Deep Learning Based Hate Speech Detection in Twitter Using Convolutional Neural Network

Sidharth K [1], Subramanian L [1], Vinod P[2], Suleiman Y. Yerima[3], , Thushara P T[1], Sreelekshmi S[1]
[1]*Computer Science Engineering Department, SCMS School of Engineering and Technology* Ernakulam, India
[2] *Department of Computer Applications, Cochin University of Science and Technology, Cochin University P.O, Cochin-682 022*
[3] *Cyber Technology Institute, Faculty of Computing, Engineering and Media*
De Montfort University, Leicester, UK
sidharthk97india@gmail.com, arun.lsubramanian@gmail.com, vinod.p@cusat.ac.in,syerima@dmu.ac.uk, thusharapt1997@gmail.com, sreelekshmi031998@gmail.com

Abstract— **Hate speech generally refers to language that is used to derogate, humiliate or express hatred towards a group of people in a manner that could be potentially harmful to them. With the growing popularity of social media platforms such as Twitter and Facebook, hate speech is on the increase at an alarming rate. Consequently, automated detection of hate speech and other harmful content using machine learning (ML) is currently an active field of research. Even though several ML-based detection methods have been proposed in previous research, most of them do not adequately distinguish hate speech from instances of offensive language. Hence, in this paper we propose a deep learning approach based on Convolutional Neural Networks (CNN) to tackle the problem by classifying tweets into three categories: hate speech, offensive speech and neutral speech. By means of experiments on a large dataset, the CNN approach is compared to Bi-directional LSTM, Linear Regression, Random Forest, Decision Tree, SVM, Gaussian Naive Bayes as well as an ensemble of ML classifiers using Stacking with XGBoost. The results show that CNN, with an overall F1 score of 0.99, outperforms all the other ML techniques. Furthermore, compared to the state-of-the-art, the proposed CNN approach improved the overall performance by 9%, whilst reducing the misclassification rate of hate speech from 40% down to 3%.**

I. INTRODUCTION

Today, social media is on the rise and has become embedded in the fabric of society. More and more people are using social media to stream live experiences, engage in various forms of discourse, share news and current affairs, interact with family and friends all around the globe, etc. This phenomenon has been fueled by increasing affordability of Internet connectivity leading to widespread access to these social media platforms worldwide.

Social media provides a place where people can feel free to express their views without being censored. Nowadays, some social media users are inclined to abuse this freedom of expression because of the perceived lack of immediate or personal consequences. As a result, the appearance of harmful content such as fake news, propaganda, racism, offensive language, discrimination, hate speech etc. is on the rise. These are particularly evident on the most popular sites like YouTube, Facebook, Twitter, etc.

Typically, when negative content is posted by users on these sites, it remains intact (sometimes for long periods of time) until a certain number of people report to the site administrator. The site administrator then checks for the validity of the reports and removes such content, if found to have violated the site's policies. With the overwhelming increase of traffic to these sites, such manual removal process has been found to be costly, in-effective and time-consuming. Hence, most of the social media giants have implemented automated mechanisms to control the spread of hate speech. Such tools work typically by scanning each word against prohibited words (from a pre-existing lexicon) in order to filter hate speech from normal content. However, such basic word scanning measures have proven less effective in the long run. People have been able to identify some techniques to bypass this detection method, and such approaches are discussed in [2]. Word changes, word-boundary changes, leetspeak are some examples of the evasion approaches being employed by savvy social media users to bypass filters.

To address these limitations, researchers have proposed automated machine learning based approaches to classify tweets with the aim of identifying hate speech. As noted by [1], distinguishing hate speech from other forms of expression can be quite challenging because hate speech lexicons may contain profane or offensive words that may be used in a different context. Indeed, the presence of profane content does not in itself signify hate speech. General profanity is not necessarily targeted towards an individual and may be used for stylistic purposes or emphasis. On the other hand, hate speech may denigrate or threaten an individual or a group of people without the use of any profanities [13]. This implies that automated detection methods designed to detect hate speech must be capable of differentiating hate speech from offensive expression. They must also possess the ability to detect hate speech that is devoid of the use of offensive language. Any automated approach implemented without incorporating such capabilities will ultimately result in a lower precision system.

Hence, in this paper we propose an approach for hate speech detection on Twitter using deep learning. Our approach tackles the problem of conflating hate speech with offensive language by implementing a multi-class detection system where machine learning models are trained to classify tweets into three categories: hate speech, offensive speech and neutral speech.

In summary the main contributions of the paper are as follows:

- We propose and evaluate a deep learning approach based on CNN for high accuracy detection of hate speech. Unlike many previous works, the method uses a multi-class approach to enable offensive expression to be distinguished from hate speech, thereby increasing precision and overall accuracy. Compared to the state of the art, our CNN model increased the overall accuracy by 9% while reducing the hate speech misclassification rate from 40% down to 3%.
- We present extensive experiments that compare deep learning approaches to conventional machine learning techniques i.e. SVM, Random Forest, Logistic Regression and Decision Trees. Additionally, a comparison is made to the ensemble technique of Stacking with XGBoost meta classifier.
- An analysis of inter-class and intra-class performance is presented, revealing inter-class word commonalities (between the hate, offensive and neutral) and their prevalence, giving further insight the misclassification performance of the machine learning models.

The rest of the paper is organized as follows: Section II discusses related works; Section III presents the methodologies including the architecture of the proposed approach, dataset pre-processing and data embedding, and the machine learning models. Section IV presents the experiments and results; Section V includes discussions, where performance of each algorithms are analyzed. Section VI contains comparative analysis of the study while Section VII presents the conclusions and future work.

II. RELATED WORKS

In [1], Davidson et al. addressed the problem of automatic hate speech detection using supervised learning. The focus of their work was on separating hate speech from other instances of offensive language as prior works failed to distinguish between the two due to lexical methods classifying all messages containing particular terms as hate speech. They used crowd-sourced hate speech lexicon to collect tweets containing hate speech keywords. Crowd sourcing is also used to label samples of the tweets into three categories: hate speech, offensive speech and neither. They used logistic regression with L1 regularization for data dimensionality reduction, after which several ML models that have been used in prior works were applied: logistic regression (LR), Naïve Bayes, Decision Trees, Random Forest and Linear SVM. Their results indicated that LR and SVM performed significantly better than others. In this paper, we utilized the annotated datasets from [1] for our experiments. However, our proposed approach is based on deep learning which ultimately resulted in significantly better classification performance than their best reported results.

Founta et al. [15] presented an eight month study of the various forms of abuse activity on Twitter. Departing from past work, they examined a wide variety of labelling schemes which cover different forms of abusive behavior. They then proposed an incremental and iterative method that utilizes crowdsourcing to annotate a large collection of tweets with a set of abuse-related labels. The annotation focused on various types of abusive or hateful language in Twitter. After statistical analysis of similarities between labels, some were merged, while others were eliminated. They ended up with two labels which they concluded to be the most representative i.e. Abusive and Hateful. Their final annotated dataset contains 80,000 Tweets which have been released to the research community.

Gaydhani et al. [16] applied N-gram and TF-IDF on different machine learning models using two different datasets from [1] and [17]. From their comparative analysis of several ML models, Logistic Regression emerged as the best with an accuracy of 95.6%. It was seen that 4.8% of the offensive tweets were misclassified as hateful. The study supports the notion that more examples of offensive language which do not contain hateful words are necessary to improve the results as discussed in [1].

Nouh et al. [5] presented a study aimed at identifying measures to detect original content in social media automatically. They identify several signals, including textual, psychological, and behavioural that together allow for the classification of radical messages. The contribution is three-fold: Analysing propaganda material published by extremist groups, and creating a text-based contextual model of original content, and building a model of psychological properties inferred from these materials. They evaluated these models on Twitter to determine the extent to which it is possible to identify radical online tweets automatically. The study shows that radical users do exhibit distinguishable textual, psychological, and behavioural properties. It found that the psychological properties were among the most distinguishing features. Additionally, the results show that textual models using vector embedding features significantly improves the detection over TF-IDF features.

Grondahl et al. [2] convey that there are numerous instances where automated hate and offensive texts are detected online, which are implemented by the social media giants such as Facebook and Twitter. Such systems emerged after people started to use social media for spreading racism, gender-based discrimination, fake news, etc. But adversary attacks to evade such detection came into existence after people realised how these systems worked. Some examples of these methods are leetspeak, changing a word, changing word boundaries (appending two words, splitting a word), etc. These attacks were successful as these systems depended mostly on individual words. It also focuses on the ways

TABLE I
RELATED WORKS

Sl No	Related Work Name	Inference Of Work
1	Davidson et al. 2017	Proposed a study to distinguish hate speech and offensive language from normal tweets, achieving an overall result of 90% accuracy. The misclassification rate was high due to the presence of certain profane words in both hate speech and offensive language categories.
2	Founta et al. 2018	Introduced a new dataset containing tweets with a variety of labels. Based on their empirical analysis of the relationship between the corresponding labels certain labels were merged or eliminated to derive the best representative dataset.
3	Gaydhani et al. 2018	Different machine learning models were used to train and test on two different datasets from Davidson et al. and Watanabe et al., out of which Logistic Regression exhibits an accuracy of 95.6%. The study concluded that more samples of certain labels are required for better analysis.
4	Nouh et al. 2019	Proposed a system of creating a text-based contextual model of radical content and building psychological and behavioural properties from the content. They Applied this on Twitter to analyse the rate of detecting radical contents. Additionally, the work proved that embedding using TF-IDF improved detection.
5	Grondahl et al. 2018	Proposed ways of adversary attacks that can be used to evade detection of hate speech in online network platforms and inferred that the model architecture has no major impact on classifier performance and that linguistic hate speech indicators are not well retained across different datasets.
6	Luu et al. 2020	Comparative analysis between traditional machine learning models and deep learning models were performed on a large dataset of comments from social media. The best result for hate speech detection was obtained using CNN with an F1 score of 0.83; traditional models failed due to oversampling in the dataset.

an adversarial attack can be made successful. The main inference of the study is that model architecture has no significant impact on classifier performance.

Luu et al. [8] conducted a study on the comparison between traditional machine learning models and deep learning models in detecting hate speech on a large dataset containing user comments on social network. The result shows that deep neural network models are better than traditional machine learning models, and the CNN model classified hate speech texts with 83.04% macro F1-score. Similar to other works, their analysis suggests that profane words and personal pronouns influenced the identification of offensive and hateful comments. The best results from the traditional models was from SVM which had an accuracy of 65.10%, and this was much lower than the results from deep neural network models. The authors concluded that lack of sufficient data in the offensive and hate categories impacted the prediction performance of the models.

The method proposed in this paper is more efficient when compared to the related works; a new organised version of the dataset was created to implement and train an effective hate speech detection system. This study shows that deep learning models perform better than traditional machine learning models, having achieved an F1 score of 0.99. Even though there was oversampling in the dataset, the model achieved a better classification rate compared to the work in [1]. Similar to previous work, from this study, it was also evident that the removal of certain profane words may lead to better results in the future works.

III. PROPOSED METHOD

A. Dataset

The dataset used for the study in this paper is obtained from [1]. This dataset contains 24,782 annotated tweets gathered from 33,458 Twitter users. The tweets have been labelled with three classes namely, 'offensive', 'hate speech' and 'neither' using a crowd funding approach as described by the authors of the paper. As shown in Table II, the dataset contains 19,189 offensive tweets (77.43%), 1,430 hate tweets (5.77%) and 4,163 neutral tweets (16.79%). The number of unique words found in each class of tweets are shown in the last column of Table II.

TABLE II
DATASET DESCRIPTION

Sl no	Name of Class	Number of samples (Descriptions)	Percentage (%)	No. of Unique Words
1	Hate	1,430	5.77%	3,575 |
2	Offensive	19,189	77.43%	15,725
3	Neutral	4,163	16.79%	9,704
	Total Descriptions	24,782		29,004

B. Architecture of ML based tweet classification system

In Fig 1, we summarize the steps involved in creating the various ML classifiers that have been studied in this paper. These consist of pre-processing of the raw data (tweets), converting the data into various inputs formats depending on the ML model, dimensionality reduction (if applicable), model training, and classification. The data is transformed into tokenized sequences for the deep learning models. While for the traditional ML models, the unigram counts of the words are produced for each tweet along with the class labels. Further dimensionality reduction is then applied to reduce the model training time.

C. Pre-Processing of Data

In NLP problems, it is a common practice to pre-process the raw data to remove irrelevant parts as noisy data will affect the performance of the algorithms [14] [19]. The following pre-processing were applied to the raw tweets in the dataset:

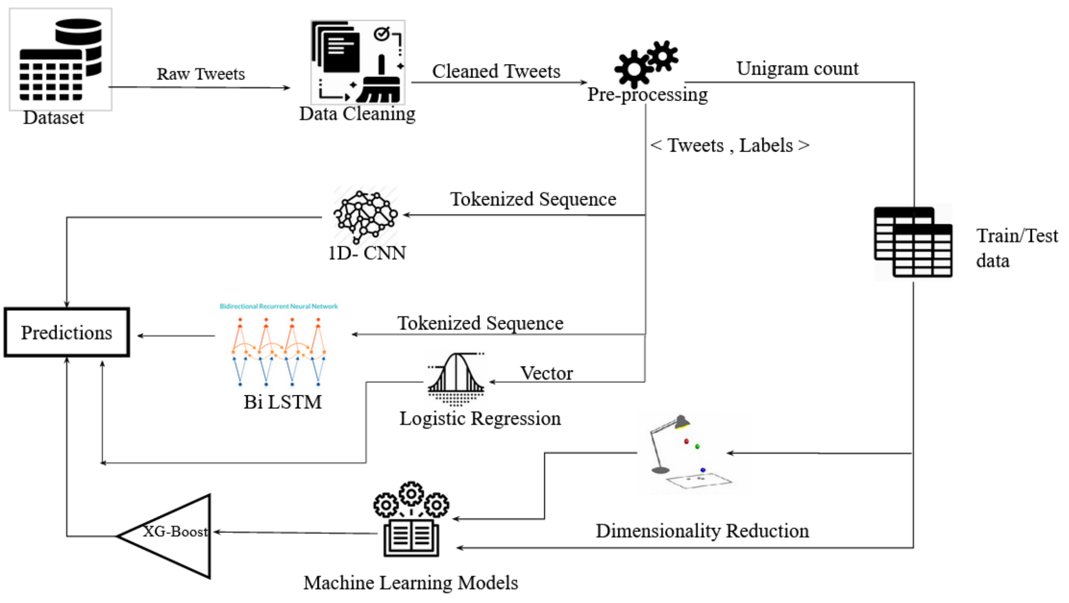

Fig. 1. Architecture diagram: summary of the stages involved in creating various machine learning models to predict the category of the tweets.

- Tokenization: This is the process of breaking the tweets smaller units or tokens for example: "I like black cars" » ['I','like','black','cars']
- Converting all tweets into lower case.
- Removing hash tags [#goodboy] and Twitter handles/mentions [@name].
- Removing punctuations, numbers, special symbols [$,%,&,*,!], stopwords and URLs [https://bitly.is/3d5Bkjx].

D. Dimensionality Reduction

Dimensionality reduction [22] techniques will remove unwanted features and reduce the size of data and thereby reduce the processing time and memory requirements. Dimensionality reduction techniques are divided into two categories: feature selection and feature extraction. Feature selection [30] will find a subset of relevant features from the feature set. Feature extraction will project the features from a higher dimension to a lower dimension and can be linear or non-linear. PCA is a linear feature extraction technique.

1) Principal Component Analysis: Principal Component Analysis(PCA) is a dimensionality reduction technique to identify the correlation between different features and to remove redundant features without information loss. It is a linear dimensionality reduction technique [4]. It is also known as "Parsimonious Summarization" [3]. The two goals of using PCA are data reduction and interpretation. Interpretation means to identify the relevant features from a large set of features. PCA will convert high dimensional data into a low dimension. The principal component is a vector showing high variance and low correlation. The number of principal components can be equal to the number of features or lesser. It is assumed that the first principal component has the most relevant information as it has more variance [27]. In the present study, n different components were taken for analysis, and we obtain the best result. Since the data is a sparse matrix, PCA was not able to produce a better result as expected.

2) TF-IDF: Term Frequency-Inverse Document Frequency, also known as TF-IDF [16], is a method to determine how important a word is in a document. Multiplying the two metrics, term frequency, and inverse document frequency gives the TF-IDF value. Term frequency implies the number of times a particular word occurs in a document. Inverse document frequency is used to determine whether a word is common or rare in the entire document set, if the value approaches 0, then that particular word is common in documents which imply that the word is irrelevant. TF-IDF thus helps in feature extraction by identifying the relevant features, the greater the value of TF-IDF, the more important the feature will be for classification.

$$T_F ID_F(t,d,D) = T_F(t,d) \times ID_F(t), \tag{1}$$

$$T_F(t,d) = |t \in d, d = t_1, t_2, ...t_n : t = t_i| \tag{2}$$

$$ID_F(t) = log(D/1 + d : t \in d). \tag{3}$$

where, d $\in D$

E. Machine Learning Models

1) Support Vector Machine: A Support Vector Machine(SVM) [20] is a supervised classification model that can be used for both classification and regression. This algorithm works on a simple strategy of separating hyperplanes. The objective of SVM is to find a hyperplane in an N-dimensional space, where N is the number of features. The hyperplane distinctly classifies the data points. There may be several hyperplanes for a given problem, and the aim is to identify the maximum margin hyperplane, i.e.

the hyperplane having maximum distance between the data points of both classes. Support vectors in SVM are the data points that are closer to the hyperplane. The main advantage of SVM is that they are most accurate in high dimensional spaces, and they are also memory efficient since they use only support vectors for decision. However, SVM requires a large amount of time in training larger datasets.

2) Decision Tree: Decision Tree models build a tree with many branches and leaf nodes depending upon the outcomes of a probable event based on the given data. Here, the attribute acts as a test case, and the corresponding edge gives the possible answer. The leaf node contains the actual output or the class label. The algorithm then finds the optimum and most probable output. The output to which the algorithm predicts also shows the optimum path from the root node to its leaf node. Decision Tree act as a predictive tool that serves many applications in the modern era of technology. It is a type of supervised learning technique in which a predictive output is constantly compared with the actual output. They are used mostly in non-linear decision-making scenarios.

3) Gaussian Näive Bayes: Näive Bayes classifier is a classifier developed using the Bayes Theorem.The theorem predicts the probability of an outcome depending upon the prior knowledge of an event using the available data [20]. In other words, using Näive Bayes increases the chance of accurately predicting the event by using the conditions or experiences from the past. Hence, it becomes easier to predict compared to other machine learning models. Mathematically, Bayes Theorem is as follows:

$$P(A|B) = (P(B|A) * P(A))/P(B) \tag{4}$$

Where,
A,B = events
P(A|B) = probability of A given B is true
P(B|A) = probability of B given A is true
P(A),P(B) = the independent probabilities of A and B

4) Random Forest: Random Forest is a supervised model that can use for classification and regression. It is an ensemble algorithm that uses other classification algorithms such as SVM, Decision Tree, etc. for making predictions. It can be called a model of many decision trees. Each of these decision trees, help in the formation of rules which will be used for prediction. If the number of decision trees is large, the more accurate will be the result. There are mainly two stages in a Random Forest algorithm, the first step is to create a Random Forest, and the next step is to make predictions with the help of it. The major advantage of Random Forest is that it avoids overfitting in many cases. During testing, it helps in identifying the most important features from the training dataset, i.e. feature engineering[12].

5) Logistic Regression: Logistic Regression is a supervised learning technique, it will find the value of the dependent variable based on a set of independent variables given. The output value might be categorical or discrete, it can be 0/1, yes/no, or True/False. Logistic Regression [11] will predict the probability of an instance to be in a class. To map predicted values into probabilities 'Sigmoid function' is used, which will map a value to another value in a range from 0 to 1. A sigmoid function is synonymously known as "logistic function". Logistic Regression applies to binary as well as multi-class classification problems [24].

6) Stacking-XGBoost: XGBoost is a tree boosting technique employed using Newton boosting [9] [10]. The origin of Boosting points to a query posed by Kearns(1988) and valiant(1989), about the possibility of converting a set of weak classifiers into a strong classifier. The invention of AdaBoost, the first boosting algorithm, makes an answer to the question. Boosting is an "ensemble learning" technique in which a set of weak learners with poor performance when stand-alone are combined to form a strong learner. The weak learners are also called base learners. Due to several systems and algorithmic optimisations, XGBoost is scalable in all situations, and this is the reason for the popularity of XGBoost. The innovations in XGBoost include:

- In order to handle sparse data, a new algorithm is introduced.
- In order to handle instance weights in tree learning a theoretically justified quantile sketch procedure.
- Parallel and distributed processing make learning faster.

XGBoost is suitable for both classification and regression problems. It works well for structured or tabular data. The two goals of using XGBoost is efficient memory usage and reduced processing time. XGBoost can perform the three gradient boosting techniques such as Gradient Boosting, Regularized Boosting, and Stochastic Boosting.

F. Neural Network Models

1) Convolutional Neural Network: A Convolutional Neural Network(CNN) [6] is a type of deep neural network which is mostly applied in image classification. The main advantage of CNN is that it automatically detects the important features without any supervision. It makes use of special convolution and pooling operations and performs parameter sharing, and this makes them universally attractive. Convolution is considered as a mathematical operation to merge two sets of information; a feature map which helps in generating unique patterns in text classification [28] is generated using a convolution filter.

2) Bi-LSTM: Bi-Directional Long Short Term Memory [7] is two RNNs working together. This model can deal with both forward and backward sequence information and helps in finding contextual meaning from tweets and improving model performance on sequence classification problems. Bi-LSTM runs the input in two ways from left to right and right to left. Both activations would be considered to calculate output at a certain time.

G. Training and Testing

Training and testing is the next step once the models are decided. The models are trained using the training data and are tested over the test data. Train- test split method is applied here for learning and prediction from the dataset. It will divide the data set into training sets and a test set. The train set contains 70% of the entire data, and the test set includes 30% of the whole data. The training set contains labelled inputs, and the test set contains unlabeled data. The random state is used to ensure that the splits are generated in the same order. The models can be used to predict the class labels of the training data, once the models are trained using the training data. The performance of each classifier can be measured using a performance matrix.

H. Model Evaluation

Accuracy, Precision, Recall, and F1 score are the measures used for model evaluation. Accuracy: It is the ratio of correctly labelled predictions to the total number of predictions. Precision: It is the ratio of correctly predicted positive objects to the total number of objects predicted as true. Recall: It is the ratio of correctly labelled positive objects to a total number of positive objects. F1 score: It is the weighted average of recall and precision.

$$Accuracy = (TP + TN)/(TP + TN + FP + FN), \quad (5)$$

$$Precision\ (P) = TP/(TP + FP), \quad (6)$$

$$Recall\ (R) = TP/(TP + FN), \quad (7)$$

$$F1score = 2 * (PR)/(P + R). \quad (8)$$

		Predicted Value	
		Positive	**Negative**
Actual Value	**Positive**	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Where,
P: Precision
R: Recall
TP(True Positive): Number of correctly labelled positive predictions.
TN(True Negative): Number of correctly labelled negative predictions.
FP(False Positive): Number of wrongly labelled positive predictions.
FN(False Negative): Number of wrongly labelled negative predictions.

IV. Experiments and Results

A. Dataset

Dataset from [1] was revised to form three CSV files which were created with unique words from each class and counting the words against all other classes. Each tweet contains the labels of their corresponding classes, namely Hate, Offensive, and Neutral. This matrix was given as the input to all base models, namely Decision tree, Gaussian Näive Bayes, Random Forest, and Support Vector Machine(SVM). Since Logistic Regression performs binary classification, here, the model uses "one versus many" approach. Hence a new CSV file is created that contains four attributes namely Tweets, Hate, Offensive, and Neutral where each tweets corresponding to its label were marked '1' and the rest labels as '0'(One-Hot Encoding). Since DNN models have their own unique feature extraction technique; the pre-processed tweets along with their class labels were directly passed to the models.

B. Hardware Software Requirements

Different software used in this work is Tensor flow, Keras Library, Natural Language Toolkit(NLTK), Sci-Kit Learn (SK-Learn), and Colab Lab by Google. Tensor Flow is an Artificial Intelligence library. That can use for building large scale deep neural networks. Keras is a neural network library that runs on the top of TensorFlow [25]. NLTK is a package containing different libraries for text processing. SK- Learn is a python library used for building different machine learning models. Colab is a cloud service provided by Google for writing efficient python programs. The operating system used is Windows, and the minimum processor requirements include 2.1GHz-3GHz Clock speed, Quad-core or Octa-core, and Intel i5, 7th Gen, or AMD Ryzen 3.

C. Performance with Machine Learning Models

1) Outcome of Support Vector Machine: The Table III shows the F1 score obtained for Support Vector Machine (SVM) upon the actual data and reduced data after performing PCA analysis. Since the dataset is large, the Support Vector Classifier(SVC) with a linear kernel is used for better performance. F1 score is used as a parameter to measure the performance when the classes are imbalanced. A high F1 score indicates high recall and precision and a low F1 score indicates low precision and recall. From Fig. 2, it is clear that the result obtained gives better precision and recall using SVM for all three classes. Here the model achieves a better F1 score for Offensive class. But while using SVM the time taken for training was too high, especially for Offensive class hence we have performed dimensionality reduction and the reduced data is given as input to the SVM. The study shows that the result reduces because the input provided is a sparse matrix; thus, PCA was unable to get a better result.

2) Inference of Decision Tree: From Table III, it is clear that the F1 score for reduced data is less than that obtained using original data for all three classes, Hate, Offensive, and Neutral. Decision tree takes two input arrays X(training data) and y(test data), it uses an "if-then-else" decision rule, the

TABLE III
PERFORMANCE WITH MACHINE LEARNING MODELS
(where "pc" determines Principal Component)

Model	Precision	Accuracy	Recall	F1 score
SVM with PCA	0.88	0.88	0.88	0.85
SVM without PCA	0.88	0.88	0.88	0.87
Decision Tree With PCA (pc = 40)	0.75	0.75	0.75	0.75
Decision Tree Without PCA	0.86	0.86	0.86	0.86
Random Forest With PCA (pc = 40)	0.82	0.82	0.82	0.78
Random Forest Without PCA	0.87	0.87	0.87	0.85
Logisitic Regression - Hate	0.97	0.97	0.97	0.97
Logisitic Regression - Offensive	0.94	0.94	0.94	0.95
Logisitic Regression - Neutral	0.97	0.97	0.97	0.97

deeper the tree then more complex the rule and fitter the model. In the case of Hate class, the F1 score without PCA is 0.87, which is higher than that obtained with PCA that is 0.76. For Offensive class, the F1 score obtained without PCA is the same as that of Hate that is 0.87. However, with PCA, the F1 score is reduced to 0.77. In the case of Neutral class, the F1 score was 0.84 without using PCA, and with PCA F1 score was decreased to 0.73.

3) Result of Random Forest: The Table III shows the F1 score value of Random Forest classifier for original data as well as reduced data. Random Forest consists of a multitude of decision trees on various sub-samples of the dataset at training time and predicting the classes, which helps in achieving better performance, control over-fitting and averaging probabilistic predictions of 10 randomised decision trees. For Hate class, the values obtained are 0.85 and 0.80 for original data and reduced data, respectively. For Offensive class, the values obtained are 0.88 and 0.80 for original data and reduced data, respectively. Similarly, for Neutral class, the values obtained are 0.84 and 0.76 for corresponding datasets, respectively.

4) Inference of Logistic Regression: Table III shows the result obtained for Logistic Regression. Since the model uses binary classification, it predicts whether the tweet is true or false corresponding to each class label. Two features used are:
1) TF-IDF with Word analyser: With an ngram range of 1-3 and maximum features of 20000.
2) TF-IDF with character analyser: With an ngram range of 3-6 and maximum features of 40000.
A Logistic Regression model with C=2 (Inverse regularisation parameter) and balanced class weight are used for analysis with F1 score considered as the output. The F1 score obtained for Hate and Neutral class is 0.97 and for Offensive class is 0.95. The result implies that the model achieves a better F1 score for all three classes.

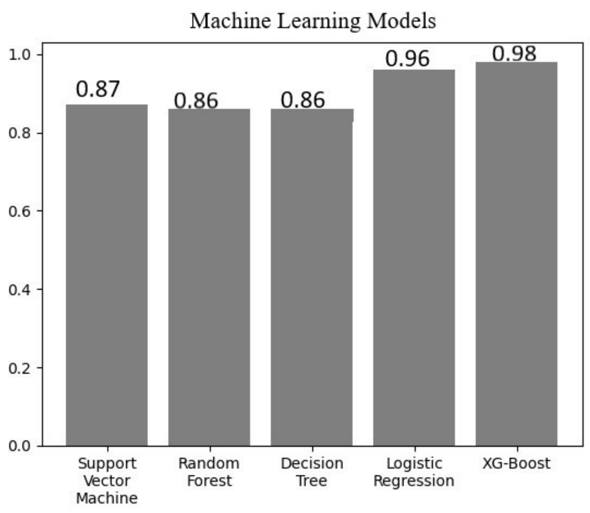

Fig. 2. F1 score-Machine Learning Models

D. Outcome of Stacking

The Table IV shows the result of the XGBoost classifier with and without TF-IDF vectorisation. XGBoost is a stacking classifier for which SVM, Random Forest, Decision Tree, and Gaussian Näive Bayes form the base classifier. The F1 score obtained for XGBoost using TF-IDF and without using TF-IDF is 0.97 and 0.98, respectively. The predictions from each base models will be an array with shape <tweets, classes> and after stacking, the shape changes to <tweets, number of models, classes>, these two matrices will be flattened to form an array of shape <tweets, number of models multiplied by classes>. TF-IDF is employed to identify relevant attributes and thereby reduce the size of the dataset. Since the dataset is a sparce matrix, XGBoost using TF-IDF was unable to perform better.

TABLE IV
RESULT OF STACKING

Classifier	Precision	Accuracy	Recall	F1 score
XGBoost	0.98	0.98	0.98	0.98
XGBoost (Using TF-IDF)	0.97	0.97	0.97	0.97

E. Traditional Machine Learning VS Deep Neural Network

Traditional machine learning requires feature engineering, which is to be identified by a domain expert to reduce the complexity of the data used and make patterns more visible for learning algorithms to work. Complex problems require extensive specialised study within each area, where the problem is broken down into parts and evaluated discretely. Whereas, in deep learning features are comprised to eliminate feature selection, where algorithms try to learn low-level features from data by itself creating n-patterns and producing high-level representation, thus reducing the challenge of feature engineering [8]. Deep learning techniques are adaptive to different domains; effective use of pre-trained models often helps achieve higher performance in a shorter period. Neural networks outperform when it comes to complex problems such as image classification, NLP, etc.

F. Result of Neural Network Models

1) Result of Bi-LSTM: Bi-directional long short term memory is two RNNs working together. This model can deal with both forward and backward sequence information, which helps in classifying tweets into three categories very effectively by learning contextual meaning from the sequences. The sequences are generated from the pre-processed tweets with a maximum sequence length of 250, padded to make each sequence of equal length, and passed to the Bi-LSTM model [23]. The model was built with an embedding layer, a bidirectional LSTM layer, and a dense layer. The embedding layer has embedding dimension 100 and maximum features of 20000. The bidirectional LSTM layer has 100 filters and a dropout of 0.2 to avoid over-fitting. The output layer, with activation SOFTMAX, consists of a dense layer with three neurons corresponding to three different classes. The model was implemented with categorical cross-entropy as the learning objective and selected the ADAM optimisation algorithm in the output layer. Table V gives the results obtained for the Bi-LSTM classifier. Bi-LSTM was the second-best classifier which gave higher value for performance measures during model evaluation. The value obtained by Bi-LSTM for Precision, Recall, and F1 score is all the same, which is 0.98.

2) Peformance of 1D-CNN: CNN is a multistage neural network architecture developed for classification. The sequences are generated from the pre-processed tweets with a maximum sequence length of 250, padded to make each sequence of equal length, and passed to 1D-CNN. The model was built with an embedding layer, two convolution layers, and two dense layers. The embedding layer has embedding dimension 100 and maximum features of 20000. Each convolutional layer has 128 filters, and the size of the filter is 5 with activation as RELU(Rectified Linear) [18]. Max pooling is employed to downsample the input representation. The output layer consists of two dense layers which improve learning and obtain stable output, the fully connected Neural network consists of a Dense layer with fifty neurons. The output layer consists of three neurons with activation SOFTMAX corresponding to three different classes. The model was implemented with categorical cross-entropy as the learning objective and selected the ADAM optimisation algorithm in the output layer. Compared to the other neural network done in the study, 1D-CNN has produced the best result in predicting the tweets with an accuracy of 0.99. Table V shows the result obtained for 1D-CNN, where the precision, recall, and F1 score obtained is 0.99.

TABLE V
RESULT OF NEURAL NETWORK MODELS

Classifier	**Precision**	**Accuracy**	**Recall**	**F1 score**
Bi-LSTM	0.98	0.98	0.98	0.98
1D-CNN	0.99	0.99	0.99	0.99

V. DISCUSSION

A. Comparative Analysis of the Classification Algorithms

From the results presented in Tables III to V, it is clear that 1D CNN deep learning model resulted in the highest F1 score of 0.99. In hate speech detection, word sequences are more important than single words. CNN specializes in unsupervised extraction of features from sequences of data and learns by creating more abstract representations of the data in the deeper layers. Hence, our proposed 1D CNN model yielded the best performance compared to all the other ML algorithms.

The second-best results (0.98 F1 score) came from the Bi LSTM and XGBoost (Meta classifier) models. The former is a deep learning model that also extracts knowledge from sequences of data, while the latter is an ensemble model using RF, SVM, GNB, and DT as base classifiers. The XGBoost model attained an F1 score of 0.97 using TF-IDF. From Fig. 2, the LR model had the next best F1 score of 0.96, followed by SVM with an F1 score of 0.87. Both RF and DT each had an F1 score of 0.86.

B. Intra-class and Inter-class Performance Analyses

A more in-depth analysis of the results obtained for each of the three classes -hate, offensive, and neutral- is given as follows.

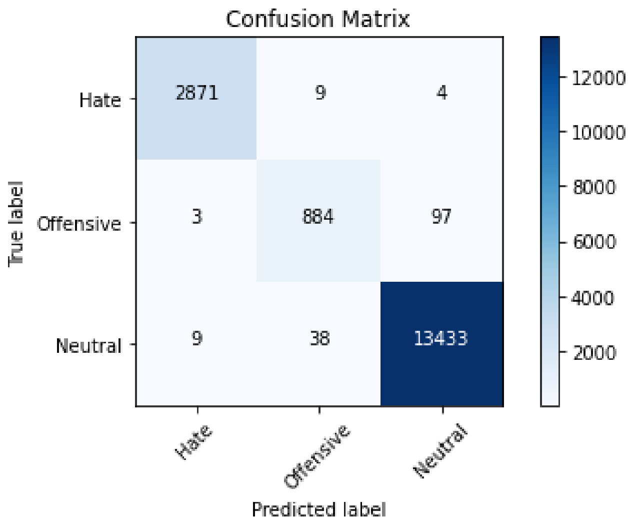

Fig. 3. Confusion Matrix of 1D-CNN

For SVM, the F1 scores for the three classes are as follows: hate-0.88, offensive-0.89, neutral-0.85. For Decision Tree, the F1 score for the three classes are as follows: hate-0.87, offensive-0.87, neutral-0.84. For Random Forest, the F1 scores for the three classes are: hate-0.86, offensive-0.88, neutral-0.84. For Logistic Regression the F1 scores were as follows: hate-0.97, offensive-0.95, neutral-0.97. From these results, it is evident that the class with the least F1-score for most models is the neutral class. On the other hand, the class with the highest F1 score for most models is the offensive class.

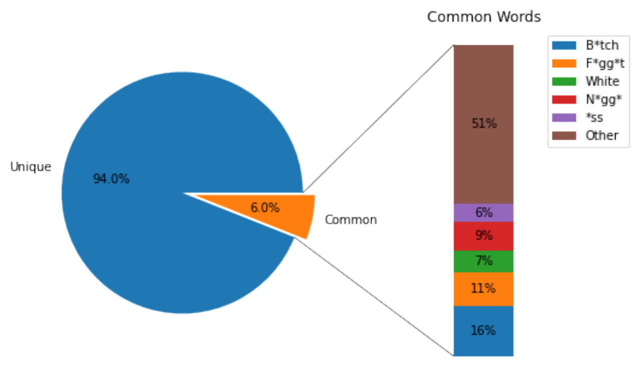

Fig. 4. Hate vs Offensive: between the hate and offensive classes, 6% of the words are common to both classes out of the total number of unique words. From the common 6%, a further breakdown of the prevalence of the overlapping words is shown.

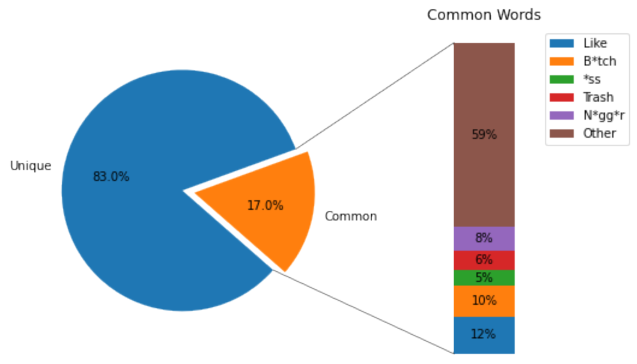

Fig. 5. Hate vs Neutral : between the hate and neutral classes, 17% of the words are common to both classes out of the total number of unique words. From the common 17%, a further breakdown of the prevalence of the overlapping words is shown.

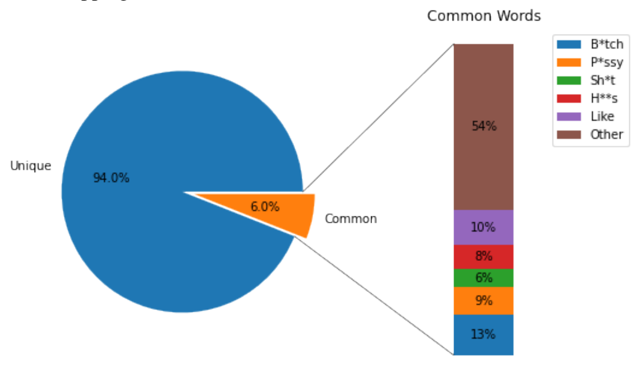

Fig. 6. Neutral vs Offensive: between the neutral and offensive classes, 6% of the words are common to both classes out of the total number of unique words. From the common 6%, a further breakdown of the prevalence of the overlapping words is shown

Fig 4 it can be seen that 6% of the words in hate and offensive classes overlap. Out of this 6%, the most frequently occurring words are B*tch, F*gg*t, White, N*gg* and *ss. From Fig 5, 17% of the words in hate and neutral classes overlap. Out of this 17%, the most frequently occurring words are Like, B*tch, *ss, Trash, and N*gg*r. From Fig 6, between neutral and offensive classes, 6% of the words are common. A further breakdown shows that the most frequent of the common words were B*tch, P*ssy, Sh*t, H**s, Like.

Fig 7 shows the word cloud (or word collage) for the hate, offensive and neutral classes. The word cloud projects all the unique words in a document into a single image with the sizes of the word being proportional to the frequency of occurrence, so that the most frequently occurring word is displayed with the largest size [26]. Fig 5 gives a clear visualization of the co-occurrence of some words in both hate and offensive classes. These figures highlight the difficulty that may arise in distinguishing between tweets from the three different classes.

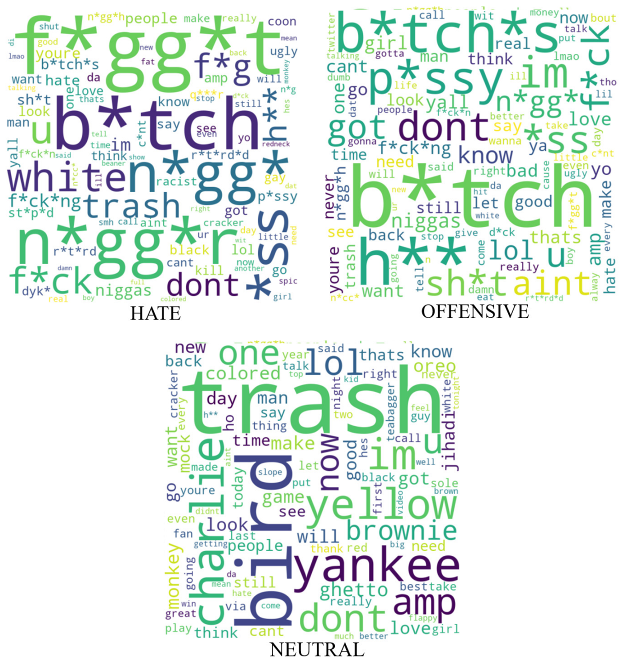

Fig. 7. Word cloud for Hate, Offensive and Neutral

Fig 4 shows that the presence of some offensive words in both offensive and hate classes could make it difficult to distinguish between hate and offensive classes in some instances. This has been discussed in [1], where a high misclassification rate between hate and offensive classes were observed for the ML classifiers studied in the paper. Indeed, an offensive tweet could be classified as hate speech or vice versa due to the presence of these overlapping words which would influence the training of the classifiers. Hence, in order to train more accurate ML classifiers from the dataset, it is important to consider utilizing models/algorithms that can be trained to infer the context which the words are used. The deep learning models (CNN and Bi LSTM) developed in this paper have the ability to capture contextual usage of these words due to having sequential data as the basic input features. This also explains why the proposed 1D CNN model performs better that the traditional ML algorithms.

VI. Comparative Analysis with Previous Work

In Table VI, the best result from [1] obtained from LR is compared to the three best models from this paper. The model which gives the best result in classifying tweets as hate, offensive, and neutral is Convolutional Neural Network(CNN) which offers an F1 score of 0.99. However, the state-of-the-artwork [1] reports an F1 score of 0.90 in classifying tweets. Hence using 1D CNN helped in obtaining an overall improvement of 9%; also, the misclassification rate of hate speech was reduced from 40% to 3%.

TABLE VI
Comparative Analysis with Previous Work

Classifiers	**Precision**	**Accuracy**	**Recall**	**F1 score**
Logistic Regression (Thomas Davidson et al., 2019)	0.91	0.90	0.90	0.90
XGBoost	0.97	0.97	0.97	0.98
Bi-LSTM	0.98	0.98	0.98	0.98
1D-CNN	0.99	0.99	0.99	0.99

VII. Conclusion and Future Work

In this paper we have presented techniques to classify Tweets into three classes, with a particular focus on detecting hate speech. The techniques improve upon previous works by developing machine learning models that can effectively distinguish hate speech from offensive speech. The approach presented in this paper is universally applicable to other types of online communications besides tweets from Twitter. In the paper we proposed a deep learning based hate speech detection system based on 1D CNN and compare this to the another deep neural network approach i.e. Bi-directional LSTM. We also developed the system based on shallow models where popular machine learning algorithms such as SVM, RF, LR, DT and GNB were applied. To improve performance, we further utilized Stacking ensemble learning with XGBoost meta classifier to combine SVM, RF, DT and GNB. For the shallow models, Stacking with XGBoost yielded the best performance but still performed below the deep learning models. The overall best performance was obtained with the 1D CNN model.

On analyzing the inter-class performance, it was found that several words were common between hate and offensive classes, hate and neutral, as well as neutral and offensive classes. These overlapping words are bound to adversely affect classification performance of trained models. In this regard, contextual usage of words become important in enabling high performance classification. Compared to other learning models, the 1D CNN model has the ability to learn from long word sequences where context can more easily be inferred. Also, with BiLSTM, forward and backward dependencies can be captured. This could explain why both deep learning models performed better than the shallow models and the ensemble model. For future work, we would consider combining CNN with Bi-directional LSTM so as to leverage their advantages for improved performance.

References

[1] Davidson, Thomas, Dana Warmsley, Michael Macy, and Ingmar Weber. "Automated hate speech detection and the problem of offensive language." arXiv preprint arXiv:1703.04009, 2017.

[2] Gröndahl, Tommi, Luca Pajola, Mika Juuti, Mauro Conti, and N. Asokan. "All You Need is" Love" Evading Hate Speech Detection." In Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security, pp. 2-12, 2018.

[3] Saleem, Haji Mohammad, Kelly P. Dillon, Susan Benesch, and Derek Ruths. "A web of hate: Tackling hateful speech in online social spaces." arXiv preprint arXiv:1709.10159, 2017.

[4] Singh, Th Bihari, Rakesh Mohanty, Mary Haobam Lalrhiatpuia, and Mohit Saini. "Aggression and Violent Behaviour: A Critical Review.", 2014.

[5] Nouh, Mariam, RC Jason Nurse, and Michael Goldsmith. "Understanding the radical mind: Identifying signals to detect extremist content on twitter." In 2019 IEEE International Conference on Intelligence and Security Informatics (ISI), pp. 98-103. IEEE, 2019.

[6] Ouyang, Xi, Pan Zhou, Cheng Hua Li, and Lijun Liu. "Sentiment analysis using convolutional neural network." In 2015 IEEE international conference on computer and information technology; ubiquitous computing and communications; dependable, autonomic and secure computing; pervasive intelligence and computing, pp. 2359-2364. IEEE, 2015.

[7] Xu, Guixian, Yueting Meng, Xiaoyu Qiu, Ziheng Yu, and Xu Wu. "Sentiment analysis of comment texts based on BiLSTM." Ieee Access 7 (2019): 51522-51532, 2019.

[8] Luu, Son T., Hung P. Nguyen, Kiet Van Nguyen, and Ngan Luu-Thuy Nguyen. "Comparison Between Traditional Machine Learning Models And Neural Network Models For Vietnamese Hate Speech Detection." arXiv preprint arXiv:2002.00759, 2020.

[9] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pp. 785-794, 2016.

[10] Nielsen, Didrik. "Tree boosting with xgboost-why does xgboost win" every" machine learning competition?." Master's thesis, NTNU, 2016.

[11] Bhargava, Kunal, and Rahul Katarya. "An improved lexicon using logistic regression for sentiment analysis." In 2017 International Conference on Computing and Communication Technologies for Smart Nation (IC3TSN), pp. 332-337. IEEE, 2017.

[12] Jotheeswaran, Jeevanandam, and S. Koteeswaran. "Feature selection using random forest method for sentiment analysis." Indian Journal of Science and Technology 9, no. 3: 1-7, 2016.

[13] Malmasi, Shervin, and Marcos Zampieri. "Detecting hate speech in social media." arXiv preprint arXiv:1712.06427, 2017.

[14] Jianqiang, Zhao. "Pre-processing boosting Twitter sentiment analysis?." In 2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity), pp. 748-753. IEEE, 2015.

[15] Founta, Antigoni-Maria, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. "Large scale crowdsourcing and characterization of twitter abusive behavior." arXiv preprint arXiv:1802.00393, 2018.

[16] Gaydhani, Aditya, Vikrant Doma, Shrikant Kendre, and Laxmi Bhagwat. "Detecting hate speech and offensive language on twitter using machine learning: An n-gram and tfidf based approach." arXiv preprint arXiv:1809.08651, 2018.

[17] Watanabe, Hajime, Mondher Bouazizi, and Tomoaki Ohtsuki. "Hate speech on twitter: A pragmatic approach to collect hateful and offensive expressions and perform hate speech detection." IEEE access 6 : 13825-13835, 2018.

[18] Agarap, Abien Fred. "Deep learning using rectified linear units (relu)." arXiv preprint arXiv:1803.08375, 2018.

[19] Jianqiang, Zhao, and Gui Xiaolin. "Comparison research on text pre-processing methods on twitter sentiment analysis." IEEE Access 5: 2870-2879, 2017.

[20] Rana, Shweta, and Archana Singh. "Comparative analysis of sentiment orientation using SVM and Naive Bayes techniques." In 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), pp. 106-111. IEEE, 2016.

[21] Brown, Alexander. "What is hate speech? Part 1: The myth of hate." Law and Philosophy 36, no. 4 : 419-468, 2017.

[22] Van Der Maaten, Laurens, Eric Postma, and Jaap Van den Herik. "Dimensionality reduction: a comparative." J Mach Learn Res 10, no. 66-71 : 13, 2009.

[23] Yoon, Joosung, and Hyeoncheol Kim. "Multi-channel lexicon integrated CNN-BiLSTM models for sentiment analysis." In Proceedings of the 29th conference on computational linguistics and speech processing (ROCLING 2017), pp. 244-253, 2017.

[24] Hosmer Jr, David W., Stanley Lemeshow, and Rodney X. Sturdivant. Applied logistic regression. Vol. 398. John Wiley Sons, 2013.

[25] Ketkar, Nikhil. "Introduction to keras." In Deep learning with Python, pp. 97-111. Apress, Berkeley, CA, 2017.

[26] Lohmann, Steffen, Florian Heimerl, Fabian Bopp, Michael Burch, and Thomas Ertl. "Concentri cloud: Word cloud visualization for multiple text documents." In 2015 19th International Conference on Information Visualisation, pp. 114-120. IEEE, 2015.

[27] Maitra, Saikat, and Jun Yan. "Principle component analysis and partial least squares: Two dimension reduction techniques for regression." Applying Multivariate Statistical Models 79 : 79-90, 2008

[28] Lee, Younghun, Seunghyun Yoon, and Kyomin Jung. "Comparative studies of detecting abusive language on twitter." arXiv preprint arXiv:1808.10245, 2018.

[29] Fortuna, Paula, and Sérgio Nunes. "A survey on automatic detection of hate speech in text." ACM Computing Surveys (CSUR) 51, no. 4 : 1-30, 2018.

[30] Unsvåg, Elise Fehn, and Björn Gambäck. "The effects of user features on twitter hate speech detection." In Proceedings of the 2nd workshop on abusive language online (ALW2), pp. 75-85, 2018.