Hate Speech Detection From Social Networking Posts using CNN and XGBoost

Ashraf Bin Shahadat
16101199
Md. Mizanur Rahman Rony
16101184
Md. Adnanul Anwar
16101005
Eialid Ahmed Joy
16101182

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
December 2019

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Ashraf Bin Shahadat
16101199

Md. Mizanur Rahman Rony
16101184

Md. Adnanul Anwar
16101005

Eialid Ahmed Joy
16101182

Approval

Of Fall, 2019 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on December 26, 2019.

Examining Committee:

Supervisor:
(Member)

Md. Golam Rabiul Alam, PhD
Associate Professor
Department Of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Mahbubul Alam Majumdar,PhD
Professor and Chairperson
Department of Computer Science and Engineering
Brac University

Abstract

The increasing growth of social networks and microblogging websites have enabled people from different backgrounds and diverse moral codes to communicate with each other quite easily. While social media promotes communication and sharing of information, these are also used to initiate heinous and negative campaigns. Social networks although discourage such act but people often use these social platforms to propagate offensive and hatred towards individuals or specific groups. Therefore,detecting hate speech has become a serious issue that needs considerable attention. The goal of this research is to detect such campaigns of hate. In this paper, two different approaches have been proposed for detecting hate and offensive language on social platforms. The paper proposes Natural language processing with CNN architecture and XGBoost classifier which will be explicitly effective for capturing the context and the semantics of hate speech. The proposed classifiers distinguish hate speech from neutral text and can achieve a higher quality of classification than current state-of-the-art algorithms.Using CNN,the accuracy that has been obtained on detecting if a tweet is offensive or neutral is 89.18% and on another dataset-containing hateful, offensive and neutral comments, the accuracy is 84.74%.The later approach of using XGBoost classifier has achieved an accuracy of 93.10% and 80.51% respectively.In addition,2333 tweets have been collected from twitter and labelled using annotators.On that dataset, using CNN model the accuracy is 76.70% and for XGBoost the accuracy is 78.14%.

Keywords: *Natural Language processing,Hatespeech,Offensive Language,Convolutional Neural Network(CNN),XGBoost.*

Acknowledgement

This is the work of Ashraf Bin Shahadat, Md.Adnanul Anwar, Md.Mizanur Rahman Rony and Eialid Ahmed Joy - students of the CSE department of BRAC University. The document has been prepared as an effort to compile the knowledge obtained by us during these four years of education and produce a final thesis which innovatively addresses one of the very urgent issues that's terrorizing our world at the moment.

All thanks to Almighty, the creator and the owner of this universe, the most merciful, beneficent and the most gracious, who provided us guidance, strength and abilities to complete this research. We are especially thankful to Md. Golam Rabiul Alam, PhD, our thesis supervisor, for his immense help, guidance and support in completion of our project. We are also thankful to the BRAC University Faculty Staffs of the Computer Science and Engineering, who have been a light of guidance for us in the whole study period at BRAC University, particularly in educating and enhancing our knowledge. Finally, we would like to express our sincere gratefulness to our beloved parents, brothers and sisters for their love and care and our friends for constant support and encouragement.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

AdaBoost Adaptive Boosting

$CBOW$ Continuous Bag Of Words

CNN Convolutional Neural Network

DFS Depth First Search

ERM Empirical Risk Minimization

GBM Gradient Boosting Machine

$NLTK$ Natural Language Toolkit

SSE Sum of Squared Errors

SVM Support Vector Machine

$TF - IDF$ term frequency–inverse document frequency

XGBoost Extreme Gradient Boosting

Chapter 1

Introduction

1.1 Background

In this era of technological advancement, social media are vastly spreading among us to communicate with each other. People are using these social networking site because of their availability, flexibility and user friendly features. Now a days we have almost 7.7 billion people all over the world among them almost 3.484 billion people are active in social media [37]. A research shows that, in 2019 the average time for using social media is 2 hour and 23 minutes in a day for every active user [3]. So any contents can spread through social media very quickly as people are very active in these platforms.

Moreover people from different culture, religion, gender, region or mentality can be found in this platform. For the flexibility and 'freedom of speech' people share their views and feelings without thinking about others. Among them some contents may hurt someone's emotion or identity. Sometime these contents are so hateful or offensive that victims take the post as a serious attack on them which often creates chaos, riot, terrorist attacks, minority suppression or suicidal attempts. These kind of incidences are increasing day by day.

In Germany, people who were anti-refugee attacked refugee through their social media activities [24]. In 2015, nine black clergy and worshippers were killed by a person. It happened on Charleston church in United States. It was kind of white supremacy goal set by a group in social media. They led that man to do some violent action [39]. In India, after 2014 when BJP came to the power many rumors occurred which lead the people to many types of communal violence [22]. In 2018, an anti-Semitic man attacked on Jewish people in Pittsburgh synagogue. This incident is known as Pittsburgh synagogue shooting. 11 people were killed by the shooter that day. Before that incident the shooter posted an anti-Semitic hatred status on Gab which is a social networking site claims more 'Freedom of speech' than other sites like Facebook and Twitter [27]. In Myanmar, Rohingya Muslims are the minority. The military and Buddhist leaders started an ethnic cleaning campaign against them. The leaders used social networking sites to accuse and blacken the Rohingya Muslims and suppressed them [14]. In 2019, New Zealand faced an atrocious event that is the Christchurch mosque shootings. The shooter was live streaming during his killing. 49 people were killed during their prayer. Later on it was found that the killer was quite active in social media. He posted his killing weapon's picture and a manifesto named "The Great Replacement" on Twitter before the incident

[35]. We can find thousands of incidents like these where spreading of hatred and offensive contents in social media are the major reason behind destruction. Day by day these violence are increasing.
Moreover hundreds of languages can be found in social media which have different complexity and structure. A simple punctuation or addition of words can change the meaning of any sentence dramatically. Moreover Word or sentence meaning vary from language to language [33]. So anything that is hate or offensive in any language can be considered as neutral or normal on other language. But the main focus of this research is to detect hatred and offensive tweets or social post on English language. For example, in 2018, Sri Lanka blocked all access into the social networking and messaging apps by accusing them to spreading an anti-Muslim riot. It was happened because the status was in Sinhala language which Facebook could not mark as hatred in time. So it spread all over the country and a deadly riot happened [29].

1.2 Research problem

Social media is a place where people of all over the world gather together in order to communicate with each other.As a result, different psychological and diversified people have the ability to communicate with each other quite easily. This often results in collision between people of different mindset. Besides, the freedom of speech gave people the opportunity to express their views without thinking much about ethics. As a result, the spreading of hatred and offensive language is increasing day by day. Although almost every social networking sites are concerned about these facts but due to the complexity of the natural language and the large number of users that are emerging day by day, it is becoming more and more challenging.Controlling and filtering all the contents is always a difficult task.Moreover,people now-a-days are using hatred or abusive words in a tricky way.People are often seen using words like 'nibba' or 'ni*ga' instead of the word 'nigga'.As the words are written in different forms often text classifier fails to recognize these word as hatred or offensive. All these factors increases the complexity of classifying hate speech from social media. Furthermore, there are limited number of publication and research works on hate speech.

1.3 Research objective

The purpose of this research work is to make social media safe and secure from all sorts of hatred and vulgar contents and comments. The main goal of this research is to classify and detect any kind of hate or offensive words or sentences from social media posts which can be harmful for any individuals or groups.This paper proposes different sets of feature extractions including word embedding,word frequency and two different models to classify hate speech.This research proposes word embedding(n-gram) with CNN as word embedding allows words to be represented based on their meanings and semantics which will allow same meaning words to have similar representation. This will help capturing unseen and tricky words. XGBoost is a recent algorithm based on gradient boosted decision trees which is famous for speed and performance.In order to give a different dimension to this research XGBoost has been used.As detecting hate speech is relatively a recent issue,there is not much related work on this topic.Hence, the researcher of this paper prepared a data set from twitter using twitter API and labelled the data with the help of nine annotators.Besides, using CNN and XGBoost for text classification specially, for hate speech detection are the approaches that are relatively new. This research intends to classify hate speech using these two different models and approaches and study their performance. The main objective of this research work are :

- To prepare a data set of hate speech from social networking site.
- To classify hate and offensive tweets/posts through machine learning model.
- To propose two different models,one on deep neural network and other on tree based model.
- To study and compare performance of the proposed models.

1.4 Scope and Limitations

Through this thesis paper we have tried to develop a system which can classify hate speech from any social media posts. The power of social media is so decisive that it can make anything viral in a blink of an eye. One can easily find different types of people in terms of their race, nationality, gender, religion etc. So spreading any hate or offensive content can be possible easily through these platform. These can lead on to big incidents like riots, suicidal attempts even terrorism. But all of these thing can be stopped from the root by classifying and detecting the hatred or offensive text from social media. The number of users in social media are increasing day by day and with the increasing number of users the number of tweets and posts are equally increasing. Controlling and monitoring this huge number of tweets and posts is a huge ask. Beside the complexity of natural language and the new of way of using hatred and offensive word make this type of research more challenging.

1.5 Thesis Report Outline

The rest of the portion of this research paper contains as following. Chapter 2 contains the background study and literature review. It will describe the existing works on this topic and different types of machine learning algorithms.
Chapter 3 is methodology which describes the structure of our total works. It will show the whole process of our proposed model with proper diagrams. This portion will contain the information on our datasets, data preprocessing, and feature extraction and classification model.
In chapter 4, results and analysis of our model will be discussed. It will talk about the results with confusion matrix and classification report. This portion will show the parameters to find out the comparison of our result and also the finding from these results. Chapter 5 will conclude our paper by summarizing our whole work.

Chapter 2

Literature Review and Related Works

Social media have been proved to be a great invention since it was invented. But as the number of users is increasing, it has become very tough to control people to keep a healthy and peaceful environment in the social media. Controlling and detecting hatred and offensive language are probably the hardest thing that the developers are facing in this modern era. There has been very little research in this particular sector. Some of the very important research works are given below. Those works are divided below based on their datasets.

2.1 Hate and Neutral texts

One of the research work is [36]. This paper mainly focuses on Indonesian Language. These authors used Twitter API to collect data from Twitter and and data collection was adjusted with the time of presidential debate which held exactly before 2019 presidential election program in Indonesia. Five different dates or schedules had fixed with five different themes and gathered 1,334,999 tweets. For Law, Human Rights, Corruption and Terrorism theme, collected tweets were 2,64,087. For Energy, Food, Infrastructure, Natural Resources and Environment theme, collected tweets were 3,28,888. For Education, Health, Employment, Social and Culture theme, collected tweets were 2,71,732. From Ideology, Government, Security and International Relation theme, 1,76,027 tweets were collected and from Economy, Social Welfare, Financial, Investment and Industry, 2,94,265 tweets were collected.
Then, selected only indonesian tweets and omitted other language's tweets. The following, punctuation, stop words, double words, special characters etc. All unnecessary information were removed and tokenized each word of cleaned tweets. After preprocessing and cleaning, Latent Dirichlet Allocation (LDA) algorithm was implemented to extract the topic of each tweet. This topic was checked whether it represented its theme or not. Then, a pre-trained sentiment analysis model was used to detect the tweet to give a polarity score like 0 for negative tweet and 1 for positive tweet. These researchers have basically implemented a model using topic modeling to identify hatred and neutral tweet.
These authors scraped Indonesian tweets from twitter to make own dataset and to train and test their model which also in case of us was needed to create our own

dataset. This paper might also be helpful for our further analyis when we will make bangla dataset.

In [30], these researchers also collected data from twitter using twitter API but slightly differently. These people selected some accounts who frequently tweet hatred things and collected 1235 tweets from those accounts and those data had been verified by the Office of Language Hall of west Java province. From 1235 tweets, 652 contained hate speech and the other 583 did not contain any hate speech. Then, these tweet data was labelled manually like 1 was labelled to a tweet if that tweet contained hate speech and 0 if that tweet did not contain hate speech. The following, data was preprocessed and TF-IDF was used on that preprocessing result to count the appearance of a word in a single sentence or a single tweet. Backpropagation neural networks was implemented and trained and tested 9 times with 9 different combinations and got the best accuracy when training and testing data was partitioned by 90% and 10%. Firstly, kept the learning rate fixed on 0.1 and tested epoch from 100, 150, 200, 250, 300 where 250 proved to be the best with 90% accuracy. Then, kept the epoch fixed to 250 and changed the learning rate for five times where 0.11 learning rate was having 90.5% accuracy for their model. After all the parameter tuning done, these group of researchers obtained an average precision of 80.664%, recall of 90.07%, and Accuracy of 89.47%.
These authors used tf-idf vectorizer like us whcih indeed shows very promising acuracy in their research and also later for our research.

One more important research is [28]. These enthusiasts worked on a binary classification problem which only says whether the data is hate or no hate. These enthusiasts divided their entire process in 4 part which were Data collection and annotation, Data preprocessing, Feature extraction and Classification and evaluation. Natural Language Toolkit (NLTK) was used for preprocessing and sci-kit learn libraries were used for creating their model. Data was consisted of public written comments of a public post and Dataset was collected from a SriLankan Colombo Telegraph website that discussed SriLankan matters. Only those articles were selected that had more than 25 comments in April and May 2017. The dataset had 1500 comments from where 1000 comments were just picked by omitting unnecessary 500 tweets. From those 1000 comments, 421 was annotated as hate and 579 was annotated as No hate. The following, preprocessed those 1000 data by using NLTK to omit the irrelevant information and tested five different classification algorithms namely Support Vector Machine (SVM), Naive Bayes classifier (NB), Logistic Regression, Decision Tree and K-Means. So basically, one unsupervised learning model was created using those five. These five algorithms were implemented on gathered dataset. After this, the dataset was splitted into two. First dataset was named as DS1000 and DS500 where DS1000 contained the whole 1000 comments that was collected at the first place and DS500 contains half of 1000 comments. DS1000 dataset was partitioned for training and testing purposes in which training dataset had 670 comments and testing dataset had the other 330 comments. From those testing dataset, 124 comments was 'Hate' and the other 206 comments was 'No Hate'. In contrast, DS500 dataset was also partitioned for training and testing purposes of the model where training dataset had 335 number of comments and testing dataset had 165 number of comments. From 165 comments of testing dataset, 'Hate' was 86 and 'No hate' was 79.

Bag Of Words (BoF) and Tf-Idf was used as feature extraction method. For BoW features, their experiment showed Naïve Bayes Classifier performed better in DS1000 dataset and had f-score of 0.709 which was better than DS500 which proves performance depends on the size of the dataset. For Tf-Idf features, again Naïve Bayes classifier performed better on DS1000 dataset compared to DS500 dataset and other supervised learning model. Naïve Bayes Classifier's f-score was 0.719. All in all, This experiment got the best accuracy from Naive Bayes classifier while Tf-Idf as feature extraction which was 71.9%.
For feature extraction, we used tf-idf like them but for classification, XGBoost was used rather than their classification algorithms to give a different dimension to our research.

These researchers from [43] tried to propose a new automatic way to detect hate speech on Facebook. Their whole research was processed into four stages - a) The Discovery Stage b) The Sensitive Social Data Collection Stage c) The Sensitive and Emotion Analysis Stage d) The Clustering Stage
Discovery Stage: In Discovery stage, some pages from Facebook were selected that usually discussed hatred things before or promoted hate speech on Facebook. From the expert of American politics, some names were identified that talks about sensitive issues very frequently. Then to crawl data from those accounts, Facebook Graph API was used to extract those pages by using the word "Like" and collected 17,176 pages, 49,968 "Like". Furthermore, 'Betweenness centrality' was also introduced to identify the most influential page (the page that discussed most about sensitive issues) among those. Among all the pages, centrality score determines only 50 pages and those were kept.
The Sensitive Social Data Collection Stage: In this stage, these group of people filtered the data. 1000 posts/per page were taken and also 1000 comments were also taken from those 50 most influential pages using Facebook API. For these 1000 posts and 1000 comments filtration, they chose two dictionaries in which first dictionary consisted of swearing and bad words or phrases from the internet and second dictionary consisted of words or phrases or sensitive topics that usually triggered specific group of people.
The Sensitive and Emotion Analysis Stage: This stage was basically further analysis of the text that was found on sensitive social data collection stage. Using sentiment and emotional analysis, they removed further unnecessary text from their data. For sentiment analysis, they used a tool named VADER (Valence Aware Dictionary for Sentiment Reasoning) which basically told, a text is positive (did not contain hate speech) or negative (contained hate speech) or neutral (contained neither positive nor negative). Emotional analysis was implemented to know the inner emotion of a positive (contained hate speech) text. For emotional analysis, they used a tool called JAMMIN.
The Clustering Stage: In this stage, the expectation was that all the unrelated text would be omitted and only the negative (contains hate speech) text would remain for the rest of their work. For the clustering, they used K-means algorithm which was basically an unsupervised learning algorithm. But before that, they tokenized their data, remove stop word and omit anything (special characters) rather than the text that is containing letters. Stemmed each word to keep only the base or root of each word. The following, TF-IDF vectorizer from Scikit-learn was used to

know the frequency of a word in a document. TF-IDF also vectorized the document and this resulting matrix from tf-idf vectorizer used as input for K-means clustering algorithm. For their experiment, these group of researchers kept k on 20. Later, this experiment performed even better when they used n-gram in their work.
These groups of authors mainly focused on clustering of the dataset. But these authors showed how n-grams changed their model's efficiency which later helped us to go with n-gram rather than unigram, bigram in our model.

2.2 Hate and Offensive texts

In this paper [40], authors proposed a fuzzy approach on cyberhate classification which outperforms other non-fuzzy and existing classifier like Support Vector Machine, Naïve Bayes classifier, Decision Trees, Deep Neural Network etc. They used four datasets of hate speech (religion, race, disability and sexual orientation) collected from twitter and implemented their model.
These researchers collected their datasets based on some specific events in which events religion, race, disability and sexual orientation were present. Therefore, they searched on twitter- "woolwich" for religion, "Obama" for race, "collins" for sexual orientation and "paralympic" for disability. At least, Four annotators were required to label the class of their dataset. Each tweet were given to the annotators based on three class ("Yes", "No", "Undecided"). Four datsets were produced by these annotators where Religion consists of 1901 tweets, Race 1876 tweets, Disability 1914 tweets and Sexual Orientation consists of 1803 tweets.
For data pre-processing like every other time, tweets were converted into lower case so that every word in in same case, stopwords, punctuation and other unnecessary information was removed from the datasets and remaining words was stemmed into base word using snowball stemmer. In feature extraction, over all four feature set was made for each dataset. Those are Bag Of Words (full feature set), Bag Of Words (subset of selected features), Doc2Vec (full feature set) and Doc2Vec (subset of selected features). When evaluating their fuzzy approach, their fuzzy approach outperforms other classifiers like Decision tree algorithm, Naïve Bayes classifier (NB), Support Vector Machine (SVM), Gradient Boosting Algorith (GBT), Deep Neural Network (DNN) comparatively. In BOF (full set), their approach performed worse than others specifically for religion. But for Doc2Vec, it performed better in both full feature set and sub set. For BOW (sub), 0.725 (average) in religion, 0.787 (highest) in race, 0.729 (highest) in disability and 0.664 (highest) in sexual orientation. For Doc2Vec (full), they got 0.522 (highest) in religion, 0.56 (highest) in race, 0.747 (highest) in disability and 0.464 (highest) in sexual orientation. ForDoc2Vec (sub), 0.508 (highest) in religion, 0.585 (highest) in race, 0.738 (highest) in disability and 0.52 (highest) in sexual orientation. Their accuracy is not the best accuracies when we consider that in terms of research perspectives. That was probably happened because of the unstructed dataset that these researchers used on their model.

2.3 Hate, Offensive and Neutral texts

Researchers of [31] focused on multi label classification like us. These authors built their model using three datasets. The First dataset contained Hateful, offensive and clean data. Second dataset contained hateful, offensive and Neither. Third Dataset consisted of Sexism, Racism and Neutral data. Both racism and sexism was considered to be hateful data. Also, for all three datasets, third class labels ('clean', 'neither', 'neither') were representing neutral. They combined their all three datasets to make a bigger dataset which was supposed to be perform better and was supposed to give them a better accuracy according to them. For testing and evaluation purposes, they divided their 25,000 data into three sets. First set was named as 'Training Set' data which had 21,000 (7,000 Hateful, 7,000 Offensive and 7,000 neutral) of tweets. Second set was named as 'Test Set' which had 2,010 (670 hateful, 670 Offensive and 670 neutral) of tweets. Third and last set was named as 'Validation set' which also had 2,010 (670 hateful, 670 offensive and 670 neutral. To continue with, data was preprocessed. In preprocessing, all unnecessary informations was omitted like having URLs in mid-sentence, starting a sentence with a user name, irrelevant expressions and many more.Then, Data was tokenized, Parts of Speech (PoS) tagged and lemmitized. For lemmitization and tokenization, OpenNLP was used but for Part-of-speech tagging, Gate Twitter PoS Tagger was chosen instead of OpenNLP because of the poor performances of OpenNLP on noisy and unstructured data. These enthusiasts also developed a small tool which extracted necessary information from hashtags (#).
Furthermore, Data was extracted from a tweet using 4 sets of features. These features were- Sentiment-Based Features, Semantic-Based Features, Unigram Features, Pattern Features.
Sentiment-Based Features: This feature basically gave polarity-based result. Polarity based result means it said which sentence contained hate speech and which did not. Suppose, there is a sentence like 'I hate you'. Sentiment based feature would give a polarity of 'I hate you' sentence which clarified this sentence contained hate speech or not.
Semantic-Based Features: Semantic based feature used to find any emphasized expression of a sentence. This was introduced to find the actual mean of those sentence which contained exclamatory symbol, question mark etc. For example, "Why aren't you going back to your home and live us in peace?". This is an offended sentence and slightly hatred but still it does not contain any hate word that can assure that it is an offended sentence. So, these symbols (exclamatory, question mark etc.) cannot just be discarded from a line. Using semantic based features, these group of researchers tried to find the changed meaning when used these sorts of expression.
Unigram Features: Unigram feature focused on explicit meaning of a sentence. For example, "I hate you". It took 'hate' from the whole sentence and assured explicit form of that sentence.
Pattern Features: Pattern feature helped to identify the implicit form of a sentence through different steps.After feature extraction, J48 graft algorithm was chosen (used to create a single variate decision trees) to implement their approach. These researchers chose J48graft algorithm because it performed better than Sup-

port Vector Machine (SVM) and Random Forest as J48graft had special parameter tuning.
For Binary classification, the offensive and hatred class was combined, named the new class as offensive and ran their model on these two classes which were 'Offensive' and 'Clean or Neutral'. Total 14,000 tweets were together in 'offensive' class and 7,000 tweets in 'clean' class. For the test set, 2680 for 'offensive' class and '1380' for 'clean' class. Then, these researchers performed the binary classification on their validation set and got the accuracy of 87.4% using all the features in 'offensive' class'.
For ternary classification, the 'offensive' class is divided into class 'offensive' and 'hate' again. 'clean' class remains same as previous. So, they had 3 classes here-'Offensive', 'hateful' and 'clean'. Each class contained 7,000 tweets. Then, they ran the ternary classification on the validation set and got the accuracy 78.4% using all the four features. It had to be less because there was always a confusion between 'hateful' and 'offensive'. Even we human make mistakes to distinguish between hate and offensive because hate is kind of offensive as well. Indeed, their accuracy fall to 78.4% for ternary classification.
To sum up, these authors basically tried give an approach that automatically detects hate speech and offended speech from twitter. They tried on binary classification and ternary classification both. In binary classification, they got the result 87.4% and in ternary classification, they obtained 78.4%. This research is similar to our research in a sense that these people worked on multi class classififcation like we did in our research.

2.4 Offensive, Hate, Neutral and Hate, Neutral texts

Another research is [15], these authors focused only on hate speech of Italian language. As per as dataset is concerned, these researchers built their own crawler to collect data from Facebook as there was no API available for Facebook like tweepy in twitter. This is something that is very new compared to other research. Author of this paper selected some high profile public web pages of politicians, artists or any newspaper written on Italian Language. These people collected 17,567 amount of data from 99 Facebook public posts and annotated them by five distinct bachelor students where majority system is considered for each comments. These researchers implemented their two classifier Support Vector Machine (SVM) and Long Short Term Memory (LSTM) from Recurrent Neural Network on two different categories of crawled dataset where first category consists of three classes Strong Hate, Weak Hate and No hate and second category consists of only two classes Hate and No Hate. For first category, 64.61% and 60.50% and for second category, 72.95% and 75.23% is the accuracy given by SVM and LSTM respectively. Cause of the such fall result for first category is very less data given for Strong Hate and Weak Hate which in case of second category decreased as more data was given for Hate. Also, one more probable reason could be less annotators. Annotators with 10-12 person could have given these researchers far better result.
To add more, these group of researchers used annotation process to create their

dataset like us but we felt to appoint 9 person for annotations because less than 9 annotators proved to be very poor for their research.

2.5 Sarcastic and Non sarcastic texts

Researchers from [44] divided their whole approach into four parts that were a) Dataset Gathering b) Data Pre-processing c) Implementation using sAtt-BLSTM convNet model d) Evaluation.

Dataset Gathering: These authors used two datasets where the first one was a balanced Dataset and Second one was a Random Dataset. In balanced dataset, they collected 15,961 tweets using tweet id from which 7994 are sarcastic tweet and 7324 are non-sarcastic. for Imbalanced dataset, they collected some random tweets. From those random tweets, 15,000 were sarcastic and 25,000 tweets were non sarcastic. Number of sarcastic tweets had to be less as it was randomly taken and the presence of sarcasm in those randomly taken tweets is very less. Sarcasm detector tool was used to build this randomly sampled dataset.

Data Pre-processing: As usual like other experiments, they also pre-processed their collected data. All the URLs, hashtags had been removed from the dataset. Tweets starting with user name, non ascii english characters all also had been removed from the dataset. All the tweets were tokenized by Natural Language Toolkit (NLTK). All tokens were converted into lower case so that every token has the same format, then porter stemming algorithm was used to go to the base of each word. For the semantic analysis of the tweets or to find the figurative text (texts that presents a different meaning because of its symbols), they focused on number of exclamation marks (!) in a tweet, number of question marks (?) in a tweet, number of full stops (.) on a line, number of capital letters and number of 'or' in the middle of a tweet. These symbolic clues are used to know the implicit form of a tweet or a sentence.

Implementation: In total, these group of people used 8 layers to build their model which were Input layer, embedding layer, BLSTM layer, attention layer, convolution layer, activation and relu layer, max-pooling layer and representation layer. Pre-processed data were fed into input layer. In the embedding layer, input maps to vectors (real values). Glove had been used to create that word vector table. Then, Bidirectional long short term memory (BLSTM) layer was used to have the data from past (input) to future and future to past where past (input) to future means forward directional and future to past means backward directional. BLSTM works very differently than Unidirectional Long short term memory or LSTM. LSTM works only with forward direction (past to future). For example, "I will go to school only if he goes with me". How unidirectional LSTM (forward) sees is that, it predicts "I will go school". Forward LSTM predicts "I will go to school" and Backward LSTM predicts "only if he goes with me". That is how BLSTM predicts well than Unidirectional LSTM. Then atten layer, convolutional layer, Relu layer, max-pooling layer and representation layer was used respectively to build their model.

Evaluation: This proposed model was evaluated on two datasets named SemEval 2015 task 11 and random tweets. For LSTM, they got the accuracy 84.89% on SemEval Dataset and 79.75% on Random Tweets. For BLSTM without attention, 86.32% got from SemEval dataset, and 81.03% got from Random tweets dataset.

For BLSTM with attention, 89.03% accuracy got from SemEval and 85.65% accuracy got from Random Tweets dataset. For convNet, 91.60% and 88.28% accuracy respectively got from SemEval and Random Tweets. From all of the models, sAtt-BLSTM convNet obtained the best accuracy compared to other model which is 97.87% (SemEval Dataset) and 93.71% (Random tweets) respectively.
These authors implemented their model on multiple datasets like us. what this proves is that any model that is trained and tested on multiple datasets is suitable for any dataset in research community and is not dependent on any single dataset. These authors implemented CNN on their work which is also similar to us.

2.6 XGBoost Related Works

Extended gradient boosting has been a very recent addition on decision trees algorithms. XGboost achieved a great response on the research community because of it high accuracy and high speed in the performance. Being New addition on research community, there is no researches done on hate speech detection till now using XGBoost and very few on text classification.

One of research work using XGBoost is [23]. These researchers implemented their work on Jigsaw toxic comment classification data. Four classification algorithms were chosen from where the first one is Logistic Regression and other three classification algorithm is very recent which are NBSVM (Naive Bayes with Support Vector Machine), extreme gradient boosting and FastText-BiLSTM (fasttext with bidirectional LSTM). Usually, transformation of dataset in any research increases the accuacy and gives a better performance but in their research, taransformation of the dataset dcreases the accuracy for Logistic Regression and Naive Bayes with Support Vector machine classifier. Better F1-score was needed on toxic data rather than on cleaned tweets as the research mainly focused on toxic comment. As per as the result is concerned, NBSVM and fasttext-BiLSTM got the f1-score of 0.8, Logistic regression obtained 0.74 and xgboost had around 0.57. FastText-BiLSTM and NBSVM performed more consistently than XGboost and Logistic Regression in most of the data transformations. But in terms of precision, XGboost achieved the highest accuuracy than others and for recall, XGBoost obtained lowest value because negative class data was not enough on the dataset.
Although XGBoost is proved to be very good and highly efficient in our research even after the data transformations.So this is totally dependent on the dataset and research sector how XGBoost is going to be performed.

Chapter 3

Methodology

In order to develop a proper machine learning model for text classification we have explored some related fields of text classification, because only hatred detection might be a very small and specific domain. After analyzing a lot of research on text classification problems like hatred detection, emotion detection from text, offensive language classification, we have found some methods of Natural Language Processing. Most popular text classification models are Neural Network, Support Vector Machines (SVM), Multinomial Logistic Regression (maximum entropy), Naive Bayes, Decision Trees, and Random Forests. Moreover, in some cases two or three models have been used together which is called ensemble approach of text classification[18]. In addition to that, methodology also varies on different feature extraction methods and data pre-processing techniques. Depending on the domain and complexity of the domain different text classification methods can be applied for efficient and best result. The practice of using Convolutional Neural Network for text classification is gaining the attention of the researchers nowadays. In our proposed model we have implemented a Convolutional Neural Network which has been tested with three different dataset to detect hatred and offensive language from social media posts in order to generalize the proposed models. Among them, two have been collected from other researchers work and one has been created using twitter API by the authors of this paper. Though Convolutional Neural Network results in a high accuracy for our both of the data-set but still we have applied another classification method XGBoost to get better accuracy. In the context of machine learning and supervised learning XGBoost added a great value on high accuracy and high speed. In the following paragraphs both of the classification method and its detailed work will be discussed.
We have discussed our methodology in four subsections.

- Data set
- Data Pre-Processing
- Creating own data-set
- Pre-Processing own data-set
- CNN based classification Model
- Classification Model using XGBoost

These four subsections are described below:

3.1 Data set

In text classification the crucial and most critical work is data set collection. Moreover, if the text classification domain is social media text analysis,then data set collection is more challenging because of the confidentiality of social media sites. Web crawling or data from Wikipedia or YouTube might be an easier way of collecting data but those data might not be well structured. Also there might not exist hatred or offensive data as much it would be in facebook or twitter. Though in Facebook we can find so many hate and offensive data but unfortunately Facebook does not share data for research purpose. On the contrary, Twitter allows researchers to collect tweets from twitter. That is why researchers collect data using Twitter API for text classification or hatred and offensive text detection. At first, we have tried to collect labeled data set, but as it is mentioned earlier, there are very few works on hatred and offensive data. Fortunately, we have found a very clean and perfect data set on hatred and offensive texts. So, in this research we have used two different data sets for our model. The first data set consists of almost 25K twitter comments and tweets which was collected and used by[13]. The second data set we have used has been collected from kaggle [12].

3.1.1 Hate-speech-offensive-language dataset by Davison

At the first place 85.4 million tweets was collected by searching with hate speech lexicon which was identified by the Internet users. Out of those huge data only 25k random data was labeled in three categories by CrowdFlower workers. Out of 24802 labeled tweets 76% are labeled with offensive, 16.6% were neutral and 7.4% of the tweets are hate speech. As the number of hate, offensive and neutral data is uneven in the dataset, in our model we have used 1535 offensive comments out of that 76% comments to make the number of offensive comments even with hate and neutral. We also used 1457 neutral and 1430 hate tweets out of those labeled comments. As training we have used total number of 3524 tweets and for testing we have used test set of 898 tweets. For the validation we have also used 20% tweets from the training set. In this paper the dataset will be called as dataset 1 or 1st dataset in the up coming sections.

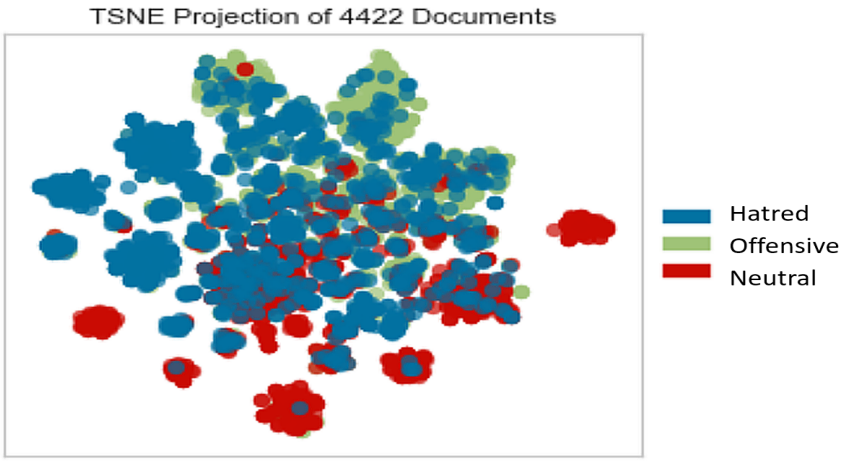

Figure 3.1: t-SNE projection based on tf-idf

Figure 3.1 represents the visualization of document similarity of the data set using t-distributed stochastic neighbour embedding(t-SNE).The text of the data set was vectored using tf-idf then the vectorized distribution of classes are visualized.

3.1.2 Toxic-comment-classification-challenge dataset of kaggle

We have collected our second dataset from kaggle which was made for a competition named toxic comment classification challenge. The dataset contains 160k comments in a CSV file named "train.csv" where every comment labeled with Toxic, Severe Toxic, Obscene, Threat, Insult, Identity_Hate. From the dataset we have collected 1405 comments whose were labeled with identity_hate and rest of the class were labeled as don't care for those comments. We have recognized those 1405 comments as hate. In the dataset there were 140k neutral data from there we have collected only 1400 neutral data for our model because we wanted to use relatively close number of hate and neutral comments to avoid biasness. In order to feed the data we have made a training set of 2269 comments and to test our model we have used 536 comments. Finally, for validation we have used 20% comments of 2269 training data. In this paper the dataset will be called as dataset 2 or 2nd dataset in the up coming sections.

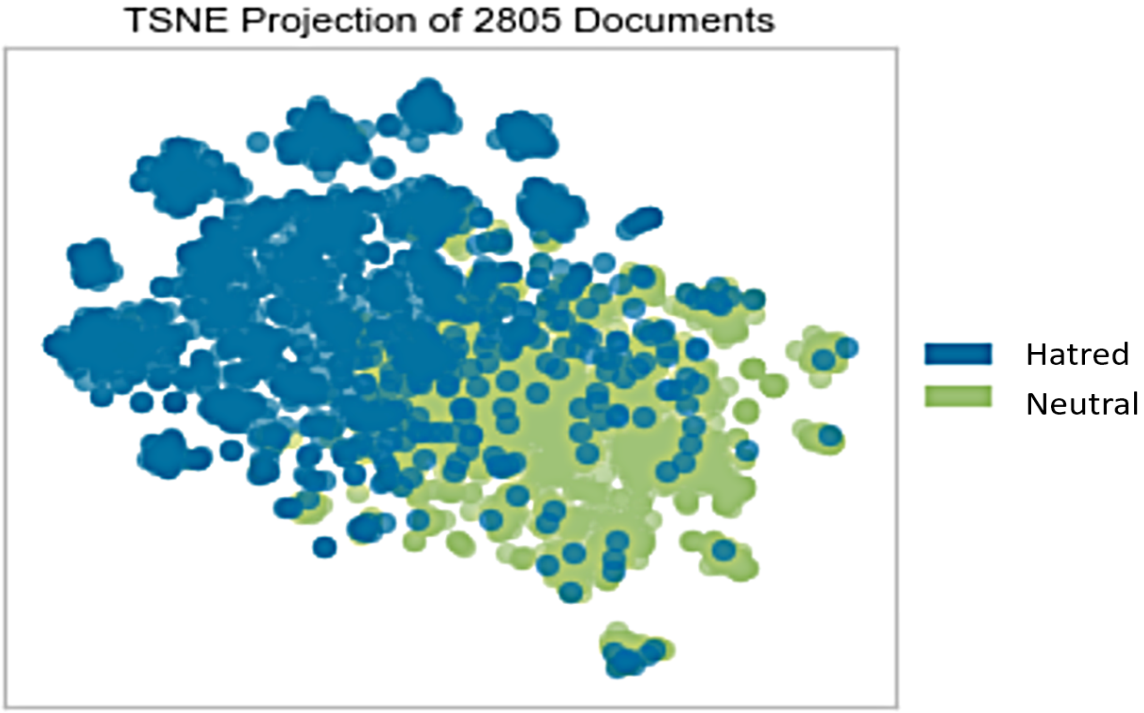

Figure 3.2: t-SNE projection based on tf-idf

Figure 3.2 is a visualization of the document similarity of this data set which is shown using t-distributed stochastic neighbour embedding(t-SNE).The text of the

data set was vectorized using tf-idf then the vectorized distribution of classes are visualized.

3.2 Data Pre-processing

As our main target is to build a model for hatred detection from social media posts and comments. To train our model with twitter data; we are required to do data pre-processing. Twitter data and any kind of data from social platform contain so many unnecessary letters, symbols, emojis and also spelling mistakes. Generally streamed tweets consist of so many hashtags, username, email or URL, mentions etc. In order to utilize tweets for training a model we must remove all the irrelevant characters. Though we have collected a cleaned twitter data where there were no emojis and URLs but in some places still there were some problems like long tweets, a lot of unnecessary words, a mixture of upper and lower case letters, punctuation. Several techniques that we have applied to process our clean data are given below:

3.2.1 Tokenization

Tokenization is the process of breaking down text into characters or words depending on the model of the tokenizer. A tokenizer splits sentences of whole data corpus into words which allows manipulating data corpus easily[1]. Moreover, machine learning models cannot parse data directly in the text classification problems so we use tokenizer. We have used NLTk library of python for tokenization which tokenize words using a regular expression. We have used tokenizer because it will help our model in the feature extraction process.

3.2.2 Punctuation removal

In any kind of social media data whether it is a comment or a tweet there might be so many punctuations which are not necessary for data analysis or hatred detection[7]. We have removed all the punctuations using regular expressions.

3.2.3 Removal of Stop Words

Stop words are the articles, auxiliaries and some of the common words that are not necessary for text analysis. NLTK have some predefined stopwords but it also allows to add some more stop words or modify the stopword list. We have applied the NLTK stopword to remove unnecessary words from the corpus.

3.2.4 Padding

Sentences in a dataset are not the same in length which increases difficulties for a neural network model as neural network model cannot process sentences of different length .To solve this problem we have used padding of 100 lengths which will convert all the sentences in the same length. But in the case of XGBoost classification model we have not done any padding.

3.3 Creating Own Dataset

3.3.1 HNO-Tweet Dataset (Own Dataset)

In order to contribute in the social media data analysis, we have also created a twitter dataset consisting 2332 tweets of hatred, neutral and offensive data. Using the twitter API we have streamed almost .25M tweets and from these tweets we have randomly selected 3000 tweets for prepossessing. As all the tweets were streamed from the twitter using twitter API not all of the 3000 randomly selected were appropriate for our dataset. After that, data cleaning process have been done for removing ambiguous comments and different language comments. As we have manually checked and read the comments we have seen some of the comments were repeating which we have defined as ambiguous. After cleaning the data, 2332 comments have been selected and these cleaned dataset then labeled offensive, neutral and hatred by manually analyzing the comments. The comment labeling for each comment was done by 9 annotators and then depending on the majority vote each comment labeled with appropriate value 0(for hatred) , 1(for offensive), 2(for neutral). After labeling we have got 423 offensive, 598 hate and 1311 neutral data. We have split our dataset into train and test data for CNN based model where train set contains 1847 data and test set contains 485 data. For validation 20% of the total dataset has been used. On the XGBoost model same number of train and test data and for the validation 25% of the total dataset have been used. In this paper the dataset will be called as dataset 3 or 3rd dataset in the up coming sections.

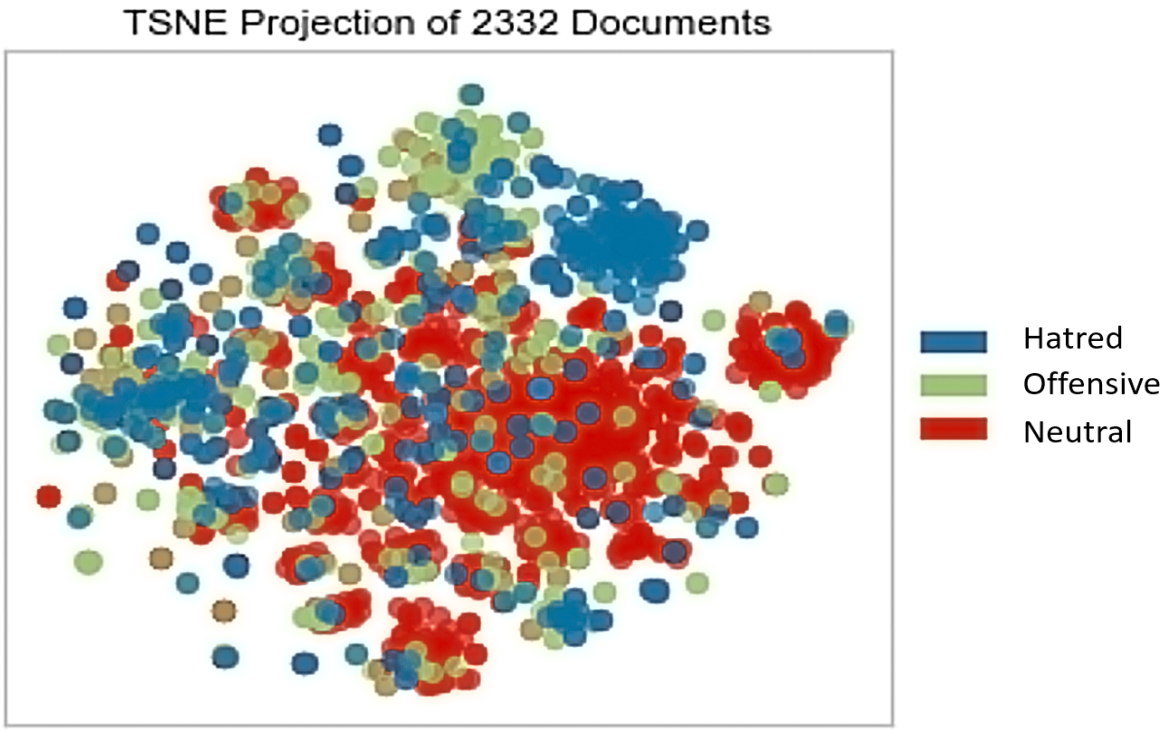

Figure 3.3: t-SNE projection based on tf-idf

Figure 3.3 illustrates the distribution of classes using t-distributed stochastic neigh-

bour embedding.The text corpus is vectorized using tf-idf and then visualized based on the distribution of classes.

3.3.2 Streaming twitter data

For preparing the dataset we have streamed tweets from twitter by using twitter API. Twitter API is a very handy tool for collecting data from twitter which allows researchers to stream tweets or collecting tweets by using user ID. In order to use twitter API a twitter application must be created by having acceptance from twitter authority. We have got the approval from twitter after applying for a twitter application and then used the twitter application for streaming twitter data. By streaming, around .25M comments have been found.

3.3.3 Data cleaning

In order to create the dataset 3000 comments have been randomly selected for cleaning data because streamed comments might be irrelevant. We have removed comments of different languages because our model is only made for English language. Also some comments which contain single words have been removed as single word sentences will increase the biasness.

3.3.4 Tweet tokenization

Earlier we have mentioned about tokenization which was text tokenization and both have a similar purpose. The reason behind mentioning tweet tokenization separately is tweets contain so many complex combinations of characters than a plain text. The existence of hashtag, mentions or usernames, informal and misspelled words increases the complexity of twitter data tokenization. Hashtags are some tags which allows a user to specify the topic of his/her post and twitter can generate a group of same hashtag tweets by using the hashtags. On the other hand, mentions and username are the user id of the twitter, where mentions starts with '@' and user id starts with some valid alphanumeric combinations. A tokenizer must generate the tokens by grouping every character in a manner that the meaning of the sentence remains exactly the same as the tweet and it is the most challenging task of tweet tokenization. In order to tokenize the corpus NLTK developed two different tokenizer where one is called word tokenizer and another is called tweet tokenizer[26]. Though the purpose of both tokenizer is same but these two have different internal functionality and slightly different output.

1. **NLTK word:** NLTK word generates tokens by separating characters on the basis of space and non-alphanumeric characters where "_" is treated as an exception of non-alphanumeric. The example given in table (3.1) provides a general overview how NLTK word works. The problem of NLTK word arise when a tweet consists some informal sentences with non-alphanumeric and alphanumeric combination, like the tweet in the example table. Word tokenizer will split the word '@bout into two different tokens '@' and 'bout' which will

generate a wrong word combination. Moreover it will also split the mentions and hashtags into different tokens like, the user id '@friend_id' will be split into '@' and 'friend_id' two tokens, because of the splitting the mention will be lost. Similarly, the hashtag 'get_a_life' will be split into '#' and 'get_a_life', as the tag has been split the tag won't be lost.

2. **Tweet tokenizer:** To overcome the problems of tweet tokenizing, NLTK made another tokenizer named NLTK tweet specially made for tokenizing tweets. The most benefiting part of using NLTK tweet is it does hold the user_name and the hashtag which allows us to group the tweets. Though in sentiment analysis or hatred detection system we avoid the username and the hashtags and other emojis but in some cases they can be used in text classification like grouping tweets, user profiling and etc. In the example on the table (3.1) we can see that tweet tokenizer can successfully tokenize the user name '@friend_id' and also preserves the hashtag as it is as'#get_a_life'. Though NLTK tokenizer successfully tokenize valid username and hashtags but still it finds difficulties in the cases where the usernames and hashtags are partially valid. As for example, if any user uses a hashtag where a non-alphanumeric character exists in the middle of the hashtag like '#b*tch', it still tokenize as one character which will be a non meaningful hashtag. As our target is to classify hatred, offensive and neutral data by analysing sentences, using tweet tokenizer will be very fine for us.

Tweet	I don't give a f*ck wh@t those $itty people s@y @bout me #get_a_life @friend_id.
NLTK word	'I', 'don't', 'give', 'a', 'f*ck', 'wh', '@', 't', 'those' '$', 'itty', 'people', 's', '@', 'y' '@' 'bout' 'me', ' #',' get_a_life', '@', 'friend_id'.
NLTK tweet	'I', 'don't', 'give', 'a', 'f*ck', 'wh', '@', 't', 'those' '$', 'itty', 'people', 's', '@y', '@bout' 'me', '#get_a_life', '@friend_id'.

Table 3.1: NLTK Word

3.4 Feature Extraction

In natural language processing and classification of data, word representation is the most important task. Word representation refers to the conversion of words into vectors which will allow a learning model to learn how to predict an unknown word's value. We have used fastText for word vectorization which was developed by Facebook AI research team. FastText is the best known word embedding library which vectorize a word not only by the word itself but also it analyses n gram character of the word.

3.4.1 Word Embedding

In this research, fastText has been used for word embedding. Now-a-days, models that are based on neural networks like Convolutional Neural Network (CNN), Deep Neural Network (DNN), Recurrent Neural Network (RNN) are getting popular. But achieving good performance from these neural network model has still been an issue that developers are facing. One of the issue is slowness of the model in both training and testing time. That inertia has to be solved in one way or another. Then, Facebook AI Research group introduced a solution called *FastText* with better time complexity and better run time in both training and testing time. FastText is recognized as most updated and is also the most recent word embedding model that has bubbled on the of mind research enthusiasts. FastText is the extended version of Word2Vec skip gram and Continuous Bag of Words (CBOW) model. In fastText, skip gram tries to predict close word of target word in a sentence whereas CBOW model takes all the words in an window and takes sum of their vectors to predict the target word of that sentence. From [10], FastText includes two additional features n-gram language model and Hierarchical Softmax which gives it faster time complexity along with the opportunity to train billions of words with a very high output space within a very short time. These researchers evaluated fastText on two different tasks named Sentiment Analysis and Tag Prediction [10] .

N-gram: Word2Vec uses Continuous Bag of Words(CBOW) model whereas Researchers of fastText thought to add Bag of n-grams feature because Bag of word gives a high time complexity. N-gram can be based on syllables, words, pairs of words and more. So it depends on the how it is being applied on an application. For example, n=3 for the word 'mysterious' means 'my, mys, yst, ste, ter, eri, rio, iou, ous, us'. Researchers used *Hashing Trick* to give fast and efficient mapping of the n-grams in the model [10].

Hierarchical Softmax: From [10], Hierarchical Softmax is used in when number of classes is high. For linear classification, the computational complexity is *O(kh)* where k derives the number of classes and h derives the dimension of the text representation. For improving this run time of linear classification, these researchers used Hierarchical Softmax and it shows the $O(hlog_2(k))))$ run time in training time which is actually better compared to *O(kh)*. It also has the advantage in finding the most probable class. In a tree, each node has the probability of reaching it from the root of that node. If a node is in *l+1* and the parent node are $n_1, n_2, n_3 \ldots\ldots\ldots n_l$ then the probability of *l+1* node is

$$P(n_{l+1}) = \prod_{i=1}^{l} P(n_i) \tag{3.1}$$

What this probability means actually is, probability of any node is always lower than probability of its parent. Using Depth First Search (DFS) and also monitoring the maximum probability in leaves gives the chance to erase any branch that is having lower probability than its previous. In test time, it shows the complexity of $O(hlog_2(k)))$ which is a great achievement if we compare it with other word

embedding model like *golve*.

To add more, FastText takes each line independently. What this means is actually if same word appears in two different lines, it takes same words two times and meaning of these words depends on sentence structure which gave model a better chance to understand the underneath meaning of any line. For example, "I hate you" and "he does not hate you rather he loves you". "Hate" word occurs two times in those two different sentences but what fastText does is that it takes both the words from those two sentences so that it does not miss any of the meaning. Unlike word2vec, fastText takes every word as composed of character n-grams. So, it sums all the vector of n-gram characters to create a vector for a word. But in case of word2vec, it treats each word a single corpus and set a vector for this individual corpus. Glove also treats word like Word2Vec. In this case, Both Word2Vec and Glove treats it's words similarly. fastText generates better word embeddings for each word because it also give vectors to rare words which it found through n-gram combinations. Because of additional n-gram features, fastText always has some extra words out of vocabulary which both word2vec and glove do not have.

We set epoch on 5, learning rate on 0.05 and thread on 4. We could have set epoch on 10, 25 etc. But we did not set those because how many number of epochs should be given that totally depends on model. We had to do parameter tuning to find the best possible combination of our model. We tried our model giving epoch 8,10,3,4 etc. but among all, we are having the best probable accuracy for epoch 5, that's why we kept our epoch on 5. Same goes for learning rate, the default learning rate for fastText is 0.5 but we tried different learning rate 0.005,0.0.3 etc. But we are having the best possible accuracy for our model when learning rate is 0.05. We kept our window size 5 which means we want our model to take 5 words before and after target word. There might be a query what will happen if a sentence size is lower than the window size of any model. For this kind of situations, window size is truncated to sentence size. For example, 'I Hate You' has the length 3. If we consider this example in terms of our model, then our model's window size will truncated from 5 to 3. We have set n-gram value of 2 to 6. This means we will take every combination of a word from 2 to 6. What this will benefit us is, it will cover maximum possible combinations of any word which will later help us to have better accuracy than others. To continue with, we also used hierarchical softmax as the number of class of our dataset is high. Our fastText model generates a vector of 100 dimensions for each word and saves them on a vector file which we later fed to CNN model.

3.5 CNN Based Classification Model

In recent years, Convolutional Neural Network has been widely used by the researchers for image processing, character recognizing, and text analysis and so on. Because of the architecture CNN learns higher order features by convolutions of the data and this gives CNN model the opportunity to learn more accurately. CNN got the attention of the researchers mainly for high accuracy in critical task, like applications of machine vision, self driving cars or autonomous cars, drones and so many. Now CNN is being used in natural language processing, sentiment analysis and other context of text classifications.

3.5.1 Basic structure of CNN

CNN is a deep learning algorithm which takes data as input and learns features from the data from a feature vector and then gives an output value by the classification layer. Convolutional Neural Network can be used in various purposes and depending on the purpose the pattern of hidden layers might be changed or evolve. The basic model of CNN has given in the figure (3.4).

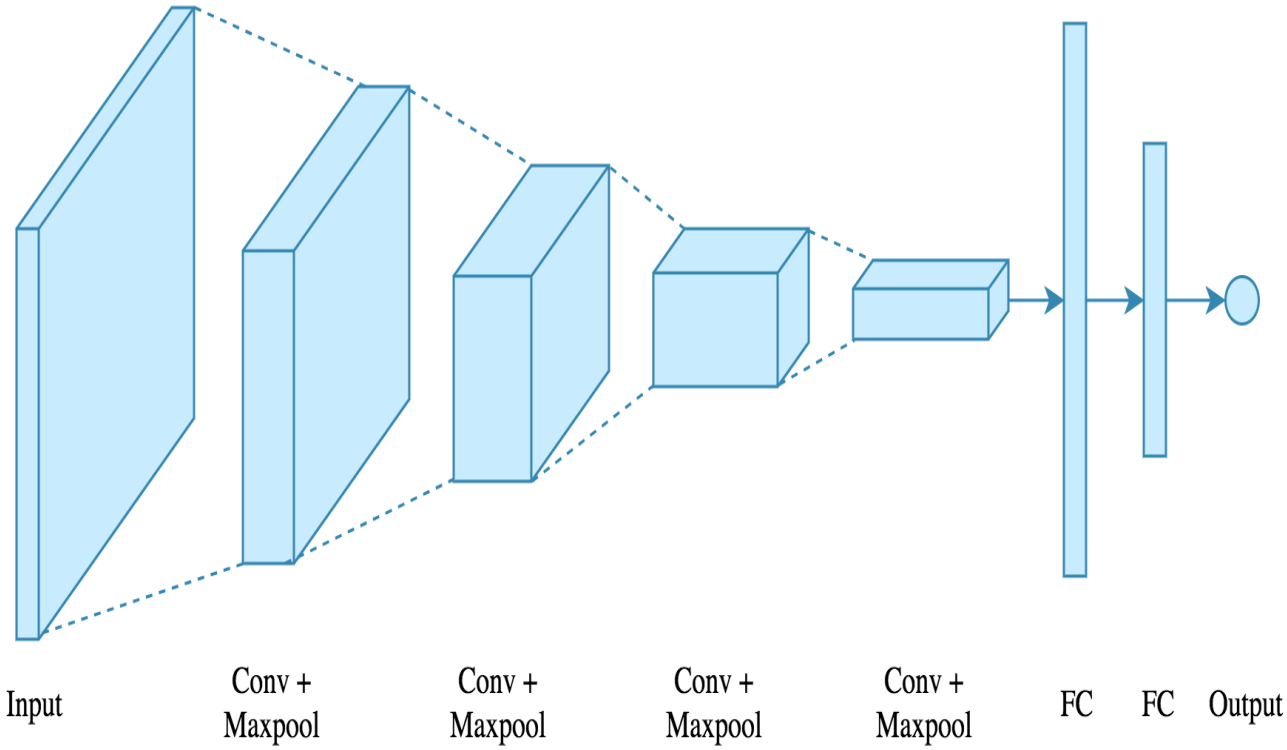

Figure 3.4: Structure of basic CNN model

A basic CNN model is mainly a multi layer neural network which may have single or combination of multiple convolution layers and pooling layers. In the figure the mentioned convolution layer is also known as feature extraction layer which extracts features from the input text. In multiple convolution layer architecture the output from previous convolution layer treats as the input for the next convolution layer. After every convolution layer the output is connected to a pooling layer which maps

the features collected from convolution layer to make the features invariant. At the end of CNN layers there is usually a fully connected layer[4].

1. **Input Layer:** For text classification the first layer which is input layer takes a matrix of text (s, k) where s refers to the length of each sentence and k refers to the dimension of the word vector. To represent all words in same length s length of padding might be used. If $x_i \in R^k$ is the vector of a word in the text where k is the dimension then x_i can be represented as

$$x_{1:n} = x_1 \oplus x_2 \oplus x_3 \oplus ..... \oplus x_n \tag{3.2}$$

2. **Convolution layer:** After input layer there will be convolution layer which filters the words applying a window size and produces new feature. Convolution is basically a mathematical operation which takes input and gives a feature map as output by applying convolution kernel. Convolutional neural networks can be manipulated by parameters used in convolution layer. One of the parameters is filter which has been mentioned as window size[17]. Filters are the function that can have the same height and width of input or it can be smaller than input (length only in 1 dimensional). Another parameter is the activation function in convolution layer which allows the model to deal with nonlinearity of the output. When an output from a layer treated as an input for next layer the output of the previous must be propagated through activation function so that an extreme output value can be used. There are some activation functions which are being used in deep learning model like, linear activation function, sigmoid function, Tanh, Hard Tanh, ReLU. Let, $w \in R^{h.k}$ is the filter and the generated feature is c_i, then equation will be

$$c_i = f(w.x_{i:i+h-1} + b) \tag{3.3}$$

$$f(x) = 1/(1 + e^{-1}) \tag{3.4}$$

In the equation (3.3) f is activation function namely sigmoid function which is used in our model. Another notation b in the equation (3.4) refers to bias. The result of applying filter on each window of a sentence will be a feature map which can be represented as

$$c = [c_1, c_2, c_3, ...c_{n-h+1}] \tag{3.5}$$

3. **Pooling layer:** After the convolution layer we will get all the features of every word for each sentence. Pooling layer is used for dimensionality reduction of the feature map. Generally, after every convolution layer there exist a pooling layer to pool out highest valued feature or the most feature from a feature map which is output of the convolution layer. Most common pooling

layer is maxpool which we have used in our model. Let p the pooling operation c_i the feature then p can be expressed as

$$p = max[c_1, c_2, c_3, ...c_n] \tag{3.6}$$

4. **Fully Connected layer:** Fully connected layer is used for getting the output from the neural network. The output from the neural network will be more efficient if we apply any regularization. Regularization used to prevent overfitting from a model. Overfitting refers to the misjudgment of a model when it has to predict an unknown data. According to[4] dropout is a mechanism which Zeros out some part of input data for preventing overfitting. We have applied dropout in our model to overcome the challenge of overfitting. In addition to that, learning rate is another hypermeter which refers to how fast the model to find out the result or goal. Very high learning rate might lead us to wrong prediction and very small learning rate will also be a problem. RMSprop[2] is a very well known effective optimizer which minimizes.

3.5.2 Our Proposed CNN Model

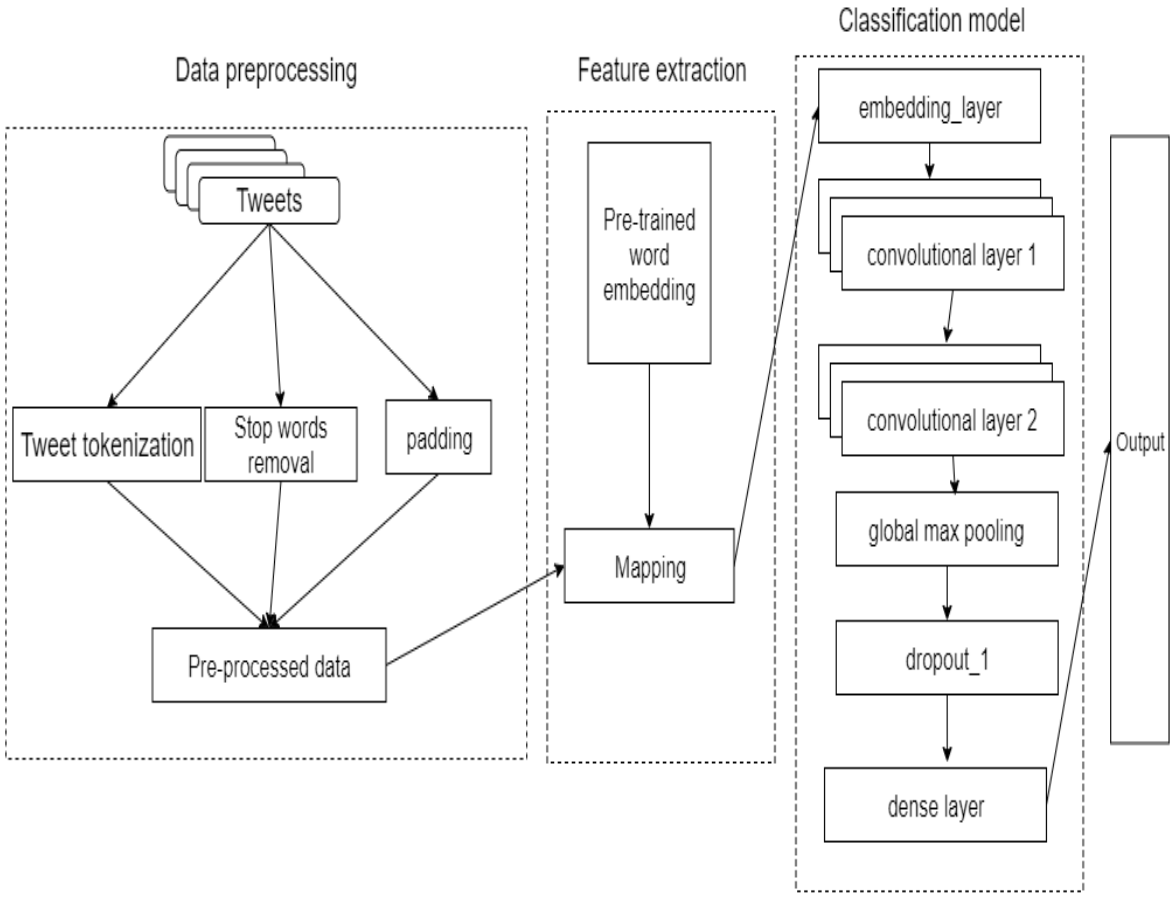

Figure 3.5: Proposed CNN model for Dataset I and Dataset II

Our classification model is based on a convolutional neural network which has 5 layers without counting input layer. We have considered our task as both multi and binary classification problem for data set I and data set II which is solved by a supervised learning model. As mentioned in the earlier part of data set, all of the three datasets were divided into training and testing. All the word of these datasets has been inputted in the feature extraction process. After that, we have inputted the vector file into the embedding layer where the layer's input length was 100 to match the dimension of the vector file. The output from the embedded layer then treated as input of convolution layer where we determined filter size of 40. We have also used Global MaxPooling which specifies the model to take maximum of each feature map. Fully connected layers have prone to overfeed that is why a dropout layer has been used to prevent over fitting. Finally, we have used the "rmsprop" optimizer to minimize the loss function of our model. The model we have used gave us the accuracy of 85% for the first dataset and 89% for the second dataset on classifying the data.We had to change the structure of the CNN model slightly for our 3rd dataset because it does not contain an even ratio of offensive,neutral and hatred data. In the CNN model for HNO-tweet (3rd dataset) after input layer,embedding layer is used in order to feed the model embedded word vector. After that a dropout layer have been used where the dropout value is 0.4. Then a convolution layer have been used in which the kernel and stride size are 5 and 2 respectively.After that gobal maxpolling have been used before using the dense layer to get the output. On contrast to the first proposed model here only one convolution layer has been used. Another modification in this CNN model is the change of activation function of convolution model, sigmoid function has been replaced by ReLU. After the learning process our CNN model has given the accuracy of 76% on test data for HNO-tweet dataset.

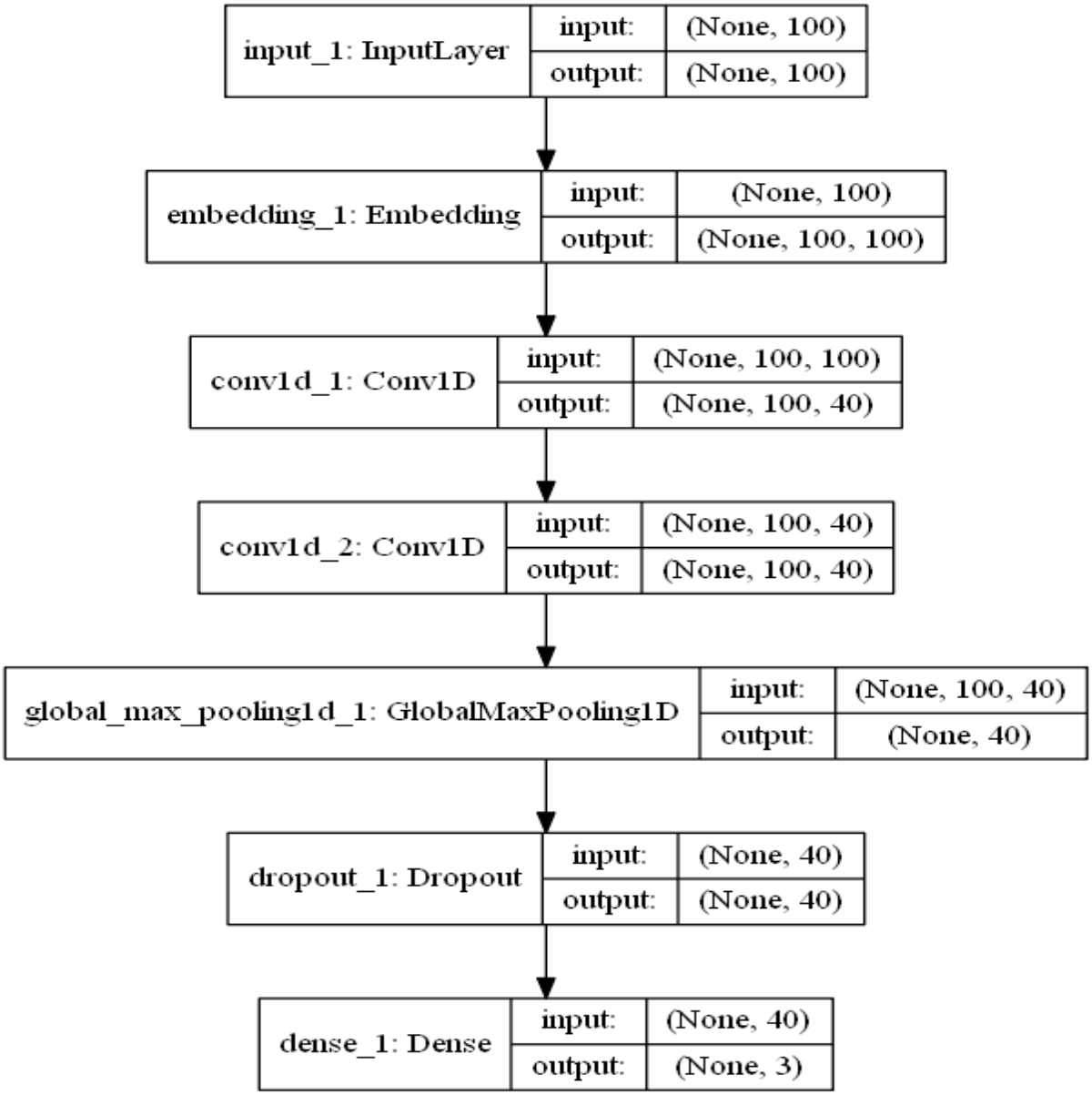

Figure 3.6: Proposed CNN Architecture

3.6 Classification Model Using XGBoost

In the context of text classification we can apply different types of machine learning models to get better results. In the earlier section of methodology we have mentioned the neural network based machine learning approach and now we are going to talk about another model that we have applied. In recent time tree boosting algorithms are getting much more attention from researchers because of its high accuracy and high speed in the performance. One of the most recent tree boosting methods is XGBoost which can be used by downloading the XGBoost library. The benefits of applying XGBoost in any supervised learning approach are so many which mainly convinced the modern researchers to apply after Tianqi Chen the inventor of XGBoost, has published the official paper[9]. Moreover, XGBoost has won all the competitions on machine learning in recent time which influences us to take a look on XGBoost.

XGBoost itself a tree boosting algorithm based on gradient boosting which follows the same tree structure that other tree boosting follows. The differences which made XGBoost superior than the other tree boosting algorithms GBM and Adabbost, is the internal functionality and the attributes of it. In the tree architecture the head node refers to the condition and depending on the condition the process splits into branches. Depending on the length of the tree length mentioned in the model the tree will go deeper. The node where tree ends will be the leaf node or the final decision node.

In a classification model XGBoost takes some labeled data as an input for supervised learning and finally predicts a targeted out by analyzing some parameters. Extracting parameters from the data and and also extracting features are most important tasks in this method. One of the most advantageous parts of Extreme gradient boosting is the application of different parameters which actually manipulates the classifier and increase the learnability or accuracy.

3.6.1 Features of XGBoost

1. **Increase the learnability:** In the cases of weak learner where the classifier provides poor accuracy and rarely shows expected results pooling boost can increase the learnability of the classifier or predictor and turns the predictor into a strong learner. The optimization functions uses in XGBoost helps the learner to increase the learnability. A weak learner always tends to predict a wrong result and it will label the output with a wrong result no matter what distribution we apply. Boosting algorithms or XGBoost finds out some low errored hypothesis from the whole hypothesis class and then generates a final hypothesis group from the selected low errored hypothesis.

2. **Regularization:** Regularization is a method which reduces the variance and sample error by manipulating the estimated coefficients[11]. The datasets that are being used in the recent times has more number of features which increases the chance of overfitting the model and it results in sample error. That is why regularization has been introduced which is called regularized regression which has been applying for generalization, early stopping, sparsity manipulating etc[38]. The most advantageous component of XGBoost that it

has built in regularization methods like Lasso regression and Ridge regression which reduces the biasness of a model.

3. **Parallel processing:** All the tree based learning model follows sequential process because in tree building process each tree builds after the completion of the previous node. But in the case of XGBoost it allows parallelization process which implies faster computation in machine learning model. In parallel processing the model uses multiple processors and multiple cores of a processor to execute. Moreover, XGBoost also supports Hadoop implementation which is actually a map reducing program[20]. Figure 3.4.1 shows the parallel processing of two different branch.

4. **Handling missing values:** While handing real world problems on analyzing data it may occur that the input may tend to be sparse. It is another facility of using XGBoost that it deals with the sparsity of the input data which has been mentioned in[9]. To handle the sparsity of the data XGBoost proposed a sparsity aware algorithm which will allow the model to identify the sparsity of any node and it will learn a default direction to make decision. In other words, every node will have a default direction which will be only applied if it finds sparsity, otherwise it will calculate the optimal path and follow the optimal path. For this algorithm XGBoost can handle all kind of sparsity by using a unified process. In the next section we will discuss about the algorithm which XGBoost applies to identify sparsity and to deal with the sparsity.

5. **Tree pruning:** Tree pruning is another advantage provided by XGBoost which allows a model to split up to a predefined level of depth in tree branch. Generally, XGBoost follows a greedy algorithm while splitting like GBM but unlike GBM it doesn't encounter a branch if it finds a negative loss. XGBoost splits up to the max_depth(a XGBoost parameter) and then starts pruning the tree backwards and removes splits beyond which there is no positive gain[32]. As for example if a split contains negative loss of -5 may followed by a split of positive loss of +7 then GBM will encounter the split. Whereas XGBoost will go deeper and it will evaluate the combined effect of +2 of the split and keep both of those.

6. **Built-in Cross-validation:** XGBoost supports cross validation which implies cross validation at each iteration in the boosting process.

3.6.2 Background of XGBoost

To help understand the implementation and inner details of XGBoost we must know the background from where the idea of XGBoost came from and the building block of the XGBoost. XGBoost is a tree based boosting algorithm which has been developed after gradient boosting and it has established itself over GBM over every aspect of machine learning problem solving. In order to understand XGBoost and its mechanism we must discuss two things, which are Boosting and Gradient Boosting.

1. **Boosting:** While solving machine learning we may face two types of problems which actually we do not face because of the new machine learning algorithms.

But in the past two types of problems faced where one is bias-complexity trade off and another is computational complexity. The bias-complexity trade off came from the idea of ERM learner and it's decomposed error. Let S a sample which has been built by an unknown distribution function D and is labeled by a target function f then a learning algorithm will give an output $h : X \to Y$ which is a predictor. The goal of the algorithm is to minimize the error for specific sample which will have minimum error $L(h)$ is known as ERM or Empirical Risk Minimization[6].

$$L_s(h) = \frac{|i \in [m] : h(x_i) \neq y_i|}{m} \tag{3.7}$$

The ERM have two different errors where one is approximation error and another is estimation error and these two errors are disproportionately dependent on each other. As a result, more expressive hypothesis class implies less approximation error but it also increases the estimation error of the ERM. Moreover, computational complexity is another issue of learning algorithms which means infeasible performance of a learning algorithm. Boosting has offered ways to deal with these complexities by turning a weak learner in to a strong learner.

2. **Gradient Boosting Machine:** Gradient boosting algorithm is a tree boosting algorithm which implies a sequential tree based model. Unlike Adaboost, GBM fits new predictor to the residual errors from the previous predictors which is a special technique for classification problems. It is ensemble approach which minimizes the loss function at each level which increases the learnability of a weak learner.

3.6.3 Data preprocessing for XGBoost model

Data preprocessing has been described briefly in the data preprocessing part of CNN model. Like CNN preprocessing we have used nltk stopwords to remove unnecessary frequent words, removed punctuation and used padding to have same length of words and sentences. In addition to that lemmatization has been while preprocessing data for XGBoost model, which is the only difference in data preprocessing from the preprocessing of CNN model.

1. **Lemmatization:** Lemmatization is a process which shortens a group of similar words into the root of the words. In other words lemmatization shortens the suffix prefix of a word and convert a word into its root form which minimizes the word space on the corpus[21]. For example, in a corpus there may exist words like able, disable, enable, ability, disability and so on. Lemmatization will shorten the suffix and prefix parts of the words disable, enable, ability and disability all the word will be referring to the root word 'able'.

3.6.4 Feature extraction for XGBoost

Though fasttext have been used as a feature extractor in the CNN model, but in XBGoost it is tested that a combined use of TF-IDF and BOW provides greater result on feature extraction. In this feature extraction approach at first a bag words from the whole dataset will be created by applying tokenization[16]. In the pre-processing part it has been mentioned that the punctuation and some of the stop words will be removed which means these characters will not be a part of a bag of words. After removal of unwanted characters whole document treated as a list of words and depending on the appearance of each word in a sentence every sentence may be represented as a vector. Depending on the output from BOW TF-IDF can be calculated where TF(Term Frequency) represents the frequency of a word in a document (sentence) and IDF(Inverse Document Frequency) represents how rare a word is across the corpus.

$$TF = \frac{Number\ of\ times\ a\ word\ appears\ in\ a\ document}{Total\ number\ of\ terms\ in\ the\ document} \tag{3.8}$$

$$IDF = log_e(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ the\ word}) \tag{3.9}$$

This is called count vectorizer technique of vector representation of words which we have applied for extraction feature for our proposed XGBoost model. Bag of words can be made by ngram or for single word count vectorizer. We have implemented word count vectorizer and depending on the dataset max_features value have been initialized with 10000. Max_features value may vary depending on the number of words present in a dataset. After fitting all three datasets in our XGBoost model it has provided accuracy of 80.51%, 93.10% and 78.14% respectively for dataset 1, dataset 2 and dataset 3.

3.6.5 XGBoost Hyperparameter Tuning (Our Proposed Model)

In machine learning models hyperparameters are the main actor that manipulate the learning process of the model and influences the model's performance. Generally, a machine learning model fitted with an amount of data and from the data it learns the parameters in the process of model training. On the other hand, hyperparameters are some predefined parameters that implies higher level properties of the model and these parameters usually define before the learning process starts. These hyperparameters entails learning capability of the model, complexity, learning rate and other higher level properties by setting different values. XGBoost learning algorithm have some hyperparameters which has been mentioned in the paper of XGBoost[9]. There are three types of hyperparameters that can manipulate the performance of XGBoost and these are given below:

General Parameters:

These parameters impacts the overall functionality of XGBoost.

1. **Booster:** Booster is the parameter which defines the type of the model at each iteration of the learning process. There are two options available to define the type of the model where one is 'Gbtree' and another is 'Gbliniear'. 'Gbtree' selects the model to be a tree based model and 'Gbliniear' selects the model to be linear[11]. In our proposed learning model of hatred detection we wished to select tree model of XGBoost and it is a default model of XGBoost that is why we didn't need to change the parameter.

2. **Silent:** Silent is function when it is activated no running message will be seen from the process which might not help the user to see what is going on the model. That is why by default the silent mode remain deactivated, when it is set to 0 it activates and stop printing any messages.

3. **Nthread:** In model advantage section it is mentioned that one of the greatest features that have made XGBoost, the fastest machine learning algorithm is the parallel processing of the model. XGBoost supports use of parallel processors and multiple cores while executing the model which increases the performance of the model. Nthread also allows to use selective numbers of cores for executing the model. By default it uses all the cores of a processor but if a user wants to use selective numbers of cores that he/she can change the value of Nthread. While running our model we have used a 12 thread process and we have used 10 threads for running our model.

4. **Verbosity:** Verbosity is proposed in place of silent when silent got deprecated. The value of verbosity remain 1 by default which prints warning messages if needed. The value of verbosity can be 0, 1, 2 or 3 uses for silent, warning, info and debug respectively. We have used the value 3 for verbosity to activate the debugging.

Booster Parameters:

As it is mentioned earlier that we may use linear model or tree model on XGBoost and depending on the model we will use Booster parameters to manipulate the model. As we are using 'Gbtree' here we will be discussing about the booster parameters of 'Gbtree'. Another reason of using and discussing only 'Gbtree' is it has outperformed linear model in every aspect.

1. **Eta or learning rate:** Eta or learning rate defines the change of step size over every iteration. In other words, learning rate is the shrinkage at every step set by the user which will define the step size for every step until the learning machine reaches the landing position. Increasing the learning rate will be computionaly good because it will reach the goal faster but at the same time because of increased step size the model will not reach the best optimum. It has been seen, to get the best result in XGBoost the learning rate must be as low as possible. We have used the default learning rate which 0.3 though the range for learning rate can be from 0 to 1.

2. **Gamma:** Gamma is pseudo regularization hyperparameter which controls the complexity of the decision tree of a model. It is also known as Lagrangian multiplier which implies minimum loss reduction to make the partition of a

leaf node[8]. Gamma is set to a specified value to determine the minimum loss acceptance. If gamma is set to a value x then the model will go to the depth of max_depth and then start to prune but it will remove the splits which are less than the value of gamma. The value of gamma varies from 0 to infinity and it very much depends on the dataset and the values of other hyperparameters used in the model. We have set the value of gamma to 1.

3. **Max_depth:** By name it refers that it is a hyperparameter which defines the maximum depth of the tree model. Depth of the tree can be vary from 0 to infinity but the user must keep in mind that XGBoost consumes much memory to train the model if the depth is high.

4. **Min_child_weight:** Min_child_weigth is also used for controlling the complexity of the model which refers to the minimum acceptance weight of new node to be added on the tree. To make the learning algorithm more conservative, higher values for min_child_weight might be used but too high value will result in underfitting. In our model we have defined the value of min_child_value as 1 though the value for this parameter may vary from 0 to infinity.

5. **Colsample_bytree:** It is a subsample parameter which defines the subsample ratio of columns while constructing the tree. Subsample ratio allows a model to randomly sampled some portion of the data to be selected from a training set which prevents overfitting. The ratio we have selected for our model is 0.9 which will randomly sampled 0.9 portion of the data from the training set.

6. **Alpha:** Alpha is a traditional regularization parameter which provides lasso regularization to control the complexity of the model and prevent overfitting. Higher value of alpha make the model more conservative that is why in our model we have initialized the alpha value to 0.1.

$$minimize(SSE + \lambda \sum_{j=1}^{p} |\beta_j|) \tag{3.10}$$

The above equation is on lasso regression where SSE means sum of squared errors.

Learning Task Parameters: Learning task parameters defines the learning objective of each steps.

1. **Objective:**Objective is a learning task hyperparameter that defines the loss function of the model to be minimized. There are many types of objective values that can be used such as, multi:softmax, reg:logistic, binary:logistic, binary:logitraw, multi:softprob and so many. In our proposed method we have used multi:softmax for 1st and 3rd dataset and for our 2nd dataset we have used binary:logitraw. As our 1st and 3rd dataset consist multiclass labels we have used multi:softmax for multiple classification. On the other hand, for the 2nd dataset linear:logitraw has been applied as the dataset contains only two classes of labels. While using multi:softmax, another parameter num_class must be defined to set the number of labels for classification and in our case num_class value was 3.

2. **Eval_metric:** The hyperparameter eval_metric used for validation data. Depending on the objective parameter XGBoost by default sets the eval_metric parameter. Also a user can define multiple eval_metric. For our model we have selected 'merror' as eval_metric for 1st and 3nd dataset as both dataset are for multi classification. For the 2nd dataset we have applied 'error' as eval_metric, though 'merror' and 'error follows the same mathematical formula but 'merror' is for multi classification and 'error' binary classification.

$$merror = \frac{Number\ of\ wrong\ cases}{Number\ of\ all\ cases} \tag{3.11}$$

3. **Base_score:** It is called global bias which defines initial prediction score of all instances. In our model we have initialised the base_score 0.1.

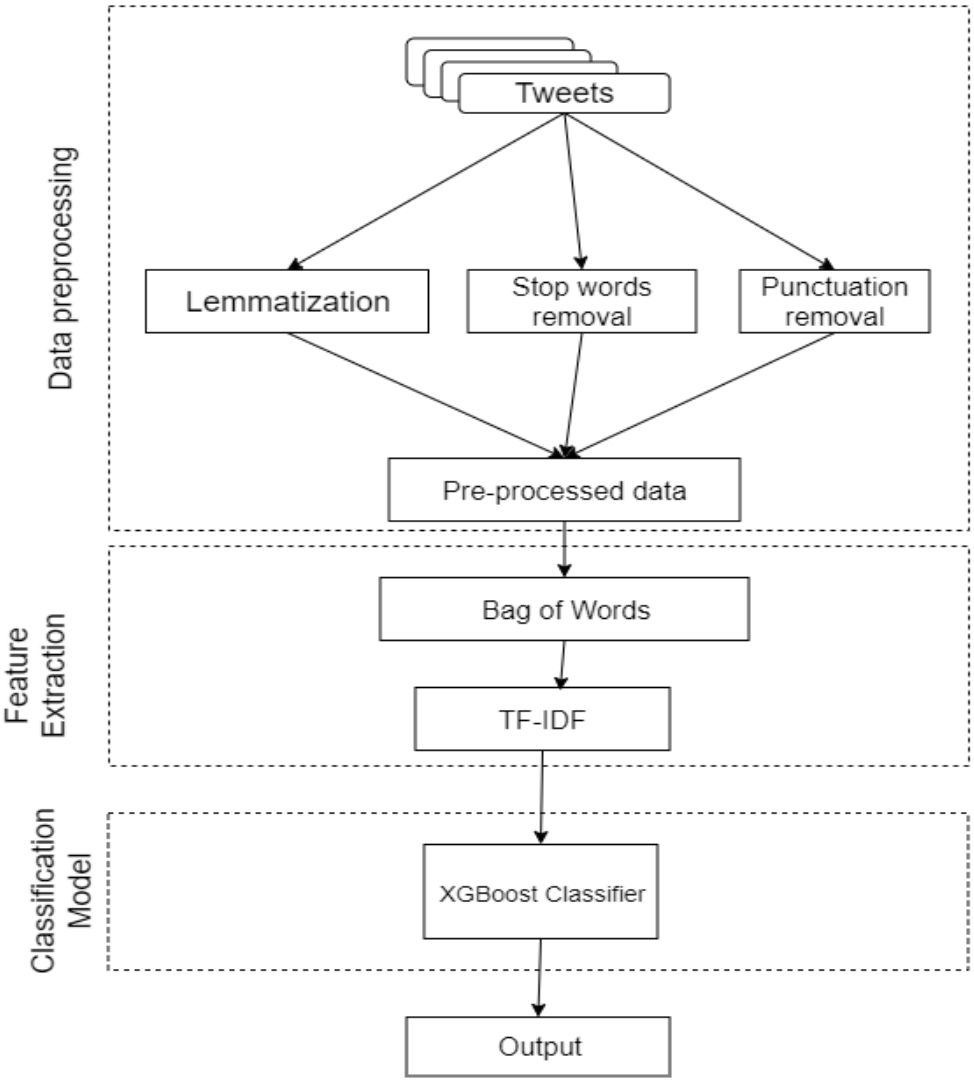

Figure 3.7: Our Proposed XGBoost Model

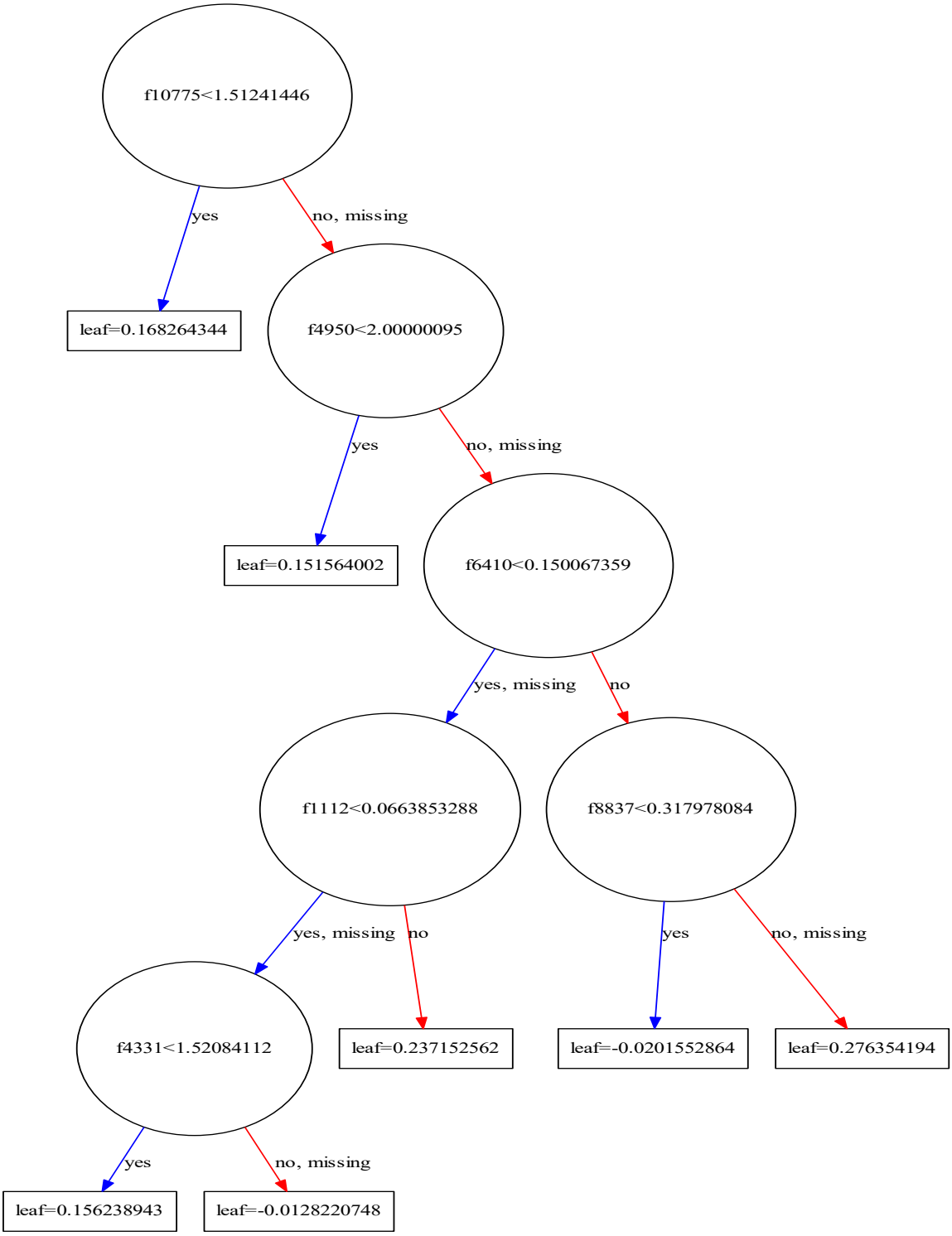

Figure 3.8: Decision tree in the model(Best Iteration)

Figure 3.7 shows the features and feature values for each split along with the leaf nodes.The variables are automatically named corresponding with the feature indices in the input tree.

Chapter 4

Result and Analysis

In this chapter, the result of our proposed models and analysis will be discussed. Here the result portion will contain the overall performance of the proposed model with some performance parameter. Analysis part will have comparison of our results, evaluation of our results and elaborate discussion on our overall performance and model.

4.1 Result

In any research or project, result is the most vital portion as it shows the outcome or findings of any research or model. In this portion, we will evaluate the results through some metrics of performance and show them using graphical representation. The major metrics that have been used here are:

- Precision
- Recall
- F-1 score
- Accuracy

4.1.1 Confusion Matrix

In classification problem, confusion matrix indicates performance measurement. It shows the confused condition of a model during prediction. The output can be two or more than two depending on the problem statement[25].

	Predicted Values		
Actual Values		Positive(1)	Negative(0)
	Positive(1)	TP	FN
	Negative(0)	FP	TN

Table 4.1: Confusion Matrix

In confusion matrix, some general terms like True positive (TP), False positive (FP), True negative (TN) and False negative (FN) are used. From these values we can calculate precision, recall and accuracy.As confusion matrix shows the overall prediction of a machine learning model, this is a very important parameter to measure performance of the model as accuracy alone is not the best measurement of performance for a model. Here we have showed six confusion matrix for three datasets using the two proposed models.

Hate-speech-offensive-language dataset by Davison :

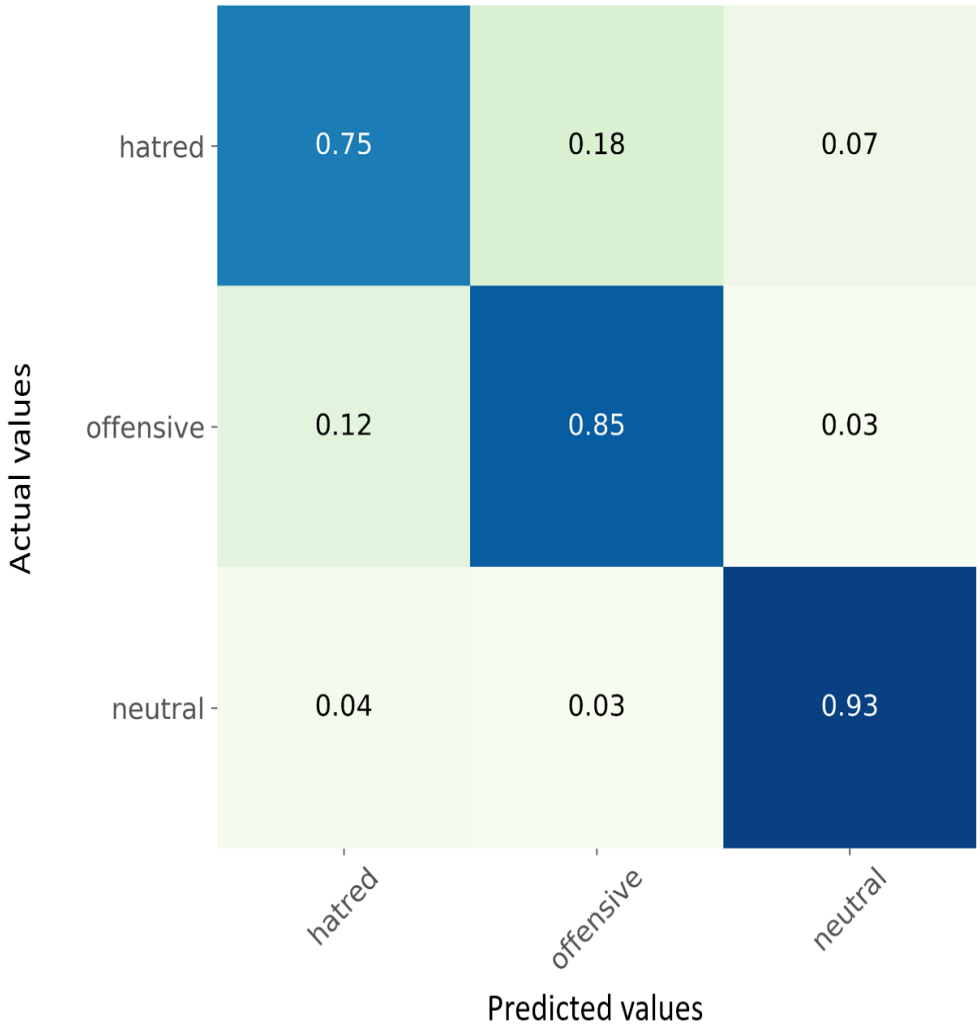

Figure 4.1: Confusion matrix on dataset I for CNN model

Figure 4.1 shows the confusion matrix for our dataset 1 where we have hatred, offensive and neutral data. So we have used multiclass confusion matrix.

Here, from the figure we can see the actual vs predicted values of the model. X axis and Y axis of this matrix shows the values of predicted label and actual label respectively. The proposed model predicted 75% as hatred, 18% of offensive and 3% of neutral where the actual value was hatred. The second row of the matrix describes the predicted values for the offensive labeled data. 12%, 85% and 3% values as hatred, offensive and neutral indicate the prediction of the model for

offensive labeled data. Similarly, the third row shows the prediction for neutral data. In this case, the model predicted 4%, 3% and 93% as hatred, offensive and neutral for neutral labeled data.

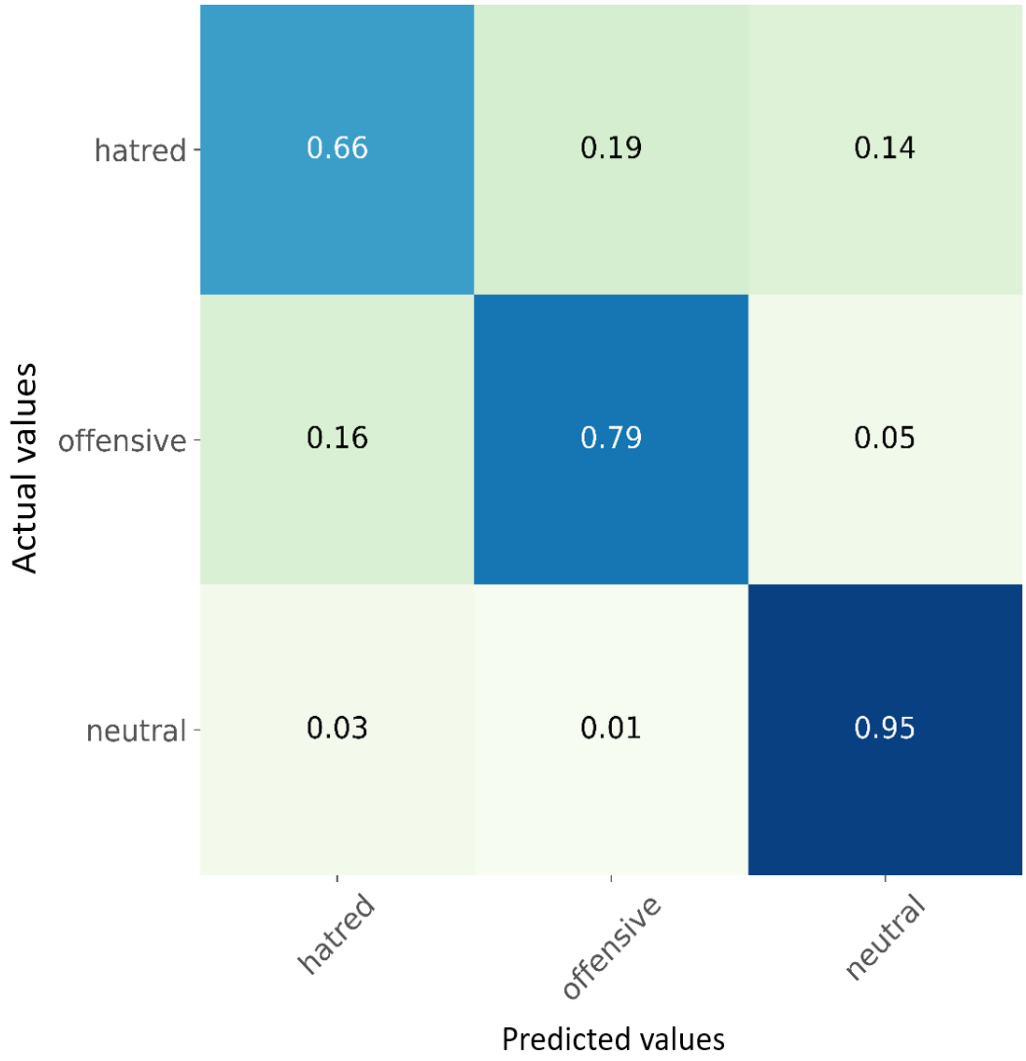

Figure 4.2: Confusion matrix for dataset I for XGBoost model

Figure 4.2 confusion matrix for our dataset 1 has been shown. This matrix shows values for the second model using XGBoost. As the dataset contains three labels of data which are hatred, offensive and neutral we have used multiclass confusion matrix.

From the figure, the matrix shows actual vs predicted values for XGBoost model. Like the previous matrix for CNN model, here X axis and Y axis of this matrix shows respectively the values of predicted label and actual label. XGBoost model predicted 66% as hatred, 19% of offensive and 14% of neutral where the original value was hatred. Similarly, the second row of the matrix describes the predicted values for the offensive labeled data. 16%, 79% and 5% values as hatred, offensive and neutral indicate the prediction of the proposed model for offensive labeled data. Moreover third row shows the prediction for neutral data. In this case, the model predicted 3%, 1% and 95% as hatred, offensive and neutral for neutral labeled data.

From these values we can see that the model using XGBoost can give a 66%, 79% and 95% prediction for hate, offensive and neutral respectively.

Toxic-comment-classification-challenge dataset of kaggle :

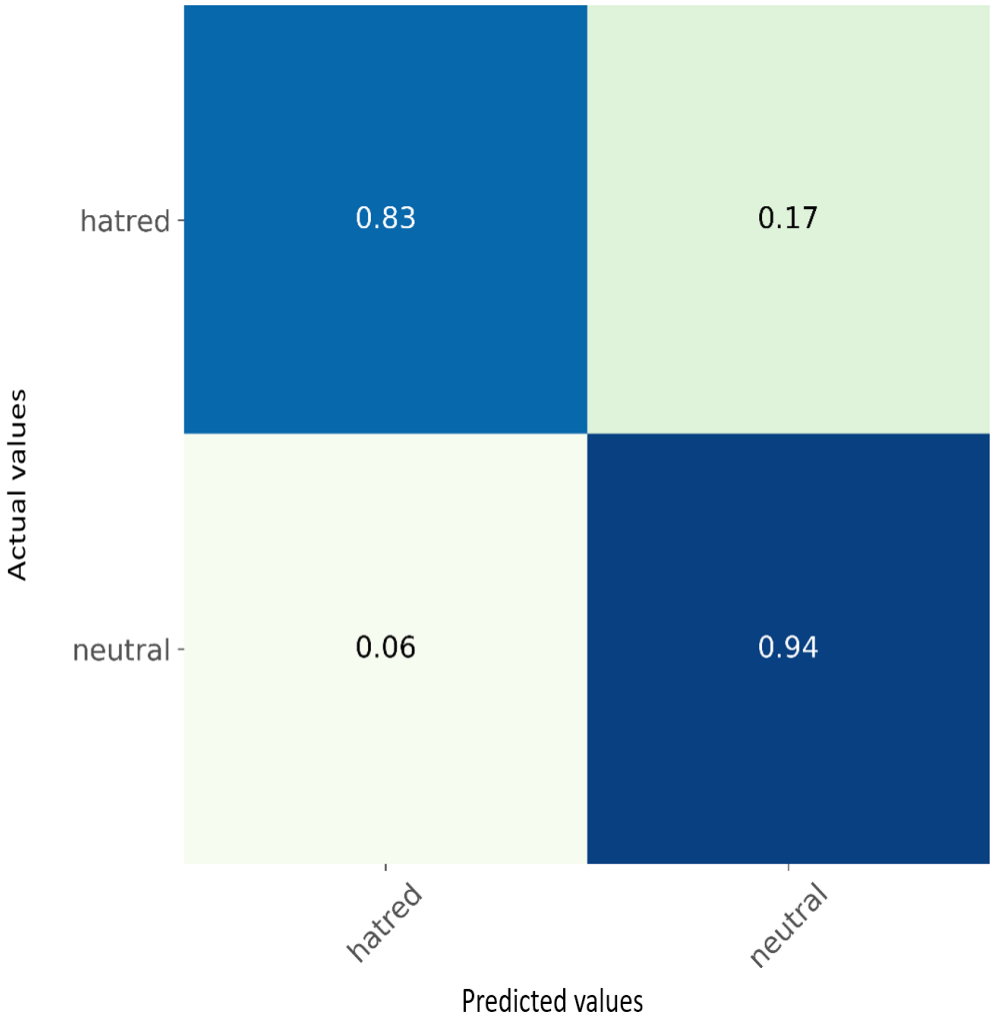

Figure 4.3: Confusion matrix on dataset II for CNN model

Figure 4.3 shows the confusion matrix for the model using CNN where we have used the second dataset. This dataset contains only hate and neutral labelled data. In the matrix, we have actual values on X axis and predicted on the Y axis [5]. The values from the matrix represents as below-
TP(true positive) : Here we get 0.83 or 83% as our true positive which indicates at 83% cases the classifier predicted as 'hatred' and actually the text was 'hatred'.
TN(true negative) : true negative means the classifier detects the tweets as negative where the actual label of the tweet is negative. The matrix shows 0.94 as true negative which means at 94% cases the classifier predicted as 'neutral' or 'non hate' for the real 'neutral' data.
FP(false positive) : shows 0.06 as FP or in 6% cases the model marked the text as

'hatred' but the text were actually 'neutral'.
FN(false negative) : On 17% cases, for neutral data the proposed model predicted them as hatred.
So the proposed CNN model can detect hatred 83% correctly from the text and for the neutral one it is 94%.

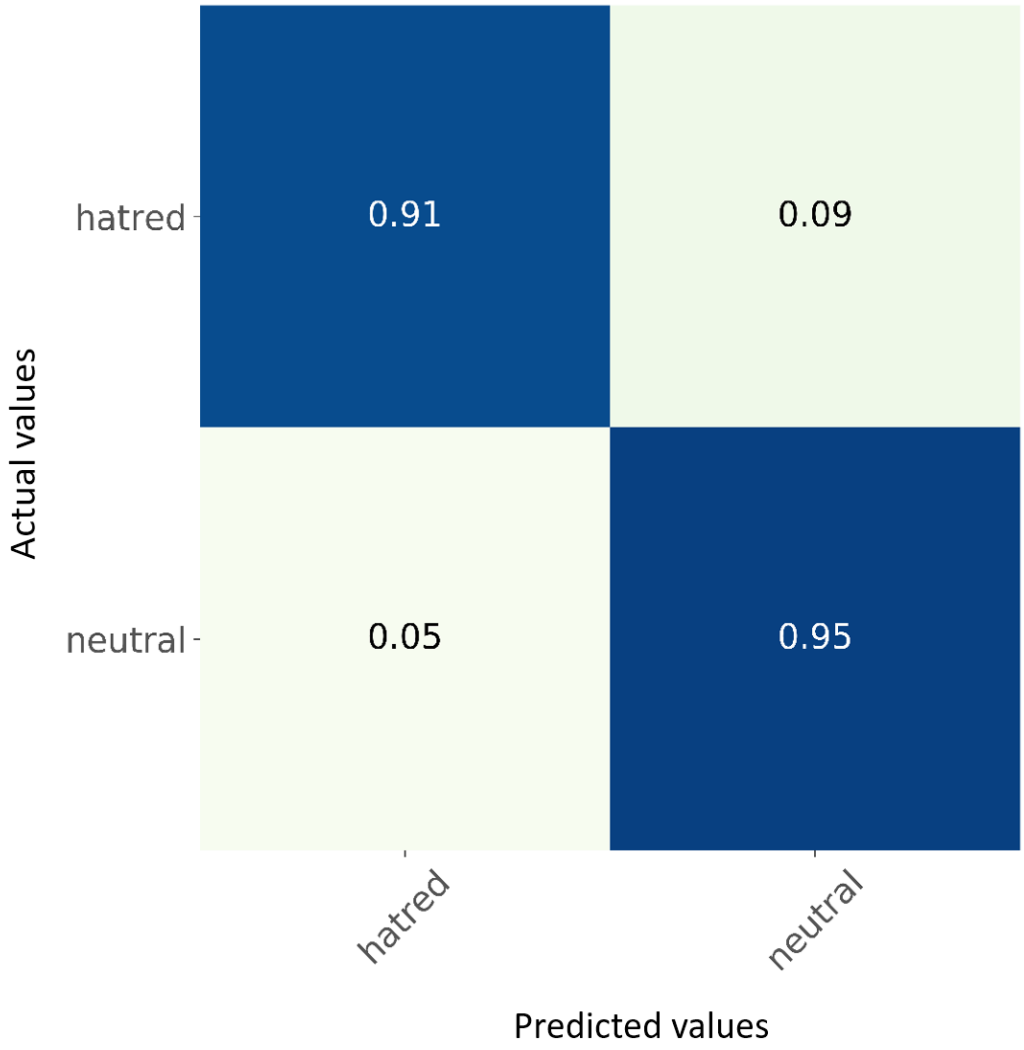

Figure 4.4: Confusion matrix for dataset II for XGBoost model

Figure 4.4 represents the confusion matrix for the second proposed model using XGBoost. We have used the second dataset where there are two label, one as hate and other as neutral. The values from the matrix indicate the overall state of confusion of the proposed model. The representation of the values are given below.
TP(true positive) : Here we get 0.91 or 91% as TP or true positive. This value indicates that 91% of cases the classifier predicted as 'hatred' and actually the text was 'hatred'.
TN(true negative) : The proposed model gave 0.95 as TN or true negative which means our model predicted as 'neutral' for the data which are actually 'neutral'.

FP(false positive) : Indicates the classifier marked the text as 'hatred' where the text was actually 'neutral' on 5% cases.
FN (false negative): Shows 9% cases the classifier predicted a 'hatred' text as 'neutral'.

HNO-Tweet Dataset (Own Dataset) :

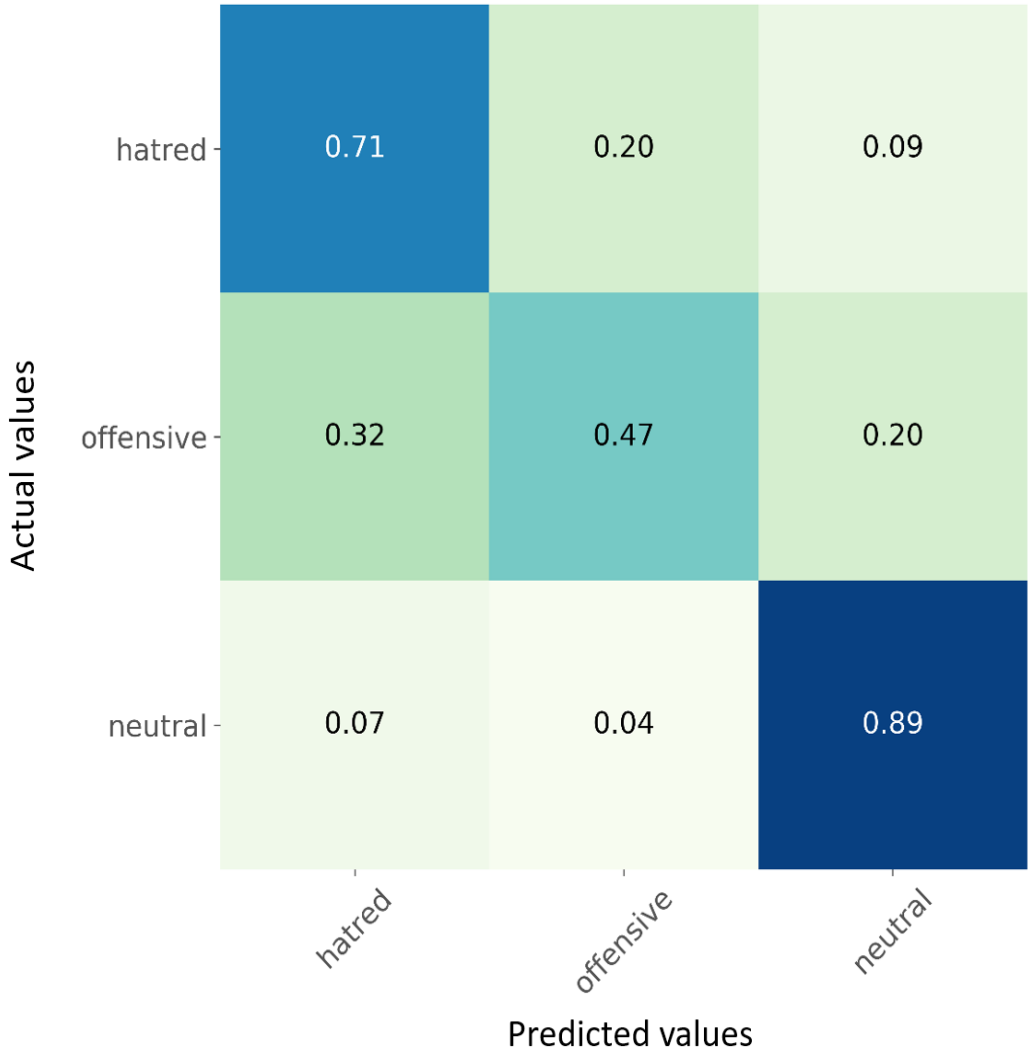

Figure 4.5: Confusion matrix for dataset III for CNN model

Figure 4.5 shows the confusion matrix for dataset 3. Here we have used the first proposed model using CNN. This dataset have three classes of data as hatred, offensive and neutral. The values in the matrix shows the overall prediction of the model indicating the performance measurement. In the following paragraph the matrix will be discussed elaborately.
The confusion matrix for dataset 3 have actual value in X axis and predicted value

in Y axis. Here the first row shows how many times the classifier predicted as hatred, offensive and neutral for the real hatred data. The values 71%, 20% and 9% indicates the cases where the actual data were hatred but the predicted values were respectively hatred, offensive and neutral. Similarly,the second row describes that the model predicted the tweets as hatred, offensive and neutral respectively 32%, 47% and 20% cases where the tweet was actually labelled as offensive. For the third row, it shows the prediction for neutral labelled tweets by the classifier and in 7%, 4% and 89% cases the data were marked as hatred, offensive and neutral data. So using dataset 3, the proposed model with CNN could find hatred, offensive and neutral data accurately in 71%, 47% and 89% cases.

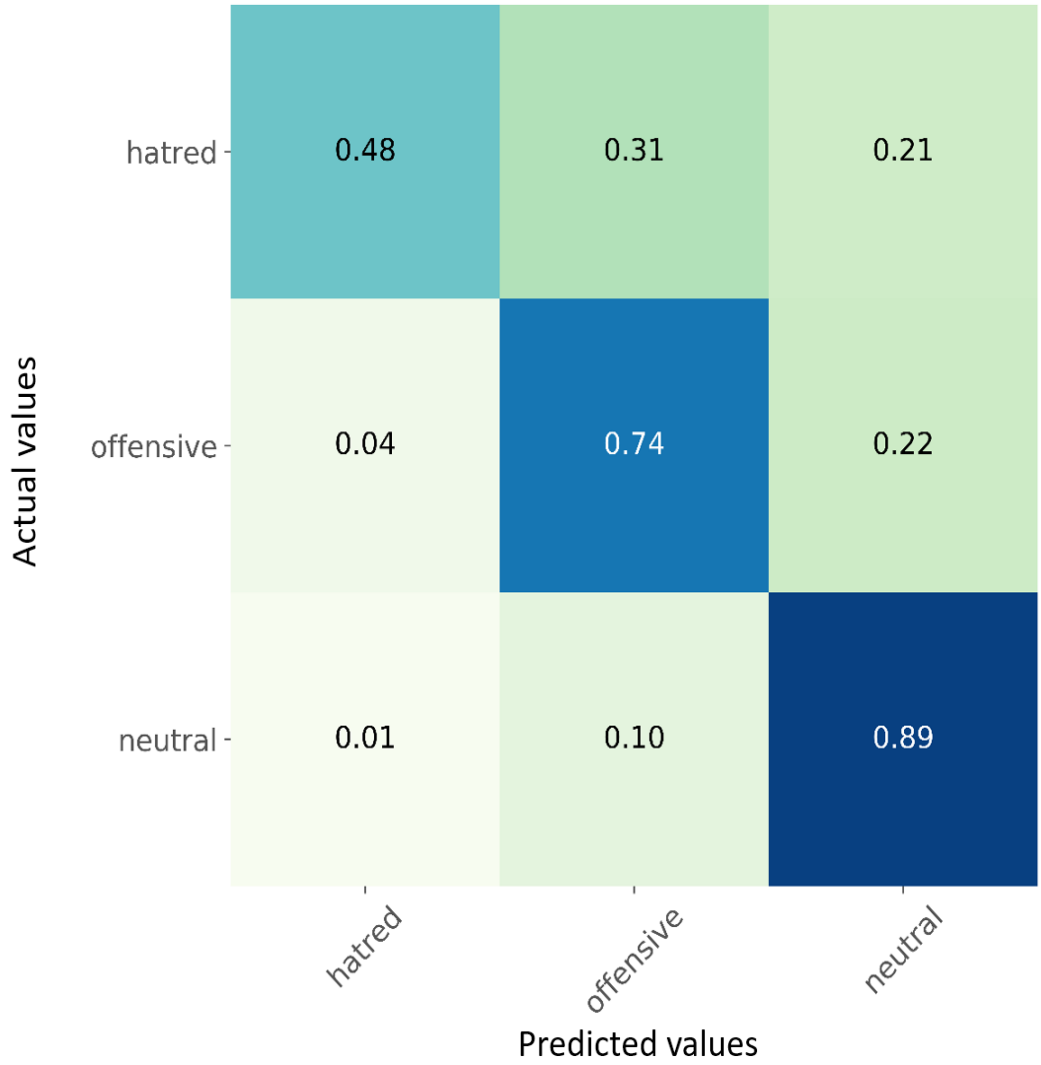

Figure 4.6: Confusion matrix on dataset III for XGBoost model

figure 4.6 represents the confusion matrix on XGBoost model for dataset 3. As dataset 3 have three classes which are hatred, offensive and neutral, a multiclass(ternary

classification) confusion matrix has been used to describe the overall prediction performance of the classifier. This will be discussed below.The figure depicts the actual value vs predicted value in a matrix form. Here, through X axis we have the actual value and Y axis represents the predicted values. Considering row wise, the first row shows that when the actual value of data is hatred, the classifier predicted them 48%, 31% and 21% of times as hatred, offensive and neutral respectively. Then the second row indicates when the actual value is offensive,the classifier marked the data as hatred, offensive and neutral in 4%, 74% and 22% cases. Similarly, the third row, which contains actual neutral labelled data,the model predicted 1%, 10% and 89% cases as hatred, offensive and neutral.

4.1.2 Classification Report

Classification report shows the quality of predictions through some calculations. More precisely, it calculates the true positive, true negative, false positive and false negative values from the confusion matrix to make classification report. In this classification report Precision, recall and f-1 score are the main metrics. Precision shows the percentage of being right in prediction. Where recall shows the total relevant results that classified correctly by the model or algorithm [41].

Precision

Precision is a popular parameter to measure the performance of any model. It shows that how much the result of a model is relevant. We can get precision by following simple equation which uses the data from confusion matrix. The equation looks like below:

$$Precison = \frac{TP}{TP + FP} \tag{4.1}$$

Using CNN and XGBoost we got precision for each class of the three dataset. They will be discussed below their respective figures.

Recall

Recall shows how much relevant results are correctly classified by a model.Recall metric calculates how many of the Actual Positives are captured through labeling it as Positive (True Positive) by the model. Like accuracy and precision, recall is also assumed to be an important parameter to analysis the performance of any model. This value come from confusion matrix and the equation looks like below.

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

We have obtained recall values for each class of the dataset on the proposed models using CNN (convolutional neural network) and XGBoost. They will be discussed individually below their respective figure.

F1-Score

F1 score is the function of precision and recall which measures the weighted average of precision and recall. F1 score calculates precision and recall at the same time by using harmonic mean instead of arithmetic mean . So this is another important index for performance evaluation [25].

$$F1 - Score = \frac{2 * recall * precision}{recall + precision} \tag{4.3}$$

We have calculated f1-score for both proposed models on three different datasets that have been mentioned earlier. The result will be elaborately described in the future section.

Hate-speech-offensive-language dataset by Davison :

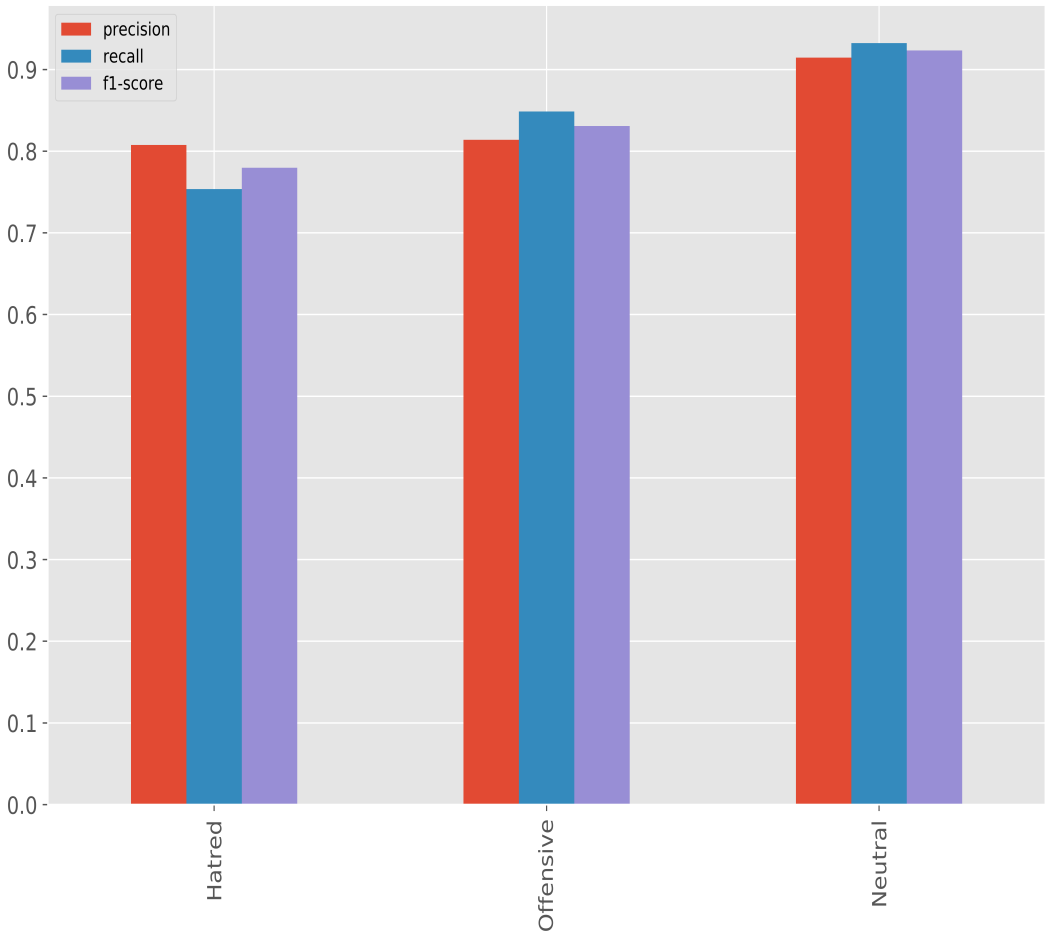

Figure 4.7: classification report with precision, recall and f1 score for dataset 1 on CNN model

Figure 4.7 shows the classification report with precision, recall and f1 score for dataset 1 using CNN model. Here, for hatred precision, recall and f1 score are 0.808, 0.754 and 0.780. Then for the offensive language precision, recall and f1 score are 0.814, 0.849 and 0.831 respectively. Similarly for neutral 0.915, 0.932 and 0.923 are the precision, recall and f1 score.

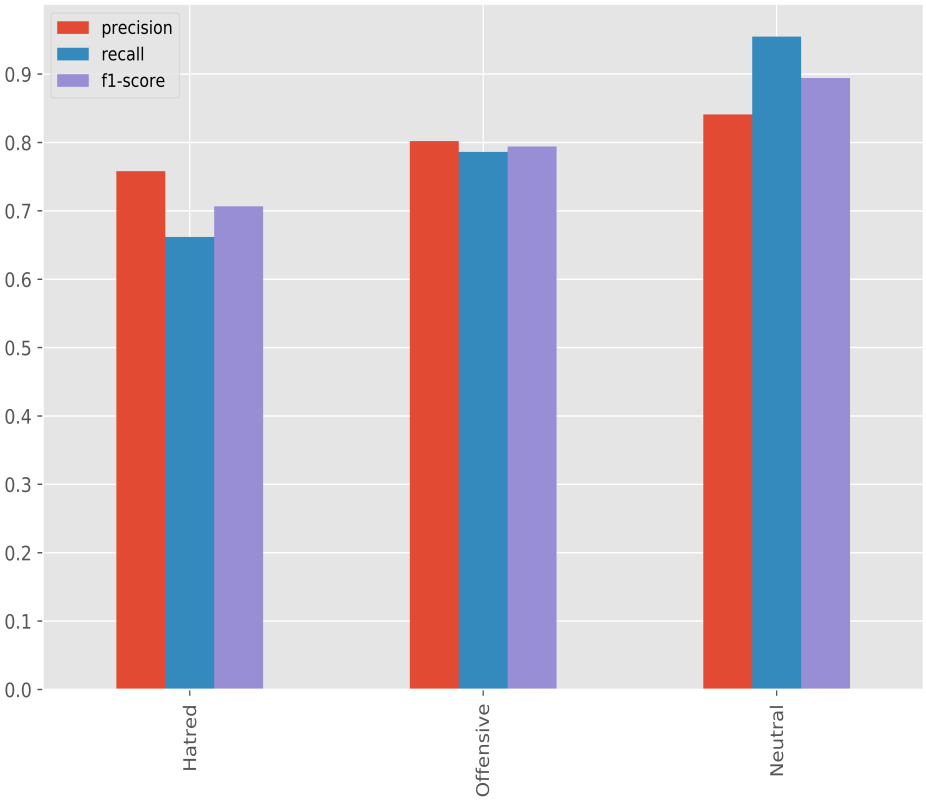

Figure 4.8: classification report with precision, recall and f1 score for dataset I on XGBoost model

Figure 4.8 represents the classification report with precision, recall and f1 score using XGBoost model on dataset 1. Here, for hatred class precision, recall and f1 score are respectively 0.758, 0.662 and 0.707. In the same way, for offensive precision, recall and f1 score are 0.802, 0.786 and 0.794. 0.841, 0.955 and 0.894 values serially indicates the precision, recall and f1 score for neutral data. High recall value has been found for the neutral data from this model which is 95.5%.

Toxic-comment-classification-challenge dataset of kaggle :

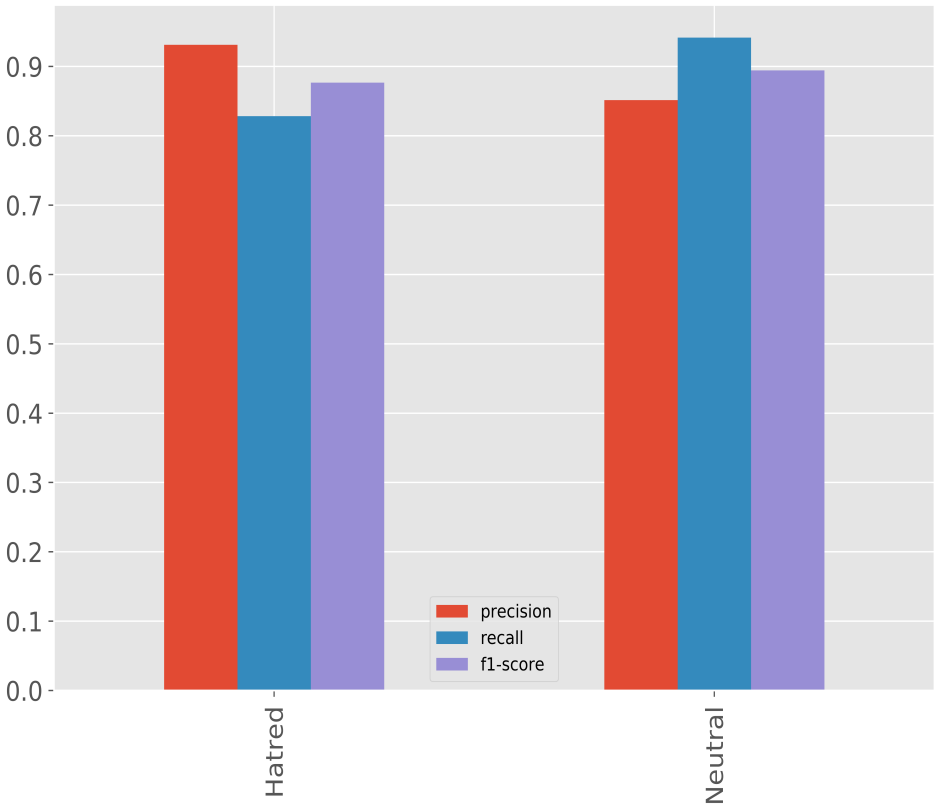

Figure 4.9: classification report with precision, recall and f1 score for dataset II on CNN model

Figure 4.9 shows the classification report with precision, recall and f1 score for dataset 2 using CNN as classification model. Here precision, recall and f1 score for hatred are 0.932, 0.828 and 0.877. Then for the neutral precision,recall and f1 score are 0.851, 0.942 and 0.894 respectively. Here the f1 score for both hatred and neutral are more than 85%.

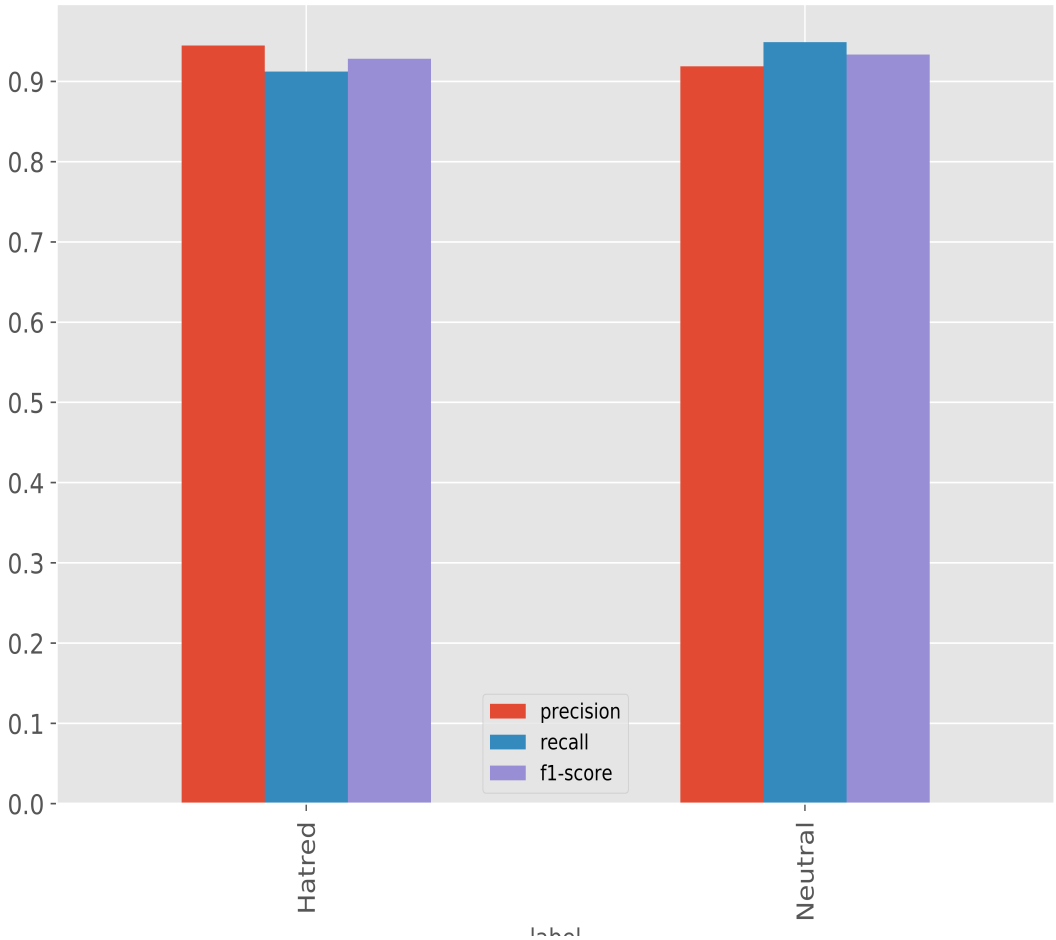

Figure 4.10: classification report with precision, recall and f1 score for dataset II on XGBoost model

Figure 4.10 describes the classification report with precision, recall and f1 score for dataset 2 using model with XGBoost. From the graph 0.945, 0.912 and 0.928 serially shows precision, recall and f1 score for hatred. Similarly, for neutral data precision, recall and f1 score are 0.919, 0.949 and 0.934. For both hatred and neutral, values are more than 90% which indicates a good result on this dataset by using XGBoost.

HNO-Tweet Dataset (Own Dataset) :

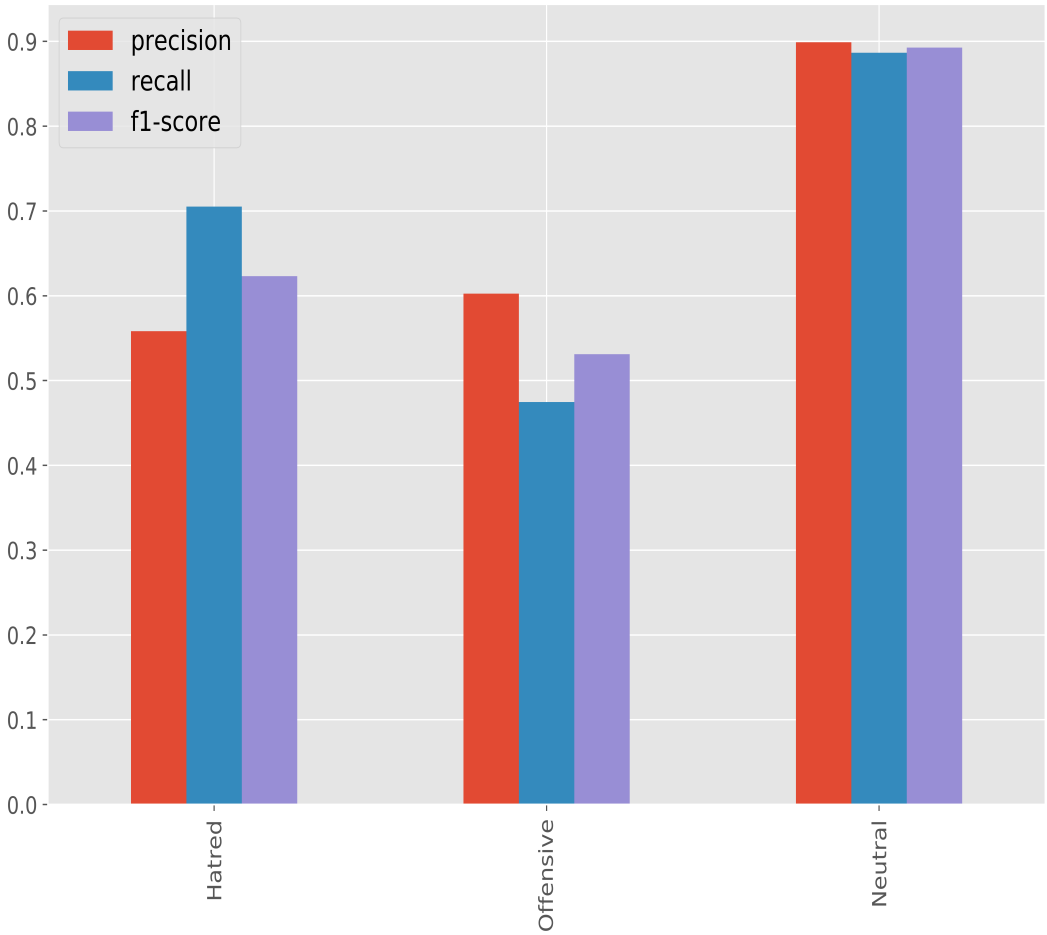

Figure 4.11: classification report with precision, recall and f1 score for dataset III on CNN model

Figure 4.11 indicates the precision, recall and f1 score through classification report for dataset 3 using model with CNN. From the graph, precision, recall and f1 score are 0.558, 0.705 and 0.623 for hatred, 0.603, 0.475 and 0.531 for offensive and at last 0.899, 0.887 and 0.893 for neutral. From the values, this model can evaluate neutral data more accurately than any other class.

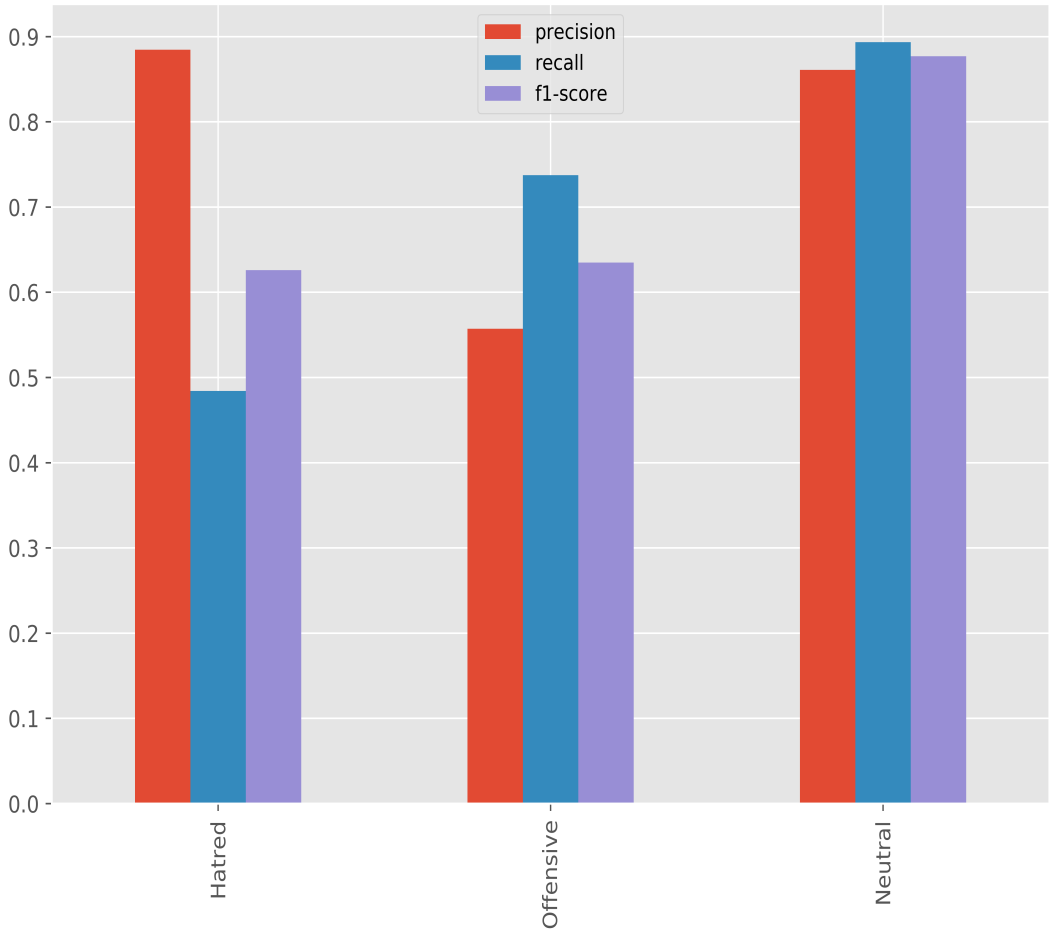

Figure 4.12: classification report with precision, recall and f1 score for dataset III on XGBoost model

Figure 4.12 describes the classification parameters like precision, recall and f1 score for dataset 3 using XGBoost model. Here 0.885, 0.484 and 0.626 values shows respectively precision, recall and f1 score for hatred data. Similarly, for offensive 0.557, 0.737 and 0.635 as precision, recall and f1 score. For neutral the values are 0.861, 0.635 and 0.877.

Accuracy

Accuracy defines the correctness of the model. It is a measure of how much a model predicted the actual thing. It is actually a ratio of rightly corrected observation to the total observation. Although accuracy is not the only parameter to evaluate any model but it is the most intuitive performance measure[19]. From confusion matrix the accuracy is calculated using the below equation:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.4}$$

Accuracy indicates how perfectly a model is working. 20% of each dataset has been used here for testing purpose and accuracy has been measured based on that testing data.

Epoch vs accuracy graph for the three dataset is given below.

Hate-speech-offensive-language dataset by Davison :

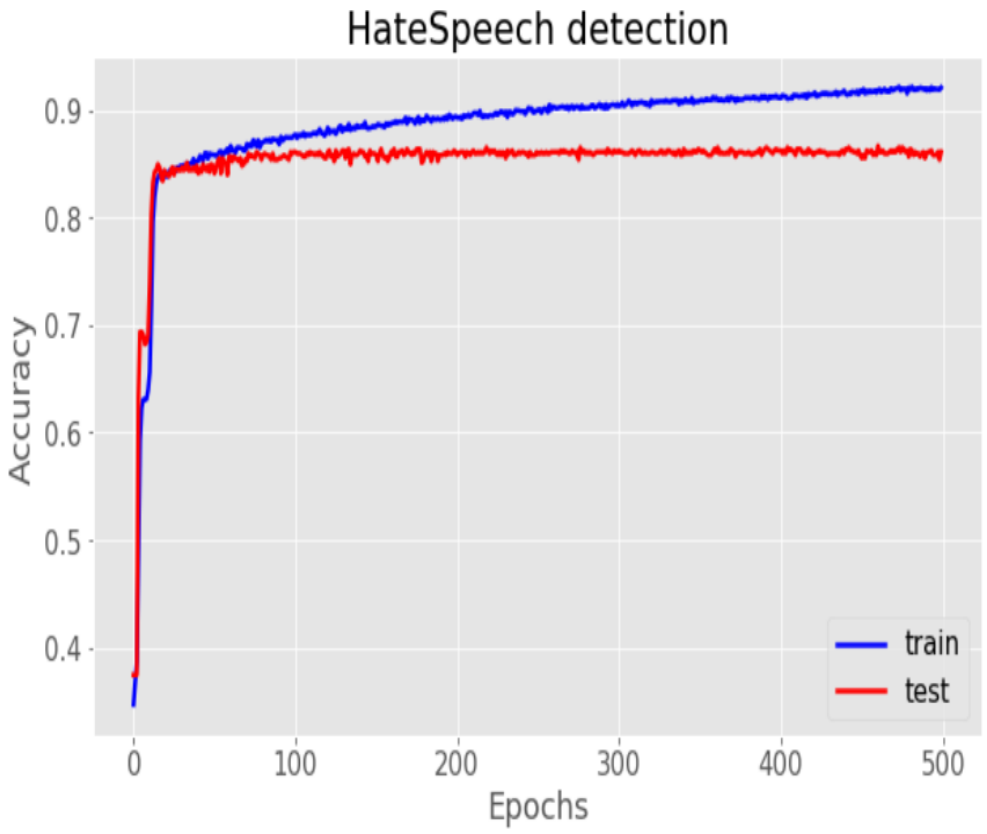

Figure 4.13: Accuracy vs epochs for dataset I on CNN model

Figure 4.13 shows the accuracy vs epochs graph for dataset 1 using CNN model. Here, 500 epochs has been used for training the model. After 500 epochs, 0.9152 of training accuracy and 0.8474 of testing accuracy has been found. However, it shows that after almost 200 epochs both the training and testing accuracy plot become straight.

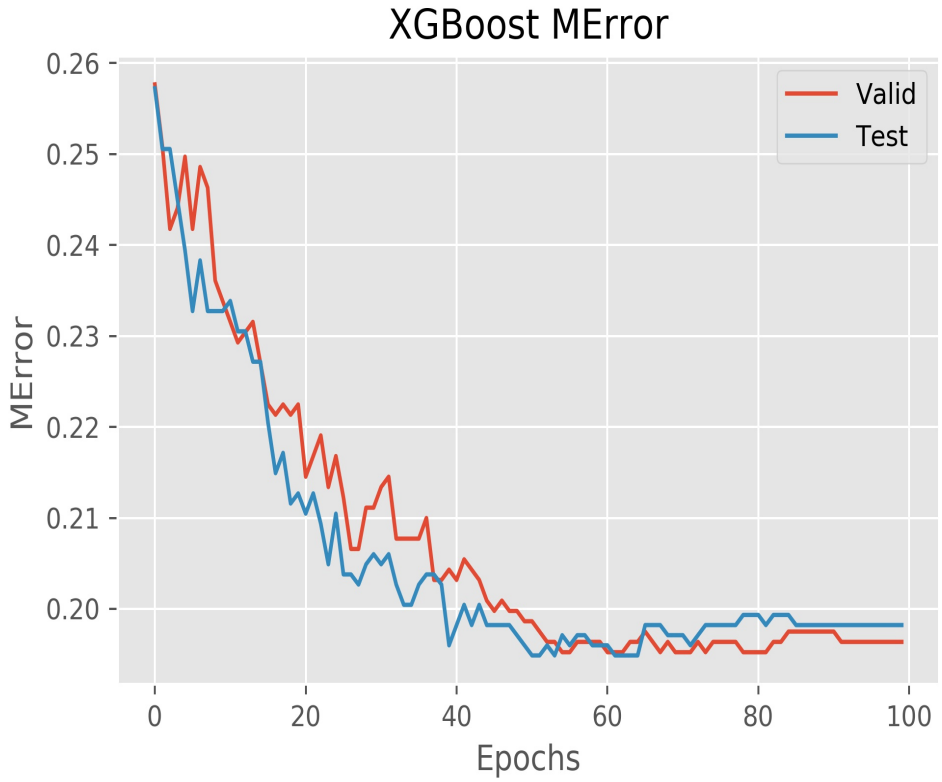

Figure 4.14: MError vs Epochs for dataset I on XGBoost Model

Figure 4.14 shows merror vs epochs graph for dataset 1 on XGBoost model . Here merror used to determine multi classification error rate, as the dataset contains labelling as hate, offensive and neutral. After 90 epochs the graph indicates a comparatively straight line for both validation and test data.

Toxic-comment-classification-challenge dataset of kaggle :

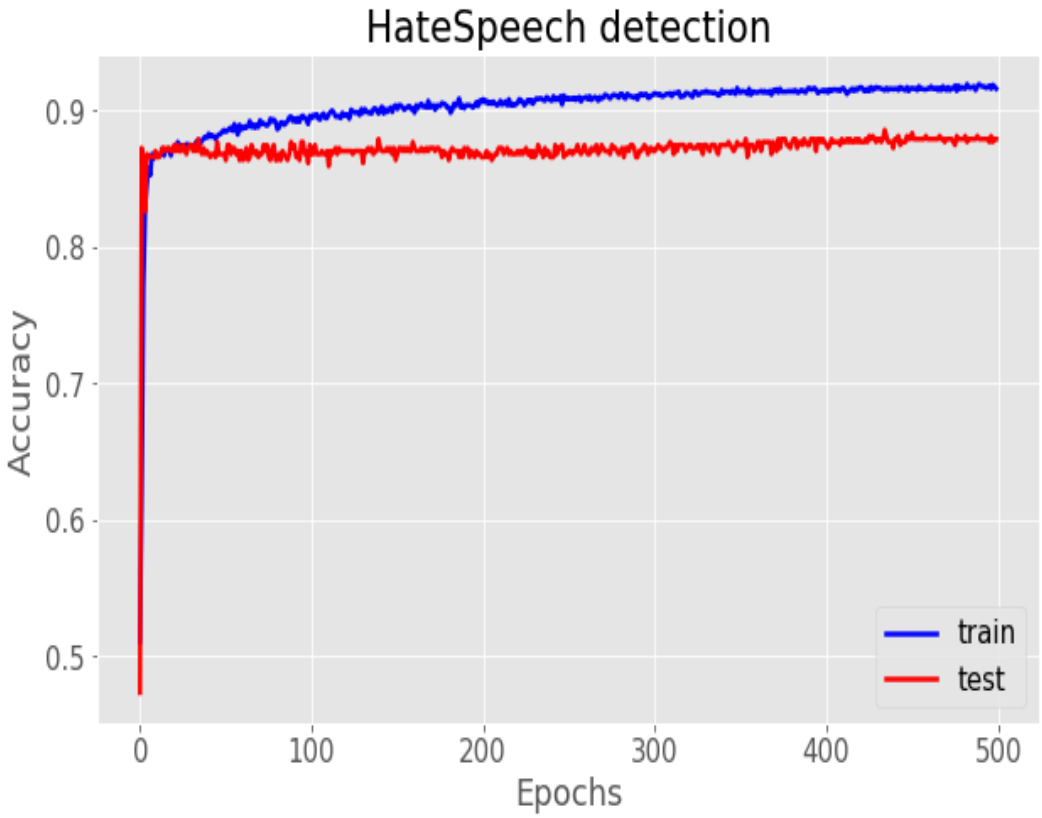

Figure 4.15: Accuracy vs epochs for dataset II on CNN model

Figure 4.15 shows the accuracy vs epochs graph for dataset 2 using the proposed CNN model. It shows the training and testing accuracy after 500 epochs or iterations. Here the result is 0.9168 for training accuracy and 0.8918 for testing accuracy after 500 epoch. Though 500 epochs have been used here but after 300 epochs the curve shows comparatively straight line for both train and test.

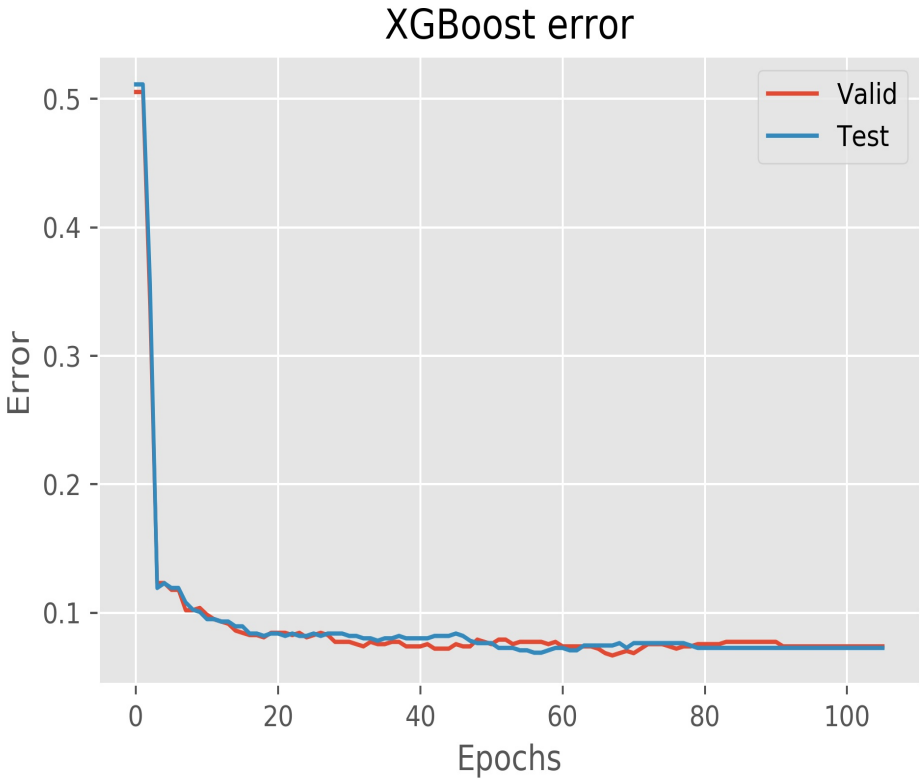

Figure 4.16: Error vs Epochs for dataset II on XGBoost Model

In figure 4.16 we can see the error vs epochs graph for dataset 2 on XGBoost model. We have used error as our parameter as the dataset have hate and neutral labelled data(binary classification). Early stopping has been used here so the model took 106 epochs to observe the comparative stable result and after 80 epochs both curves of valid and test gave stable result. The validation error is 0.077 or 7.7%. So the valid accuracy for the model is 92.25% and the test accuracy is 93.10%.

HNO-Tweet Dataset (Own Dataset) :

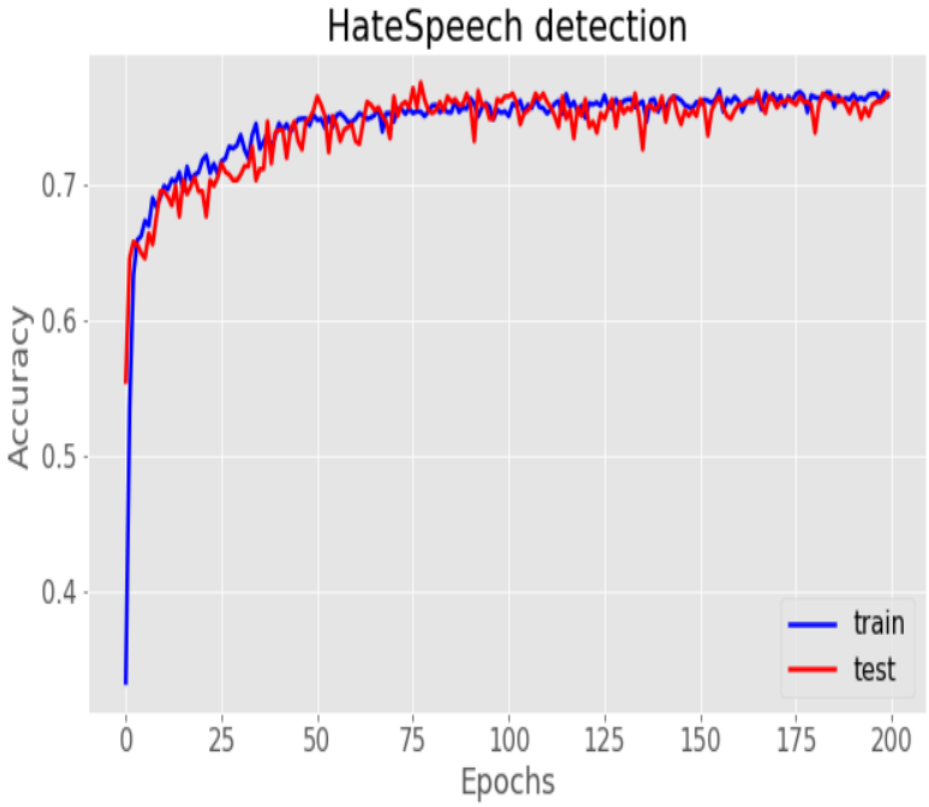

Figure 4.17: Accuracy vs Epochs for dataset III on CNN model

Figure 4.17 shows the accuracy vs epochs graph for the CNN model using dataset 3. It shows after 200 epochs,the obtained accuracy for training and testing is 0.7742 and 0.7670 respectively. The plotted graph shows some unusual behaviour comparatively to the other two previously used datasets.The reason perhaps can be due to the fact that this dataset contain some new corpus or words that are not present on our pre-trained word embedding.

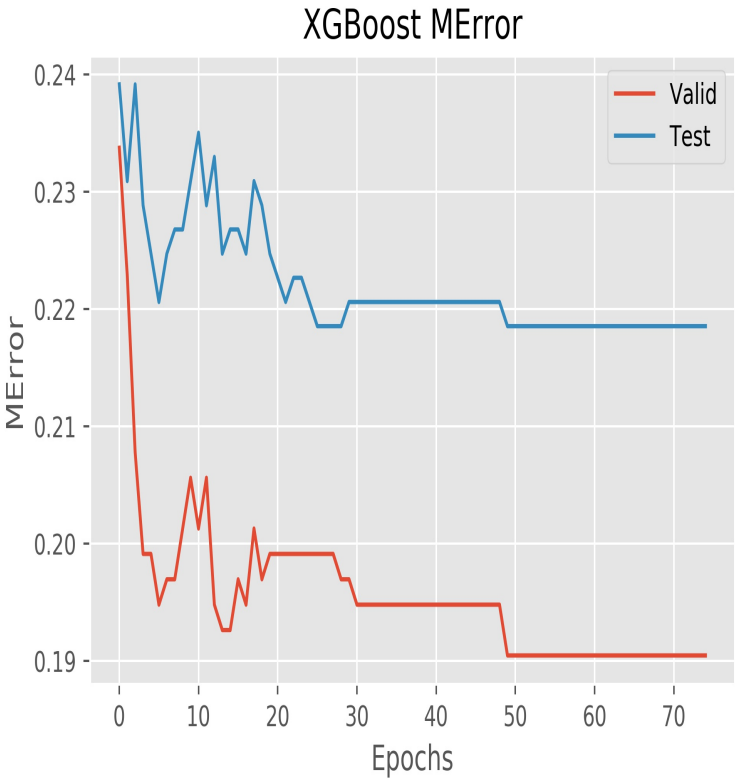

Figure 4.18: MError vs Epochs for dataset III on XGBoost Model

In figure 4.18 we can see the merror vs epochs graph for our dataset 3 on XGBoost model. We have used merror parameter to determine multi classification error rate. For both curves after 50 epochs, it shows comparatively straight line for both valid and test data .The accuracy that has been achieved is 80% for validation and 78.14% for testing.

4.2 Analysis

In this part of the paper, overall model and performance comparison will be discussed. Then based on comparison and performance, a discussion for improvement and which model is performing how in which situation will be shown.
We proposed two models for hate speech detection from social media post or comments. One is using CNN (convolutional neural network) and another one is by using XGBoost. Then we took three sets of data especially from twitter to evaluate our model. Here comparative performance tables have been given which shows a comparative overview of model performance.

Dataset 1		Hate	Offensive	Neutral
CNN	Precision	0.808	0.814	0.915
	Recall	0.754	0.849	0.932
	F1-Score	0.780	0.831	0.923
	Testing Accuracy	84.74%		
XGBoost	Precision	0.758	0.802	0.841
	Recall	0.662	0.786	0.955
	F1-Score	0.707	0.794	0.894
	Testing Accuracy	80.51%		

Table 4.2: performance comparison between models using CNN and XGBoost for dataset 1

For dataset 1 and dataset 2 we have used 500 epochs to develop the accuracy for the proposed CNN model. We get 0.780, 0.831 and 0.923 f1 score for hatred, offensive_language and neutral in dataset 1. On this dataset, the accuracy is 84.74% .Then we have used our second data set containing 2805 of labelled data with hate as labelled 0 and neutral as labelled 1. We get 0.884 and 0.899 f1 score for neutral and hatred. This time we get 0.8918 or 89.18% of testing accuracy.
Our third dataset contains more than 2300 of labeled data as hatred, offensive and neutral. We have used 200 epochs to develop the performance of our model. We get 0.623, 0.531 and 0.893 f1-score for hate, offensive and neutral.
On these similar datasets, XGBoost classifier has been used. We get 0.707, 0.794 and 0.894 as f1-score for hatred,offensive and neutral respectively on the first dataset. Then on the second dataset applying the same model, we get 0.928 and 0.934 as f1-score for hatred and neutral.
For the last dataset which we have created from twitter has given 0.626, 0.635 and 0.877 as f1-score for hatred, offensive and neutral data.

Dataset 2		Hate	Neutral
CNN	Precision	0.931	0.851
	Recall	0.828	0.942
	F1-Score	0.877	0.894
	Testing Accuracy	89.18%	
XGBoost	Precision	0.945	0.919
	Recall	0.912	0.949
	F1-Score	0.928	0.934
	Testing Accuracy	93.10%	

Table 4.3: performance comparison between models using CNN and XGBoost for dataset 2

Dataset 3		Hate	Offensive	Neutral
CNN	Precision	0.558	0.603	0.899
	Recall	0.705	0.475	0.887
	F1-Score	0.623	0.531	0.893
	Testing Accuracy	76.70%		
XGBoost	Precision	0.885	0.557	0.861
	Recall	0.484	0.737	0.635
	F1-Score	0.626	0.635	0.877
	Testing Accuracy	78.14%		

Table 4.4: performance comparison between models using CNN and XGBoost for dataset 3

Through this research work, we have tried to develop an optimized and accurate system to detect hate speech from social media post. In order to do that we have gone through some research works to obtain the perfect model. After studying and doing a handful research we have tried to develop two models using CNN (convolutional neural network) and XGBoost to detect hate speech from social media post.

We have used three different datasets on two proposed models.Among the datasets, two of them have been collected and one was created by us which was discussed earlier. Using CNN model,after parameter tuning and tweaking, we got almost 85% and 89% accuracy on both of the collected datasets.We have tried to use same portion of samples for hate,offensive and neutral data in order to remove the classifier biasness . On this two dataset,precision,recall and F-1 score that has already been discussed in previous portion were good enough. In most of the cases the model detects hate or neutral or offensive from the text with more than 85% of accuracy.

For the dataset that we have created (mentioned as dataset 3), we have changed some parameters of the proposed CNN model which is discussed on section 3.5.2. The dataset contains an improper portion of hate,offensive and neutral labelled samples. So our proposed CNN model which worked fine for both of the previous dataset has been tweaked for better result. The architecture was almost the same.But some parameters have been tweaked.

Then we proposed another model,XGBoost classifier. Using XGBoost we got the accuracy of 80.51% and 93.1% for the 1st two datasets and for the third dataset we got 78.14%.

After analysing the results, we observed that for the second dataset, we got a better result in both models. Especially in XGBoost, we found all the result parameters like accuracy, precision, recall and f1-score was more than 90%. Even in CNN model we got a better result for this dataset. In section 3.1 where Data Set was discussed, we saw the tf-idf based t-SNE projection in which dataset 2 has showed the better distribution of classes. This is one of the reason why dataset 2 has given a better result compared to the other two datasets.Besides,the other two datasets were related to ternary classification(3 class) whereas dataset 2 has binary classification(2 class).This could be another reason for better result.

From these results, we can remark that both of the proposed models are performing well in all three different datasets. In machine learning for any classifiaction problem, dataset is a big concern. A good dataset is essential and recommended for any classification problem. We can see that both of the proposed models has worked fine for dataset 1 and dataset 2.It showed a promising result. However, for the third dataset, although the accuracy is good,but the precision,recall and f-1 score is not up to the mark.As the distribution of the classes for datset 3 is not well uniformed and the dataset contains different portion of samples for hate,offensive and neutral, the precision,recall and f-1 score is not up to the mark.

The two proposed models have obtained a good and desired result in terms of performance. But in general, CNN model has performed better for ternary classification and XGBoost has given a better result on binary classification .Another remarkable finding of our research is that XGBoost is better in terms of speed compared to CNN. The results on three different dataset shows that the proposed models are generalized and can fit well for any datasets.

There are few works found on related topics with the same dataset and models. Though the proposed models,working methodology, using and selection of dataset are not fully same with the existing works but comparison with existing model and proposed model in terms of some result parameters are given here.

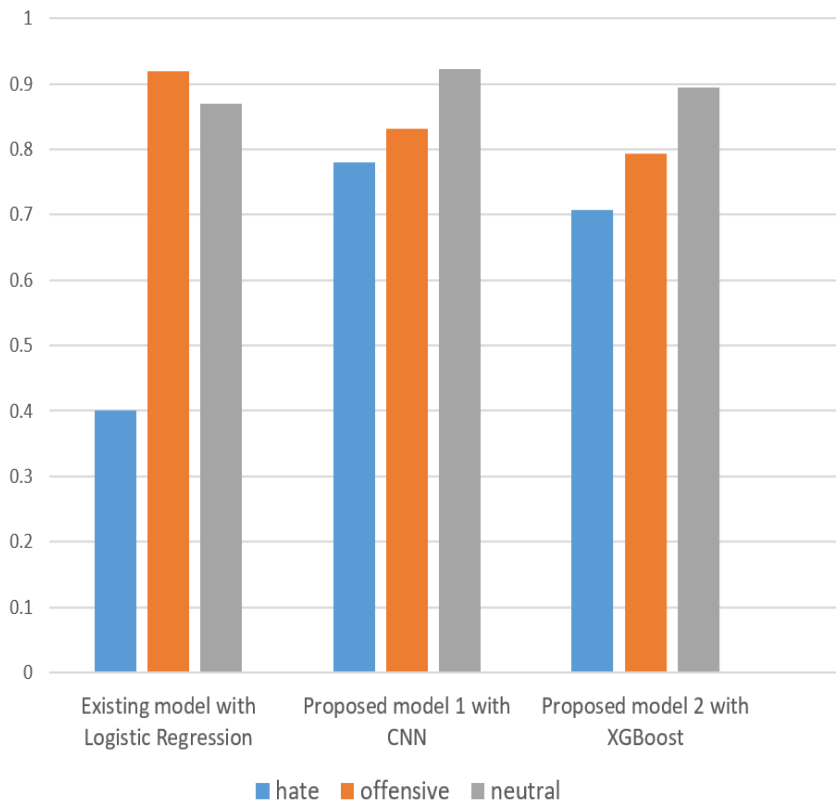

Figure 4.19: Comparison with F1 score of proposed models and existing model for dataset I

Figure 4.19 shows the comparison of F1 score for the proposed model with existing work for dataset 1. Dataset 1, which is known as t-Davidson dataset, there are three classes namely hate, offensive and neutral. This dataset has been used by[34] and the researchers have used logistic regression with L2 regularization.

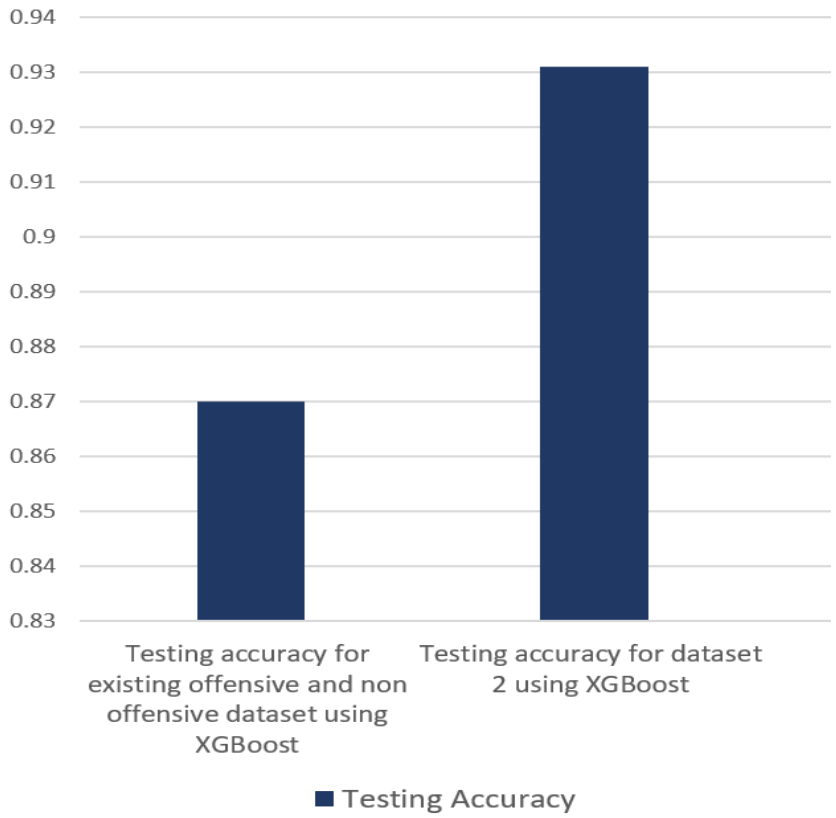

Figure 4.20: Comparison with existing dataset and dataset 2 on testing dataset using XGBoost model for binary classification

Figure 4.20 depicts the comparison of XGBoost model on binary classification. Here the researcher[42] have used XGBoost model for binary classificatin on a dataset containing offensive and non offensive labelled data. Here the figure shows comparison of our proposed XGBoost model on dataset 2 with the the work of [42].

Chapter 5

Conclusion

This research mainly focuses on detecting hatred and offensive language from social media posts.The main challenge that is often overlooked while detecting hate speech is the use of shortcuts and trivial changes in the main keyword that is considered as hate word.Because of the complexity of natural language as well as the usage of slightly changed words to represent the same meaning are the factors that make it more challenging. The purpose of this paper is to overcome these challenges in an efficient way .In this paper,two different approaches have been proposed for identifying hate speech on social media.In the first approach a pre-trained word embedding using fastText is used on CNN architecture.As fastText uses n-gram character level information,it is useful to preserve the meaning of shorter words that may show up as n-gram of other words.Beside,Our purpose was to introduce a different dimension on detecting hate speech so we have used word embeddings and CNN combination.In our later approach,we have used BOW then TF-IDF and XGBoost classifier.We have used three different datasets for more generalizing our models . On the first dataset which contains hate,offensive and neutral,the accuracy that has been obtained is 85% and 81% using CNN and XGBoost respectively. On another dataset which contains hate and neutral comments,the accuracy is 89% and 93% respectively.Besides, a dataset was created using twitter API and was labelled by the help of nine annotators.On that dataset using CNN and XGBoost, the accuracy that have been accomplished on test data are 77% and 78% respectively.Although the precision,recall and f-1 score on hate and offensive is not quite remarkable on that dataset as the count of hate and offensive labelled data is not that much compared to neutral. In general, The paper proposes a new approach on detecting hate and offensive language from social media which yields promising results.

Future Work

For the future work, We will consider the following things in order to expand the work as there is still some scope.

- Creating a Bangla dataset from twitter and detecting hate speech.
- Creating a large dictionary of hate word.
- A word tokenizer which will be effective for tokenizing tweets that contain hidden profanity.

Bibliography

[1] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*, 1st. O'Reilly Media, Inc., 2009, ISBN: 0596516495, 9780596516499.

[2] T. Tieleman and G. Hinton, "Rmsprop: Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning", *Tech. Rep., Technical report*, p. 31, 2012.

[3] D. Hendricks, "Complete history of social media: Then and now", *Small Business Trends*, vol. 8, 2013.

[4] Y. Kim, "Convolutional neural networks for sentence classification", in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751. DOI: 10.3115/v1/D14-1181. [Online]. Available: https://www.aclweb.org/anthology/D14-1181.

[5] K. Markham, "Simple guide to confusion matrix terminology", *data school*, vol. 25, 2014.

[6] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. New York, NY, USA: Cambridge University Press, 2014, ISBN: 1107057132, 9781107057135.

[7] P. Burnap and M. L. Williams, "Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making", 2015.

[8] 2016. [Online]. Available: https://medium.com/data-design/xgboost-hi-im-gamma-what-can-i-do-for-you-and-the-tuning-of-regularization-a42ea17e6ab6.

[9] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system", in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, San Francisco, California, USA: ACM, 2016, pp. 785–794, ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939785.

[10] E. Grave, T. Mikolov, A. Joulin, and P. Bojanowski, "Bag of tricks for efficient text classification", in *EACL*, 2016.

[11] A. JAIN, *Complete guide to parameter tuning in xgboost (with codes in python)*, 2016. [Online]. Available: https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/.

[12] 2017. [Online]. Available: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data.

[13] T. Davidson, D. Warmsley, M. W. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language", *CoRR*, vol. abs/1703.04009, 2017. arXiv: 1703.04009. [Online]. Available: http://arxiv.org/abs/1703.04009.

[14] A. Davis, *How social media spurred myanmar's latest violence*, Sep. 2017. [Online]. Available: https://iwpr.net/global-voices/how-social-media-spurred-myanmars-latest.

[15] F. Del Vigna, A. Cimino, F. Dell'Orletta, M. Petrocchi, and M. Tesconi, "Hate me, hate me not: Hate speech detection on facebook", Jan. 2017.

[16] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification", in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 427–431. [Online]. Available: https://www.aclweb.org/anthology/E17-2068.

[17] J. Patterson and A. Gibson, *Deep Learning: A Practitioner's Approach*. Beijing: O'Reilly, 2017, ISBN: 978-1-4919-1425-0. [Online]. Available: https://www.safaribooksonline.com/library/view/deep-learning/9781491924570/.

[18] Z. Wang and Z. Qu, "Research on web text classification algorithm based on improved cnn and svm", *2017 IEEE 17th International Conference on Communication Technology (ICCT)*, pp. 1958–1961, 2017.

[19] *Confusion matrix in machine learning*, Feb. 2018. [Online]. Available: https://www.geeksforgeeks.org/confusion-matrix-machine-learning/.

[20] S. S. Dhaliwal, A. A. Nahid, and R. Abbas, "Effective intrusion detection system using xgboost", *Information*, vol. 9, p. 149, 2018.

[21] S. Frenda and B. Somnath, *Deep analysis in aggressive mexican tweets*, 2018. [Online]. Available: http://hdl.handle.net/2318/1676282.

[22] A. Gowen and M. Sharma, "Rising hate in india", *Washington Post*, 2018.

[23] F. Mohammad, "Is preprocessing of text really worth your time for online comment classification?", *ArXiv*, vol. abs/1806.02908, 2018.

[24] K. Müller and C. Schwarz, "Fanning the flames of hate: Social media and hate crime", *Available at SSRN 3082972*, 2018.

[25] S. Narkhede, "Understanding confusion matrix", *URL: https://towardsdatascience.com/understanding-confus ion-matrix-a9ad42dcfd62. Accessed*, vol. 10, 2018.

[26] J. Risch and R. Krestel, "Aggression identification using deep learning and data augmentation", in *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 150–158. [Online]. Available: https://www.aclweb.org/anthology/W18-4418.

[27] K. Roose, "On gab, an extremist-friendly site, pittsburgh shooting suspect aired his hatred in full", *New York Times*, 2018.

[28] N. D. T. Ruwandika and A. R. Weerasinghe, "Identification of hate speech in social media", in *2018 18th International Conference on Advances in ICT for Emerging Regions (ICTer)*, Sep. 2018, pp. 273–278. DOI: 10.1109/ICTER.2018.8615517.

[29] M. Safi, *Sri lanka accuses facebook over hate speech after deadly riots*, Mar. 2018. [Online]. Available: https://www.theguardian.com/world/2018/mar/14/facebook-accused-by-sri-lanka-of-failing-to-control-hate-speech.

[30] N. Setyadi, M. Nasrun, and C. Setianingsih, "Text analysis for hate speech detection using backpropagation neural network", Dec. 2018, pp. 159–165. DOI: 10.1109/ICCEREC.2018.8712109.

[31] H. Watanabe, M. Bouazizi, and T. Ohtsuki, "Hate speech on twitter: A pragmatic approach to collect hateful and offensive expressions and perform hate speech detection", *IEEE Access*, vol. 6, pp. 13 825–13 835, 2018, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2806394.

[32] 2019. [Online]. Available: https://xgboost.readthedocs.io/en/latest/parameter.html#parameters-for-tree-booster.

[33] D. Crystal and R. H. Robins, *Meaning and style in language*, Jan. 2019. [Online]. Available: https://www.britannica.com/topic/language/Meaning-and-style-in-language.

[34] T. Davidson, D. Bhattacharya, and I. Weber, "Racial bias in hate speech and abusive language detection datasets", *arXiv preprint arXiv:1905.12516*, 2019.

[35] R. Evans, "Shitposting, inspirational terrorism, and the christchurch mosque massacre", *Bellingcat, March*, vol. 15, 2019.

[36] T. Febriana and A. Budiarto, "Twitter dataset for hate speech and cyberbullying detection in indonesian language", in *2019 International Conference on Information Management and Technology (ICIMTech)*, vol. 1, Aug. 2019, pp. 379–382. DOI: 10.1109/ICIMTech.2019.8843722.

[37] S. Kemp, "Digital 2019: Global internet use accelerates", *Retrieved from wearesocial: https://wearesocial. com/blog/2019/01/digital-2019-global-internet-useaccelerates*, 2019.

[38] N. Kumar, *Advantages of xgboost algorithm in machine learning*, 2019. [Online]. Available: http://theprofessionalspoint.blogspot.com/2019/03/advantages-of-xgboost-algorithm-in.html.

[39] Z. Laub, "Hate speech on social media: Global comparisons", *Council on Foreign Relations*, vol. 7, 2019.

[40] H. Liu, P. Burnap, W. Alorainy, and M. L. Williams, "A fuzzy approach to text classification with two-stage training for ambiguous instances", *IEEE Transactions on Computational Social Systems*, vol. 6, no. 2, pp. 227–240, Apr. 2019, ISSN: 2373-7476. DOI: 10.1109/TCSS.2019.2892037.

[41] A. Long, *Understanding data science classification metrics in scikit-learn in python*, Feb. 2019. [Online]. Available: https://towardsdatascience.com/understanding-data-science-classification-metrics-in-scikit-learn-in-python-3bc336865019.

[42] M. Ramakrishnan, W. Zadrozny, and N. Tabari, "Uva wahoos at semeval-2019 task 6: Hate speech identification using ensemble machine learning", in *Proceedings of the 13th International Workshop on Semantic Evaluation*, 2019, pp. 806–811.

[43] A. Rodríguez, C. Argueta, and Y. Chen, "Automatic detection of hate speech on facebook using sentiment and emotion analysis", in *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, Feb. 2019, pp. 169–174. DOI: 10.1109/ICAIIC.2019.8669073.

[44] L. H. Son, A. Kumar, S. R. Sangwan, A. Arora, A. Nayyar, and M. Abdel-Basset, "Sarcasm detection using soft attention-based bidirectional long short-term memory model with convolution network", *IEEE Access*, vol. 7, pp. 23 319–23 328, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2899260.