CSE-4022 NATURAL LANGUAGE PROGRAMMING

OFFENSIVE TWEET DETECTION USING BI-DIRECTIONAL LONG SHORT-TERM MEMORY

submitted in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE ENGINEERING

By

TEAM 18
SLOT: B1+TB1

NAME	REGISTRATION NUMBER
Soham Sengupta	19BCE0857
Sanjay M	19BCE2207

B.Tech. Computer Science and Engineering

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

November, 2022

EXECUTIVE SUMMARY

The term hate speech is comprehended as any type of speech, writing or behavioral communication that attacks or uses pejorative or discriminatory language with reference to an individual or a group based on who and what they are, in other words, on the basis of their religion, skin color, country of origin, race, descent, gen der or other identity factor. With the growing social user media userbase, the interactions between people are sometime aggressive when there is a difference of opinion. In this project, we are developed a custom Bi-Directional LSTM architecture for detecting offensive tweets. The proposed method uses a dataset which has a 3-class classification of each tweet acquired using twitter API and manually labelled by people. And a custom Bi-directional LSTM model is made and the preprocessed data is fed into for training. And with the test data the model is evaluated based on various metrics.

CONTENTS PAGE NO

List of Abbreviations

LSTM	Long short-term memory
GRU	Gated Recurrent Units
NN	Neural Network
RNN	Recurrent Neural Network
NLTK	Natural Language Took Kit
GloVe	Global Representation of Word Vectors

1.Introduction :

1.1 Objective

Whenever a tweet is posted is extracted using an API, one must able to classify the tweet as "Hate Speech", "Offensive" or a "Neutral" tweet. The system when fed with a tweet it should using deep learning method the system should classify the tweet as hate speech, offensive or neutral. The aim of the project is to get a dataset that has as three classes as mentioned above with their corresponding tweets. Further with this data, we aim to create custom architecture Bi-Directional LSTM which gives a proper accuracy. The architecture should be defined in a way so that the model doesn't overfit or underfit and it yields a proper working accuracy. Previous project that used LSTM or RNN as their model gave lot of false positive results. Therefore, the key aim of the project is to create a Bi-Directional LSTM model which predicts whether the given tweet is hate speech, offensive or neither of those with a less false negatives.

1.2 Motivation

On the Internet, something unusual has always been happening since social media was created. Hate speech directed toward persons of a certain gender or ethnicity is common on social media, and these online abominations have real-world effects, such as the spread of fear and hatred across communities. Radical anti-female or anti-male hate speech is popular among internet trolls in India, which is a cultural hotspot, and the widespread use of these insulting sentiments has resulted in a nationwide gender issue. In summary, Internet prejudice has aided the spread of hate cultures. Hate speech is only effective because it may instill discriminatory beliefs without being observed, and we devised a strategy to counteract this. The primary reason we decided to take on this challenge is to prevent hate in our society.

1.3 Background

The general study that are necessary for the project are to learn about LSTM, BiDirectional LSTM, and ways to optimize the models and all the tools related to it. Long Short-Term Memory (LSTM) networks are a sort of recurrent neural network that may learn order dependence in sequence prediction challenges. This is a requirement in a variety of complicated issue domains, including machine translation, speech recognition, and others. Deep learning's LSTMs are a complicated topic. It can be difficult to grasp the concept of LSTMs and how words like bidirectional and sequence-tosequence apply to the area. A bidirectional LSTM, often known as a bi-LSTM, is a sequence processing model that consists of two LSTMs, one of which takes input in one way and the other in the other. Bi-LSTMs effectively increase the quantity of data available to the network, giving the algorithm better context. GRU employs the so-called update gate and reset gate to overcome the vanishing gradient problem of a regular RNN. In essence, there are two vectors that determine what data should be sent to the output. They are unique in that they can be trained to retain knowledge from the past without having to wash it away over time or delete information that is unrelated to the forecast. A word embedding is a learnt text representation in which words with related meanings are represented similarly. One of the significant achievements of deep learning on tough natural language processing problems may be this way to expressing words and documents. RMSprop combines the idea of simply using the gradient's sign with the idea of tailoring the step size to each weight individually. So, rather than looking at the gradient's magnitude, we'll look at the step size that's been established for that particular weight. Keras is an open-source software library for artificial neural networks that includes a Python interface. Keras serves as a user interface for TensorFlow. Keras supported a variety of backends up until version 2.3, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML. The Natural Language Toolkit, or NLTK for short, is a Python-based collection of modules and applications for symbolic and statistical natural language processing in English. The programming language used to implement all the necessary library is python.

2.Project Description and Goals

2.1 General Description of Algorithms

Bi-Directional LSTM:

The BRNN principle is to divide the neurons of a standard RNN into two paths, one for positive time (ahead states) and the other for negative time (back states) (backward states). The outputs of those two states are not related to the inputs of the states in the opposite manner. The right diagram depicts the general structure of RNN and BRNN. Unlike normal RNN, which requires delays for incorporating future information, input information from the past and future of the current time frame can be included by using two-time directions. Because the two directional neurons do not interact, BRNNs can be taught using comparable algorithms as RNNs. Back-propagation across time, on the other hand, necessitates the usage of extra processes because updating input and output layers cannot be done simultaneously. The following are the general training procedures: Forward states and backward states are passed first in forward pass, followed by output neurons. Output neurons are passed first in the reverse pass, followed by forward and backward states. The weights are updated after the forward and backward passes are completed.

Softmax:

Softmax is a function rather than a loss. It compresses a vector in the range (0, 1), and the sumof the elements is 1. It is applied to the score of the results. Elements can be thought of as class probabilities because they represent a class. The Softmax function cannot be applied independently to each si, since it depends on all elements of s. For any given class si, the Softmax function can be calculated as:

$$f(s)_i = \frac{\exp s_i}{\sum_j^C \exp s_j}$$

Where sj are the scores inferred by the net for each class in C. Note that the Softmax activation for a class si depends on all the scores in s.

Categorical Cross-Entropy Loss:

Categorical cross-entropy is a loss function that is utilized in multi-class classification tasks.

These are tasks where an example can only belong to one out of various possible categories, and the model should decide which one. The categorical cross-entropy is well suited to classification tasks, since one example can be considered to belong to a specific category with probability 1, and to other categories with probability 0. It is a Cross-Entropy loss plus a Softmax activation. If we use this loss, we will train a CNN to output a probability over the C classes for every image. It is used for multi-class classification.

$$CE = -\log\left(\frac{\exp s_p}{\sum_j^C \exp s_j}\right)$$

Where Sp is the CNN score for the positive class.

RMSprop Optimizer:

The gradient descent technique with momentum is nearly equivalent to the RMSprop optimizer. The RMSprop optimizer limits oscillations in the vertical direction. As a result, by raising our learning rate, our algorithm will be able to take greater horizontal steps and reach a faster convergence. The distinction between gradient descent and RMSprop lies in the method of calculating the gradients.

2.2 Existing Methods

General sentiment analysis of the tweets can classify as positive or negative sentences, which won't be much sufficient to classify a tweet as a hate speech or offensive. Classical method of hate speech detection uses manually encoded word features. Lexical detection methods tend to have low precision because they classify all messages containing particular terms as hate speech and previous work using supervised learning has failed to distinguish between the two categories. That is classifying a tweet as hate or offensive based on certain words. Not all tweets that contain certain words are offensive. With the help of word embedding supervised machine learning techniques like support vector machine were used on the word embedding with target classes. 11.6% of the total tweets were flagged as offensive. As RNN as efficient for NLP and sequence-based data, the classified dataset is converted into word embedding and RNN were trained and used to helps to classify data as offensive or not.

2.3 Gap Analysis and Proposed Method

We propose a custom Bi-directional architecture for offensive tweet detection. When compared to RNN, LSTM and Bi-LSTM have a short-term memory, which helps to identify textual pattern in the data rather than focusing on just a bag of words. With LSTM bidirectional in nature the textual pattern identified will be more and the classification will have less false positive rates. The data pre-processing included removing the special characters, one hot encoding the target variables for the corresponding tweet. And Tokenization of the words are done, and Using GloVe word embedding from Stanford, all the words from the current vocabulary are converted in to vector which can be used to train the Bi-Directional LSTM model. The data is padded to match the input layer of the custom architecture model. The data is split into training and testing and the custom model is trained until a specified metrics is achieved. The input will be a vectorized words of the tweet and the output vector will be of 3 and the output classes are 3. And with the test data, evaluation of the model is performed based on some evaluation metrics and analyzed.
Bi-Directional LSTM has the ability to understand the context of an embedded matrix in both the direction when compared to GRU and RNN, LSTM is more thorough for larger datasets such as the dataset used in this particular project. With this feature, more ambiguous data that starts with a bad context and end in a positive way. Thus, more false positives can be reduced. This gives our proposed model an edge over the other existing methodologies.

3.Technical Specification

3.1Dataset Description:

By using Twitter API 25000+ tweets are extracted, and manually labelled by people whether the tweet is offensive, hate speech or neither of those. And a column contains the count of how many people have voted for the tweet. The amount of hate speech or offensiveness is denoted in the range of 1 – 10. This dataset is acquired from Cornell University, USA. Each tweet in the dataset has all the special characters along with twitter's common notations, for e.g., RT for Re-Tweet. Around 76% (2/3) of the data is flagged as hate or offensive speech, which will enhance the model performance, by providing various and deeper textual pattern for hate speech or offensive tweet.

3.2 Tools and Libraries Used:

NumPy: NumPy is a Python library that adds support for huge, multi-dimensional arrays and matrices, as well as a large number of high-level mathematical functions to operate on these arrays.

Pandas: Pandas is a data manipulation and analysis software package for the Python programming language. It includes data structures and methods for manipulating numerical tables and time series, in particular. It's open-source software with a threeclause BSD license. The word "panel data" is an econometrics word for data sets that comprise observations for the same persons over multiple time periods.

Matplotlib: Matplotlib is a graphing package for Python with NumPy, the Python numerical mathematics extension. It provides an object-oriented API for embedding charts into applications utilizing GUI toolkits such as Tkinter, wxPython, Qt, or GTK. There's also a procedural "pylab" interface built on a state machine (like OpenGL) that's meant to look like MATLAB, however it's not recommended.

TensorFlow:

TensorFlow is a machine learning software library that is free and open-source. It can be used for a variety of applications, but it focuses on deep neural network training and inference. TensorFlow is a dataflow and differentiable programming-based symbolic math toolkit.

NLTK: The Natural Language Toolkit, or NLTK for short, is a Python-based collection of modules and applications for symbolic and statistical natural language processing in English.

Keras: Keras is an open-source software library for artificial neural networks that includes a Python interface. Keras serves as a user interface for TensorFlow. Keras supported a variety of backends up until version 2.3, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML.

Google Colab: Colab notebooks let you blend executable code and rich text, as well as graphics, HTML, LaTeX, and more, in a single document. Your Colab notebooks are saved in your Google Drive account when you create them. You can quickly share your Colab notebooks with coworkers or acquaintances, allowing them to provide comments or even make changes.

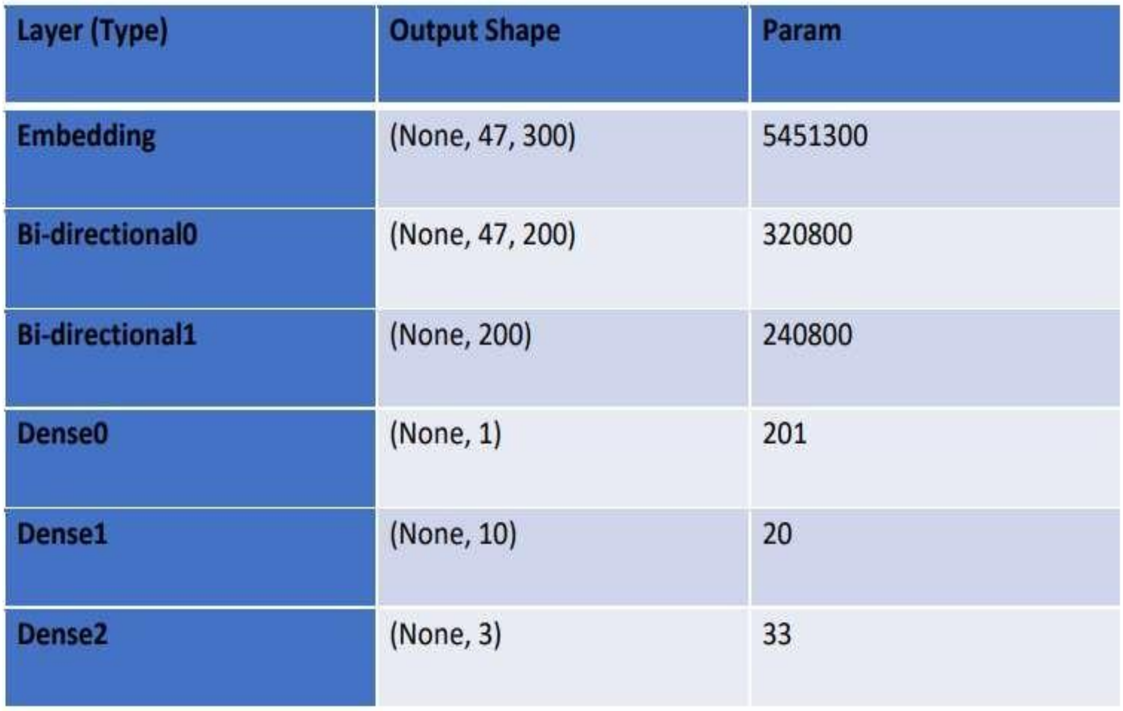
3.3 Usage of the Tools:

Once the dataset is collected the dataset is imported into the working colab using pandas library all the data wrangling process are done using Pandas and the text to vectorized process are done using the NLTK library. And the vector is operated using the NumPy library. And finally,the model definition and training and testing are done using Keras with TensorFlow as Backend. Finally, all the results are visualized using matplotlib.

4. Design Approach and Details

4.1 Bi-Directional LSTM Model:

The proposed architecture consists of 6 layers, where the initial layer is an embedding layer to get the vectorized form of the words and sentences. And there are two Bi-Directional layers with the dimensions of (,47, 200) and (,200). These Bi-Directional layers helps to identify the context in both the direction of the sentences. There is dense layer with one node as from the previous layer there will be only one value. And a dense layer with 10 nodes to extract the further features. And final layer consists of 3 nodes as the number of classes to be predicted is 3 they are "Hate Speech", "Offensive Tweet" and "Neither".



Layer (Type)	Output Shape	Param
Embedding	(None, 47, 300)	5451300
Bi-directional0	(None, 47, 200)	320800
Bi-directional1	(None, 200)	240800
Dense0	(None, 1)	201
Dense1	(None, 10)	20
Dense2	(None, 3)	33

Table 1: Architecture of the Proposed Model

4.2 LITERATURE REVIEW:

No.	AUTHOR AND YEARS OF REFERENCE	TITLE	ABSTRACT	METHODOLOGY	LIMITATIONS

1	ARAUJO DE SOUZA, Gabriel and DA COSTA ABREU, Marjory (2020)	Automatic offensive language detection from Twitter data using machine learning and feature selection of metadata	In theory, the use of social media was proposed so we could share our views online, keep in contact with loved ones or share good moments of life. However, the reality is not so perfect, so you have people sharing hate speechrelated messages, or using it to bully specific individuals, for instance, or even creating robots where their only goal is to target specific situations or people. Identifying who wrote such text is not easy and there are several possible ways of doing it, such as using natural language processing or machine learning algorithms that can investigate and perform predictions using the metadata associated with it.	1.Data extraction and normalisation 2. Naive Bayes Classifier 3. Support Vector Machine	For the Data Type A, the Linear SVM demonstrated a greater difficulty in parameter setting to find satisfactory results. They tested to change the configuration of the alpha parameter with the values 0.1, 0.01 and 0.001 besides configuring the max epochs to 100 and 500, but in all possible combinations of theses parameters, the algorithm always classified any text as offensive.
2	Aya Elouali, Zakaria Elberrichi , Nadia Elouali (2019)	Hate Speech Detection on Multilingual Twitter Using Convolutional Neural Networks	Hate speech detection on Twitter is often treated in monolingual (in English generally) ignoring the fact that Twitter is a global platform where everyone expresses himself with his natal language. In this paper, they created a model which, taking benefits of the advantages of neural networks, classifies tweets written in seven different languages (and even those that contains more than one language at the same time) to hate speech or non hate speech.	• Baseline Methods: use different word representations (TFIDF, Bag of Words, Char ngrams) with traditional classifiers • DNNs + GBDTs: Combine the neural networks used in second experiment with the classifier GBDTs (Gradient Boosted Decision Trees). DNNs are used for feature extraction for GBDTs.	Multiple layers of data cleaning is required. Also, as the words are more , as many parameters and experiments have to be made and checked which makes this work a time consuming one.

3	Reza Farahbakhsh, Noël Crespi (2020)	Hate speech detection and racial bias mitigation in social media based on BERT model	In this paper, we first introduce a transfer learning approach for hate speech detection based on an existing pre-trained language model called BERT (Bidirectional Encoder Representations from Transformers) and evaluate the proposed model on two publicly available datasets that have been annotated for racism, sexism, hate or offensive content on Twitter. Next, we introduce a bias alleviation mechanism to mitigate the effect of bias in training set during the finetuning of our pre-trained BERT-based model for hate speech detection.	To analyze the ability of the BERT transformer model on the identification of hate speech, we describe the mechanism used in the pretrained BERT model at first. BERT is a multilayer bidirectional transformer encoder trained on the English Wikipedia and the Book Corpus containing 2,500M and 800M tokens, respectively.	To train the model, we need to split Waseemdataset and Davidson-dataset into training, validation and test sets. Considering the disparate distribution of tweets in different classes described , it is justifiable that we are dealing with imbalanced datasets .
4	Aditya Gaydhani , Vikrant Doma , Shrikant Kendre and Laxmi Bhagwat (2018)	Detecting Hate Speech and Offensive Language on Twitter using Machine Learning: An N-gram and TFIDF based Approach	Using Twitter dataset, we perform experiments considering n-grams as features and passing their term frequency-inverse document frequency (TFIDF) values to multiple machine learning models. We perform comparative analysis of the models considering several values of n in n-grams and TFIDF normalization methods.	1)Data preprocessing 2)Feature extraction 3)Normalisation of TFIDF 4)Cross validation of each combination of feature parameters.	the model does not account for negative words present in a sentence. Improvements can be done in this area by incorporating linguistic features.

5	Muhammad Okky Ibrohim (2019)	Identify Abusive and Offensive Language in Indonesian Twitter using Deep Learning Approach	This paper implements a deep learning approach to enhance the performance when identifying abusive but not offensive or an offensive language. We use Long Short-Term Memory (LSTM) with word embedding because our literature study found that LSTM with word embedding is good for text classification (both for English or Indonesian text classification).	In this research, we use the LSTM network with word embedding, as done in [12]. The LSTM [11] is one of Recurrent Neural Network (RNN) variant, which falls into deep learning approach.	we suggest using embedding built on the large-sized social media corpus to capture more semantics in the learning process. This might be needed to handle the high noise or variety of writing in the dataset.
6	György Kovács, Pedro Alonso & Rajkumar Saini (2021)	Challenges of Hate Speech Detection in Social Media	The detection of hate speech in social media is a crucial task. The uncontrolled spread of hate has the potential to gravely damage our society, and severely harm marginalized people or groups. A major arena for spreading hate speech online is social media. This significantly contributes to the difficulty of automatic detection, as social media posts include paralinguistic signals (e.g. emoticons, and hashtags), and their linguistic content contains plenty of poorly written text.	There are many layers to the difficulty of automatically detecting hateful and/or offensive speech, particularly in social media. Some of these difficulties being closely related to the shortcomings of keywordbased approaches. For one, words can be obfuscated in many different ways, both in an intentional attempt to avoid automatic content moderation , or as a consequence of the use of social media for communication	Another challenge to consider is that of imbalanced data. While the spread of hateful and offensive content is a serious problem in social media, it is fortunately still true that this content only constitutes a small fraction of all content.

7	Zeerak Waseem, Zirk Howey (2016)	Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter	On social media, hate speech in the form of racist and sexist comments is all too frequent. As a result, many social media platforms address the issue of recognising hate speech, although hate speech is defined in a variety of ways and is difficult to define.	In order to pick the most suitable features, we perform a grid search over all possible feature set combinations, finding that using character n-grams outperforms using word ngrams by at least 5 F1-points (60.42 vs. 69.86) using similar features.	s. While the problem is far from solved, we find that using a character n-gram based approach provides a solid foundation. Demographic information, apart from gender, brings little improvement, but this could be due to the lack of coverage
8	Hajime Watanabe, Mondher Bouaziz, Tomoaki Ohtsuki (2018)	Hate Speech on Twitter A Pragmatic Approach to Collect Hateful and Offensive Expressions and Perform Hate Speech Detection	language aimed to a certain group of individuals who share a same interest property, whether it's their gender or not (i.e., sexism),their race or ethnic group. While most online social networks and microblogging services prohibit hate speech, the sheer scale of these networks and websites makes controlling all of their material nearly difficult.	Weka presents variety of classifiers organized into groups based on the type of the algorithm (e.g., decision treebased, rulebased, etc.).To evaluate the performance of classification, we use 4 different key performances indicators (KPIs) which are the percentage of true positives, the precision.	we will try to build a richer dictionary of hate speech patterns that can be used, along with a unigram dictionary, to detect hateful and offensive online texts. We will make a quantitive study of the presence of hate speech among the different genders, age groups and regions, etc

9	Shammur A. Chowdhury, Soon-gyo Jung, Hind Almerekhi & Bernard J. Jansen (2020)	Developing an online hate classifier for multiple social media platforms	Although researchers have found that hate is a problem across multiple platforms, there is a lack of models for online hate detection using multi-platform data. To address this research gap, we collect a total of 197,566 comments from four platforms: YouTube, Reddit, Wikipedia, and Twitter, with 80% of the comments labeled as non-hateful and the remaining 20% labeled as hateful.	In this study, we train different models using various algorithms (described in “Experimental design and evaluation” section) along with different feature representations (presented in “Discussion” section) and compare their performance. In addition to the different algorithm performance, we also evaluated the performance of the models using two baseline models: a keyword-based classifier (KBC).	the challenge posed by the limitations of available labeled data. This means that our experiments and suggestions address this problem. This also means that our discussion of other problems regarding hate speech detection are limited, and the purpose of this discussion is mostly to provide proper context.

10	Hajung Sohn, Hyunju Lee. (2019)	MCBERT4HATE: Hate Speech Detection using Multichannel BERT for Different Languages and Translatio ns	As manual hate speech detection by human annotators is both costly and time consuming, there are needs to develop an algorithm for automatic recognition. Transfering knowledge by fine-tuning a pre-trained language model has been shown to be effective for improving many downstream tasks in the field of natural language processing. The Bidirectional Encoder Representations from Transformers (BERT) is a language model that is pretrained to learn deep bidirectional representations from a large corpus.	This study was aimed at developing an effective model for automatic hate speech detection. The proposed model integrates the hidden features of separate BERT models trained on different languages. This approach is supposed to effectively capture different semantic representation of different languages. In addition, we investigated the effect of translations as auxiliary sentences for sentence classification.	Translations from machine translation models have many errors.

4.2 ARCHITECTURE DIAGRAM:

The proposed architecture flow of the work is shown in the below figure. First the dataset is extracted and manually labelled. Further all the special character are removed from the sentences and data wrangling has been carried out. The labels are one hot encoded. And all the stop words have been removed. From there the words have been tokenized. Then using the GloVe model from Stanford, the words has been vectorized. Training and Testing sets have been split. The architecture of the model is defined and the model have been trained on 6 the training data with custom callbacks to avoid overfitting. Further the model is evaluated on the test set and with some custom performance metrics. And with the performance score graphs have been plotted for visual analysis.

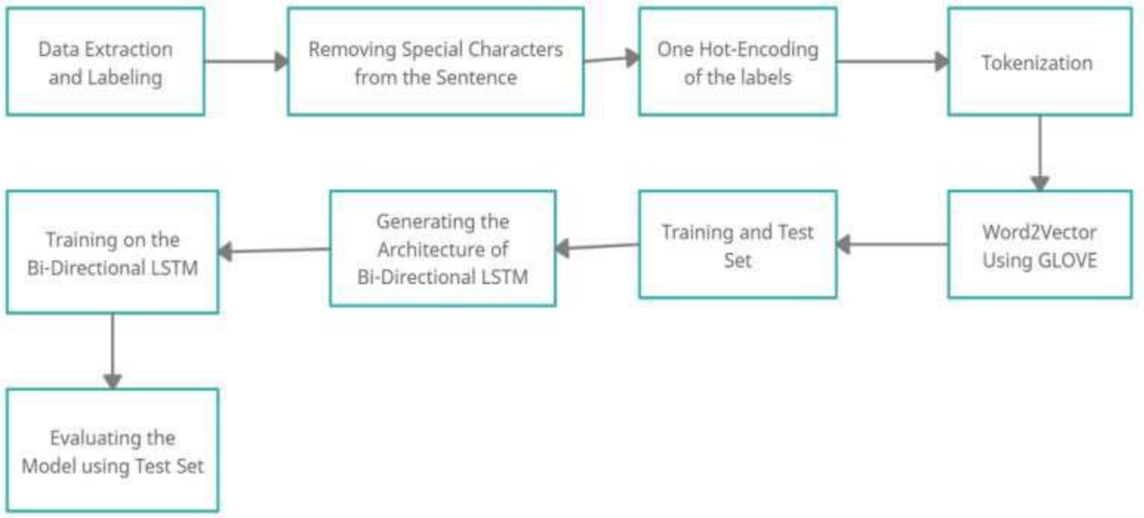

Data Extraction and Labeling
Removing Special Characters from the Sentence
One Hot-Encoding of the labels
Tokenization
Word2Vector Using GLOVE
Training and Test Set
Generating the Architecture of Bi-Directional LSTM
Training on the Bi-Directional LSTM
Evaluating the Model using Test Set

Fig 1: Architecture Diagram of the Project

5.SCHEDULE,TASKS AND MILESTONE

5.1 List of Tasks

1. Background Study
2. Literature Survey
3. Dataset Collection
4. Data Wrangling
5. Tokenization, Word Embedding, TextToSequence
6. Model Definition and Training
7. Model Hyperparameters Tuning
8. Model Evaluation
9. Result Analysis

5.2Gantt Chart

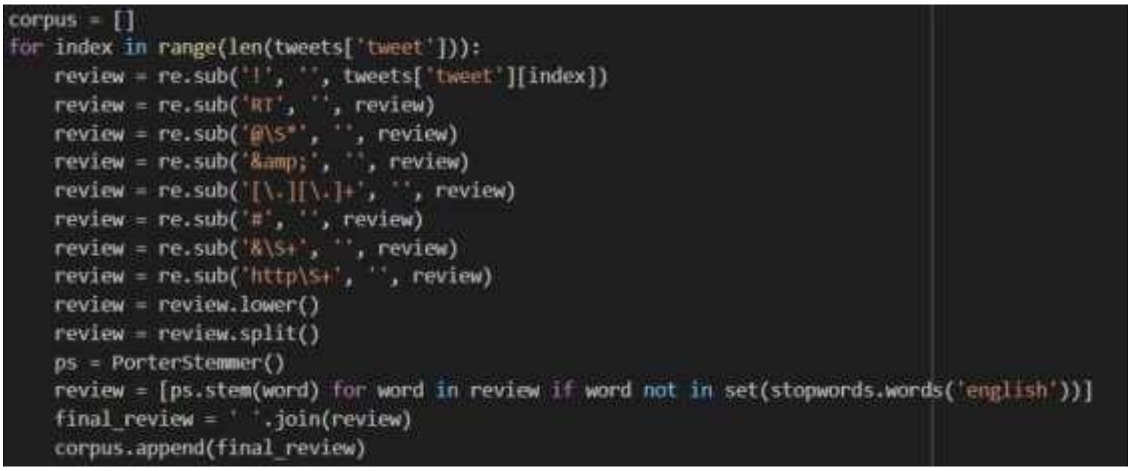
First, we did a background study on hate speech, it’s frequency and studied the effect it has on people. Then we listed out different algorithms and models to solve this problem. We carefully compared them all to get the most efficient model. We then searched through some of the existing models and projects to improvise our idea. After looking at several different datasets we decided to go with clowdflower’s dataset as it satisfied our requirements. We then transformed and mapped data from one "raw" data form into another format with the intent of making it more appropriate and valuable for a variety of downstream purposes. We then turned these data into non-sensitive data called tokens that can be used in our bidirectional LSTM model. Word embedding techniques were also applied. Text to sequence was done. With 7 everything ready, we defined our model and trained it. We then choose a set of optimal hyperparameters for our learning algorithm followed by model evaluation that aims to generalize the accuracy of the model on future data. Finally, we collected the results and analyzed our results to improve our model and define a scope of improvement.

5.3 Complete Implemented Process

- Data imported as Features and Labels.
- Special Characters from the data has been removed.
- The entire data has been split into training and test set.
- The target values have been one hot encoded
- The words have been tokenized and vocabulary length of the dataset has been found.
- Word embedding has been done using the GloVe word to vector from Stanford.
- The embedded words have been put into a sequence vector, so that it can be fed into the model
- The entire pre-processing of the data is done.
- Model is defined and trained with custom call-backs to attain best accuracy.
- Training and Validation accuracy has been plotted to see the model performance

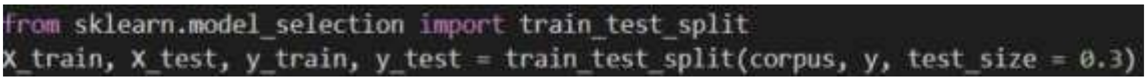
6.PROJECT IMPLEMENTATION SCREENSHOTS

1) Performance of Data Wrangling Process



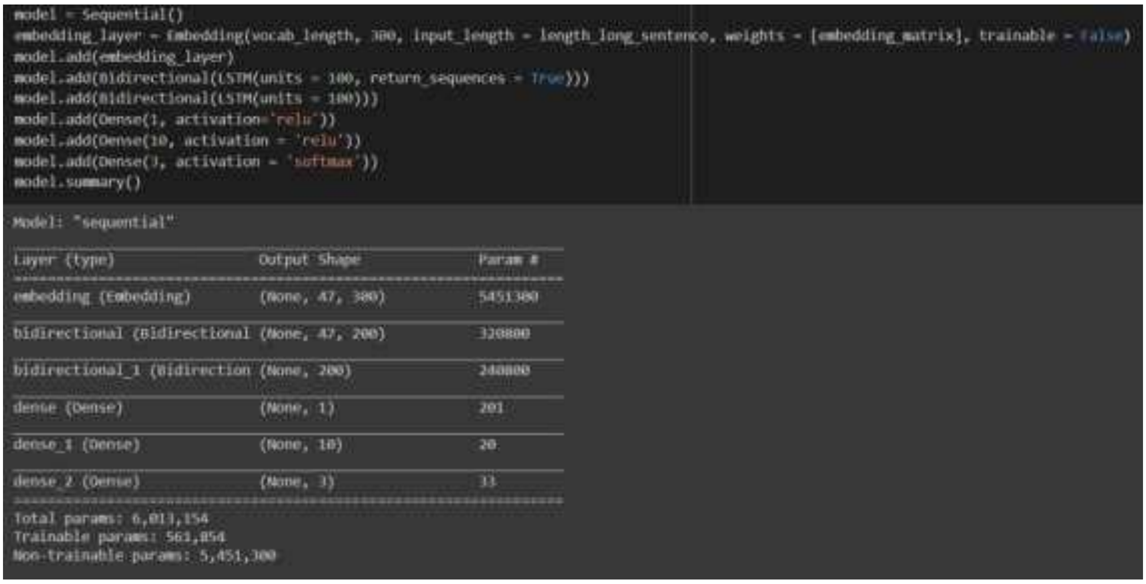
```
corpus = []
for index in range(len(tweets['tweet'])):
    review = re.sub('!', '', tweets['tweet'][index])
    review = re.sub('RT', '', review)
    review = re.sub('@\S*', '', review)
    review = re.sub('&amp;', '', review)
    review = re.sub('[\.][\.]+', '', review)
    review = re.sub('#', '', review)
    review = re.sub('&\S+', '', review)
    review = re.sub('http\S+', '', review)
    review = review.lower()
    review = review.split()
    ps = PorterStemmer()
    review = [ps.stem(word) for word in review if word not in set(stopwords.words('english'))]
    final_review = ' '.join(review)
    corpus.append(final_review)
```

2) The entire dataset has been split into training and testing.



```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(corpus, y, test_size = 0.3)
```

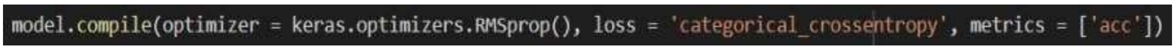
3) The model has been defined using Keras Callbacks.



```
model = Sequential()
embedding_layer = Embedding(vocab_length, 300, input_length = length_long_sentence, weights = [embedding_matrix], trainable = False)
model.add(embedding_layer)
model.add(Bidirectional(LSTM(units = 100, return_sequences = True)))
model.add(Bidirectional(LSTM(units = 100)))
model.add(Dense(1, activation='relu'))
model.add(Dense(10, activation = 'relu'))
model.add(Dense(3, activation = 'softmax'))
model.summary()
```

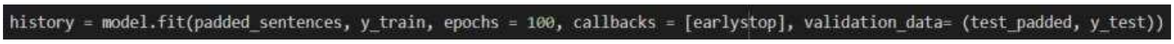
```
Model: "sequential"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
embedding (Embedding)       (None, 47, 300)           5451300
_________________________________________________________________
bidirectional (Bidirectional (None, 47, 200)          320800
_________________________________________________________________
bidirectional_1 (Bidirection (None, 200)              240800
_________________________________________________________________
dense (Dense)               (None, 1)                 201
_________________________________________________________________
dense_1 (Dense)             (None, 10)                20
_________________________________________________________________
dense_2 (Dense)             (None, 3)                 33
=================================================================
Total params: 6,013,154
Trainable params: 561,854
Non-trainable params: 5,451,300
```

4) The model has been complied using the RMPprop Optimizer and Categorical Cross entropy Loss.



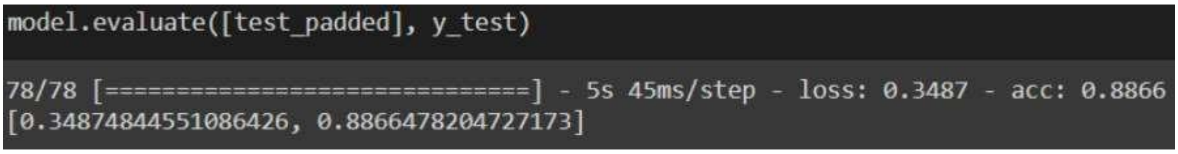
```
model.compile(optimizer = keras.optimizers.RMSprop(), loss = 'categorical_crossentropy', metrics = ['acc'])
```

5) The model has been started training



```
history = model.fit(padded_sentences, y_train, epochs = 100, callbacks = [earlystop], validation_data= (test_padded, y_test))
```

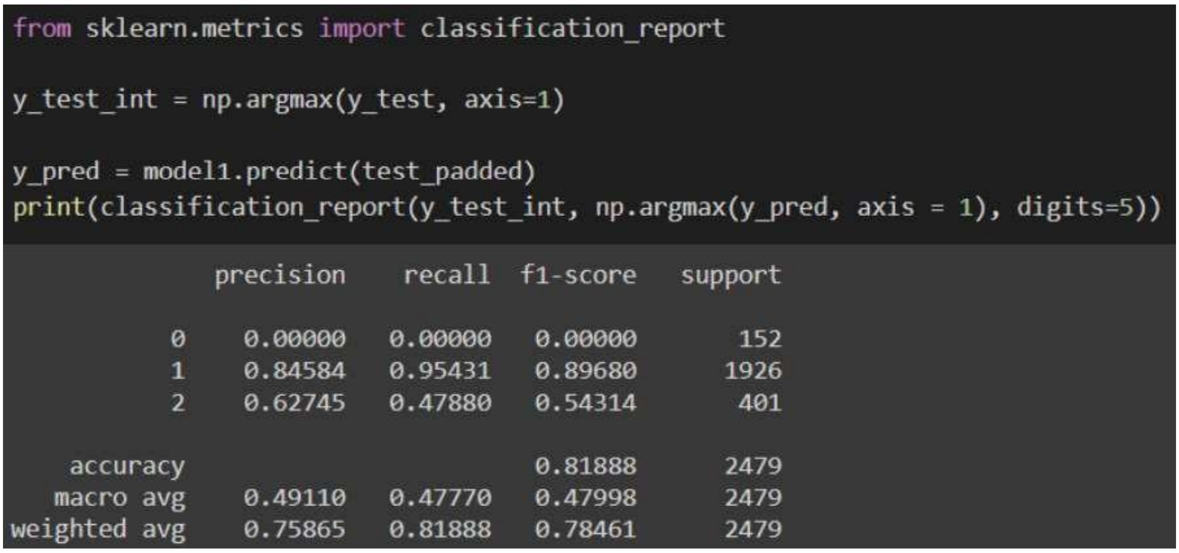
6) The model has been evaluated and achieved and validation accuracy score of 0.8866



```
model.evaluate([test_padded], y_test)
```

```
78/78 [==============================] - 5s 45ms/step - loss: 0.3487 - acc: 0.8866
[0.34874844551086426, 0.8866478204727173]
```

7) Precision, Recall, F1 Score has been generated for the model.



```
from sklearn.metrics import classification_report

y_test_int = np.argmax(y_test, axis=1)

y_pred = model1.predict(test_padded)
print(classification_report(y_test_int, np.argmax(y_pred, axis = 1), digits=5))
```

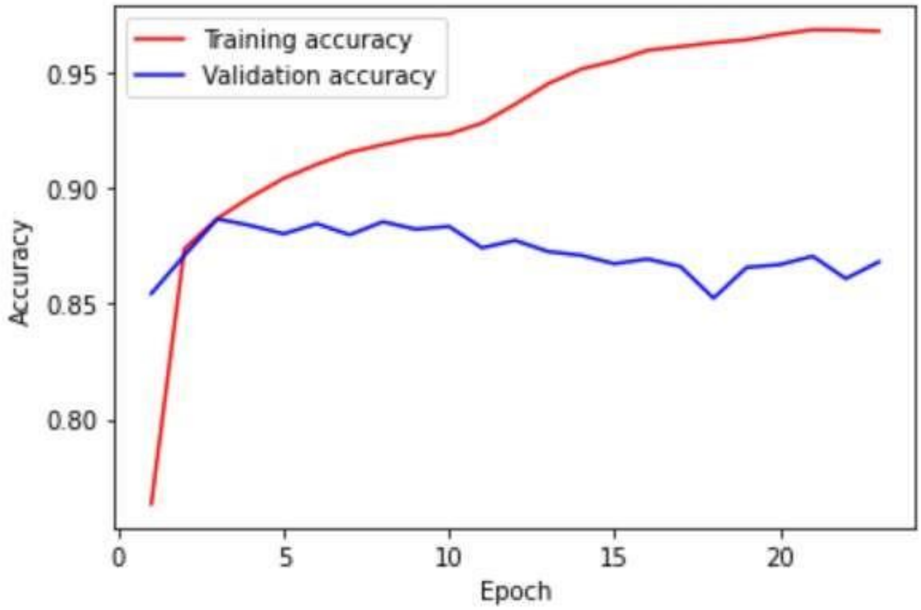
```
              precision    recall  f1-score   support

           0    0.00000   0.00000   0.00000       152
           1    0.84584   0.95431   0.89680      1926
           2    0.62745   0.47880   0.54314       401

    accuracy                        0.81888      2479
   macro avg    0.49110   0.47770   0.47998      2479
weighted avg    0.75865   0.81888   0.78461      2479
```

CODE:

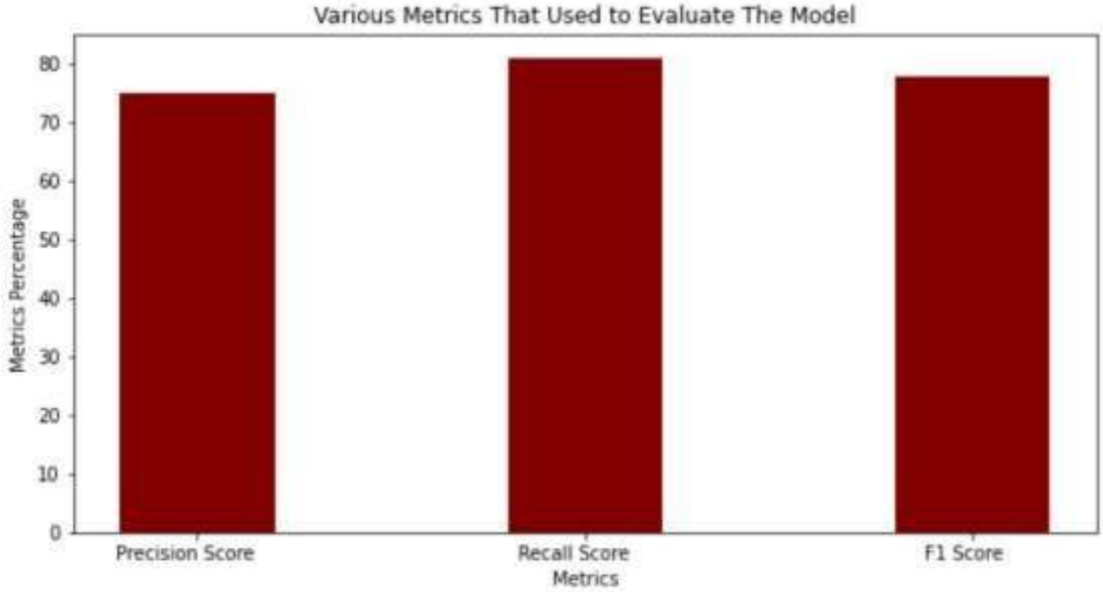
7.RESULT AND DISCUSSION

The proposed Bi-Directional architecture got an accuracy score of 0.8866 in the validation set, which has the 30 percent of the entire dataset. A custom call back was setup and the number of epochs the model ran was 23. The training accuracy the model acquired for the corresponding validation accuracy score is 0.9692. The weighted average Precision Score for the model with the particular test set is 0.75865, the weighted Recall score is 0.81888, the weighted F1 Score is 0.78461. For a dataset with 70-30 split these performance metrics denote that the model is not overfitted and this precision, recall and f1 score denotes that the number of false positive cases are less. From the graph it is clear that the model is headed towards overfitting as the training accuracy is keep on increasing and the validation accuracy is saturated over a period of epoch as the data is more ambiguous and the variety of data is not sufficient for the model to train. The custom callback set up has been efficiently stopped the model from over fitting and stop the model at the saturation point. This model has been able to work with more ambiguous data which require the contextual analysis from both the direction of the sentence as the precision, recall and f1 score are sufficient for a 7030 split.

Plot of Training and Validation Accuracy W.r.t Epoch



Plot of Performance Metrics



8.SUMMARY

In this project we got a dataset from Crowdflower company which has been labeled the tweet data into 3 classes namely “Hate Speech”, “Offensive Tweet” and “Neither of Those”. With this dataset, data wrangling process like removing special characters, removing stop words. Then the words are tokenized. Further these tokenized words are converted in to text to sequences. Then using the Stanford GloVe word to vector model all the words in the founded vocabulary have been converted into vectors. And finally, word embedding matrix with custom padding are made. A total of 70-30 split has been made for training and testing. Then the proposed architecture is defined and trained using custom callbacks. With 23 epochs the model achieved and training accuracy of 0.9692 and a validation accuracy of 0.8866 have been achieved. The weighted average Precision Score for the model with the particular test set is 0.75865, the weighted Recall score is 0.81888, the weighted F1 Score is 0.78461. For a dataset with 70-30 split these performance metrics denote that the model is not overfitted and this precision, recall and f1 score denotes that the number of false positive cases are less. This model will be helpful to identify individuals in social networks who are spreading hate speech and offending anyone online can be identified.

9.REFERENCES

1. https://www.researchgate.net/publication/347757898_A_Systematic_Literature_Review_of_Different_Machine_Learning_Methods_on_Hate_Speech_Detection
2. https://www.researchgate.net/publication/342263430_Bidirectional_Long_Short_Term_Memory_Method_and_Word2vec_Extraction_Approach_for_Hate_Speech_Detection
3. https://www.mdpi.com/2076-3417/10/23/8614/pdf
4. https://arxiv.org/pdf/1809.08651
5. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0221152
6. https://shura.shu.ac.uk/26018/1/IEEE_WCCI_Artificial_Intelligence.pdf
7. https://link.springer.com/article/10.1007/s42979-021-00457-3#Sec15
8. https://eprints.whiterose.ac.uk/128405/8/chase.pdf
9. https://thesai.org/Downloads/Volume11No8/Paper_61Automatic_Hate_Speech_Detection.pdf
10. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0221152#pone.0221152.ref009