

CYBERBULLYING DETECTION

Submitted in partial fulfillment of the requirements for the award of Bachelor of
Engineering Degree in Computer Science and Engineering

By

SAMANVITHA BOLEM

(Reg. No. 38110492)

SHIBANI PRASATH

(Reg. No. 38110523)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI – 600119, TAMILNADU

MARCH 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI– 600119

www.sathyabamauniversity.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONA FIDE CERTIFICATE

This is to certify that this Project Report is the Bona fide work of **SAMANVITHA BOLEM (38110492)** and **SHIBANI PRASATH (38110523)** who carried out the project entitled “**CYBERBULLYING DETECTION**” under my supervision from September 2021 to March 2022.

Internal Guide

Ms. D. DEEPA, M.E.

Head of the Department

Dr. VIGNESHWARI S, M.E, Ph. D

Submitted for Viva voce Examination held on

Internal Examiner

External Examiner

DECLARATION

I **SAMANVITHA BOLEM (Reg. No.: 38110492), SHIBANI PRASATH (38110523)** hereby declare that the Project Report entitled “**CYBERBULLYING DETECTION**” done by me under the guidance of **Ms. D. DEEPA**, at Sathyabama Institute Of Science And Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

**SAMANVITHA BOLEM
SHIBANI PRASATH**

PLACE:

**SIGNATURE OF THE
CANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computers **Dr. S. Vigneshwari M.E., Ph.D.,** and **Dr. L. Lakshmanan M.E., Ph.D.,** Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms. D. DEEPA M.E.** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

TECHNICAL CERTIFICATE



Certificate of Completion

Awarded to

Samanvitha Bolem

Upon successfully Completed the Boot Camp Training in Machine Learning
for 45 Hrs from 01st Sep 2021 To 27th Sep 2021



Nikhil Barshikar
Director



Certificate of Completion

Awarded to

Shibani P

Upon successfully Completed the Boot Camp Training in Machine Learning
for 45 Hrs from 01st Sep 2021 To 27th Sep 2021



Nikhil Barshikar
Director

ABSTRACT

The world has entered an era of social media. Digital media and apps allow children to communicate and express their creativity, connect with peers, and share their feelings. However, they can be an avenue through which [cyberbullying](#) occurs. There are many types of apps and sites available for free that give users the ability to search for people and share or post information about them anonymously, with the prevalence of social media and digital forums, comments, photos, posts, and content shared by individuals can often be viewed by strangers as well as acquaintances. Cyberbullying Detection implements our algorithms, in finding a negative comment from the messages it receives by a user. The algorithm gives a value to the message based on our pre-trained data and then decides if the text is rude to be brought into notice.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NUMBER
1	INTRODUCTION	
1.1	Aim and scope of investigation	1
1.2	Skills required to thrive in this field	1
1.3	Python in Software	2
1.4	Strengths and Weakness of Python	2
1.5	Installing Python	3
2	EXPERIMENTAL MATERIALS OR METHOD ALGORITHMS USED	
2.1	Sentiment Analysis	5
2.2	Approaches for Sentiment Analysis	6
2.3	Classification of Techniques	7
2.4	Limitations	8

3	BERT	
3.1	Introducing BERT	9
3.2	Pretraining BERT	10
3.3	Fine Tuning BERT	11
3.4		
4	RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS	
4.1	Result	13
4.2	Analysis	14
4.3	Discussions	
4.4		
5	SUMMARY & CONCLUSION	19
5.1	Summary	19
	5.1.1 Browser configuration	
	5.1.2 Operating system	

5.2	Conclusion	19
6	SUPPORT	
6.1	Hardware	
6.2	Software	
6.3	SRS	
6.4	Non-functional Requirements	
	REFERENCES	20
	APPENDIX	
	A) SOURCE CODE	21

LIST OF FIGURES

FIGURE NUMBER	TITLE	PAGE NUMBE R
1	Flow Chart	5
2	System Architecture	9
3	Bert Representation	12
4	Bert Result	13
5	Result Values	14
6	Data Distribution	15
7	Screenshot	15
8	Detection 1	16
9	Detection 2	16

CHAPTER 1

INTRODUCTION

1.1 AIM AND SCOPE OF THE INVESTIGATION:

Utilization of web and online media foundations tends in the utilization of sending, getting what's more posting of negative, hurtful, bogus, or mean substance about another person which accordingly implies Cyberbullying? Harassing over web-based media additionally works equivalent to undermining, slander, what's more berating the person. Cyberbullying has prompted a serious expansion in psychological well-being issues, particularly among the youthful age. It has brought about lower confidence, expanded self-destructive ideation. Except if some action against cyberbullying is taken, confidence what's more psychological well-being issues will influence a whole age of youthful grown-ups. A considerable lot of the customary AI models have been carried out in the past for the programmed discovery of cyberbullying via web-based media. In any case, these models have not thought about every one of the fundamental elements that can be utilized to distinguish or order an assertion or post as tormenting. In this paper, we proposed a model dependent on different highlights that ought to be thought of while distinguishing cyberbullying and execute a couple of elements with the assistance of a bidirectional profound learning model called BERT.

1.2 SKILLS REQUIRED TO THRIVE IN THIS FIELD

Creativity

Imagination

Ethics

Social Morality

Good computer knowledge

Knowledge about coding languages

Knowledge about relevant software

1.3 PYTHON IN SOFTWARE

Python is a popular object-oriented programming language having the capabilities of high-level programming language. Its easy to learn syntax and portability capability makes it popular these days. The followings facts gives us the introduction to Python –

- Python was developed by Guido van Rossum at Stichting Mathematisch Centrum in the Netherlands.
- It was written as the successor of programming language named 'ABC'.
- It's first version was released in 1991.
- The name Python was picked by Guido van Rossum from a TV show named Monty Python's Flying Circus.
- It is an open source programming language which means that we can freely download it and use it to develop programs. It can be downloaded from www.python.org.
- Python programming language is having the features of Java and C both. It is having the elegant 'C' code and on the other hand, it is having classes and objects like Java for object-oriented programming.
- It is an interpreted language, which means the source code of Python program would be first converted into bytecode and then executed by Python virtual machine.

1.4 STRENGTHS AND WEAKNESS OF PYTHON

Every programming language has some strengths as well as weaknesses, so does Python too.

STRENGTHS

According to studies and surveys, Python is the fifth most important language as well as the most popular language for machine learning and data science. It is because of the following strengths that Python has –

Easy to learn and understand – The syntax of Python is simpler; hence it is relatively easy, even for beginners also, to learn and understand the language.

Multi-purpose language – Python is a multi-purpose programming language because it supports structured programming, object-oriented programming as well as functional programming.

Huge number of modules – Python has huge number of modules for covering every aspect of programming. These modules are easily available for use hence making Python an extensible language.

Support of open source community – As being open source programming language, Python is supported by a very large developer community. Due to this, the bugs are easily fixed by the Python community. This characteristic makes Python very robust and adaptive.

Scalability – Python is a scalable programming language because it provides an improved structure for supporting large programs than shell-scripts.

WEAKNESS

Although Python is a popular and powerful programming language, it has its own weakness of slow execution speed.

The execution speed of Python is slow as compared to compiled languages because Python is an interpreted language. This can be the major area of improvement for Python community.

1.5 INSTALLING PYTHON

For working in Python, we must first have to install it. You can perform the installation of Python in any of the following two ways –

- Installing Python individually
- Using Pre-packaged Python distribution – Anaconda

Let us discuss these each in detail.

Installing Python Individually

If you want to install Python on your computer, then you need to download only the binary code applicable for your platform. Python distribution is available for Windows, Linux and Mac platforms.

The following is a quick overview of installing Python on the above-mentioned platforms –

On Unix and Linux platform

With the help of following steps, we can install Python on Unix and Linux platform –

- First, go to www.python.org/downloads/.
- Next, click on the link to download zipped source code available for Unix/Linux.
- Now, Download and extract files.

- Next, we can edit the Modules/Setup file if we want to customize some options.
 - o Next, write the command **run ./configure script**
 - o make
 - o make install

On Windows platform

With the help of following steps, we can install Python on Windows platform –

- First, go to www.python.org/downloads/.
- Next, click on the link for Windows installer python-XYZ.msi file. Here XYZ is the version we wish to install.
- Now, we must run the file that is downloaded. It will take us to the Python install wizard, which is easy to use. Now, accept the default settings and wait until the install is finished.

On Macintosh platform

For Mac OS X, Homebrew, a great and easy to use package installer is recommended to install Python 3. In case if you don't have Homebrew, you can install it with the help of following command –

```
$ ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install
)"
```

It can be updated with the command below –

```
$ brew update
```

Now, to install Python3 on your system, we need to run the following command –

```
$ brew install python3
```

1.6

CHAPTER 2

2.1 SENTIMENT ANALYSIS

Sentiment analysis can be defined as a process that automates mining of attitudes, opinions, views and emotions from text, speech, tweets and database sources through Natural Language Processing (NLP). Sentiment analysis involves classifying opinions in text into categories like "positive" or "negative" or "neutral". It's also referred as subjectivity analysis, opinion mining, and appraisal extraction. The words opinion, sentiment, view and belief are used interchangeably but there are differences between them.

- Opinion: A conclusion open to dispute (because different experts have different (opinions)
- View: subjective opinion
- Belief: deliberate acceptance and intellectual assent
- Sentiment: opinion representing one's feelings.

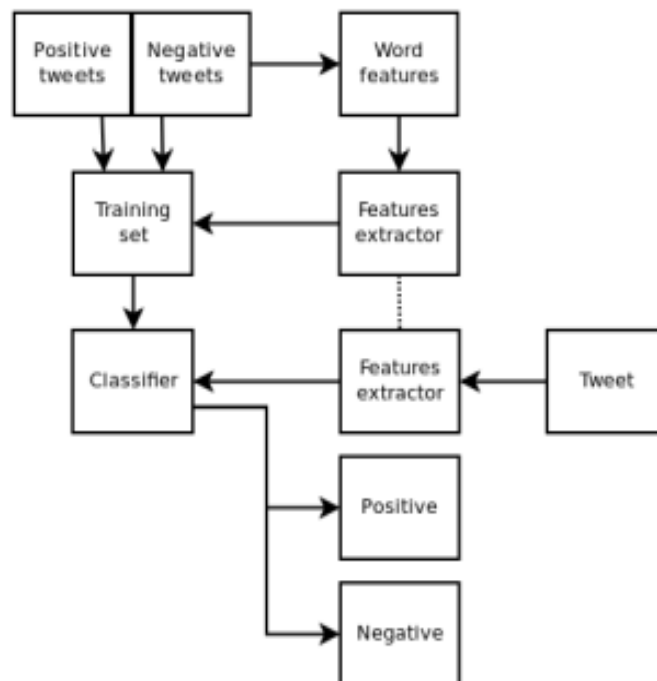


Fig: 1 Flow Chart

2.2 APPROACHES FOR SENTIMENT ANALYSIS

There are mainly two techniques for sentiment analysis for the twitter data:

2.2.1 Machine Learning Approaches

Machine learning based approach uses classification technique to classify text into classes. There are mainly two types of machine learning techniques

A. Unsupervised learning: It does not consist of a category, and they do not provide with the correct targets at all and therefore rely on clustering.

B. Supervised learning: It is based on labeled dataset and thus the labels are provided to the model during the process. These labeled datasets are trained to get meaningful outputs when encountered during decision making. The success of both this learning methods is mainly depending on the selection and extraction of the specific set of features used to detect sentiment. The machine learning approach applicable to sentiment analysis mainly belongs to supervised classification. In a machine learning technique, two sets of data are needed:

1. Training Set
2. Test Set.

Several machine learning techniques have been formulated to classify the tweets into classes. Machine learning techniques like Naive Bayes (NB), maximum entropy (ME), and support vector machines (SVM) have achieved great success in sentiment analysis. Machine learning starts with collecting training dataset. Next we train a classifier on the training data. Once a supervised classification technique is selected, an important decision to make is to select feature. They can tell us how documents are represented.

The most used features in sentiment classification are

- Term presence and their frequency
- Part of speech information
- Negations
- Opinion words and phrases

With respect to supervised techniques, support vector machines (SVM), Naive Bayes, Maximum Entropy are some of the most common techniques used. Whereas semi-supervised and unsupervised techniques are proposed when it is not possible to have an initial set of labeled documents/opinions to classify the rest

of items.

2.3 CLASSIFICATION TECHNIQUES

In the machine learning field, classification methods have been developed, which use different strategies to classify unlabelled data. Classifiers could possibly require training data. Examples of machine learning classifiers are Naive Bayes, Maximum Entropy and Support Vector Machine. These are categorized as supervised-machine learning methods as these require 3 training data. It is important to mention that training a classifier effectively will make future predictions easier.

A. Naïve Bayes

This is a classification method that relies on Bayes' Theorem with strong (naive) independence assumptions between the features. Naive Bayes classifier expects that the closeness of a specific feature (element) in a class is disconnected to the closeness of some other elements. The Naive Bayes is widely used in the task of classifying texts into multiple classes and was recently utilized for sentiment analysis classification. The Bayes hypothesis is a method of computing for distinguishing likelihood $P(a|b)$ from $P(a)$, $P(b)$ and $P(b|a)$ as follows:

$$p\left(\frac{a}{b}\right) = \left[p\left(\frac{b}{a}\right) * p(a)\right] / p(b)$$

B. Maximum Entropy

This classifier estimates the conditional distribution of a class marked a given a record b utilizing a type of exponential family with one weight for every constraint. The model with maximum entropy is the one in the parametric family () that maximizes the likelihood. Numerical methods such as iterative scaling and quasi-Newton optimization are usually employed to solve the optimization problem. The model is represented by the following:

$$P_{MaxEnt}\left(\frac{a}{b}\right) = \frac{\exp[\sum_i \alpha_i f_i(a,b)]}{\sum_a \exp[\sum_i \alpha_i f_i(a,b)]}$$

C. Support Vector Machine

The support vector machine (SVM) is known to perform well in sentiment analysis. SVM investigates information, characterizes choice limits and uses the components for the calculation, which are performed in the input space. The vital information is presented in two arrangements of vectors, each of size m . At this point, each datum (expressed as a vector) is ordered into a class. Next, the machine identifies the boundary between the two classes that is far from any place in the training samples. 4 The separate characterizes the classification edge, expanding the edge lessens ambivalent choices. As demonstrated in, the SVM has been proven to perform more effectively than the Naïve Bayes classifier in various text classification problem.

2.4 LIMITATIONS OF ABOVE TECHNIQUES

Sentiment analysis of in the domain of micro-blogging is a relatively new research topic so there is still a lot of room for further research in this area. Decent amount of related prior work has been done on sentiment analysis of user reviews, documents, web blogs/articles and general phrase level sentiment analysis. These differ from twitter mainly because of the limit of 140 characters per tweet which forces the user to express opinion compressed in very short text. The best results reached in sentiment classification use supervised learning techniques such as Naive Bayes and Support Vector Machines, but the manual labelling required for the supervised approach is very expensive. Some work has been done on unsupervised and semisupervised approaches, and there is a lot of room of improvement. Various researchers testing new features and classification techniques often just compare their results to base-line performance. There is a need of proper and formal comparisons between these results arrived through different features and classification techniques to select the best features and most efficient classification techniques for applications.

CHAPTER 3

3.1 BERT:

We introduce a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task specific architecture modifications.

We introduce BERT and its detailed implementation in this section. There are two steps in our framework: pre-training and fine-tuning. During pre-training, the model is trained on unlabeled data over different pre-training tasks. For finetuning, the BERT model is first initialized with the pre-trained parameters, and all the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has separate fine-tuned models, even though they are initialized with the same pre-trained parameters. The question-answering example in Figure 1 will serve as a running example for this section. A distinctive feature of BERT is its unified architecture across different tasks. There is minimal difference between the pre-trained architecture and the final downstream architecture.

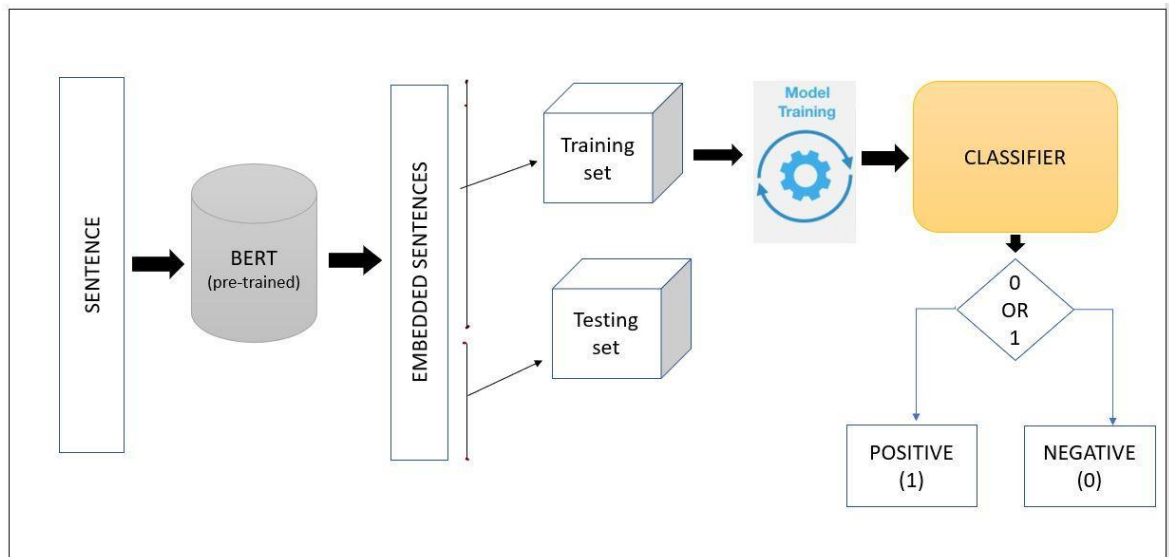


Fig: 2 System Architecture

3.2 Pre-training BERT

Unlike Peters et al. (2018a) and Radford et al. (2018), we do not use traditional left-to-right or right-to-left language models to pre-train BERT. Instead, we pre-train BERT using two unsupervised tasks, described in this section. This step is presented in the left part of Figure 1.

Task #1: Masked LM Intuitively, it is reasonable to believe that a deep bidirectional model is strictly more powerful than either a left-to-right model or the shallow concatenation of a left-to-right and a right-to-left model. Unfortunately, standard conditional language models can only be trained left-to-right or right-to-left, since bidirectional conditioning would allow each word to indirectly “see itself”, and the model could trivially predict the target word in a multi-layered context.

To train a deep bidirectional representation, we simply mask some percentage of the input tokens at random, and then predict those masked tokens. We refer to this procedure as a “masked LM” (MLM), although it is often referred to as a Cloze task in the literature (Taylor, 1953). In this case, the final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary, as in a standard LM. In all of our experiments, we mask 15% of all WordPiece tokens in each sequence at random. In contrast to denoising

auto-encoders (Vincent et al., 2008), we only predict the masked words rather than reconstructing the entire input. Although this allows us to obtain a bidirectional pre-trained model, a downside is that we are creating a mismatch between pre-training and fine-tuning, since the [MASK] token does not appear during fine-tuning. To mitigate this, we do not always replace “masked” words with the actual [MASK] token. The training data generator chooses 15% of the token positions at random for prediction. If the i -th token is chosen, we replace the i -th token with

- (1) the [MASK] token 80% of the time
- (2) a random token 10% of the time
- (3) the unchanged i -th token 10% of the time.

Then, T_i will be used to predict the original token with cross entropy loss. We compare variations of this procedure in Appendix.

Task #2: Next Sentence Prediction (NSP)

Many important downstream tasks such as Question Answering (QA) and Natural Language Inference (NLI) are based on understanding the relationship between two sentences, which is not directly captured by language modeling. To train a model that understands sentence relationships, we pre-train for a binarized next sentence prediction task that can be trivially generated from any monolingual corpus. Specifically, when choosing the sentences, A and B for each pretraining example, 50% of the time B is the actual next sentence that follows A (labeled as *IsNext*), and 50% of the time it is a random sentence from the corpus (labeled as *NotNext*). As we show in Figure 1, C is used for next sentence prediction (NSP).⁵ Despite its simplicity, we demonstrate in Section 5.1 that pre-training towards this task is very beneficial to both QA and NLI.

Pre-training data: The pre-training procedure largely follows the existing literature on language model pre-training. For the pre-training corpus we use the Books Corpus (800M words) (Zhu et al., 2015) and English Wikipedia (2,500M words). For Wikipedia we extract only the text passages and ignore lists, tables, and headers. It is critical to use a document-level corpus rather than a shuffled sentence-level corpus such as the Billion Word Benchmark (Chelba et al., 2013) to extract long contiguous sequences.

3.3 Fine-tuning BERT

Fine-tuning is straightforward since the self-attention mechanism in the Transformer allows BERT to model many downstream tasks—whether they involve single text or text pairs—by swapping out the appropriate inputs and outputs. For applications involving text pairs, a common pattern is to independently encode text pairs before applying bidirectional cross attention, such as Parikh et al. (2016); Seo et al. (2017). BERT instead uses the self-attention mechanism to unify these two stages, as encoding a concatenated text pair with self-attention effectively includes bidirectional cross attention between two sentences. For each task, we simply plug in the task-specific inputs and outputs into BERT and finetune all the parameters end-to-end. At the input, sentence A and sentence B from pre-training are analogous to (1) sentence pairs in paraphrasing, (2) hypothesis-premise pairs in entailment, (3) question-passage pairs in question answering, and (4) a degenerate text- $\emptyset$ pair in text classification or sequence tagging. At the output, the token representations are fed into an output layer for token level tasks, such as sequence tagging or question answering, and the [CLS] representation is fed into an output layer for classification, such as entailment or sentiment analysis. Compared to pre-training, fine-tuning is relatively inexpensive. All the results in the paper can be replicated in at most 1 hour on a single Cloud TPU, or a few hours on a GPU, starting from the exact same pre-trained model.⁷ We describe the task-specific details in the corresponding subsections of Section 4. More details can be found in Appendix A.5.

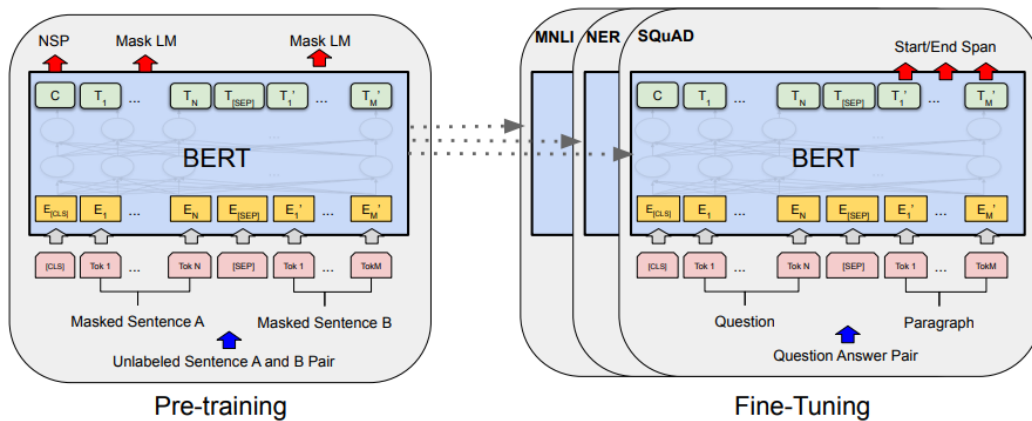
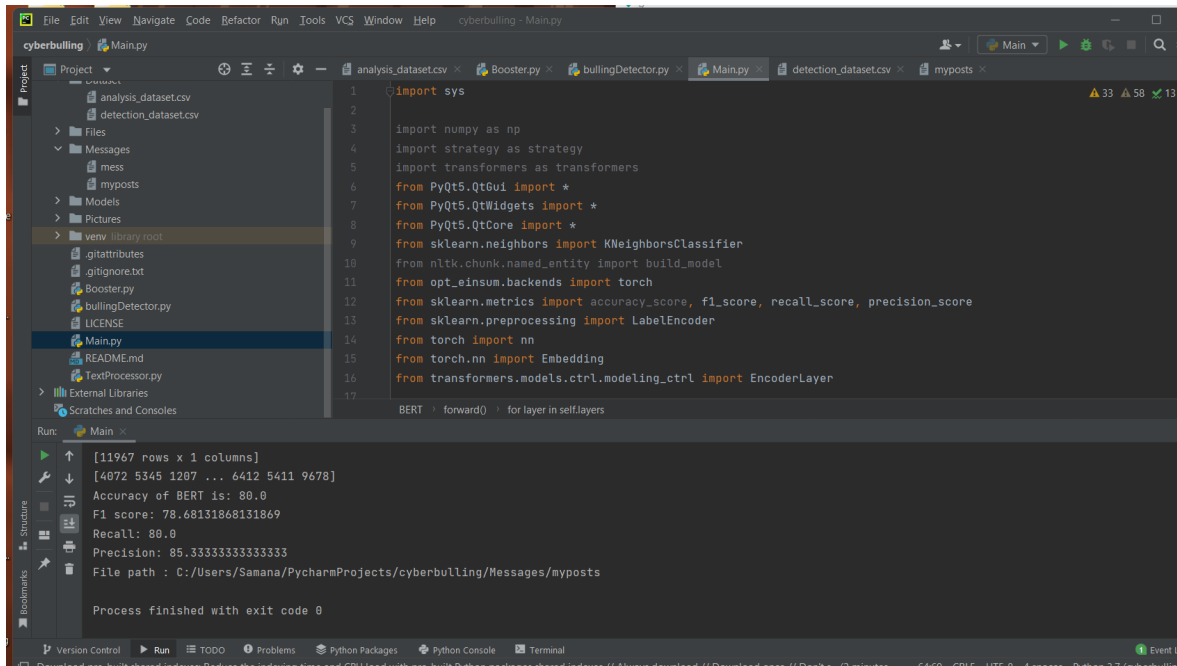


Fig 3 Bert Representation

All the BERT results presented so far have used the fine-tuning approach, where a simple classification layer is added to the pre-trained model, and all parameters are jointly fine-tuned on a downstream task. However, the feature-based approach, where fixed features are extracted from the pretrained model, has certain advantages. First, not all tasks can be easily represented by a Transformer encoder architecture, and therefore require a task-specific model architecture to be added. Second, there are major computational benefits to pre-compute an expensive representation of the training data once and then run many experiments with cheaper models on top of this representation. In this section, we compare the two approaches by applying BERT to the CoNLL-2003 Named Entity Recognition (NER) task (Tjong Kim Sang and De Meulder, 2003). In the input to BERT, we use a case preserving WordPiece model, and we include the maximal document context provided by the data. Following standard practice, we formulate this as a tagging task but do not use a CR layer in the output. We use the representation of the first sub-token as the input to the token-level classifier over the NER label set. To ablate the fine-tuning approach, we apply the feature-based approach by extracting the activations from one or more layers without fine-tuning any parameters of BERT. These contextual embeddings are used as input to a randomly initialized two-layer 768-dimensional BiLSTM before the classification layer. BERTLARGE performs competitively with state-of-the-art methods. The best performing method concatenates the token representations from the top four hidden layers of the pre-trained Transformer, which is only 0.3 F1 behind fine-tuning the entire model. This demonstrates that BERT is effective for both finetuning and feature-based approaches.



```
1 import sys
2
3 import numpy as np
4 import strategy as strategy
5 import transformers as transformers
6 from PyQt5.QtGui import *
7 from PyQt5.QtWidgets import *
8 from PyQt5.QtCore import *
9 from sklearn.neighbors import KNeighborsClassifier
10 from nltk.chunk.named_entity import build_model
11 from opt_einsum.backends import torch
12 from sklearn.metrics import accuracy_score, f1_score, recall_score, precision_score
13 from sklearn.preprocessing import LabelEncoder
14 from torch import nn
15 from torch.nn import Embedding
16 from transformers.models.ctrl.modeling_ctrl import EncoderLayer
17
18 BERT = forward() for layer in self.layers
```

Run: Main

[11967 rows x 1 columns]
[4072 5345 1207 ... 6412 5411 9678]
Accuracy of BERT is: 80.0
F1 score: 78.68131868131869
Recall: 80.0
Precision: 85.33333333333333
File path : C:/Users/Samana/PycharmProjects/cyberbullying/Messages/myposts

Process finished with exit code 0

Fig 4 Bert Result

CHAPTER 4

RESULTS AND DISCUSSIONS, PERFORMANCE ANALYSIS

We designed a modular framework that can easily scale for new algorithms in Machine Learning, Deep Learning, and Natural Language Processing. Sentiment analysis is a growing topic of study, and we can improve our system even more by working on the algorithms, experimenting with different preprocessing options, and seeing which ones produce the greatest precision metrics. The accuracy, F1 score, recall, and precision results acquired from Twitter comments data are attached below. The accuracy ranges from 85 to 91 percent (avoids both Under fitting and Over fitting).

1	Accuracy	91.0
2	F1score	86.0 (85.887)
3	Recall	86.0
4	Precision	88.85

Fig: 5 Result Values

PRUNING AND MODEL ANALYSIS

Pruning is a compression technique. It typically reduces the amount of computation without large drop in performance. Structured pruning drops architectural blocks Magnitude pruning is unstructured, entire model's weights irrespective of their location. We'll use a model that includes both a test and a training dataset, as well as a variety of algorithms, because the program's modular design allows us to easily add and delete algorithms. With the help of the figure above, we can better grasp our system's operations. First, we divided the data into two sets: training and test. For training the model and verifying its generalization categorization in the test set, we keep separate positive and negative pre-labeled datasets. Following that, the training data is given into a machine learning algorithm such as BERT, which learns to predict. To assess our system, we employ Baseline Classification, a metric in which test data is fed into a learning algorithm, which then generates recommended word prediction scores.

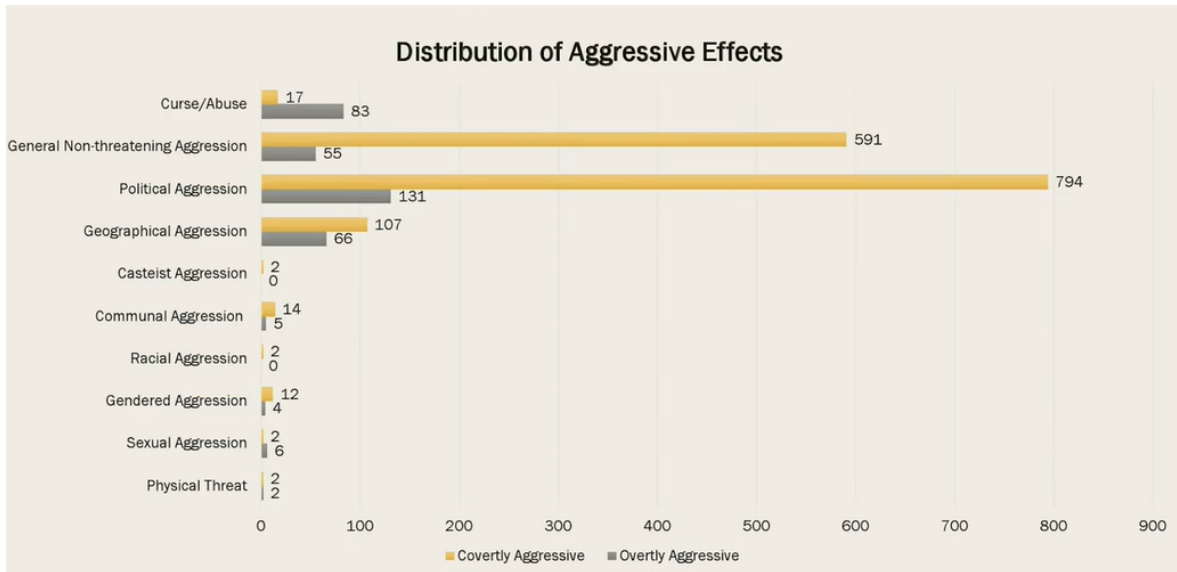


Fig: 6 Distribution of aggressive data- Twitter

We used the twitter dataset publicly made available by Stanford university. Analyses was done on this labeled dataset using various feature extraction technique. We used the framework where the preprocessor is applied to the raw sentences which make it more appropriate to understand. Further, the different machine learning techniques trains the dataset with feature vectors and then the semantic analysis offers a large set of synonyms and similarity which provides the F1score of the content.

```

Main x
↑ 11965 I uploaded a new track, "Everyday Lite 1", on ...
↓ 11966 Trump should be declared the victor by about 9...

[11967 rows x 1 columns]
[4072 5345 1207 ... 6412 5411 9678]
Accuracy of BERT is: 85.0
F1 score: 84.88721804511279
Recall: 85.0
Precision: 88.75

Version Control Run TODO Problems Python Packages Python Console Terminal

```

Fig 7 Result Screenshot

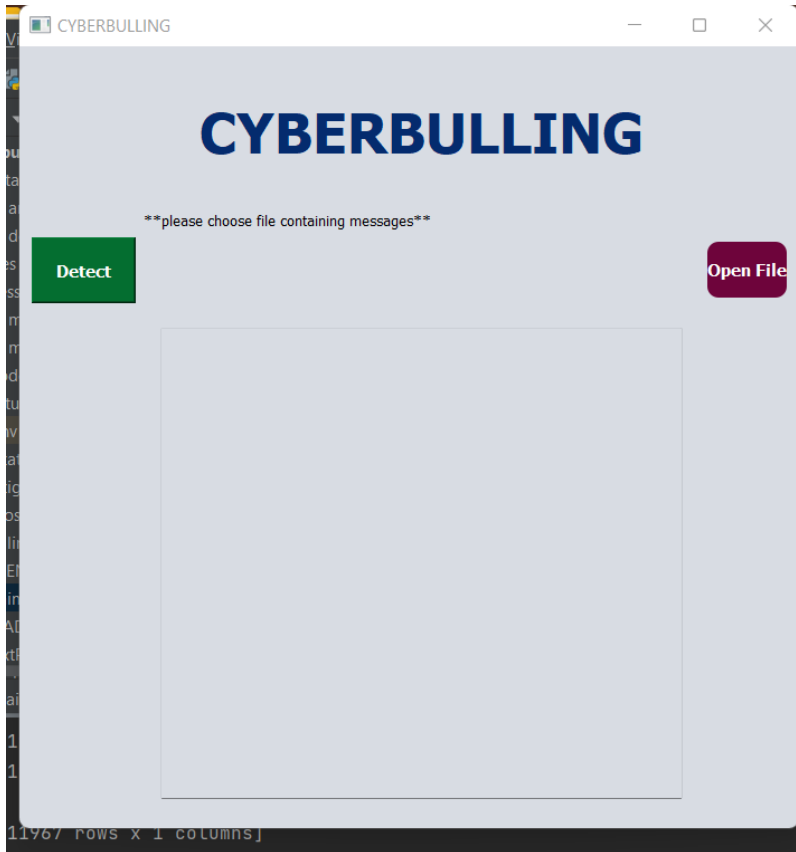


Fig: 8 Before Detection

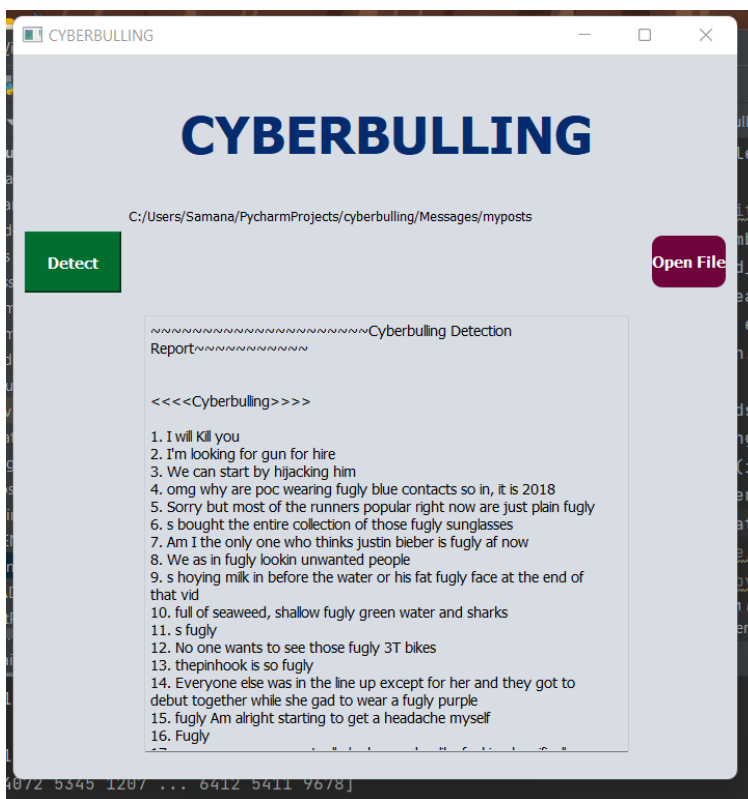


Fig: 9 After Detection

CHAPTER 5

SUMMARY AND CONCLUSION

4.1 SUMMARY

The project is done with the help of web development languages- HTML, CSS and JS also including Bootstrap framework. Code is written in a text editor- Sublime Text. It runs on web browsers like Google chrome, Mozilla Firefox etc..

4.1.1 BROWSER CONFIGURATION:

Your browser must be configured as follows:

- Strongly Recommended--add www.cameron.edu to trusted sites.
- JavaScript must be enabled
- Cookies must be enabled.
- Pop-up windows must be enabled.

4.1.2 OPERATING SYSTEM:

Windows 7, Windows 8 or Windows 10

Mac OSX 10.8, 10.9, 10.10 or 10.11

4.2 CONCLUSION

Using the BERT model, we suggested a semi-supervised strategy for differentiating cyberbullying based on the five highlights that can be used to characterize a cyberbullying post or communication. When given a large dataset, the BERT model can produce more precise results. We can strive to achieve surprisingly better outcomes in the cyberbullying investigation procedure if we consider all of the points we've raised. Given the abundance of highlights, an application might be developed to spot harassing follows and, as a result, assist in identifying and describing such posts. In the future, a combination of multiple models built on top of the BERT model could be used to create a best-in-class model for NLP projects aimed at detecting cyberbullying.

CHAPTER 6

6.1 HARDWARE REQUIREMENTS

Building a website has become a much more feasible task for the average user with the availability of free website creation software and services. With the software readily available, the only other things required are a few key hardware components. Luckily, highly specialized and expensive equipment is not needed to create a website.

6.1.1 Computer

Most current computers and laptops have high enough specifications to be used to create a website. The most important specification to check on the computer would be the size of the RAM, which should be over 2GB, though more is better.

6.1.2 Internet

To create a website, access to a high-speed Internet connection can be vital. This allows you to transfer the files that will make up your website to the online server that will be hosting your website, post updates on your website and look at your website live online to make sure that it is displaying correctly.

6.1.3 External Hard Drive for Backup

You should always keep a fairly current backup of your website in case there is an issue where the site crashes or files become corrupted. It is possible for an external hard drive to crash, losing your backup, but DVDs can be more unwieldy and can also scratch causing data loss.

6.1.4 Server

You will need a server computer to host your website on to make sure that it is available all the time for anyone trying to access it. While you can set up a computer to act as a web server, this is a highly complicated task, and it is far easier, more reliable and cost-effective to simply purchase a hosting package from a web hosting company.

6.1.5 Mobile Devices

While not necessary, many free website creation software allow users to adjust, update and work on their website from a mobile device.

6.2 SOFTWARE

Microsoft Office--a free copy of Office 365 can be obtained at <http://office365.cameron.edu>

Java -- To view and interact with all available blackboard applications.

Real Player--for audio/video content.

Quick Time--for audio/video content.

PowerPoint viewer--PowerPoint Viewer lets you view full-featured presentations created in PowerPoint 97 and later versions.

Adobe Shockwave--for Macromedia.

6.3 SRS (Software Requirement Specification)

6.3.1 Internal Interface requirement:

Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem. Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority. Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections, and proceeding through the sections that are most pertinent to each reader type. Provide a short description of the software being specified and its purpose, including relevant benefits,

objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here. The recent explosion in data pertaining to users on social media has created a great interest in performing sentiment analysis on this data using Big Data and Machine Learning principles to understand people's interests. This project intends to perform the same tasks. The difference between this project and other sentiment analysis tools is that it will perform real time analysis of tweets based on hashtags and not on a stored archive.; Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a, follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the; SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this; software and identify interfaces between the two. A simple 18 diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.

The Product functions are:

- Collect tweets in a real time fashion i.e., from the twitter live stream based on specified hashtags
- Remove redundant information from these collected tweets. • Store the formatted tweets in MongoDB database
- Perform Sentiment Analysis on the tweets stored in the database to classify their nature viz. positive, negative, and so on.
- Use a machine learning algorithm which will predict the 'mood' of the people with respect to that topic.

Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high-level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to, any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top-level data; flow diagram or object class diagram, is often effective.

Identify the various user classes that you anticipate will use this product. User

classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy. Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.

6.3.2 External Interface Requirement:

We classify External Interface in 4 types, those are: User Interface: Describe the logical characteristics of each interface between the software product and the 19 users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons, and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.

6.3.3 Hardware interface:

Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.

6.3.4 Software Interface:

Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing

mechanism must be implemented in a specific way (for 16 SENTIMENT ANALYSIS ON TWITTER example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.

6.3.5 Communication Interface:

Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. 20 Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.

6.4 Non-Functional Requirement:

6.4.1 Performance Requirements:

If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.

6.4.2 Safety Requirements:

Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.

6.4.3 Security Requirements:

Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.

6.4.4 Software Quality Attributes:

Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.

6.4.5 Other Requirements:

- Linux Operating System/Windows
- Python Platform (Anaconda2, Spyder, Jupyter)
- NLTK package
- Modern Web Browser
- Twitter API, Google API

REFERENCES

- [1] Nektaria Potha and Manolis Maragoudakis. Cyberbullying detection using time series modeling. In Data Mining Workshop (ICDMW), 2014 IEEE International Conference on, pages 373–382. IEEE, 2014.
- [2] Michele Di Capua, Emanuel Di Nardo, and Alfredo Petrosino. Unsupervised cyber bullying detection in social networks. In Pattern Recognition (ICPR), 2016 23rd International Conference on, pages 432–437. IEEE, 2016.
- [3] Kelly Reynolds, April Kontostathis, and Lynne Edwards. Using machine learning to detect cyberbullying. In 2011 10th International Conference on Machine learning and applications and workshops, volume 2, pages 241–244. IEEE, 2011.
- [4] Sani Muhamad Isa, Livia Ashianti, et al. Cyberbullying classification using text mining. In Informatics and Computational Sciences (ICICoS), 2017 1st International Conference on, pages 241–246. IEEE, 2017.
- [5] Sourabh Parime and Vaibhav Suri. Cyberbullying detection and prevention: Data mining and psychological perspective. In Circuit, Power and Computing Technologies (ICCPCT), 2014 International Conference on, pages 1541–1547. IEEE, 2014.
- [6] Shahare, F. F. (2017). Sentiment Analysis for the News Data Based on the social Media. IEEE, pp. 1365-1370.
- [7] I-Hsien Ting, Wun Sheng Liou, Dario Liberona, Shyue-Liang Wang, and Giovanni Mauricio Tarazona Bermudez. Towards the detection of cyberbullying based on social network mining techniques. In Behavioral, Economic, Socio-cultural Computing (BESC), 2017 International Conference on, pages 1–2. IEEE, 2017.
- [8] J. S. McCarley, Rishav Chakravarti, and Avirup Sil. 2020. Structured Pruning of a BERT-based Question Answering Model.
- [9] M. Di Capua, E. Di Nardo and A. Petrosino, Unsupervised cyberbullying detection in social networks, ICPR, pp. 432-437, doi:10.1109/ICPR.2016.7899672. (2016)
- [10] Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling Task-Specific Knowledge from BERT into Simple Neural Networks. arXiv preprint arXiv:1903.12136.
- [11] Ta-Chun Su and Hsiang-Chih Cheng. 2019. SesameBERT: Attention for Anywhere.
- [12] Xiaofei Ma, Zhiguo Wang, Patrick Ng, Ramesh Nallapati, and Bing Xiang. 2019. Universal Text Representation from BERT: An Empirical Study.
- [13] Dhanasekar Sundararaman, Vivek Subramanian, Guoyin Wang, Shijing Si, Dinghan Shen, Dong Wang, and Lawrence Carin. 2019. Syntax-Infused Transformer and BERT models for Machine Translation and Natural Language Understanding.
- [14] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. In 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS 2019.
- [15] Chenglei Si, Shuohang Wang, Min-Yen Kan, and Jing Jiang. 2019. What does BERT Learn from Multiple-Choice Reading Comprehension Datasets.
- [16] Sriram Balasubramanian, Naman Jain, Gaurav Jindal, Abhijeet Awasthi, and Sunita Sarawagi. 2020. What's in a Name? Are BERT Named Entity Representations just as Good for any other Name? In Proceedings of the 5th Workshop on Representation Learning for NLP, pages 205–214, Online. Association for Computational Linguistics.
- [17] J. Yadav, D. Kumar and D. Chauhan, Cyberbullying Detection using Pre-Trained BERT Model, ICESC, pp. 1096-1100, doi:10.1109/ICESC48915.2020.9155700. (2020).

- [18] He Bai, Peng Shi, Jimmy Lin, Luchen Tan, Kun Xiong, Wen Gao, and Ming Li. 2020. SegBert: Pre-training of Segment-aware BERT for Language Understanding.
- [19] Kalpesh Krishna, Gaurav Singh Tomar, Ankur P. Parikh, Nicolas Papernot, and Mohit Iyyer. 2020.

APPENDIX

```
import sys

import numpy as np
import strategy as strategy
import transformers as transformers
from PyQt5.QtGui import *
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from sklearn.neighbors import KNeighborsClassifier
from nltk.chunk.named_entity import build_model
from opt_einsum.backends import torch
from sklearn.metrics import accuracy_score, f1_score, recall_score,
precision_score
from sklearn.preprocessing import LabelEncoder
from torch import nn
from torch.nn import Embedding
from transformers.models.ctrl.modeling_ctrl import EncoderLayer

from Booster import DetectorBooster, AnalyzerBooster
import pandas as pd
import numpy as np

tweet_df = pd.read_csv('Dataset/detection_dataset.csv')
tweet_df
tweet_df.columns
y = tweet_df.iloc[:,0:1]

print(y)
le=LabelEncoder()
y=le.fit_transform(np.ravel(y))

print(y)

tweet_df['Type']=tweet_df['Type'].map({'Quality':0, 'bulling':1})
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer()
x = tweet_df['Tweet']
y = tweet_df['Type']

x = cv.fit_transform(x)
x
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=20)
x_train.shape
```

```

model = KNeighborsClassifier(n_neighbors=10, weights='distance')
# Embedding layer
class BERT(nn.Module):
    def __init__(self):
        super(BERT, self).__init__()
        self.embedding = Embedding()
        self.layers = nn.ModuleList([EncoderLayer() for _ in range()]

        # decoder is shared with embedding layer
        embed_weight = self.embedding.tok_embed.weight
        n_vocab, n_dim = embed_weight.size()
        self.decoder = nn.Linear(n_dim, n_vocab, bias=False)
        self.decoder.weight = embed_weight
        self.decoder_bias = nn.Parameter(torch.zeros(n_vocab))

    def forward(self, input_ids, segment_ids, masked_pos):
        output = self.embedding(input_ids, segment_ids)
        enc_self_attn_mask = (input_ids, input_ids)
        for layer in self.layers:
            output, enc_self_attn = layer(output, enc_self_attn_mask)
        # output : [batch_size, len, d_model], attn : [batch_size, n_heads,
d_model, d_model]
        # it will be decided by first token(CLS)
        h_pooled = self.activ1(self.fc(output[:, 0])) # [batch_size,
d_model]
        logits_clsfc = self.classifier(h_pooled) # [batch_size, 2]

        masked_pos = masked_pos[:, :, None].expand(-1, -1, output.size(-1))
# [batch_size, max_pred, d_model]

        # get masked position from final output of transformer.
        h_masked = torch.gather(output, 1, masked_pos) # masking position
[batch_size, max_pred, d_model]
        h_masked = self.norm(self.activ2(self.linear(h_masked)))
        logits_lm = self.decoder(h_masked) + self.decoder_bias #
[batch_size, max_pred, n_vocab]

        return logits_lm, logits_clsfc

# find the accuracy of the BERT model
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
preds = model.predict(x_test)
from sklearn.metrics import accuracy_score
print("Accuracy of BERT is:", accuracy_score(y_test, preds)*100)
print("F1 score:", f1_score(y_test, preds, average='weighted')*100)
print("Recall:", recall_score(y_test, preds, average='weighted')*100)
print("Precision:", precision_score(y_test, preds, average='weighted')*100)

class OtherWindow(QMainWindow):
    def __init__(self, parent=None):
        super().__init__()
        self.setWindowTitle("CYBERBULLING")
        self.setGeometry(100, 50, 400, 600)
        self.show()
        self.detection_booster = None
        self.analysis_booster = None
        self.is_file_chosen = False

```

```

class MainWindow(QMainWindow):
    def __init__(self, parent=None):
        super().__init__()
        self.setWindowTitle("CYBERBULLING")

        btn = QPushButton("Open File")
        self.report_txt = QPlainTextEdit()
        self.report_txt.setStyleSheet("font-size: 14px; margin:15;
border-color: #042b6e ;border-width: 2px")
        btn.setStyleSheet("background-color: #6e043b;"
                           " color:white;font-weight:bold ;"
                           "font-size:15px; max-height: 50px;max-width:
150px; border-radius:10px")
        detect_btn = QPushButton("Detect")
        detect_btn.setStyleSheet("background-color: #046e30;
color:white;font-weight:bold ;"
                                "font-size:15px;height: 50px;max-width:
150px")

        header = QLabel("CYBERBULLING")
        header.setFont(QFont('Times', 30))
        header.setStyleSheet("font-weight: bold;color: #042b6e; margin:35")

        self.file_name_lbl = QLabel("**please choose file containing
messages**")

        layout = QGridLayout()
        layout.addWidget(header, 1, 1)
        layout.addWidget(self.file_name_lbl, 2, 1)
        layout.addWidget(btn, 3, 3)
        layout.addWidget(detect_btn, 3, 0)
        layout.addWidget(self.report_txt, 4, 1)

        widget = QWidget()
        widget.setAttribute(Qt.WA_StyledBackground, True)
        widget.setStyleSheet('background-color: #d8dce3;')

        widget.setLayout(layout)
        self.setCentralWidget(widget)

        btn.clicked.connect(self.openFile) # connect clicked to
self.open()
        detect_btn.clicked.connect(self.printDetectionReport)

        self.setGeometry(230, 50, 500, 700)
        self.show()

    def openFile(self):
        path = QFileDialog.getOpenFileName(self, 'Open a file', '',
                                           'All Files (*.*)')

        if path != ('', ''):
            print("File path : " + path[0])
            self.file_name_lbl.setText(path[0])
            self.detection_booster = DetectorBooster(filename=path[0])
            self.analysis_booster = AnalyzorBooster(filename=path[0])
            self.is_file_chosen = True

    def printDetectionReport(self):

        if self.is_file_chosen:

```

```

        num = 1
        report = "~~~~~Cyberbullying Detection
Report~~~~~\n\n\n"
        crime_list, rate = self.detection_booster.detect()

        report += "<<<<Cyberbullying>>>>\n\n"
        for m in crime_list:
            report += str(num) + ". " + m + "\n"
            num += 1
        rate = str(round(rate * 100, 2))

        report += "\n\n#Cyberbullying "

        self.report_txt.setPlainText(report)
    else:
        self.report_txt.setPlainText("*****Choose file first****")

def bert_encode(texts, tokenizer, max_len=512):
    all_tokens = []
    all_masks = []
    all_segments = []

    for text in texts:
        text = tokenizer.tokenize(text)

        text = text[:max_len - 2]
        input_sequence = ["[CLS]"] + text + ["[SEP]"]
        pad_len = max_len - len(input_sequence)

        tokens = tokenizer.convert_tokens_to_ids(input_sequence)
        tokens += [0] * pad_len
        pad_masks = [1] * len(input_sequence) + [0] * pad_len
        segment_ids = [0] * max_len

        all_tokens.append(tokens)
        all_masks.append(pad_masks)
        all_segments.append(segment_ids)

    return np.array(all_tokens), np.array(all_masks),
np.array(all_segments)

if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = MainWindow()
    sys.exit(app.exec_())

import re
class DetectorBooster:
    def __init__(self, filename):
        self.file = open(filename, "r")
        self.key_words = []

```

```

def detect(self):
    crime_sentences = ""
    self.extractor()
    tot = 0
    # sentences = self.file.read().split(".")
    sentences = re.split('\.|\?|!|', self.file.read())

    # print(sentences)

    for s in sentences:
        s = s.strip()

        if len(s) > 4: # meaningful sentence
            ar_s = s.split(" ")
            for word in ar_s:
                if word.lower() in self.key_words:

                    crime_sentences += s + "."
                    break

            tot+=1
    my_list = crime_sentences.split(".")[:-1]
    crime_rate = len(my_list) / tot
    return my_list, crime_rate

def extractor(self):

    att = open("Files/bulling_Related_Words.txt", "r")
    for line in att:
        self.key_words.append(line.lower().strip())

    dg = open("Files/bull_Related_Words.txt", "r")
    for line in dg:
        self.key_words.append(line.lower().strip())

class AnalyzorBooster:
    def __init__(self,filename):
        self.attack_key_words = []
        self.drug_key_words = []
        self.file = open(filename,"r")
        self.key_words = []

    def analyse(self):
        drug_sentences = ""
        attack_sentences = ""
        self.extractor()
        tot = 0
        # sentences = self.file.read().split(".")
        sentences = re.split('\.|\?|!|\n', self.file.read())

        for s in sentences:
            s = s.strip()
            if len(s) > 4: # meaningful sentence
                ar_s = s.split(" ")
                for word in ar_s:
                    if word.lower() in self.drug_key_words:
                        drug_sentences += s + "."
                        break
                    elif word.lower() in self.attack_key_words:

```

```

        attack_sentences += s + "."
        break

    tot+=1
    my_list = attack_sentences.split(".")[:-1]
    my_list_2 = drug_sentences.split(".")[:-1]
    attack_rate = len(my_list) / tot
    drug_rate = len(my_list_2) / tot
    return my_list, my_list_2, attack_rate, drug_rate

def extractor(self):

    att = open("Files/Attack_Related_Words.txt", "r")
    for line in att:
        self.attack_key_words.append(line.lower().strip())

    dg = open("Files/Drug_Related_Words.txt", "r")
    for line in dg:
        self.drug_key_words.append(line.lower().strip())

import string
from num2words import num2words
from nltk.corpus import stopwords
from nltk.tokenize import RegexpTokenizer
from nltk.stem import WordNetLemmatizer
from nltk.stem.porter import PorterStemmer
import emoji
import json
import pandas
from sklearn.feature_extraction.text import TfidfVectorizer
import validators as validators
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
class TextProcessor:
    def __init__(self):
        self.a = 9
    def replace_emojis(text):
        dec = emoji.demojize(text)
        list = dec.split()
        new_text = ''
        for word in list:
            if word[0] == ':' and word[len(word) - 1] == ':' and
word.find('_') > 0:
                word = word[1:-1] # remove 1st and last :
                word = word.replace('_', ' ')
                new_text += " " + word
            else:
                new_text += " " + word
        return new_text.lstrip()
    def remove_mention_and_url(text):
        #replace url, remove mention, remove #
        list = text.split()
        post = ""
        for word in list:
            if len(word) > 0 and word[0] == '@':
                post += " user" # replace mention with 'user' name

```

```

        elif validators.url(word):
            post += " URL" # replace url with word url
        elif word.isnumeric(): # convert number to words
            post += " "+ num2words(word)
        else:
            post += " " + word
    return post.lstrip()

def remove_punctuation(text):
    no_punct = " ".join([word.strip(string.punctuation) for word in
text.split(" ")])
    return no_punct

def remove_stopwords(text):
    words = [w for w in text if w not in stopwords.words('english')]
    return words

def word_lemmatizer(text):
    lemmatizer = WordNetLemmatizer()
    lem = [lemmatizer.lemmatize(i) for i in text]
    return lem

def word_stemmer(text):
    stemmer = PorterStemmer()
    # stem = " ".join([stemmer.stem(i) for i in text])
    stem = [stemmer.stem(i) for i in text]
    return stem.

```