# Predicting the Type and Target of Offensive Posts in Social Media

Submitted in partial fulfilment of the requirements of the degree of
**Bachelor of Technology**

by

**Ranjeet Kumar 177250**

**Avadhoot Hede 177209**

**Deepak Maurya 177215**

**Supervisor:**
**Dr.Balaprakasa Rao Killi**
Assistant Professor
**NIT warangal**

**Computer Science and Engineering**
**NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL**

**2020-21**

# APPROVAL SHEET

The project work entitled **"Predicting the Type and Target of Offensive Posts in Social Media"** by **Ranjeet Kumar(177250)**,**Avadhoot Hede(177209)** and **Deepak Maurya(177215)** is approved for the degree of Bachelor of Technology in Computer Science and Engineering.

**Examiners**

__________________________

__________________________

__________________________

**Supervisor**

[illegible]

__________________________

**Dr.Balaprakasa Rao Killi**
Assistant Professor

**Chairman**

__________________________

**Prof.P Radha Krishna**
Department of Computer Science and Engineering

Date:______________________

Place:______________________

# DECLARATION

We declare that this written submission represents our ideas in our own words and where other's ideas or words have been included we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and can also evoke penal action from the sources which have thus not been properly cited or from where proper permission has not been taken when needed.

Ranjeet Kumar

Ranjeet Kumar-177250

[illegible]

Avadhoot Hede-177209

Deepak Maurya

Deepak Maurya-177215
Date: 9th May,2021

# CERTIFICATE

This is to certify that the project work entitled **“Predicting the Type and Target of Offensive Posts in Social Media”** is a bonafide record of work carried out by **Ranjeet Kumar(177250), Avadhoot Hede (177209)** and **Deepak Maurya(177215)**, submitted to the faculty of “Computer Science and Engineering Department”,in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in “Computer Science and Engineering” at National Institute of Technology, Warangal during the academic year 2020-21.

**Dr.Balaprakasa Rao Killi**
Assistant Professor
Project Guide
Department of CSE
NIT Warangal

**Prof.P.Radha Krishna**
Head of the Department
Department of CSE
NIT Warangal

# Abstract

Posting offensive contents on social media has been a concern over the last few years. As offensive content has become so common on social media, there have been a lot of research to identify potential messages. This has created many problems because of the immense reputation of the social media websites like Twitter and Facebook.The real objective lies in the fact that the model uses and step up the prediction of offensive posting contents in order to make smoother the path of the appropriate actions and limitations of the offensive tweets.

We replicate this work hierarchically, identify the type and target of the offensive tweets on social media. We will use the publicly available OLID dataset 2019 (Offensive Language Identification Dataset) for the research project, a new dataset of tweets described with offensive content using an three-dimensional annotation system performance of several different types of machine learning and deep learning models in OLID.

We also did a comprehensive analysis of comparisons of various sampling techniques, Feature Extraction Methods and Model Building Algorithms. We here used SVM(Support Vector Machine), BiLSTM(Bidirectional Long-Short Time Memory) as well as BERT(Bidirectional Encoder Representation from Transformer ) as the models to be trained using the given dataset.

# List of Figures

# List of Tables

# Contents

# Chapter 1

# Introduction

## 1.1 Text Classification

Text Classification is a method of classifying the texts into already defined labels/categories. For example, we can classify blogs into many different categories like Politics, education or Sports, etc., emails into non-spam or spam emails, etc. It may be performed with the aid of Natural Language Processing using several different classification Algorithms like CNN, SVM and BiLSTM etc.

## 1.2 Natural Language Processing

NLP stands for natural language processing. This is a learning field of artificial intelligence that focuses on empowering computers to make them understand human language. The problem in understanding human language is that it is not a collection of rules and interpreting the context of a conversation and reading between the lines of a completely different game.

Although, the latest developments in Machine Learning and Deep Learning with the assistance of Neural Networks as well as easy to be used python models has unlocked for us the way to write our code to make computers interpret complicated Human Languages.

## 1.3 Offensive posts detection

Offensive content is rife in the media and is, therefore, a serious concern for online communities, government agencies as well as social media platforms. One of the most general methods used to address this problem is to train the system to detect offensive contents, which may be present at that time. we propose a three-level hierarchical descriptive schema that categorizes the following given three levels:

A Offensive Language Detection

B Categorization of Offensive Language

C Offensive Language Target Identification

We apply the schema on Offensive Language Identification Dataset (OLID), a large and new dataset of English posts with very high quality annotated type and target of offensive posts.

We do the experiments on the OLID dataset by using several different machine learning and deep learning models for each level of the annotation.

- SVM
- BiLSTM
- BERT+LSTM

# Chapter 2

# Problem Statement

This chapter explains the definition of the problem. This section elaborates on the motivation to choose the topic of offensive posts discovery over social media and the reason for selecting OLID data for our problem followed by selecting a small activity within the dataset that selects the project. The following section will describe OLID data and give us an understanding of the feature and distribution of the targets.

## 2.1 Problem Model and Formulation

These days with the escalating use of social media platforms such as Twitter and Facebook, we frequently watch people abusing the freedom of speech. Some people try to use this forum to post offensive tweets about a group or individual. It can also harm the mentality of the communities or individuals. By looking at the susceptibility of the topic, our goal is to address the problem of finding offensive languages on social media using cutting-edge methods in Natural Language Processing, Machine Learning and Deep Learning. To continue our topic, we select the Offensive Language Identification Dataset (OLID) got released in 2019. There are three subtasks of the new dataset that includes detection, type and targets prediction in offensive tweet respectively.

In the OLID database, we use the schema of classification into three categories to differentiate between whether the language is offensive or not (A), its types (B), and its targets (C).

1. **Level A: Offensive language Detection**
   It differentiates among the following type of tweets:
   i. **Not Offensive (NOT)**: Post without offending or insulting.
   ii. **Offensive (OFF)**: Tweets containing any kind of inappropriate language(insult) or targeted, indecent or indirect offense .It includes threats, insults, and posts that contain foul or abusive language.

2. **Level B: Categorization of Offensive Language**
   It discriminates the following types of offense:
   i. **Targeted Insult (TIN)**: Post that contains insults / threats to individuals, groups, or others.
   ii. **Untargeted (UNT)**: Posts containing unintentional swearing. Typically obscene posts are not targeted, but contain inappropriate language.

3. **Level C: Offensive Language Target Identification**
   It discriminates the threats targets:
   i. **Individual (IND)**: This could be a celebrity, a named person or an unnamed participant in a conversation. Insults and threats of this type are frequently described as cyberbullying.
   ii. **Group (GRP)**: Posts addressed to a group of persons deemed to be united by the same nationality, gender or sexual regulation, political affiliations, religious beliefs, or any other general practice. Most of the threats and insults against the group are similar to what is often taken as hate speech content.
   iii. **Other (OTH)**:The target for this offensive post is not among the above two categories (e.g. state, organization, event, or problem)

## 2.2 Dataset Description

The OLID dataset defines each tweet indicator/post in the 3 level annotation system which means that each event has a label with 3 related classes. The first level specifies if the post is annoying (OFF) or not (NOT). The second level of the identifier represents the types of attack twitter that can be targeted with insults (TIN) or un-target (UNT). The third level recognizes the targets of the most categorized tweet, group (GRP), individual (IND) or other (OTH).

OLID dataset is a collection of 14,100 defined posts that are available. The training dataset contains 13240 commentary posts while the test division contains 860 tweets. Every problem can be constructed as 3 small tasks to find offensive contents, identify the types of offensive contents and make categorize targets. The distribution of the OLID database is shown in the table and the distribution of the training set is shown below in the order of the annotations.

Table 2.1: Dataset Overview

| ***A*** | **B** | **C** | **Train** | **Test** | **Total** |
|---|---|---|---|---|---|
| OFF | TIN | IND | 2407 | 100 | 2507 |
| OFF | TIN | OTH | 395 | 35 | 430 |
| OFF | TIN | GRP | 1074 | 78 | 1152 |
| OFF | UNT | - | 524 | 27 | 551 |
| NOT | - | - | 8840 | 620 | 9460 |
| ALL | | | 13240 | 860 | 14100 |

1. **For sub-task A** The distribution for each class for training at level A is given below
   - OFF – 4480 tweets(33.23%)
   - NOT - 8840 tweets(66.76%)
2. **For sub-task B** The distribution for each class for training at level B is given below
   - TIN - 3876 tweets(88.09%)
   - UNT - 524 tweets(11.90%)
3. **For sub-task C** The distribution for each class for training at level C is given below
   - IND – 2407 tweets(62.10%)
   - GRP – 1074 tweets(27.70%)
   - OTH – 395 tweets(10.19%)

# Chapter 3

# Related Work

Sentiment classification regarding offensive posts or toxic comments has been intensively researched in recent years, generally in the context of social media data where researchers have to build various ML models to overcome the problem of sentiment analysis of toxic comments. There are lots of research paper have been published for sentiment analysis of toxic comment we will discuss some papers related to our task.

Offensive Comment classification research firstly began with Yin et al's application of combining TF-IDF with contextual features [1]. The performance of the TF-IDF model with a simple TF-IDF model had compared and resulted in a 6% improvement in the F1 score of the model classifier on chat-style datasets[2].

The paper[3] describes experimental results that apply a support vector machine [4] to a benchmark dataset to train sentiment classification. N-gram and different weighting schemes were used to extract the most classical features. It also searches for chi-square weighting features to select informational features for classification. Experimental analysis suggests that the selection of the chi-square feature can lead to a significant improvement in classification accuracy.

Yu and Wang [5] have proposed a refined word vectors model which may be used on pre-trained word vectors (e.g. Word2vec[6] and Glove[7]) to improve the capture of emotional information. The refinement model was based on adjusting the vector word representations so that they were close to both identical and emotional words and very distant from the different words. Word embedding from refined models (Re), Re (Word2vec), and Re (GloVe) respectively, advanced Word2Vec, as well as GloVe by 1.7% and 1.5%, rated above all categories of classified binary editing, and both developed at 1.6%

of the fine is the division of grain. These results indicate a potential improvement in our model that may result in further development of word vectors (to make use of the refinement model) that incorporates more semantic details.

Chu and Jue [4] compare the effectiveness of various deep learning strategies for this problem, especially using character and word embedding. They have tested the performance of RNN[8] with LSTM[9] and word vectors. paper[10][11] presents sentiment analysis of social media using BiLSTM with a multi-head attention mechanism. Lots of sentiment tasks are performed by BiLSTM.

In this paper[12], BERT[13] model was build annotated datasets(provided by HASOC 2019 Competitions) containing posts from social media in English, German, and Hindi (including code-mixing) .making a multilingual deep learning model for classification of hate speech and offensive language in social media.

Zampieri, S. Malmasi sees the problems of offensive language detection over social media as universal alternately focusing on some kind of offensive content over the internet as seen in last studies. They emerged with a new three-levels dataset of an annotation program that allows the researcher to dig deeper into the topic. Subtask A to determine if the posts are offensive or not. Subtask B also recognizes the type of offenses if untargeted or targeted. Then, Subtask C classifies the offensive posts into one of three categories individual, group, or other[14].

we continue to expand the previous analysis to test the effectiveness of ML and deep learning methods: SVM, LSTM, Bi-directional LSTM and BERT. Here we continue to extend the previous analysis and investigating the effectiveness of different machine learning and deep learning methods: SVM, LSTM, BILSTM and BERT.

# Chapter 4

# Design

## 4.1 SVM

SVM is one of the most controlled and versatile machine learning algorithms.It is used to do the work of classification and regression but here we will talk about the work of classification. Normally moderate data set is preferred.[4]

### 4.1.1 Working Mechanism

The main goal of SVM is to get the right hyperplane that separates the data points equally into two parts by enlarging this margin. To distinguish two categories of data points, there are several hyperplanes to choose from. Our goal is to find a plane with a higher limit, i.e. a higher distance among the data points for both categories. Increasing the margin gives some reinforcements so that the new data points are separated with greater confidence Flying planes is the decision-making parameters that help separate data points. Datapoint falling on any side of a hyperplane may be caused by different categories. Also, the size of the hyperplane is dependant on several factors.

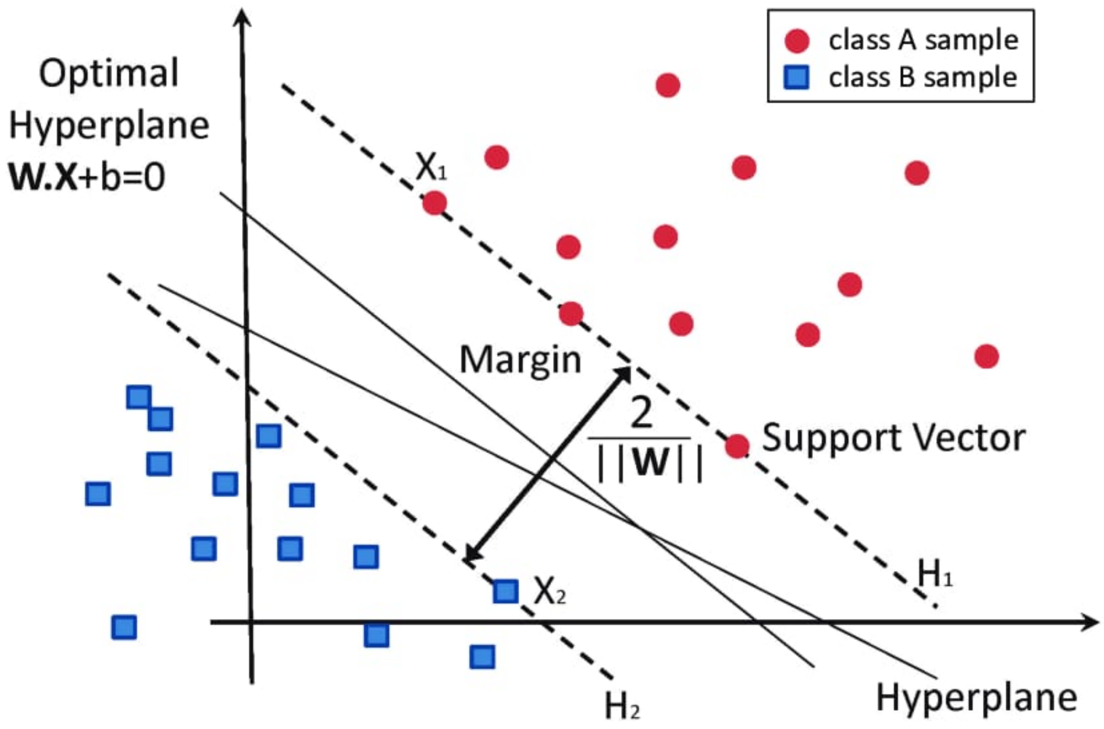


Figure 4.1: Classification of data by SVM
Source: Refer [15]

When the predicted values and real values are of same sign,the cost is 0.If not, then we calculate the amount of the loss. We also include standard parameters like regularization for performing the cost function. The purpose of the regularization is to measure margin enlargement and loss.

$$min_w \lambda \|w\|^2 + \sum_{i=1}^{n} (1 - y_i \langle x_i, w \rangle)_+ \tag{4.1}$$

Now that we have the loss function, we take the output in part in terms of weight gain gradients. Using gradients, we can update our instruments.

$$\frac{\delta}{\delta w_k} \lambda \|w\|^2 = 2\lambda w_k \tag{4.2}$$

$$\frac{\delta}{\delta w_k} (1 - y_i \langle x_i, w \rangle)_+ = \begin{cases} 0, & \text{if } y_i \langle x_i, w \rangle \geq 1 \\ -y_i x_{ik}, & \text{else} \end{cases} \tag{4.3}$$

Where there is no differentiation, i.e. the model accurately predicting the phase of our data points, we should only modify the weights using a gradient from the standard parameters.

$$w = w - \alpha \cdot (2\lambda w) \tag{4.4}$$

Where there is a variance, i.e. our model commits an error in the prediction of our datapoint section, we involve the loss and the regularization for performing the gradient updates.

$$w = w + \alpha \cdot (y_i \cdot x_i - 2\lambda w) \tag{4.5}$$

### 4.1.2 Algorithm

**1.** read the dataset file

**2.** applying preprocessing on dataset( using nltk library)

**3.** extracting feature from preprocessed dataset

**4.** split the dataset into training and testing

**5.** specify learning rate and optimizers for SVM model

**6.** training the SVM model

**7.** testing the SVM model

**8.** print classification report

## 4.2 BiLSTM

The traditional RNN uses a backpropagation (BPTT) training algorithm. As training progresses, the remaining errors which need to be corrected will reduce significantly, leading to a slight recovery of network weights, which do not show the effect of long-term memory. So, Hochreiter et al. developed LSTM in 1997 [9].It is a special type of RNN model which solves the problem of vanishing gradient [16] of RNN models.

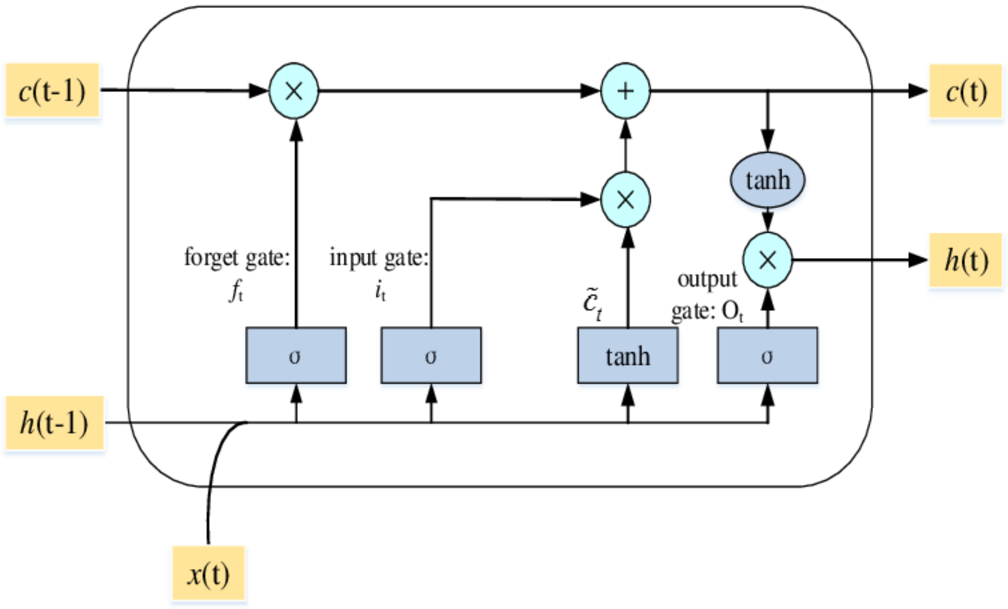


Figure 4.2: Structure of LSTM unit
Source: Refer [17]

### 4.2.1 LSTM

Compared to RNN[8], LSTM adds up a cell state c , and manages c's value by three gate units: input gate $i_t$ ,forget gate $f_t$ and output gate $o_t$. Depending on the current input $w_t$ and the last hidden state $h_{t-1}$ , the forget gate is calculated using the sigmoid function to ascertain what informations the cell state $c_t$ at the present moment would take over from the cell state $c_{t-1}$ at the last moment, as given in equation (1)

$$f_t = \sigma(q_f[h_{t-1}, w_t] + b_f) \tag{1}$$

The input gate is being used to modify the status of the cell, and any information that needs to be stored is available with sigmoid function, as mentioned in the equation (2), the temporary cell state $\tilde{c}_t$ is calculated by the tanh function, as in the equation (3), then adds product of $i_t$ and $\tilde{c}_t$ and product of $f_t$ and $c_{t-1}$ to get $c_t$ , as in the given equation (4)

$$i_t = \sigma(q_i[h_{t-1}, w_t] + b_i) \tag{2}$$

$$\tilde{c}_t = \tanh(q_c[h_{t-1}, w_t] + b_c) \tag{3}$$

$$c_t = i_t * \tilde{c_t} + f_t * c_{t-1} \tag{4}$$

The output gate finds out the effect of $c_t$ on the hidden state $h_t$ , as in the equation (5), then gets the output $h_t$ from the unit depending on $o_t$ and $c_t$ , as in the equation (6)

$$o_t = \sigma(q_o[h_{t-1}, w_t] + b_o) \tag{5}$$

$$h_t = o_t * \tanh(c_t) \tag{6}$$

where $\sigma$ is the sigmoid function,b is bias vector and q is weight matrix.

### 4.2.2 BiLSTM Advantage

Long Short term memory only captures sequence's historical information, but that's not sufficient. If one can access the future information the way one would do in the past, this will be very useful as now you have full information of sequence. The bidirectional LSTM contains a forward and a backward LSTM layer. The working principle of the operation is given as follows: the forward layer records the past details of sequence; the backward layer records the future information of the sequence. These two layers are connected to a single output layer. The great highlight of the construction is that the sequence details are considered in full. Let's consider that the input time t is the word embedding weight, at time t-1, the output of this forward hidden unit is $\overrightarrow{h_{t-1}}$, and the output of the backward hidden unit is $\overleftarrow{h_{t+1}}$, Then the output of backward and hidden unit at time t is given as follows :

$$\overrightarrow{h_t} = L(w_t, \overrightarrow{h_{t-1}}, c_{t-1}) \tag{7}$$

$$\overleftarrow{h_t} = L(w_t, \overleftarrow{h_{t+1}}, c_{t+1}) \tag{8}$$

where $L()$ indicates the operation of the hidden layer of the LSTM.The forward output vector is $\overrightarrow{h_t}$ and the backward output vector is $\overleftarrow{h_t}$, and these should be merged to get the text features. It should also be noted that H refers to the number of the hidden layer cells:

$$H_t = \overrightarrow{h_t}||\overleftarrow{h_t} \tag{9}$$

### 4.2.3 Functional Diagram

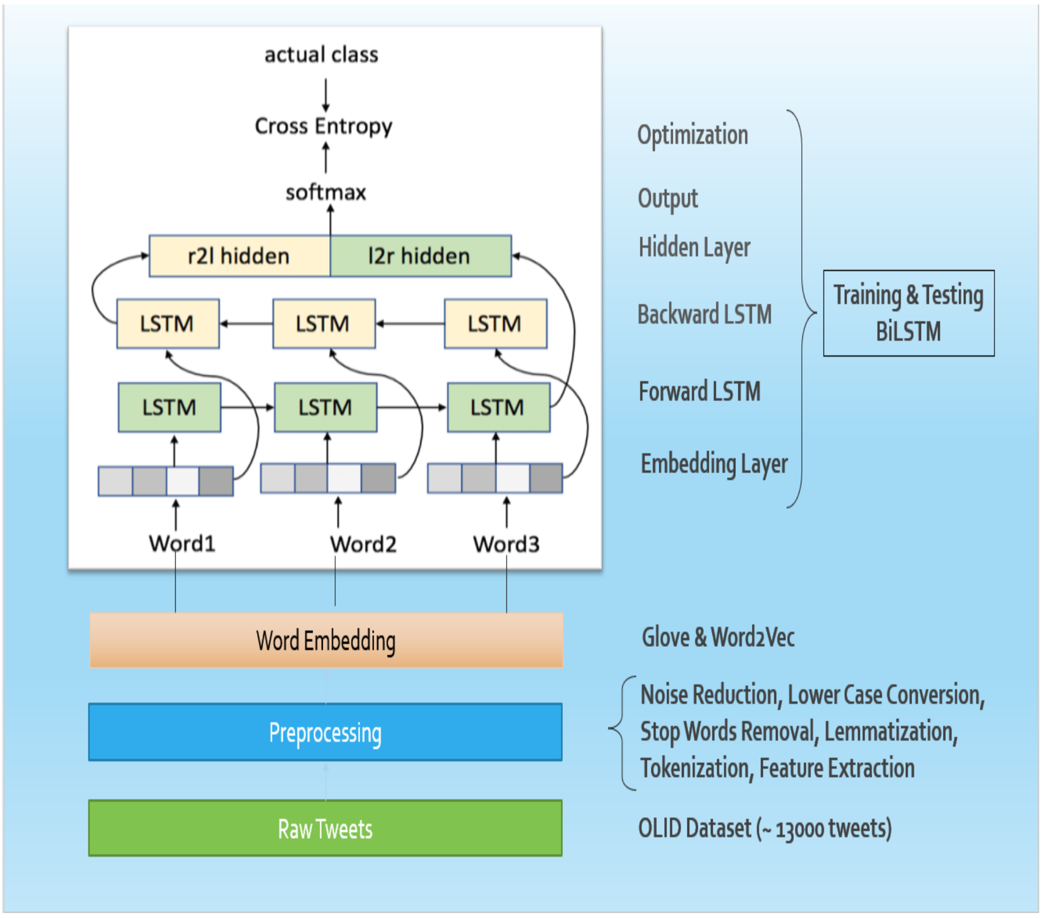


Figure 4.3: Function Design of BiLSTM Model

### 4.2.4 Algorithm

1. read the data set file
2. preprocess the dataset using nltk library
3. load the pre-trained Glove or Word2Vec word vector:

    i. If the current word is present in the model,return the respective word vector;

    ii. If the word is not present in the model,then return 0;

    iii. Now, check the shape of the dataset.

4. split the dataset into training and testing
5. Input training dataset:
    i. Build a Sequential Model
    ii. Add two BILSTM layer (number of LSTM unit = 32 or 64)
    iii. Add a Dense layer (number of neurons = 100)
    iv. Add classification layer, for binary using sigmoid and for multilabel using softmax
    v. Choose a optimizer for training the model
    vi. train the model
6. Input test data set:
    i. predict the output on test dataset
    ii. print precision,recall,accuracy,loss

## 4.3 BERT

BERT represents the Bidirectional Encoder from Transformers[18].

BERT is considered for pretrained language representations, which means we train a common objective model of "understanding language" in large corpus text (like Wikipedia) and use the model to perform sub-NLP tasks which we care about (like Sentiment Analysis). BERT surpasses former approaches because it is the first and the most advanced, most advanced NLP training system.

Unsupervised states that BERT is trained using the plain corpus only, which is predominant because a large amount of clear text information is publicly available over the internet in many languages.

Already-trained presentations may be out of context, and contextual presentations may continue to be inconsistent or paralyzed in both cases. Non-contextual models like GloVe or Word2vec produce the same representation of "word embedding" for every word in the vocabulary, so the bank will have the same representation in the bank. Content models produce individual word representations based on the other words in the sentence.

BERT is built on the latest work in pre-trained state presentations including Semi-supervised Sequence Learning, ELMo, Generative Pre-Training, and ULMFit but more importantly, the types are not compatible and misunderstood. It means that each word is formed by the word only using the words on its left or right. For example, in this sentence "I made a bank deposit" the unidirectional representation of bank is dependant only on I made a but not deposit. Some previous work combines the representations from different left-context or right-context models but in only a "shallow" manner. BERT represents "bank" using both contexts left and right contexts — I made a deposit — start from the very down of a deep neural network, so it's in-depth bidirectional.

### 4.3.1 BERT Architecture

The BERT Architecture is built over Transformers. We presently have two varieties given: BERT Base: 12 layers (transformer blocks), 12 attention heads, and 110 million borders Big BERT: 24 layers (transformer block), 16 attention heads, and 340 million borders.

Here we have used the BERT BASE model for our specific work. It contains a stack of 12 encoders that generate embedded content of sentences or words.

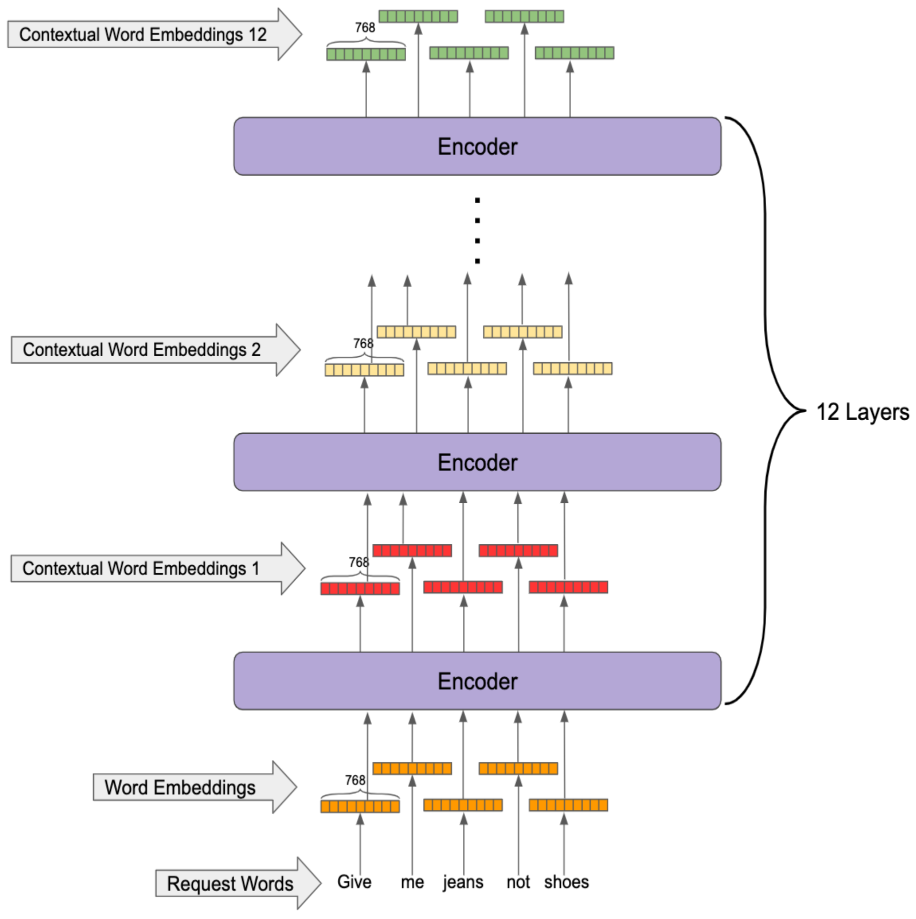


Figure 4.4: BERT Base Model Configuration
Source: Refer [19]

#### 4.3.1.1 Encoder

The encoder analyzes the input sequence and compiles the data into a context vector (also known as a contextual embedding). This representation is an outline of the definition of the whole sequence of the source. The Encoder contains a very important Self-attention layer that keeps the score of the relationship between different words in sequence and the feed-forward layer. around them, followed by Layer-Normalisation

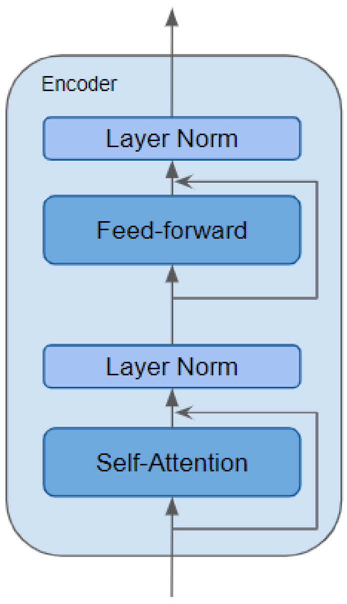
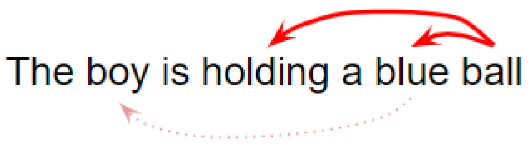


Figure 4.5: Encoder Structure
Source: Refer [20]

1. **Self Attention -** The key to the Transformer's ground-breaking performance is its use of Attention[10].While processing a word, Attention allows the model for focussing on other words present in the input that are closely related to that word.

   The Transformer architecture uses self-attention by relating every word in the input sequence to every other word. eg. Consider two sentences:

   i. The cat drank the milk because it was hungry.
   ii. The cat drank the milk because it was sweet.

   In the first sentence, the word 'it' refers to 'cat', while in the second it refers to 'milk. When the model processes the word 'it', self-attention gives the model more information about its meaning so that it can associate 'it' with the correct word.

2. **Feed forward Layer -** It takes input from the self-care layer. sending its output to the top of the next Encoder.

3. **Layer Normalization -** Normalising activates the previous layer of each instance provided in a batch independently, rather than passing a batch such as Batch Normalization. e.g. applies a change that keeps the expression within each example close to 0 and the normal opening deviation is closer to 1.

### 4.3.2 Working Mechanism

BERT uses Transformer, an attention-grabbing method that studies the relationship of status among words in a text. With its vanilla method, Transformer incorporates given two distinct modes - a text encoder and a decoder that generates performance predictions. Since the purpose of BERT is to produce a language model, only the encoding method is required.

In contrast to directing model, which reads text inputs sequentially (from left to right or right to left), the encoder reads the word sequence simultaneously. It is therefore bidirectional, even if it can be right to say that it is incorrect. This feature allows the models to read the words' context based on the surrounding environment.

### 4.3.3 Functional Diagram

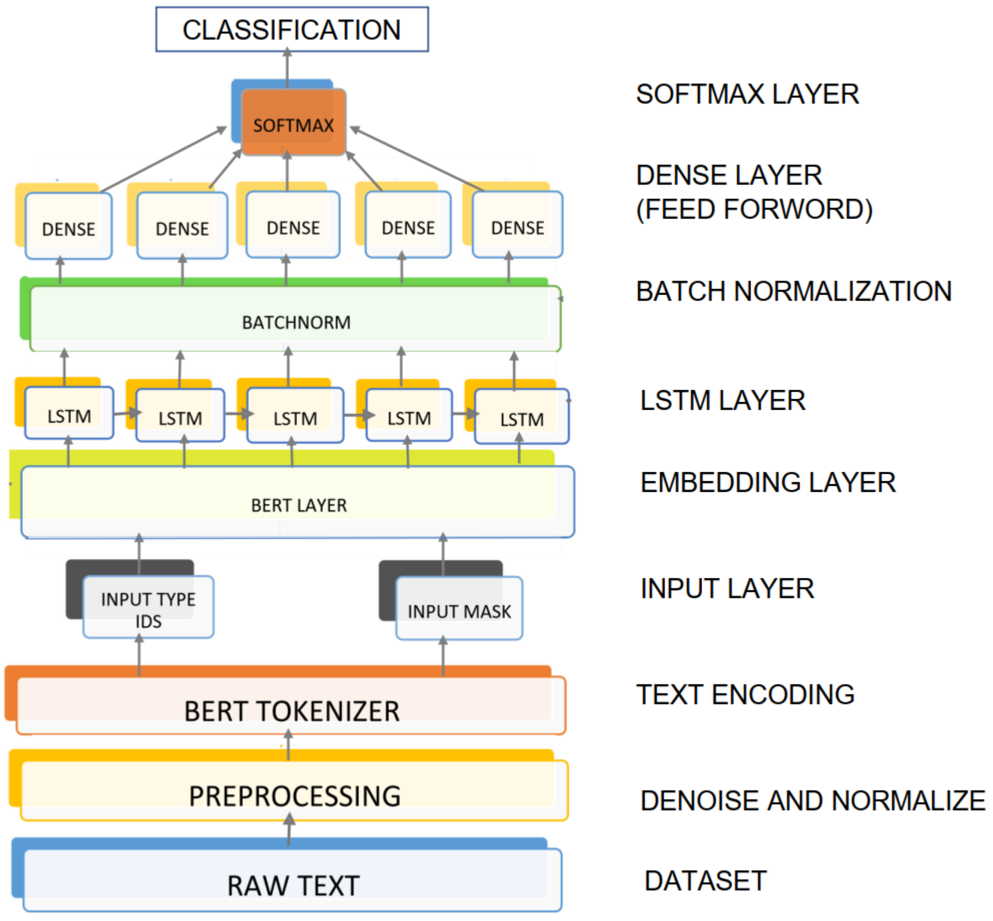


Figure 4.6: BERT with additional LSTM

### 4.3.4 Algorithm

1. read the data set file
2. preprocess the dataset using nltk library
3. Encode the sequence of dataset using Bert Tokenizer
4. Build a pipeline of dataset for training

5. Input train data set:
    i. Build a Sequential Model
    ii. Add Bert layer for Contextual Embedding
    iii. Add LSTM layer (number of lstm cell )
    iv. Apply Batch Normalization
    v. Add a Dense layer (feed forword network)
    vi. Add a softmax function for classification purpose
    vii. Choose a optimizer for training the model(i.e. Adam)
    viii. train the model
6. Input test data set:
    i. predict the output on test dataset
    ii. print precision,recall,accuracy,loss

# Chapter 5

# Implementation

This chapter starts with data preprocessing for our dataset making our data more insight full, followed by different feature engineering techniques and finally the model detail which we have implemented.

## 5.1 Data Preprocessing

Data preprocessing is the heart of any NLP task. It removes the noise and unnecessary information making our dataset more confined and meaningful. There are various kinds of methods to preprocess text data.

Here we have mentioned the different steps of preprocessing for our text data.

### 5.1.1 Lower Case Conversion

To lowercase, the text data to have all the data in a uniform format and to make sure “NLP” and “nlp” are treated as the same. We will use a python lower() function that converts all uppercase strings to lowercase strings.

### 5.1.2 Removing Punctuation

As punctuation doesn’t add any extra information or value. Hence removal of all such instances will help reduce the size of the sequence and save memory space for faster operation. The simplest way to do this is by using the regular expression and replace() function in Python.

### 5.1.3 Removing Stopwords

Stop words are very frequent words that have less meaning or no meaning in comparison to other keywords We will use the nltk library for stopwords to achieve this operation.

### 5.1.4 Standardizing text

Most of the text data is in the form of either customer reviews, blogs, or tweets, where there is a high chance of people using abbreviations and short words, saving time in typing, and convey the same meaning. This may help the downstream process to easily understand and resolve the semantics of the text. We use our custom dictionary to look for short words and abbreviations.

### 5.1.5 Tokenizing text

Tokenization splits the text into the smallest meaningful units. There is a sentence tokenizer and a word tokenizer. word tokenizer which is a mandatory step in text preprocessing for any kind of analysis. There are many libraries to perform tokenization like TextBlob, NLTK, and Spacy. We will use word_tokenize from the NLTK library

### 5.1.6 Stemming

Stemming is a process of extracting a root word. For example, “go,” “goes,” and “going” are stemmed into “go”.

### 5.1.7 Lemmatizing

Lemmatization is a process of extracting a root word from its vocabulary. For example, “bad,” “worse” or “worst” is lemmatized into bad.

- The stemmed word for caring is car.
- The lemmatized word for caring is care.

Lemmatization has better results in comparison to stemming. So we have chosen lemmatization over stemming in our data preprocessing steps. we will use WordNet Lemmatizer from the nltk library.

### 5.1.8 Handling Hashtag, URL Link and USER Tag

we will remove all hashtag and URL links. Replace the user tag with a particular word because these have no semantic meaning and have no effect on our sentiment analysis. We will not remove the user tag completely because this can help in predicting whether a post is targeted or not. So we will convert the user tag to AT_USER by using regular expressions.

### 5.1.9 Removing Special Characters

They don't have many contributions to the sentiment of the tweet. Special characters are all non-alphanumeric types. We can remove them by using a standard regular expression pattern.

### 5.1.10 Number Conversion

They have no contribution to predicting sentiment. We can remove them or convert them to the corresponding word. Using the inflect library we will replace every number with a word.

### 5.1.11 Removing Emoji

Although they convey some meaning like a laugh, cry, sad. But for the toxic or offensive posts, they have almost no effect. Emoji is represented by Unicode characters. So we remove all emoji in our raw tweets by using a re pattern.

### 5.1.12 Removing White Space

After performing different preprocessing steps extra spaces may add. So we will remove those whitespaces to increase computational efficiency. We can remove additional white spaces by using a standard regular expression pattern.

## 5.2 Feature Extraction

We need feature extraction from text for every NLP task because our model doesn't understand text data. So we will convert text data to some feature vector using the feature extraction method. There are lots of feature extraction methods (converting text to features). Some traditional feature engineering methods are One Hot encoding, Count vectorizer, N-grams, Co-occurrence matrix, TF-IDF, and Word embedding. We have used TF-IDF and Word embedding. So we discuss these more in the following section.

### 5.2.1 TF-IDF

TF-IDF stands for Term Frequency-Inverse Document Frequency. In All other feature engineering methods except word embedding, if a particular word appears in all documents then it has more importance. That will not give a good feature vector for our analysis.
TF-IDF‘s main objective is to show how important a word is to a document in a dataset and so that normalized words occur frequently in all text documents.

#### 5.2.1.1 Term Frequency (TF)

It is the ratio of the number of a word present in a sequence to the size of the sequence. TF is simply capturing the weightage of the word without considering the length of the document. I.e. a word having a frequency of 4 while sentence size is 15 words is not the same case as sentence size is 50 words. The first one should get more weightage than the second case. So TF solves this case.

#### 5.2.1.2 Inverse Document Frequencey(IDF

IDF can be calculated for every word using the formula

$$IDF = \log(\frac{N}{n})$$

Where N is a number of rows and n is the number of rows in which the word was present. It measures the rarity of a term. Frequent words like ‘the’ show up in almost every document but rare words will not show up in all documents. If any word appears in every doc then that word will not be important to us since it will not help to classify. IDF handles this problem.

TF-IDF is the multiplication of TF and IDF so both problems are handled which makes our prediction relevant.

### 5.2.2 Word Embedding

Every other feature engineering method has not shown good results in capturing the meaning of words. They fail very badly because they consider only

the frequency of words. For semantic relation how close these words appear in a sentence. These problems were solved by word embedding.
Word embedding captures the contextual or semantic meaning of words mapping these to real value word vectors. Word embedding uses neural networks to train models and so it learns weight which can be represented as the word vector. Similar words have similar word vector representations.

#### 5.2.2.1 Word2Vec

It is a deep learning google framework to train word embedding. Using all words of the corpus and predicting the closest words. for every word in the corpus creates word vectors. It outperforms every other feature engineering method for word similarity. There is a pre-trained model of over 100 billion words for word2vec.

### 5.2.3 Glove

Glove considers both local and global statistics (word co-occurrence) to get word vectors. It is developed and maintained by Stanford University. Currently, several pre-trained word vectors are having different dimensions like 100,200 and 600.
we have to use Glove pretrained word vector of dimension 200 for word embedding to the dataset.

## 5.3 Handling Imbalance

After exploring the dataset we have found that unbalanced labeling was 33% offensive and 66% not offensive post. One label is twice the other label. Before feeding this dataset as the input you have to minimize this unbalanced otherwise your model's output will be a bias toward majority classes(labels). Figure 5.1 shows the class label imbalance of the OLID training dataset.

Table 5.1: Imbalance in Dataset

| ***A*** | **B** | **C** | **Train** | **Test** | **Total** |
|---|---|---|---|---|---|
| OFF | TIN | IND | 2407 | 100 | 2507 |
| OFF | TIN | OTH | 395 | 35 | 430 |
| OFF | TIN | GRP | 1074 | 78 | 1152 |
| OFF | UNT | - | 524 | 27 | 551 |
| NOT | - | - | 8840 | 620 | 9460 |
| ALL | | | 13240 | 860 | 14100 |

There are various kinds of techniques to overcome this problem. We will use the resampling technique to balance the class label in the dataset.

### 5.3.1 Under-sampling Technique

As the name suggests here we will decrease our class label by having more numbers to match the class label having fewer numbers in the dataset.

Random under-sampling is a method that removes data of class labels having more numbers until the distribution of both class labels is balanced. In this case, the size of our dataset decreases because we already have a small dataset So we will not choose this method because the model will be trained very poorly.

### 5.3.2 Over-sampling Technique

As the name suggests we will increase our class label having less number to rebalance our dataset until both class label distribution have similar value. The random over-sampling technique increases the class label by randomly duplicating having less number until distribution becomes equal. This may lead to overfitting to our model. We have used this technique to balance our dataset.

## 5.4 Implementing Model

We have built models of both types of machine learning and deep learning. The deep learning model performs better than the machine learning model. We have used SVM which is a machine learning model and BiLSTM and BERT which is a deep learning model.
Model created for classification of an offensive post, their type and target:

### 5.4.1 SVM

SVM is a supervised machine learning algorithm that can be used for both regression and classifications. SVM's objective is to build the hyperplane so that marginal width becomes maximum.SVM performs very well for higher-dimensional data. The combination of TF-IDF with SVM makes the model predict class labels for text data.

### 5.4.2 BiLSTM

It stands for bidirectional long short term memory.LSTM has a cell state which stores long-term information which flows through all LSTM units. It has a forget gate that tells how much previous information to be dropped and adding it with the input gate calculates the next hidden state.

BiLSTM consists of two LSTM layers, one moving forward to store past information and another moving backward to store future information, and combining both information, it captures the overall context of the sequence. In this model pre-trained BERT model is used as an embedding layer to capture the semantic and contextual meaning of sequence and pass it to the LSTM layer. Now our model can understand sequence very well.

### 5.4.3 BERT+LSTM

BERT is based on transformer architecture. The transformer consists of an encoder and a decoder layer. Encoder captures the contextual meaning of sequence and Decoder is used for translation for the given target sequence. Because we are not going to translate so we remove the decoder layer. And group stack of the encoder which makes our BERT model. Because the model was trained on a high corpus of text, it can achieve state-of-the-art performance. By fine-tuning a pre-trained BERT model and adding an LSTM layer on top of it we perform classification for our dataset.

## 5.5 Evaluation Metrics:

Evaluation metrics are used for measuring the performance of our machine learning models and deep learning models. The metrics should be selected attentively based on the test data given. For example, if we select the accuracy of the unequal test that can be used it can provide the best possible accuracy, but in reality, the accuracy of most class dominates the accuracy of the minority category. So here the best metric would be an F1-score or a matrix of confusion to get a real understanding of the performance of the classes in the test database. To test the effectiveness of our machine learning models and deep learning to find offensive posts, we will use each category's Recall, Precision, and F1-score. The performance of the model was tested with the Macro F1 score as the test dataset is not balanced.

Some important terms to understand the further concepts better are following:

- True Positive (TP)-Count of tuples in majority class which are correctly predicted as majority class.
- False Positive (FP)-Count of tuples in minority class which are incorrectly predicted as majority class.
- True Negative (TN)-Count of tuples in minority class which are correctly predicted as a minority class.
- False Negative (FN)-Count of tuples in majority class which are incorrectly predicted as a minority class.

Evaluation metrics which we will use in determining the performance of our classification model on the OLID test dataset are given below:-

1. Confusion Matrix: Confusion Matrix is the representation in table format for evaluating the performance of a classification model about the real and predicted values. It gives the accuracy of every class which is given in the targets. It compares the actual target values with the predicted values by the classification model.

2. Recall: Recall is the measure of the true positivity rate. It says that how one may be sure that an instance that belonged to a particular class is accurately classified as positive. Given below is the formula for the precision of a class

$$Recall = \frac{TruePositive}{(TruePositive + FalseNegative)}$$

3. Precision: Precision is the measure of the truly positive predicted value out of all predicted as positives. It says that how one may be sure that an instance that was predicted as a particular class is belonging to that category. Given below is the formula for the precision of a class

$$Precision = \frac{TruePositive}{(TruePositive + FalsePositive)}$$

4. F1-Score: F1-Score is defined as the weighted average of precision and recall for a given label. Both precision and recall value must be higher to get F1-Score higher for a class. Its worst value is 0 and its best value is 1. The F1-score is calculated as:

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

5. Classification report: It gives an accurate understanding of recall, precision, and f1 score divided into existing classes in the dataset. With our problem since the dataset is imbalanced, then we do not have the general accuracy, but the assessment of this discriminatory classification may be based on the confusion matrix (indicating the accuracy for each category). And we will be considering f1 scores in all 2 classes that say how much the precision and recall values are balanced. In addition, this report also makes an average of the weight of precision and recall and f1 score in each category in the target category. We will therefore consider the weighted average score and the macro ratio of f1 scores in both classes as one of the test metrics.

# Chapter 6

# Results and Discussion

We have implemented a three-level hierarchical model to carry out each task of classification of tweets. We created a model based on Support Vector Machine (SVM), a model based on BiLSTM, and one based on BERT. Now, here we will try to do the comparative analysis of all three models based on Accuracy, Precision, Recall, Weighted F1-score, and macro F1-score for each level separately. All this analysis will help us to come up with the best model for predicting the offensive tweets with our dataset OLID.

## 6.1 Level-1 Task Classification Result:

Here, the tweets are classified into two labels offensive (OFF) and Not-offensive (NOT). After calculating the performance of the models we implemented and trained, the following results were obtained:-

Table 6.1: Result table for the Level-A classification

| | **NOT** | | | **OFF** | | | **Weighted Average** | | |
|---|---|---|---|---|---|---|---|---|---|
| **Model** | **P** | **R** | **F1** | **P** | **R** | **F1** | **P** | **R** | **F1** |
| **SVM** | 0.75 | 0.97 | 0.84 | 0.83 | 0.33 | 0.47 | 0.77 | 0.76 | 0.72 |
| **BiLSTM** | 0.80 | 0.80 | 0.80 | 0.77 | 0.77 | 0.77 | 0.79 | 0.79 | 0.79 |
| **BERT** | 0.88 | 0.95 | 0.91 | 0.88 | 0.73 | 0.80 | 0.88 | 0.88 | 0.87 |

Table 6.2: F1-Macro and Accuracy of models at Level-A

| ***Model*** | **F1-Macro** | **Accuracy** |
|---|---|---|
| **SVM** | 0.66 | 0.76 |
| **BiLSTM** | 0.79 | 0.79 |
| **BERT** | **0.86** | **0.88** |

Here, we get the BERT model as the best model in both accuracy(0.88) and Macro-F1 score(0.86). BiLSTM and BERT models performed much better than SVM in F1-macro score and also are better than SVM in accuracy. Though SVM accuracy is also good, it does not classify the offensive posts as offensive in most of the cases giving recall of offensive class very less as 0.33.

## 6.2 Level-2 Task Classification Result:

Here, the tweets are classified into two labels - targeted(TIN) and untargeted(UNT). After calculating the performance of the models we implemented and trained, the following results were obtained:-

Table 6.3: Result table for the Level-B classification

| | **TIN** | | | **UNT** | | | **Weighted Average** | | |
|---|---|---|---|---|---|---|---|---|---|
| Model | **P** | **R** | **F1** | **P** | **R** | **F1** | **P** | **R** | **F1** |
| **SVM** | 0.88 | 1.00 | 0.94 | 0.00 | 0.00 | 0.00 | 0.78 | 0.88 | 0.82 |
| **BiLSTM** | 0.91 | 0.85 | 0.88 | 0.83 | 0.90 | 0.87 | 0.88 | 0.87 | 0.87 |
| **BERT** | 0.86 | 0.91 | 0.88 | 0.89 | 0.83 | 0.86 | 0.87 | 0.87 | 0.87 |

Table 6.4: F1-Macro and Accuracy of models at Level-B

| ***Model*** | **F1-Macro** | **Accuracy** |
|---|---|---|
| **SVM** | 0.47 | **0.88** |
| **BiLSTM** | **0.87** | 0.87 |
| **BERT** | **0.87** | 0.87 |

Here, we get the BERT and BiLSTM models performing very well in F1-Macro(0.87) which is far better than SVM(0.47). Though SVM accuracy(0.88) is better than BiLSTM(0.87) and also BERT(0.87), it is performing very badly in classifying untargeted posts giving a very bad F1-macro score.

## 6.3 Level-3 Task Classification Result:

Here, The tweets are classified into two labels - group(GRP), individual (IND), and other(OTH). After calculating the performance of the models we implemented and trained, the following results were obtained:-

Table 6.5: Result table for the Level-C classification

| | **GRP** | | | **IND** | | | **OTH** | | |
|---|---|---|---|---|---|---|---|---|---|
| **Model** | **P** | **R** | **F1** | **P** | **R** | **F1** | **P** | **R** | **F1** |
| **SVM** | 0.68 | 0.48 | 0.56 | 0.72 | 0.93 | 0.81 | 0.00 | 0.00 | 0.00 |
| **BiLSTM** | 0.77 | 0.81 | 0.79 | 0.86 | 0.69 | 0.76 | 0.80 | 0.91 | 0.85 |
| **BERT** | 0.00 | 0.00 | 0.00 | 0.61 | 0.26 | 0.36 | 0.90 | 0.98 | 0.94 |

Table 6.6: Weighted Average,F1-Macro and Accuracy of models at Level-C

| | **Weighted Average** | | | | |
|---|---|---|---|---|---|
| **Model** | **P** | **R** | **F1** | **F1-Macro** | **Accuracy** |
| **SVM** | 0.64 | 0.71 | 0.66 | 0.46 | 0.71 |
| **BiLSTM** | 0.81 | 0.80 | 0.80 | **0.80** | 0.80 |
| **BERT** | 0.86 | 0.89 | 0.86 | 0.43 | **0.89** |

Here, we get the BERT model as the best model in accuracy(0.89) and the BiLSTM model as the best model in Macro-F1 score(0.80). SVM model's accuracy is very poor in both accuracy(0.71) and macro F1-score(0.46). BERT is performing the worst in the macro-F1 score(0.43). BiLSTM performs decently in Accuracy(0.80). SVM model can't identify the other-targeted posts whereas the BERT model faces problems in identifying Group-targeted posts.

# Chapter 7

# Conclusion and Future Work

The present chapter begins with the key inferences we have processed, in terms of understanding, problem description and dissimilar solution approach for this problem. We later conclude the section with potential areas for future work which will lead to a meaningful contribution in the research spectrum.

## 7.1 Conclusions

Listed are the main conclusions to contribute to the problem of finding abusive language on online platforms:-

1 We inspected various strategies for managing inequality in the training set, but in our database; randomized sampling efficiently reduces the difference between the majority and the class of people with precision accuracy and gives high F1 scores.

2 Immense evaluation of texts from preceding work in the field of abusive language discovery has aided us in a detailed exploration of the OLID dataset.

3 Lastly, the testing and investigation on our different models lead to the fact that BERT has achieved the best performance of obtaining a high F1 score of 0.84 on the given data.

4 We have conducted an in-depth analysis of the comparison of different ML and deep learning models to further evaluate the dataset and conclude out an important point on which the successive work can continue with the data and results obtained through the present project.

## 7.2 Future Work

Listed are few suggestions for further tackling the research problem.

- The model can be made more complex by increasing the number of layers and changing the configuration to achieve better results given time and resources.

- Indigenous speech is vastly complex and a few posts may not seem annoying on the face but are the case upon analyzing by a person's annotation. In the future an inspection to find out the synthetic and semantic features and their combination with other features. Problem solutions could be improved drastically.

- We would like to create a model for a multilingual dataset.

- We also aim to fine tunning the hyperparameters for both ML and Depp learning models to find the best values for these hyperparameters. Accuracy can be improved.

- Random sampling has done well with the data with the problem of the imbalanced dataset. It also has the disadvantage of taking a longer training time. An accurate or better approach can be devised.

# Bibliography

[1] Cheng-Hui Huang, Jian Yin, and Fang Hou. A text similarity measurement combining word semantic information with tf-idf method. *Chinese Journal of Computers*, 34:856–864, 05 2011.

[2] Dawei Yin, Zhenzhen Xue, Liangjie Hong, Brian Davison, April Edwards, and Lynne Edwards. Detection of harassment on web 2.0. 01 2009.

[3] Nurulhuda Zainuddin and Ali Selamat. Sentiment analysis using support vector machine. In *2014 International Conference on Computer, Communications, and Control Technology (I4CT)*, pages 333–337, 2014.

[4] M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, 1998.

[5] Liang-Chih Yu, Jin Wang, K. Robert Lai, and Xuejie Zhang. Refining word embeddings using intensity scores for sentiment analysis, 2018.

[6] Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[7] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[8] Alex Sherstinsky. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *CoRR*, abs/1808.03314, 2018.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[10] Fei Long, Kai Zhou, and Weihua Ou. Sentiment analysis of text based on bidirectional lstm with multi-head attention. *IEEE Access*, PP:1–1, 09 2019.

[11] Lutfiye Seda Mut Altin, Àlex Bravo Serrano, and Horacio Saggion. LaSTUS/TALN at SemEval-2019 task 6: Identification and categorization of offensive languages on social media using attention-based Bi-LSTM models. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 672–677, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics.

[12] Tharindu Ranasinghe, Marcos Zampieri, and Hansi Hettiarachchi. Brums at hasoc 2019: Deep learning models for multilingual hate speech and offensive language identification. 12 2019.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[14] P. Nakov S. Rosenthal N. Farra M. Zampieri, S. Malmasi and R. Kumar. Predicting the type and target of offensive posts in social media. *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1:1415–1420, 2019.

[15] Esperanza García-Gonzalo, Zulima Fernández-Muñiz, Paulino José García Nieto, Antonio Bernardo Sánchez, and Marta Menéndez Fernández. Hard-rock stability analysis for span design in entry-type excavations with learning classifiers. *Materials*, 9(7), 2016.

[16] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6:107–116, 04 1998.

[17] Xiaofeng Yuan, Lin Li, and Yalin Wang. Nonlinear dynamic soft sensor modeling with supervised long short-term memory network. *IEEE Transactions on Industrial Informatics*, PP:1–1, 02 2019.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[19] Reza Sohrabi,from Multithreaded. Give me jeans not shoes: How bert helps us deliver what clients want. Available at:

https://multithreaded.stitchfix.com/assets/posts/2019-07-07-give-me-jeans/bert.png, 2019.

[20] Ketan Doshi. Transformers explained visually (part 1): Overview of functionality. Availble at: https://miro.medium.com/max/306/1*THykpgtL058A9EpkstnUJQ.png. 2020.

# Acknowledgement

Its our privilege and honor to express our gratitude to all the respected personalities who have guided us, inspired us and supported us in the smooth and successful completion of our project. We thank our beloved faculty supervisor Dr.Balaprakasa Rao Killi, Assistant Professor, Department of Computer Science and Engineering(CSE), National Institute of Technology, Warangal, for his persistent supervision, guidance, suggestions and encouragement during this project. He has motivated us during the low times and gave us courage to move ahead positively.

We are grateful to Prof.P.Radha Krishna, Head of the Department, Computer Science and Engineering, National Institute of Technology, Warangal for his moral support to carry out this project. We are also thankful to the Project Evaluation Committee, for their strenuous efforts to evaluate our projects. We wish to thank all the staff members in the department for their kind cooperation and support given throughout our project work. We are thankful to all of our colleague friends who have given valuable suggestions and help in all stages of the development of the project.

Ranjeet Kumar (177250)
Avadhoot Hede(177209)
Deepak Maurya(177215)