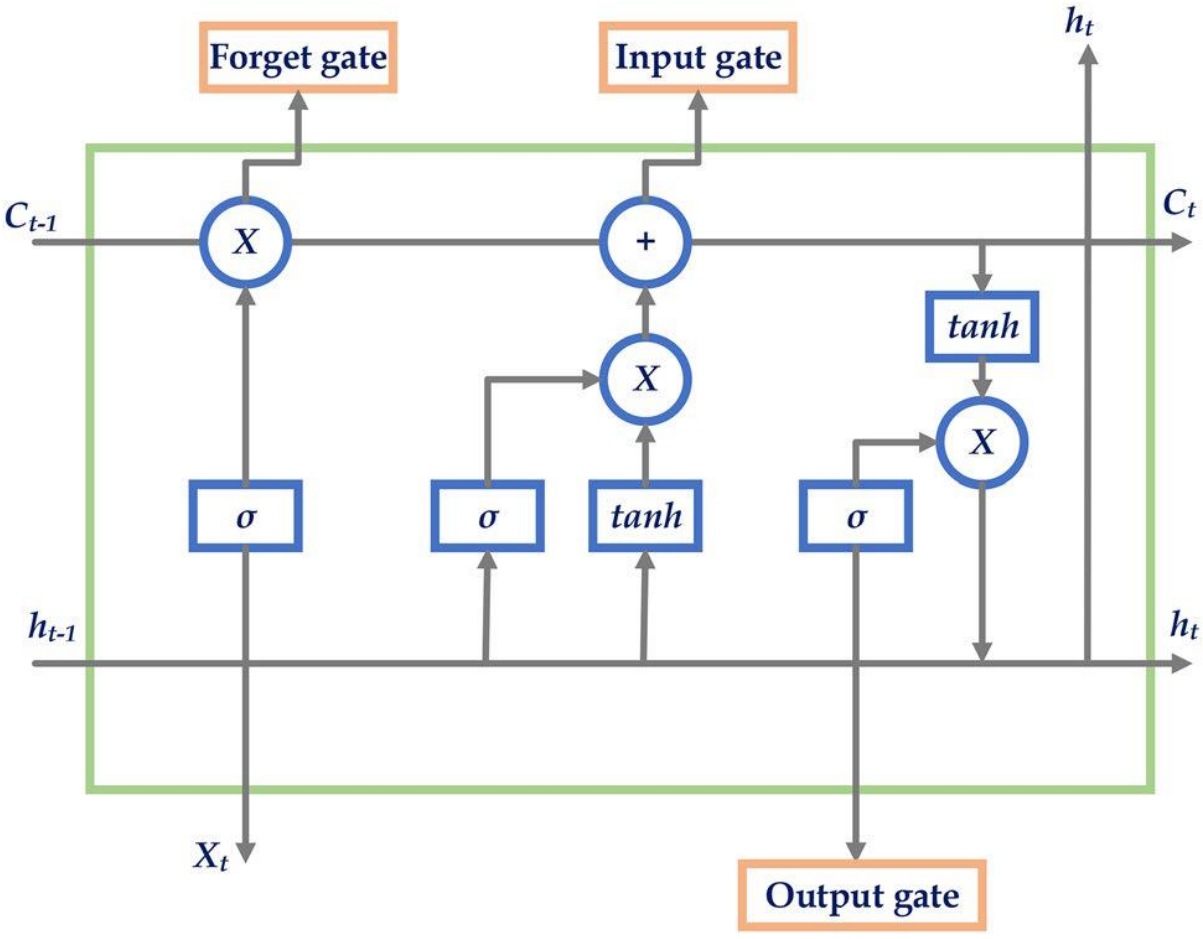


Fig. 4.1 Long short term memory architecture. [9]

Recurrent neural networks

Due to the vanishing gradient problem, conventional Neural Networks (NN) do not provide us a satisfactory result when implemented on time series data. In 1982 John Hopfield implemented RNN to address the above-mentioned subject matter. See Fig. 4.2 represents the structure of RNN model.

RNN is better with the trends of using NN learn over a timeframe. RNN was used to forecast serial data such as actions in a video based on past events, voice audio, text events, etc. Figure4 demonstrates the operating configuration for the RNN. In the figure, Xt Its weight vector stands for the hidden layer and represents the output layer weight vector; Et Is the output layer Weight Vector, Dt denotes matrix for the input word. Time stamp on the hidden layer t is measured by using Equation (9).

$$X_t = \sigma(A \times D_t + C \times X_{t-1}), \quad (9)$$

$$E_t = \sigma(B \times X_{t-1}) \quad (10)$$

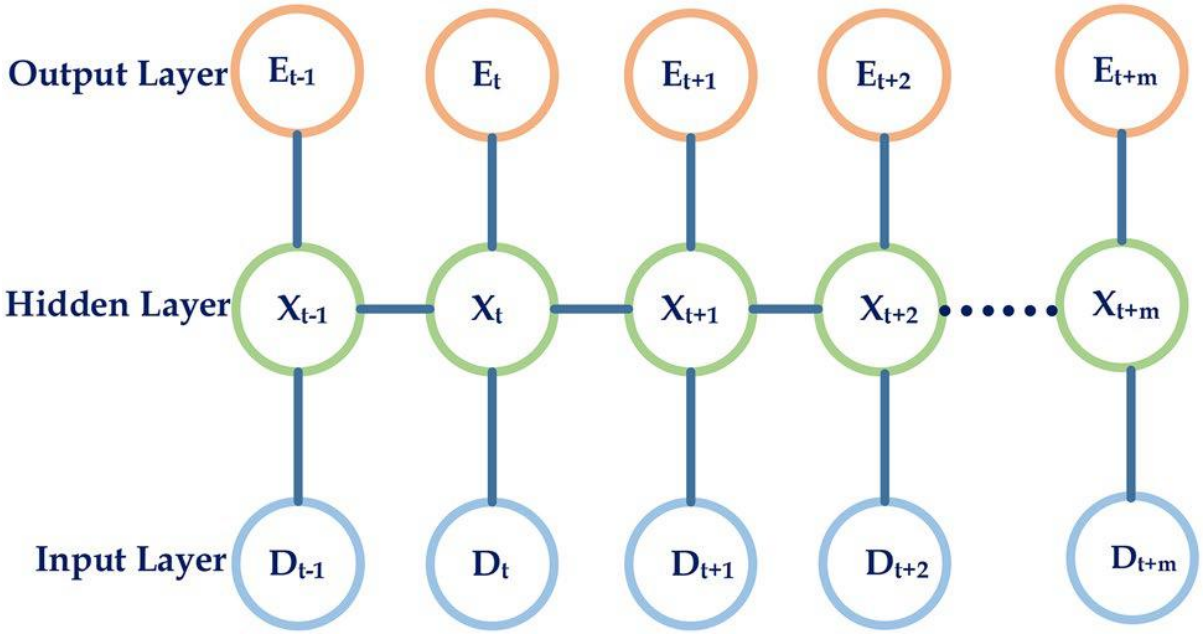


Fig. 4.2 Working model of RNN. [9]

Gated recurrent unit (GRU)

We have applied GRU as the latest variant of RNN designed to deal with short-term memory problems that are similar to LSTM. Note that GRU does not have a cell state and makes use of a hidden state to carry information. It consists of two gates: a reset gate rt and an update gate zt represented by Eqs. (11), (12). The update gate performs similar functions of the forget gate and an input gate of an LSTM with a responsibility to choose which information should be dropped or included. The reset gate determines the amount of the previous data be forgotten since GRU has fewer gates compared to LSTM, which speeds up the training process.

$$z^t = \sigma(w_z^t . x^t + U_z^t . h_{t-1} + b_z^t), \quad (11)$$

$$r^t = \sigma(w_r^t . x^t + U_r^t . h_{t-1} + b_r^t), \quad (12)$$

where zt denotes update gate, (·) represents the sigmoid function, w, U and b: parameter matrices and vector, ht denotes the output vector, xt denotes the input vector.

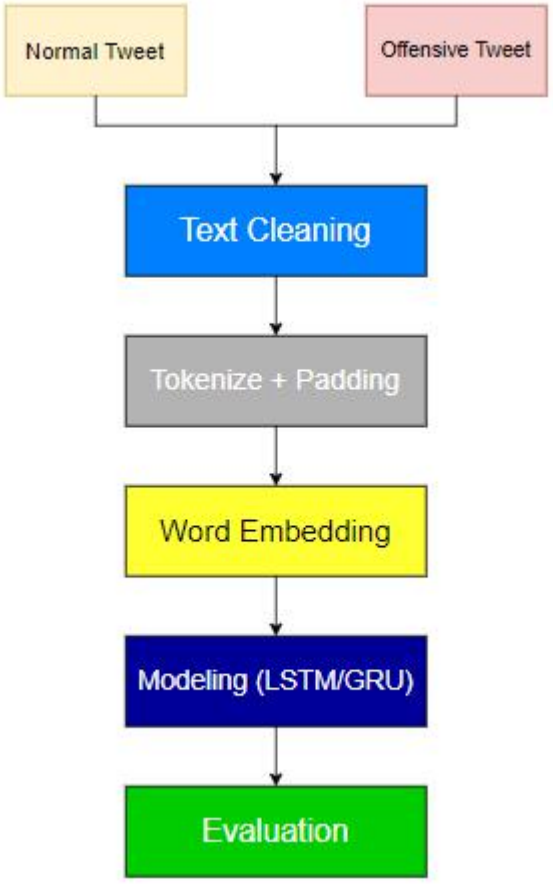


Fig. 4.3 Model Layout

Implementation

Dataset

```
import numpy as np
import pandas as pd
import seaborn as sb
import tensorflow as tf
import math
import matplotlib.pyplot as plt
import re
import string
from nltk import word_tokenize
from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score, accuracy_score, precision_score, recall_score, make_scorer
from tensorflow.keras.preprocessing import text, sequence
from tensorflow.keras.layers import LSTM,Bidirectional,GRU, Dense, Embedding, Dropout
from tensorflow.keras.models import Sequential

%matplotlib inline
file=pd.read_csv('Dataset for Detection of Cyber-Trolls.csv')
file
```

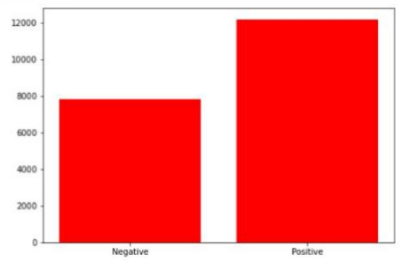
	Tweets	Label
0	Get fucking real dude.	1
1	She is as dirty as they come and that crook ...	1
2	why did you fuck it up. I could do it all day...	1
3	Dude they dont finish enclosing the fucking s...	1
4	WTF are you talking about Men? No men thats n...	1
...	...	...
19995	I dont. But what is complaining about it goi...	0
19996	Bahah yeah i&;m totally just gonna&; get pis...	0
19997	hahahahaha >:) im evil mwahahahahahahahahaha	0
19998	What&;s something unique about Ohio? :)	0
19999	Who is the biggest gossiper you know?	0

20000 rows × 2 columns

```
from collections import Counter
c= Counter(file.Label)
c
```

```
Counter({1: 7822, 0: 12179})
```

```
y= c.values()
x=["Negative","Positive"]
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.bar(x,y,color='r')
plt.show()
```



Pre-Processing

Text Cleaning

```
import re
import string
def remove_punct(text):
    text_nopunct = ''
    text_nopunct = re.sub('['+string.punctuation+']', '', str(text))
    return text_nopunct
file['Cleaned_Tweets'] = file['Tweets'].apply(lambda x: remove_punct(x))
file.head()
```

Tokenization

```
tokens = file['Cleaned_Tweets'].map(word_tokenize)
file['Tokenized'] = [' '.join(t) for t in tokens]
tokens
```

```
def lower_token(tokens):
    return [word.lower() for word in tokens]

tokens_lowered=[lower_token(t) for t in tokens]
file['tokens_lowered']=tokens_lowered
file.head()
```

Stop-words removal

```
stoplist=stopwords.words('english')
def removeStopWords(tokens):
    return [w for w in tokens if w not in stoplist]
def removenumeric(tokens):
    return [w for w in tokens if not w.isdigit()]
filtered_words=[removeStopWords(t) for t in tokens_lowered]
filtered_words=[removenumeric(t) for t in filtered_words]
file['Text_Final'] = [' '.join(t) for t in filtered_words]
file['Filtered_words'] = filtered_words
file.head(8)
```

Pre-processed dataset

	Tweets	Label	Cleaned_Tweets	Tokenized	tokens_lowered	Text_Final	Filtered_words
0	Get fucking real dude.	1	Get fucking real dude	Get fucking real dude	[get, fucking, real, dude]	get fucking real dude	[get, fucking, real, dude]
1	She is as dirty as they come and that crook ...	1	She is as dirty as they come and that crook ...	She is as dirty as they come and that crook Re...	[she, is, as, dirty, as, they, come, and, that...	dirty come crook rengel dems fucking corrupt j...	[dirty, come, crook, rengel, dems, fucking, co...
2	why did you fuck it up. I could do it all day...	1	why did you fuck it up I could do it all day ...	why did you fuck it up I could do it all day t...	[why, did, you, fuck, it, up, i, could, do, it...	fuck could day lets hour ping later sched writ...	[fuck, could, day, lets, hour, ping, later, sc...
3	Dude they dont finish enclosing the fucking s...	1	Dude they dont finish enclosing the fucking s...	Dude they dont finish enclosing the fucking sh...	[dude, they, dont, finish, enclosing, the, fuc...	dude dont finish enclosing fucking showers hat...	[dude, dont, finish, enclosing, fucking, showe...
4	WTF are you talking about Men? No men thats n...	1	WTF are you talking about Men No men thats no...	WTF are you talking about Men No men thats not...	[wtf, are, you, talking, about, men, no, men, ...	wtf talking men men thats menage thats gay	[wtf, talking, men, men, thats, menage, thats,...
5	Ill save you the trouble sister. Here comes a ...	1	Ill save you the trouble sister Here comes a b...	Ill save you the trouble sister Here comes a b...	[ill, save, you, the, trouble, sister, here, c...	ill save trouble sister comes big ol fuck fran...	[ill, save, trouble, sister, comes, big, ol, f...
6	Im dead serious.Real athletes never cheat don...	1	Im dead seriousReal athletes never cheat dont...	Im dead seriousReal athletes never cheat dont ...	[im, dead, seriousreal, athletes, never, cheat...	im dead seriousreal athletes never cheat dont ...	[im, dead, seriousreal, athletes, never, cheat...
7	...go absolutely insane.hate to be the bearer ...	1	go absolutely insanehate to be the bearer of b...	go absolutely insanehate to be the bearer of b...	[go, absolutely, insanehate, to, be, the, bear...	go absolutely insanehate bearer bad newsloldon...	[go, absolutely, insanehate, bearer, bad, news...

Preparing train and test dataset

```
xtrain, xtest, ytrain, ytest = train_test_split(file['Text_Final'], file['Label'], shuffle=True, test_size=0.2)

training_sentence_lengths = [len(tokens) for tokens in xtrain]
print("Max sentence length in train is %s" % max(training_sentence_lengths))
max_len=max(training_sentence_lengths)

test_sentence_lengths = [len(tokens) for tokens in xtest]
print("Max sentence length in test is %s" % max(test_sentence_lengths))
```

```
Max sentence length in train is 904
Max sentence length in test is 667
```

```
max_words = 10000
tokenizer = text.Tokenizer(num_words = max_words)

tokenizer.fit_on_texts(xtrain)

xtrain_seq = tokenizer.texts_to_sequences(xtrain)
xtest_seq = tokenizer.texts_to_sequences(xtest)

xtrain_pad = sequence.pad_sequences(xtrain_seq)
xtest_pad = sequence.pad_sequences(xtest_seq)
word_index = tokenizer.word_index
len(xtrain_pad[6])
```

```
142
```

Using Glove word embeddings

```
#Loading the Glove Vectors
embedding_vectors = {}
with open('glove.twitter.27B.100d.txt','r',encoding='utf-8') as glove_file:
    for row in glove_file:
        values = row.split(' ')
        word = values[0]
        weights = np.asarray([float(val) for val in values[1:]])
        embedding_vectors[word] = weights
```

```
emb_dim = 100
if max_words is not None:
    vocab_len = max_words
else:
    vocab_len = len(word_index)+1
embedding_matrix = np.zeros((vocab_len, emb_dim))
oov_words = []
for word, idx in word_index.items():
    if idx < vocab_len:
        embedding_vector = embedding_vectors.get(word)
        if embedding_vector is not None:
            embedding_matrix[idx] = embedding_vector
        else:
            oov_words.append(word)
```

```
embedding_matrix[2]
```

```
array([ 1.8435e-01,  3.4638e-01,  3.6504e-01,  6.0354e-01, -2.2940e-01,
       -1.0155e-01,  4.6474e-01,  2.8357e-01,  1.9533e-01,  6.3805e-01,
       -2.2669e-02,  2.1714e-01, -4.4768e+00, -2.5935e-01,  6.7712e-01,
       -4.0085e-02, -8.9188e-04, -3.3866e-01, -4.4386e-01, -5.1624e-01,
       -2.7588e-01,  2.2449e-01,  5.8658e-03, -8.5680e-02,  6.8822e-01,
        4.4768e-02, -4.4996e-01, -2.4830e-01, -2.6020e-01, -3.5478e-01,
       -9.5430e-01, -2.3096e-01, -4.5752e-02,  9.0701e-02, -6.2813e-01,
        5.6478e-01, -4.2195e-01,  3.6337e-01,  1.5341e-01,  1.0405e-01,
       -1.1739e+00,  1.0603e-01, -2.6939e-01, -1.8659e-01, -1.3978e-01,
        7.7009e-02,  6.1475e-01,  5.2478e-01,  4.0327e-01,  7.7527e-01,
       -4.6076e-01,  3.6190e-01, -2.5888e-01, -1.0429e-01,  1.9468e-01,
        2.4984e-01, -7.2952e-01, -1.3537e-01,  4.9564e-01, -1.4137e-01,
       -1.5075e-01, -2.2521e-01,  3.6637e-01, -1.9326e-01, -1.5961e-01,
       -2.6602e-01, -2.4458e-01, -2.2380e-02,  1.2608e-01,  2.0559e-01,
        1.2733e-01,  7.2176e-01,  5.0610e-02, -1.5285e-01, -4.2749e-01,
        3.4278e-02,  4.1792e-01, -3.8507e-02,  8.5392e-02,  1.4612e-01,
        1.9571e+00,  1.9714e-02, -5.1073e-01, -7.6858e-01, -4.5990e-01,
        2.0151e-01, -2.6207e-01, -3.8744e-01, -3.7503e-01, -1.7158e-01,
       -1.7789e-01,  9.9983e-02, -1.9365e-02,  1.4310e-01,  2.3853e-01,
        7.6340e-01, -4.3485e-01,  4.3300e-01,  3.1190e-01,  3.5100e-01])
```

Implementing LSTM model

```
#model LSTM

lstm_model = Sequential()
lstm_model.add(Embedding(vocab_len, emb_dim, trainable = False, weights=[embedding_matrix]))
lstm_model.add(LSTM(128, return_sequences=True))
lstm_model.add(Dropout(0.2))
lstm_model.add(LSTM(64, return_sequences=True))
lstm_model.add(Dropout(0.2))
lstm_model.add(LSTM(32, return_sequences=True))
lstm_model.add(Dropout(0.2))
lstm_model.add(LSTM(16, return_sequences=False))
lstm_model.add(Dropout(0.2))
lstm_model.add(Dense(32, activation='relu'))
lstm_model.add(Dropout(0.1))
lstm_model.add(Dense(1, activation = 'sigmoid'))
lstm_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(lstm_model.summary())
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 100)         1000000
_________________________________________________________________
lstm (LSTM)                  (None, None, 128)         117248
_________________________________________________________________
dropout (Dropout)            (None, None, 128)         0
_________________________________________________________________
lstm_1 (LSTM)                (None, None, 64)          49408
_________________________________________________________________
dropout_1 (Dropout)          (None, None, 64)          0
_________________________________________________________________
lstm_2 (LSTM)                (None, None, 32)          12416
_________________________________________________________________
dropout_2 (Dropout)          (None, None, 32)          0
_________________________________________________________________
lstm_3 (LSTM)                (None, 16)                3136
_________________________________________________________________
dropout_3 (Dropout)          (None, 16)                0
_________________________________________________________________
dense (Dense)                (None, 32)                544
_________________________________________________________________
dropout_4 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 33
=================================================================
Total params: 1,182,785
Trainable params: 182,785
Non-trainable params: 1,000,000
_________________________________________________________________
None
```

Implementing BLSTM model

```
#model BLSTM

blstm_model=Sequential()
blstm_model.add(Embedding(vocab_len, emb_dim, trainable = False, weights=[embedding_matrix]))
blstm_model.add(Bidirectional(LSTM(128, return_sequences=True)))
blstm_model.add(Dropout(0.2))
blstm_model.add(Bidirectional(LSTM(64, return_sequences=True)))
blstm_model.add(Dropout(0.2))
blstm_model.add(Bidirectional(LSTM(32, return_sequences=True)))
blstm_model.add(Dropout(0.2))
blstm_model.add(Bidirectional(LSTM(16, return_sequences=False)))
blstm_model.add(Dropout(0.2))
blstm_model.add(Dense(32, activation='relu'))
blstm_model.add(Dropout(0.1))
blstm_model.add(Dense(1,activation='sigmoid'))
blstm_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(blstm_model.summary())
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, None, 100)         1000000
_________________________________________________________________
bidirectional (Bidirectional (None, None, 256)         234496
_________________________________________________________________
dropout_5 (Dropout)          (None, None, 256)         0
_________________________________________________________________
bidirectional_1 (Bidirection (None, None, 128)         164352
_________________________________________________________________
dropout_6 (Dropout)          (None, None, 128)         0
_________________________________________________________________
bidirectional_2 (Bidirection (None, None, 64)          41216
_________________________________________________________________
dropout_7 (Dropout)          (None, None, 64)          0
_________________________________________________________________
bidirectional_3 (Bidirection (None, 32)                10368
_________________________________________________________________
dropout_8 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 32)                1056
_________________________________________________________________
dropout_9 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 33
=================================================================
Total params: 1,451,521
Trainable params: 451,521
Non-trainable params: 1,000,000
_________________________________________________________________
None
```

Implementing GRU model

```
#GRU Model
emb_dim = embedding_matrix.shape[1]
gru_model = Sequential()
gru_model.add(Embedding(vocab_len, emb_dim, trainable = False, weights=[embedding_matrix]))
gru_model.add(GRU(128, return_sequences=True))
gru_model.add(Dropout(0.2))
gru_model.add(GRU(64, return_sequences=True))
gru_model.add(Dropout(0.2))
gru_model.add(GRU(32, return_sequences=True))
gru_model.add(Dropout(0.2))
gru_model.add(GRU(16, return_sequences=False))
gru_model.add(Dropout(0.2))
gru_model.add(Dense(32, activation='relu'))
gru_model.add(Dropout(0.1))
gru_model.add(Dense(1, activation = 'sigmoid'))
gru_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(gru_model.summary())
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, None, 100)         1000000
_________________________________________________________________
gru (GRU)                    (None, None, 128)         88320
_________________________________________________________________
dropout_10 (Dropout)         (None, None, 128)         0
_________________________________________________________________
gru_1 (GRU)                  (None, None, 64)          37248
_________________________________________________________________
dropout_11 (Dropout)         (None, None, 64)          0
_________________________________________________________________
gru_2 (GRU)                  (None, None, 32)          9408
_________________________________________________________________
dropout_12 (Dropout)         (None, None, 32)          0
_________________________________________________________________
gru_3 (GRU)                  (None, 16)                2400
_________________________________________________________________
dropout_13 (Dropout)         (None, 16)                0
_________________________________________________________________
dense_4 (Dense)              (None, 32)                544
_________________________________________________________________
dropout_14 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 33
=================================================================
Total params: 1,137,953
Trainable params: 137,953
Non-trainable params: 1,000,000
_________________________________________________________________
None
```

Experimental Results and Analysis

After feeding all three of our models with the pre-processed dataset we get the following results:

1. LSTM model

```
batch_size = 64
epochs  = 10
lstm = lstm_model.fit(xtrain_pad, ytrain, validation_data=(xtest_pad, ytest), batch_size = batch_size, epochs = epochs)
```

```
Epoch 1/10
250/250 [==============================] - 45s 166ms/step - loss: 0.5925 - accuracy: 0.6676 - val_loss: 0.5086 - val_accuracy:
0.7240
Epoch 2/10
250/250 [==============================] - 38s 153ms/step - loss: 0.4985 - accuracy: 0.7414 - val_loss: 0.4852 - val_accuracy:
0.7505
Epoch 3/10
250/250 [==============================] - 39s 155ms/step - loss: 0.4730 - accuracy: 0.7657 - val_loss: 0.4771 - val_accuracy:
0.7540
Epoch 4/10
250/250 [==============================] - 38s 152ms/step - loss: 0.4443 - accuracy: 0.7872 - val_loss: 0.4546 - val_accuracy:
0.7645
Epoch 5/10
250/250 [==============================] - 38s 151ms/step - loss: 0.4196 - accuracy: 0.8029 - val_loss: 0.4436 - val_accuracy:
0.7835
Epoch 6/10
250/250 [==============================] - 38s 151ms/step - loss: 0.3837 - accuracy: 0.8269 - val_loss: 0.4257 - val_accuracy:
0.7995
Epoch 7/10
250/250 [==============================] - 38s 150ms/step - loss: 0.3494 - accuracy: 0.8464 - val_loss: 0.4395 - val_accuracy:
0.8062
Epoch 8/10
250/250 [==============================] - 38s 150ms/step - loss: 0.3109 - accuracy: 0.8661 - val_loss: 0.4091 - val_accuracy:
0.8102
Epoch 9/10
250/250 [==============================] - 38s 153ms/step - loss: 0.2763 - accuracy: 0.8869 - val_loss: 0.3940 - val_accuracy:
0.8310
Epoch 10/10
250/250 [==============================] - 39s 156ms/step - loss: 0.2379 - accuracy: 0.9061 - val_loss: 0.3717 - val_accuracy:
0.8562
```

```
#plot accuracy for lstm
plt.figure(figsize=(15, 7))
plt.plot(range(epochs),lstm.history['accuracy'])
plt.plot(range(epochs),lstm.history['val_accuracy'])
plt.xlabel('Number of epochs')
plt.ylabel('Accuracy')
plt.legend(['training_acc', 'validation_acc'])
plt.title('Accuracy')
```

```
Text(0.5, 1.0, 'Accuracy')
```

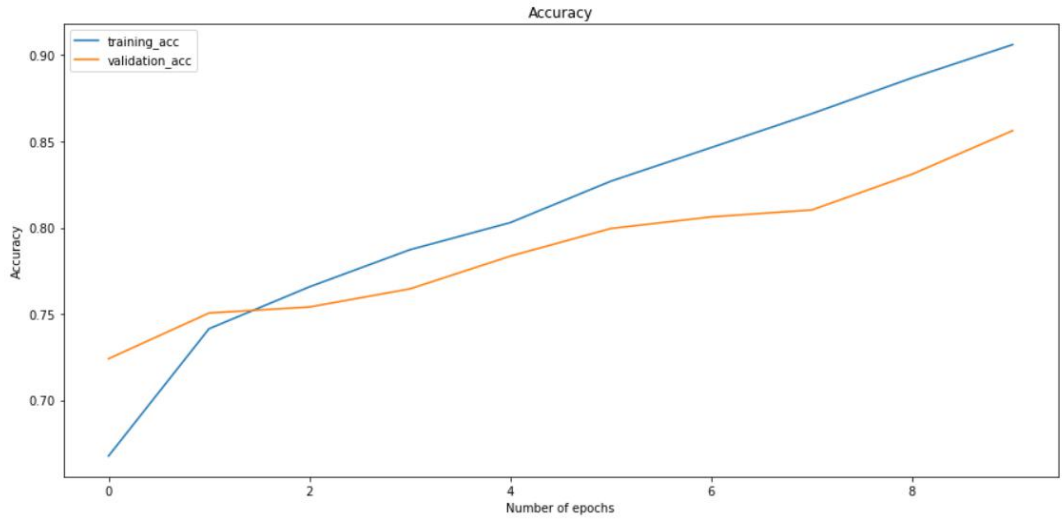


Fig. 6.1

2. BLSTM model

```
batch_size = 64
epochs=10
blstm=blstm_model.fit(xtrain_pad, ytrain, validation_data=(xtest_pad, ytest), batch_size = batch_size, epochs = epochs)
```

```
Epoch 1/10
250/250 [==============================] - 79s 316ms/step - loss: 0.2174 - accuracy: 0.9146 - val_loss: 0.3894 - val_accuracy:
0.8575
Epoch 2/10
250/250 [==============================] - 78s 314ms/step - loss: 0.1886 - accuracy: 0.9267 - val_loss: 0.3943 - val_accuracy:
0.8662
Epoch 3/10
250/250 [==============================] - 78s 313ms/step - loss: 0.1610 - accuracy: 0.9389 - val_loss: 0.4022 - val_accuracy:
0.8650
Epoch 4/10
250/250 [==============================] - 80s 319ms/step - loss: 0.1520 - accuracy: 0.9419 - val_loss: 0.3617 - val_accuracy:
0.8842
Epoch 5/10
250/250 [==============================] - 79s 318ms/step - loss: 0.1275 - accuracy: 0.9509 - val_loss: 0.4137 - val_accuracy:
0.8857
Epoch 6/10
250/250 [==============================] - 78s 312ms/step - loss: 0.1152 - accuracy: 0.9553 - val_loss: 0.3735 - val_accuracy:
0.8928
Epoch 7/10
250/250 [==============================] - 77s 306ms/step - loss: 0.1030 - accuracy: 0.9608 - val_loss: 0.4084 - val_accuracy:
0.8910
Epoch 8/10
250/250 [==============================] - 77s 307ms/step - loss: 0.0903 - accuracy: 0.9643 - val_loss: 0.4338 - val_accuracy:
0.8935
Epoch 9/10
250/250 [==============================] - 77s 306ms/step - loss: 0.0926 - accuracy: 0.9646 - val_loss: 0.4634 - val_accuracy:
0.8930
Epoch 10/10
250/250 [==============================] - 78s 311ms/step - loss: 0.0796 - accuracy: 0.9683 - val_loss: 0.5479 - val_accuracy:
0.8748
```

```
#plot accuracy for blstm
plt.figure(figsize=(15, 7))
plt.plot(range(epochs),blstm.history['accuracy'])
plt.plot(range(epochs),blstm.history['val_accuracy'])
plt.xlabel('Number of epochs')
plt.ylabel('Accuracy')
plt.legend(['training_acc', 'validation_acc'])
plt.title('Accuracy')
```

```
Text(0.5, 1.0, 'Accuracy')
```

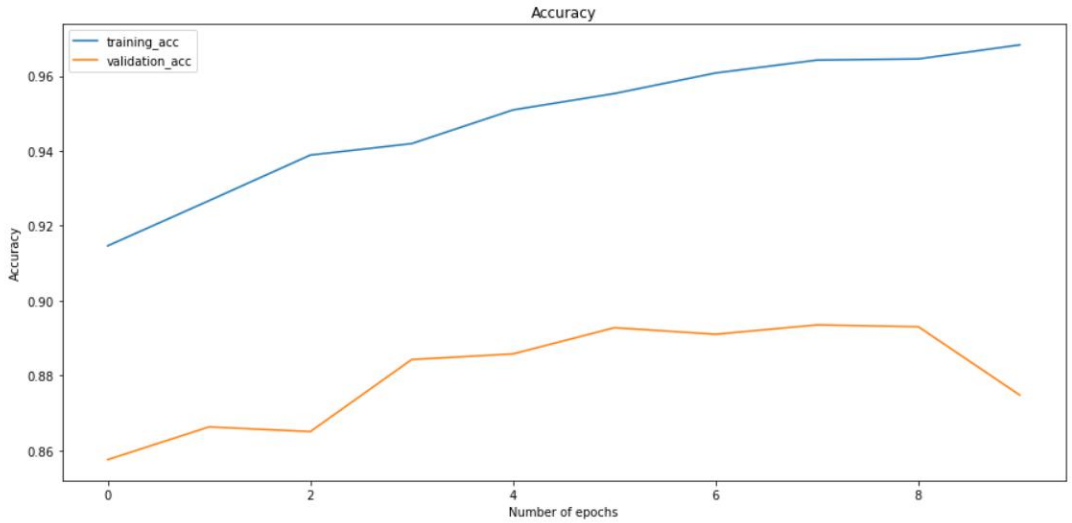


Fig. 6.2

3. GRU model

```
batch_size = 64
epochs=10
history_gru=gru_model.fit(xtrain_pad, ytrain, validation_data=(xtest_pad, ytest), batch_size = batch_size, epochs = epochs)
```

```
Epoch 1/10
250/250 [==============================] - 39s 143ms/step - loss: 0.5569 - accuracy: 0.6960 - val_loss: 0.4996 - val_accuracy:
0.7315
Epoch 2/10
250/250 [==============================] - 34s 138ms/step - loss: 0.4927 - accuracy: 0.7480 - val_loss: 0.4866 - val_accuracy:
0.7552
Epoch 3/10
250/250 [==============================] - 35s 141ms/step - loss: 0.4668 - accuracy: 0.7699 - val_loss: 0.4735 - val_accuracy:
0.7565
Epoch 4/10
250/250 [==============================] - 37s 148ms/step - loss: 0.4476 - accuracy: 0.7804 - val_loss: 0.4551 - val_accuracy:
0.7747
Epoch 5/10
250/250 [==============================] - 36s 145ms/step - loss: 0.4210 - accuracy: 0.8007 - val_loss: 0.4450 - val_accuracy:
0.7750
Epoch 6/10
250/250 [==============================] - 37s 146ms/step - loss: 0.3917 - accuracy: 0.8242 - val_loss: 0.4296 - val_accuracy:
0.7855
Epoch 7/10
250/250 [==============================] - 39s 157ms/step - loss: 0.3531 - accuracy: 0.8484 - val_loss: 0.4007 - val_accuracy:
0.8250
Epoch 8/10
250/250 [==============================] - 37s 146ms/step - loss: 0.3086 - accuracy: 0.8699 - val_loss: 0.3968 - val_accuracy:
0.8298
Epoch 9/10
250/250 [==============================] - 36s 143ms/step - loss: 0.2637 - accuracy: 0.8942 - val_loss: 0.4194 - val_accuracy:
0.8223
Epoch 10/10
250/250 [==============================] - 36s 142ms/step - loss: 0.2308 - accuracy: 0.9127 - val_loss: 0.3883 - val_accuracy:
0.8487
```

```
#plot accuracy for gru
plt.figure(figsize=(15, 7))
plt.plot(range(epochs), history_gru.history['accuracy'])
plt.plot(range(epochs), history_gru.history['val_accuracy'])
plt.xlabel('Number of epochs')
plt.ylabel('Accuracy')
plt.legend(['training_acc', 'validation_acc'])
plt.title('Accuracy')
```

```
Text(0.5, 1.0, 'Accuracy')
```

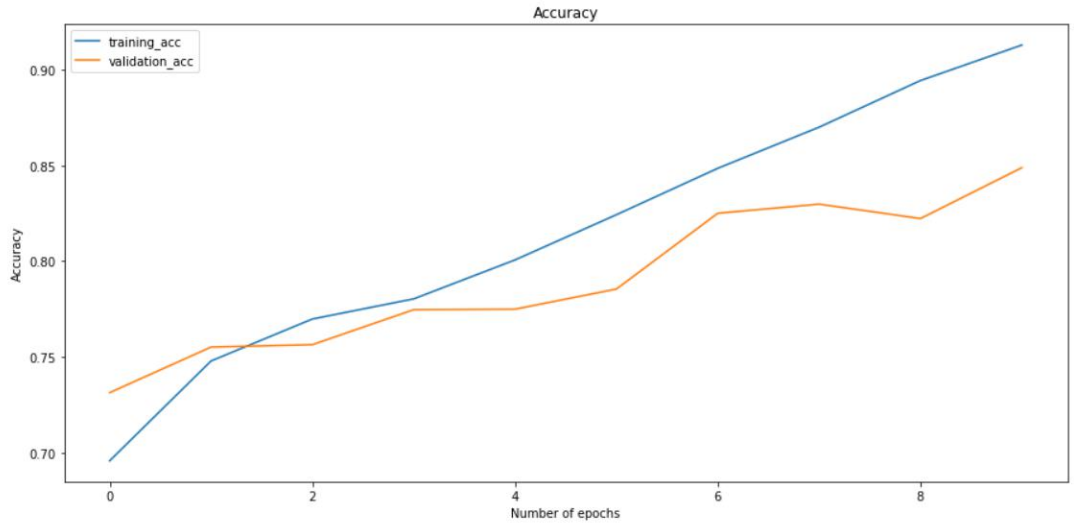


Fig. 6.3

Here is the accuracy graph of all three models combined:-

```
#plot accuracy
plt.figure(figsize=(15, 7))
plt.plot(range(epochs), lstm.history['accuracy'])
plt.plot(range(epochs), blstm.history['accuracy'])
plt.plot(range(epochs), history_gru.history['accuracy'])
plt.xlabel('Number of epochs')
plt.ylabel('Accuracy')
plt.legend(['lstm_acc', 'blstm_acc', 'gru_acc'])
plt.title('Accuracy')
```

```
Text(0.5, 1.0, 'Accuracy')
```

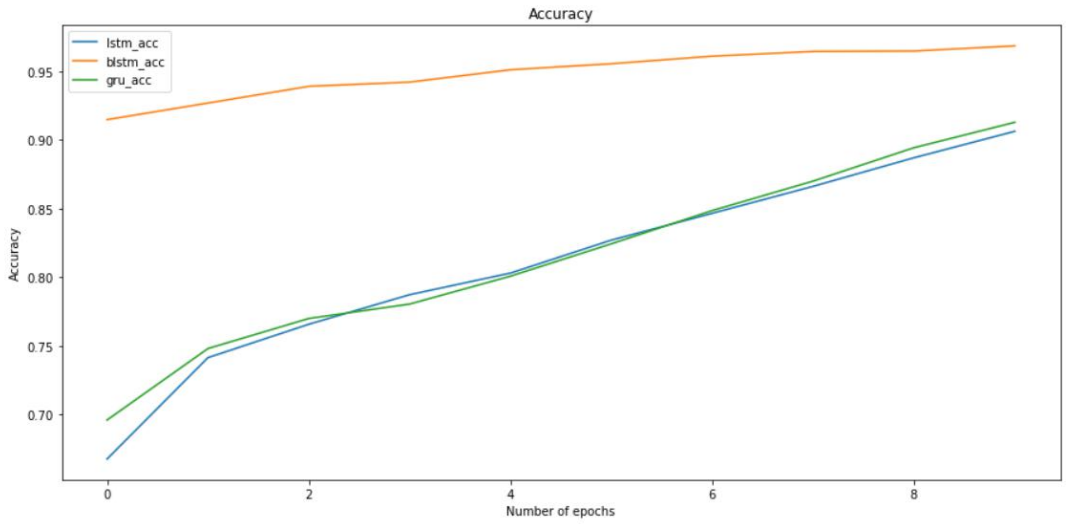


Fig. 6.4

```
plt.figure(figsize=(15, 7))
plt.plot(range(epochs), lstm.history['val_accuracy'])
plt.plot(range(epochs), blstm.history['val_accuracy'])
plt.plot(range(epochs), history_gru.history['val_accuracy'])
plt.xlabel('Number of epochs')
plt.ylabel('Validation Accuracy')
plt.legend(['lstm_acc', 'blstm_acc', 'gru_acc'])
plt.title('Validation Accuracy')
```

```
Text(0.5, 1.0, 'Validation Accuracy')
```

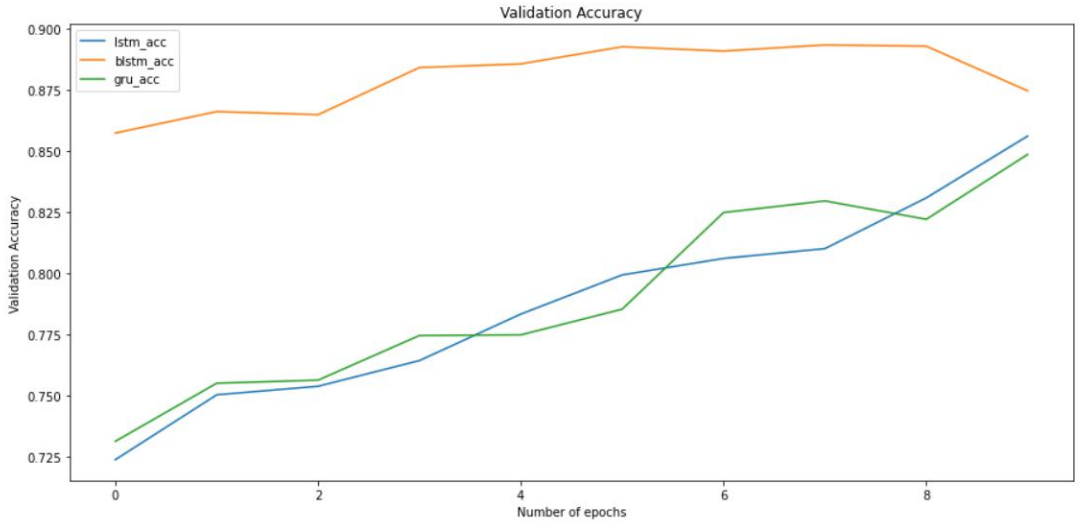


Fig. 6.5

For result analysis we have confusion matrix for the three models, these are as follows:

LSTM

```
lstm_pred_xtest=np.rint(lstm_model.predict(xtest_pad))
cm=tf.math.confusion_matrix(labels=ytest, predictions=lstm_pred_xtest)
import seaborn as sb
plt.figure(figsize=(10,7))
sb.heatmap(cm,annot=True , fmt='d')
plt.xlabel('Predicted')
plt.ylabel('True')
```

```
Text(69.0, 0.5, 'True')
```

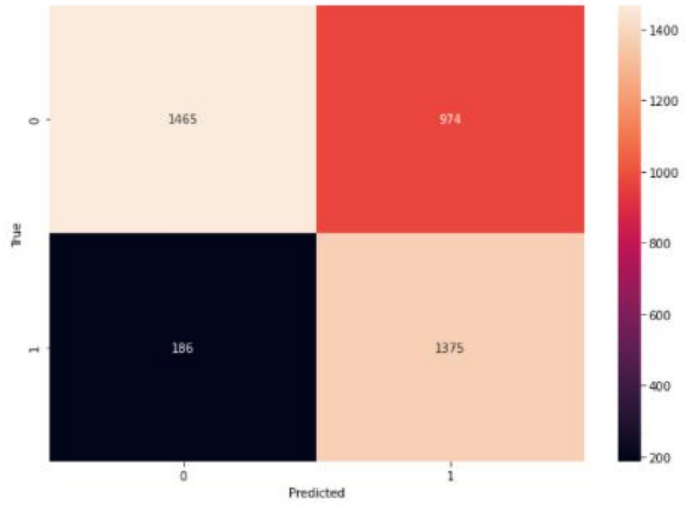


Fig. 6.6

BLSTM

```
blstm_pred_xtest=np.rint(blstm_model.predict(xtest_pad))
cm=tf.math.confusion_matrix(labels=ytest, predictions=blstm_pred_xtest)
import seaborn as sb
plt.figure(figsize=(10,7))
sb.heatmap(cm,annot=True , fmt='d')
plt.xlabel('Predicted')
plt.ylabel('True')
```

```
Text(69.0, 0.5, 'True')
```

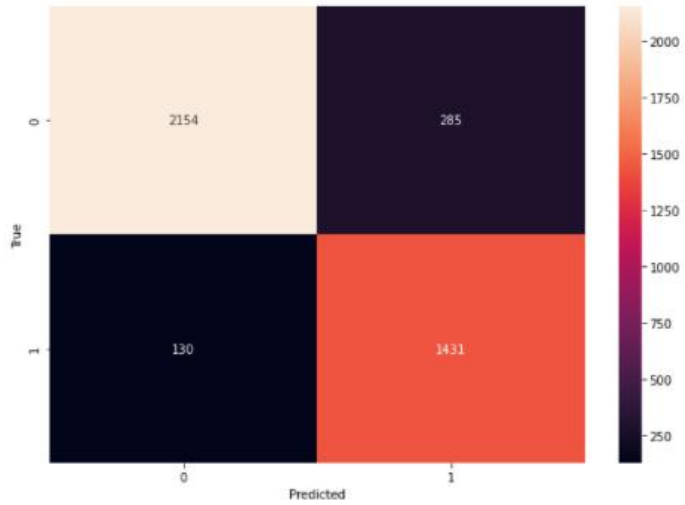


Fig. 6.7

GRU

```
gru_pred_xtest=np.rint(gru_model.predict(xtest_pad))
cm=tf.math.confusion_matrix(labels=ytest, predictions=gru_pred_xtest)
plt.figure(figsize=(10,7))
sb.heatmap(cm,annot=True , fmt='d')
plt.xlabel('Predicted')
plt.ylabel('True')
```

```
Text(69.0, 0.5, 'True')
```

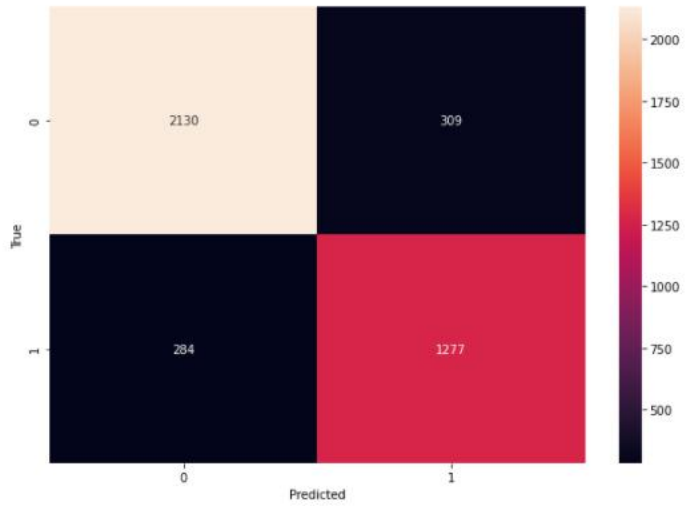


Fig.6.8

Conclusion and Future Scope

The advent of information and networking technology has created the good, the bad, and the ugly in online communication responses. These responses are often abused and have caused irreparable emotional damage that most often led to depression and suicide on innocent individuals when they were unable to speak out to get help from different agencies or family members. Meanwhile, some researchers have previously discussed this issue with traditional machine learning models, however most of these models built in these experiments can be applied to one social network at a time. In this paper, our novel methodology is compared with the latest research on using deep learning-based models to make their way in the detection of cyberbullying incidents. Our proposed LSTM utilizes doubled input gates, output gates, and forget gates in comparison to the traditional LSTM in use. This achievement gives better accuracy. However, our proposed model has more computational complexity and cost in terms of performance. This project takes a closer look at resolving the limitations of previous studies with better identification efficiency when directly compared to standard versions. Furthermore, our project has performed an empirical analysis to determine the effectiveness and performance of deep learning algorithms in detecting insults in Social Commentary. Three deep learning models, namely LSTM, GRU, and BLSTM are used in the experiments. Data pre-processing steps are applied that include: text cleaning, tokenization, stemming, as well as lemmatization to remove and stop words in the communication chain from getting to gullible users. The data from the pre-processing step is later passed through clean textual data and directly into deep learning algorithms for prediction. We can conclude that the BLSTM model achieved high accuracy and F1-measure scores in comparison to RNN, LSTM, and GRU. In the future, we shall integrate our deep learning approach with automatic detection by tracking and object identification mechanism through the application of the next phase of artificial intelligence (AI) 2.0 interface to aid law enforcement agencies in the smart city to curb the menace of cyberbullying.

References

Online:

[1] S. Arshad, "Sentiment Analysis / text classification using RNN(bi-LSTM)(recurrent neural network)," *Medium*, 21-Sep-2019. [Online]. Available: https://medium.com/@saad.arshad102/sentiment-analysis-text-classification-using-rnn-bi-lstm-recurrent-neural-network-81086dda8472.

[2] S. Shekhar, "LSTM for text classification," *Analyticsvidhya.com*, 14-Jun-2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/lstm-for-text-classification/.

[3] A. Lee, "Painless text classification using RNN," *Level Up Coding*, 07-May-2021. [Online]. Available: https://levelup.gitconnected.com/painless-classification-model-using-rnn-b90cb0982543.

[4] "Text classification with an RNN," *Tensorflow.org*. [Online]. Available: https://www.tensorflow.org/text/tutorials/text_classification_rnn.

[5] "Before you continue to YouTube," *Youtube.com*. [Online]. Available: https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi.

[6] "Before you continue to YouTube," *Youtube.com*. [Online]. Available: https://www.youtube.com/playlist?list=PLeo1K3hjS3us_ELKYSj_Fth2tIEkdKXvV.

[7] "Before you continue to YouTube," *Youtube.com*. [Online]. Available: https://www.youtube.com/playlist?list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI.

Technical Report:

[8] L. Ketsbaia, B. Issac, and X. Chen, "Detection of hate tweets using machine learning and deep learning," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2020.

[9] C. Iwendi, G. Srivastava, S. Khan, and P. K. R. Maddikunta, "Cyberbullying detection solutions based on deep learning architectures," *Multimed. Syst.*, 2020.

[10] M. Mahat, "Detecting cyberbullying across multiple social media platforms using deep learning," in *2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, 2021.

[11] S. Paul, S. Saha, and M. Hasanuzzaman, "Identification of cyberbullying: A deep learning based multimodal approach," *Multimed. Tools Appl.*, 2020.

[12] J. Wang, K. Fu, and C.-T. Lu, "SOSNet: A graph convolutional network approach to fine-grained cyberbullying detection," in *2020 IEEE International Conference on Big Data (Big Data)*, 2020.

[13] D. Mukhopadhyay, K. Mishra, K. Mishra, and L. Tiwari, "Cyber bullying detection based on twitter dataset," in *Machine Learning for Predictive Analysis*, Singapore: Springer Singapore, 2021, pp. 87–94.

[14] V. Banerjee, J. Telavane, P. Gaikwad, and P. Vartak, "Detection of cyberbullying using deep neural network," in *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, 2019.

Journal Article:

[15] J. A. Cornel *et al.*, "Cyberbullying detection for online games chat logs using deep learning," in *2019 IEEE 11th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, 2019.

[16] H. Kumar Sharma, K. Kshitiz, and Shailendra, "NLP and machine learning techniques for detecting insulting comments on social networking platforms," in *2018 International Conference on Advances in Computing and Communication Engineering (ICACCE)*, 2018.

[17] J. Yadav, D. Kumar, and D. Chauhan, "Cyberbullying Detection using Pre-Trained BERT Model," in *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, 2020.

[18] J. Wu, M. Wen, R. Lu, B. Li, and J. Li, "Toward efficient and effective bullying detection in online social network," *Peer Peer Netw. Appl.*, vol. 13, no. 5, pp. 1567–1576, 2020.

Dataset:

[19] DataTurks, "Tweets dataset for detection of Cyber-trolls." .Avaiable:https://www.kaggle.com/dataturks/dataset-for-detection-of-cybertrolls