Twitter Hate Speech Recognition

Karthik Akshay Kamalesh
Student Id:1140893
Department of Computer Science
Lakehead University
Thunder Bay, Canada
kamaleshk@lakeheadu.ca

Ajay Ramasubramanian
Student Id:1153753
Department of Computer Science
Lakehead University
Thunder Bay, Canada
aramasub@lakeheadu.ca

***Abstract*—Twitter is one of the most important social networking services which is used to post and send messages known as "tweets" by the users. Other users on a global scale can read these tweets. Every day, people from various educational backgrounds and cultures express their perspectives on numerous parts of life. As the use of social media grows, everyone appears to believe that they may say or post whatever they want. As a result, there have been conflicts among people. This results in hate speech, which uses aggressive, violent or offensive language based on race or ethnic group(racism) or gender (sexism). Therefore, automated classification can be done to detect hate speech from Twitter. In this approach, we use various Machine learning models such as Knn, Naive Bayes, Logistic regression, SVM, CNN to compare the accuracy and classify whether the tweet is hateful or not.**

I. Introduction

According to the report, there are around 250 million users on Twitter, and 500 million tweets are posted each day. People around the world share their ideas and give their opinion. Owing to the differences in origins, cultures, and beliefs, many people use angry and abusive words while conversing with others from various backgrounds. This hate speech cannot be controlled using a manual detection process because it is tedious and time-consuming. So, Machine learning models are used to detect the hate words and classify them. We will build a machine learning model that can classify a tweet into two classes: hate speech or not hate speech. Different models such as KNN, Naive Bayes, Logistic regression, SVM, CNN are used to classify and compare the accuracy.

We will be using different Vectorizing techniques to extract the features to work on and different Machine learning techniques to perform classification. To evaluate the performance of all the classifiers, we use K-fold Cross-validation on our dataset and evaluation metrics like Accuracy, Precision, Recall, and F1 score. To provide detailed analysis, we use Confusion matrix.

II. Literature Review

A similar study was conducted by Fatahillah [1] on analyzing Twitter hate speech detection. The dataset they used is in the Indonesian language, which contains only 200 tweets. They used Naive Bayes classifier to classify tweets, and they got around 93% accuracy on the training sample. The problem with this paper is that it has fewer datasets for training and no testing dataset available..

Pariyani [2] used the tf-idf and bag of words algorithms. To categorize hate speech in tweets, they used machine learning methods such as SVM, Logistic Regression, and Random Forest. Based on their findings, they can infer that utilizing data without preprocessing and machine learning models with default settings, Random Forest with a bag of words performs best with 0.6580 F1 Score and 0.9629 Accuracy Score. However, as previously stated, getting the best accuracy is insufficient when dealing with an unbalanced class dataset. Therefore, F1 score was deployed, which is fairly low for data without preprocessing. To enhance this, they made use of certain preprocessing processes as well as grid search to find the optimal parameter for the machine learning model. With 0.7488 F1 Score and 0.9668 Accuracy Score after preprocessing and utilizing grid search, SVM with Tf-IDF offers the greatest result with 0.7488 F1 Score and 0.9668 Accuracy Score.

The use of deep neural network architectures for the task of hate speech identification was examined by Badjatiya [3]. They were discovered to surpass existing approaches by a large margin. When deep neural network models like Glove were mixed with gradient boosted Neural Networks, the best accuracy values were obtained. It was discovered that CNN paired with glove embedding provided the highest level of accuracy.

III. Methodology and Justifications

We chose the Machine learning algorithms that delivered the best results in earlier research after analysing numerous articles relevant to Hate speech identification and also implemented them using Deep Neural networks algorithm. Finally, for all of these methods, we will present a comparison and analysis. Going forward,we will be seeing the Architecture of our proposed system.

A. Dataset

The Dataset is collected from the open-source Kaggle (Detecting hatred tweets, provided by Analytics Vidhya). The dataset contains two subsets: train data which contains 46745 records, and test dataset which contains 10000 records. The training data is classified as 0 and 1, with 0 indicating a non-hateful tweet and 1 indicating a hateful tweet.

B. Dataset Preprocessing

Data may contain useless, missing, redundant, or inconsistent data that cannot provide greater accuracy, hence lowering the model's performance. It is a critical step that must be completed before sending the data to the machine learning model.
The preprocessing stage are:

1) Tokenizing: Tokenizing is the process of breaking down a statement into words. This is used to expand our dataset's vocabulary. This vocabulary is used to represent each tweet in our dataset, and it is determined by the algorithm we use, such as TD-IDF or count vectorizer.

2) Stop words: Stop words are terms that have little meaning or are unnecessary in most phrases, such as a,the,of. As a result, these stop words must be removed or else misclassification will occur.

3) Stemming: Stemming is the process of removing a word's prefix or suffix in order to create a similar term in a common form. For example, if we disregard the tense, the words processing, process, and processed all have the same meaning, hence it is necessary to convert all of these words into a similar form. Porter stemmer is used here for this well-known English stemmer.

4) Lemmatization: Lemmatization considers the context and transforms the term into its meaningful base form, known as a Lemma. The same word might have several different Lemmas at times. Lemmatization, for example, would properly identify the base form of 'caring' as 'care,' but stemming would remove the 'ing' element and change it to car.

C. Feature Extraction

To train a machine learning model, it is critical to give it with a list of attributes that are more valuable than others for prediction. This process of extracting characteristics is known as feature extraction. For this project, we use both count vectorizer and TF-IDF vectorizer from tweets, which are then fed into a machine learning classifier to categorize tweets into two groups.

1) Count Vectorizer: CountVectorizer tokenizes (divides sentences into words) the text while also performing basic preprocessing. It eliminates all punctuation marks and lowercases all of the text. A vocabulary of recognized words is developed, which will subsequently be utilized to encode unknown material. An encoded vector with the complete vocabulary's length and an integer count for the number of times each word appears in the text is returned.

2) TF-IDF Vectorizer: TFIDF (term frequency-inverse document frequency) evaluates the importance of a word in a collection or corpus to a document. Term Frequency (TF) is a score based on the frequency with which the term appears in the present document.

$$\text{TF} = \frac{\text{(Total number of times word t appears in a document)}}{\text{(Total number of terms in the paper)}}$$

The inverse document frequency (IDF) scores the rare of a term across the document. It is calculated as follows:

$$\text{IDF} = \log \frac{\text{(total number of documents)}}{\text{(number of documents containing word t)}}$$

$$\text{TF - IDF} = \text{TF} * \text{IDF}$$

Scores are weighing where all the words are not equally important. The scores have the effect of emphasizing distinct words (containing relevant information) in a particular document. Thus, the IDF of a rare term is likely to be high, whereas the IDF of a frequent term is likely to be low.

3) Glove: For the Convolutional neural network, both the count vectorizer and TF-IDF vectorizer are insufficient, so we use Glove's vectorizing technique. The gloVe is a weighted least-squares model with a log-bilinear objective. Glove's training goal is to learn word vectors with a dot product equal to the words' co-occurrence probability logarithm. This goal correlates (the logarithm of) ratios of co-occurrence probabilities with vector differences in the word vector space since the logarithm of a ratio equals the difference of logarithms. This information is also represented as vector differences since these ratios can contain some type of meaning. As a result, the generated word vectors are excellent at word analogy tasks.

D. Train and Test Split

1) K fold cross validation: The training set is divided into k subsets .The following procedure is followed for each of the k "folds":

a) A model is trained using the folds as training data.
b) The model is then validated using the remaining data (i.e., it is used as a test set to compute a performance measure such as accuracy).

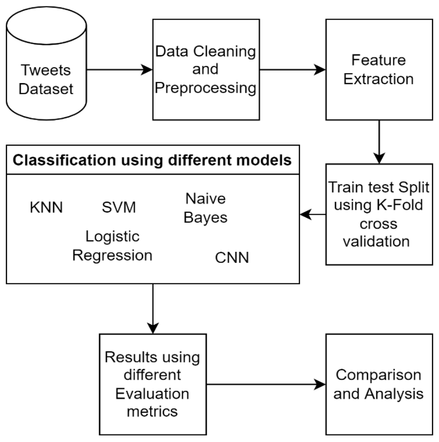
Tweets Dataset
Data Cleaning and Preprocessing
Feature Extraction
Train test Split using K-Fold cross validation
Classification using different models
KNN
SVM
Naive Bayes
Logistic Regression
CNN
Results using different Evaluation metrics
Comparison and Analysis

Fig. 1. System Overview Diagram.

The average of the values computed in the loop becomes the performance metric supplied by k-fold cross-validation. This method is computationally expensive, but it does not waste a lot of data (unlike setting an arbitrary validation set), which is a big plus in applications like inverse inference when the number of samples is minimal.

E. Models

Predictive modelling is the process of creating a model that can generate predictions. In order to produce such predictions, such a model often contains a machine learning algorithm that learns certain attributes from a training dataset. Historical data is utilised to construct and refine a model, which is then used to new data to provide predictions. Scikit-learn is a free Python machine learning package. It comprises a variety of algorithms [7] [9].

1) K-nearest neighbour: KNN is one of the most basic non-parametric (no assumptions) classifiers (KNN). It is a lazy learning algorithm that does not involve any type of learning. KNN assigns a new observation to the class with the most votes based on k-nearest neighbours. In this stage, a Euclidean-like distance is applied. To identify the best number of k-values, a cross-validation approach can be utilized. The KNN technique is a sort of passive learning that is easy to apply and is especially well suited to the quick processing of large amounts of input with high flux. Furthermore, the KNN method may be enhanced by substituting a kernel function for the standard Euclidean distance.

2) Support Vector Machine: SVM translates lower-dimensional input into higher-dimensional data using nonlinear or linear mapping. It looks for the linear optimal dividing hyperplane within this new dimension to distinguish the tuples between the sets [4]. A hyperplane always discriminates information from two array scans for sufficient nonlinear scaling to a sufficiently high dimension. With the use of support vectors, the SVM locates this hyperplane. The instances that are closer to the margins are called support vectors. There is no limit to how many separating lines can be drawn here. The goal is to classify the "highest" one with the fewest errors on previously encountered tuples [5].

3) Logistic Regression: A widely used Supervised Classification Algorithm is Logistic Regression. It allows us to predict whether or not an observation belongs to a given class using a simple method. In its basic form, Logistic Regression is a Binary Classifier. As a result, the target vector can only have two values. In the formula for the Logistic Regression Algorithm (also known as a Sigmoid Function), we have a Linear Model that is integrated into a Logistic Function:

$$\text{P} = \frac{1}{1 + e^{-m+nX}}$$

where P is the probability of the text belonging to the considered class, m and n the parameters to be learned, and e denotes Euler's Number.

4) Multinomial Naive Bayes: We divide the edit into two categories (whether it is mean tweets or not) by calculating the likelihood as a posterior probability and assigning it to the appropriate class, assuming that each change is independent of the others (the Naive Bayes Assumption). In the Naive Bayes approach, each class is assumed to be independent of the others. It is based on Bayes' theorem, which allows for the calculation of explicit probabilities for any hypothesis.

$$\text{P(h/T) = P(h) * P(T—h) / P(T)}$$

where P(h) denotes prior probability of hypothesis h, P(T) is prior probability of training data T, P(T—h) is probability of T given h, and P(h—T) is probability of h given T.

5) Convolution Neural Network (CNN): CNN is constituted of three layers: convolution layer, pooling layer and fully connected layer.There are 192 filters in the convolutional layer, with a window size of 2. The filters in this layer slide across the embedded matrix. The filter's width will be the same as the embedded matrix's width, but the filter's height may vary. The ReLu function is then employed as an activation function. Padding keeps the input and output lengths consistent. The output is sent to the Maxpooling layer, which reduces the output's dimensionality. The feature extraction is done by these two layers. The features are then communicated to the fully linked layer, which performs the classification. The result will be normalised using the Softmax function.

F. Evaluation Metrics

The ultimate purpose of employing evaluation metrics is to determine how well a machine learning model will perform when dealing with unknown data [6] [8].

1) Accuracy: The number of correct predictions divided by the total number of input samples is known as accuracy. It only works when there are an equal number of samples in each class.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total predictions}}$$

2) Precision: The number of correct positive results divided by the number of positive results predicted by the classifier is what is known as precision.

$$\text{Precision} = \frac{\text{TruePositives}}{\text{TruePostives + FalsePositives}}$$

3) Recall: The number of correct positive results divided by the total number of relevant samples (all samples that should have been detected as positive) is known as recall.

$$\text{Recall} = \frac{\text{TruePositives}}{\text{TruePostives + FalseNegatives}}$$

4) f1-score: The weighted average of Precision and Recall is the F1 Score. As a result, this score takes into account both false positives and false negatives. When the data is unbalanced, this method is employed.

$$\text{F1Score} = 2 * \frac{(\text{Recall * Precision})}{(\text{Recall + Precision})}$$

IV. Experiments

In this experiment, we are considering the ”tweets” column to see if it can detect mean tweets. Missing values are checked in the new dataset, but none are found. Lowercasing was used to remove special characters, punctuation, hyperlinks, and the HTML tag from the dataset. We also tried using lemmatization and stemming to further preprocess the data but it reduces the performance of the learning models. Using Five-fold (80 -20 split) Cross-Validation, the dataset is separated into train and validation data. On the training data, the TFIDF with n-gram with n=(1,3) is utilized for vectorization. The models mentioned are trained on training data and evaluated on validation data using the evaluation metrics to get the training accuracy and then finally it is evaluated on the test dataset to get the testing accuracy. We use the sklearn library for all the models except CNN. With alpha 1.0, the NB uses the default laplace smoothing values. The SVC uses a linear kernel with a regularisation parameter of 1.0 by default.The CNN model is built with the Python tensorflow module. To avoid overfitting, the model includes three convolution layers with relu and Dropout. With a learning rate of 0.0001, the Adam optimizer is utilised. The softmax activation function is used in the output layer.

V. Results

Using different vectorizers, the output of the Machine Learning models is compared based on the evaluation metrics mentioned in section III F. Table 1 and Table 2 show the experimental results of 5 Fold Cross-Validation over training data. It is seen that SVM and LR performed better than the other models with 94 percent accuracy when the count vectorizer was used and SVM dominated in the case of the TF-IDF vectorizer.
Table 3 and Table 4 show the experimental results of the testing data. We can see that SVM gets 94 percent accuracy in both the vectorizing technique but whereas Logistic Regression performs better-using Count vectorizer and gives an overall accuracy of 95 percent. Table 5 shows the results of the CNN technique using Glove and it gives an overall of 93 percent accuracy.

TABLE I
TRAIN RESULTS FOR COUNT VECTORIZER USING 5-FOLD CROSS VALIDATION

ML Model	Fold1	Fold2	Fold3	Fold4	Fold5	Avg
KNN	0.81	0.88	0.88	0.89	0.89	0.87
SVM	0.94	0.93	0.93	0.93	0.94	0.94
NB	0.91	0.91	0.91	0.92	0.92	0.91
LR	0.94	0.94	0.94	0.94	0.94	0.94

TABLE II
TRAIN RESULTS FOR TF-IDF VECTORIZER USING 5-FOLD CROSS VALIDATION

ML Model	Fold1	Fold2	Fold3	Fold4	Fold5	Avg
KNN	0.67	0.77	0.76	0.77	0.77	0.75
SVM	0.94	0.93	0.93	0.94	0.94	0.94
NB	0.92	0.92	0.91	0.92	0.92	0.92
LR	0.93	0.93	0.92	0.93	0.93	0.92

TABLE III
TEST RESULTS USING COUNT VECTORIZER

ML Model	Accuracy	Precision	Recall	f1-score
KNN	0.89	0.90	0.88	0.89
SVM	0.94	0.95	0.94	0.94
Multinomial Naive Bayes	0.92	0.91	0.92	0.91
Logistic Regression	0.95	0.95	0.94	0.94

TABLE IV
TEST RESULTS USING TF-IDF VECTORIZER

ML Model	Accuracy	Precision	Recall	f1-score
KNN	0.80	0.84	0.77	0.78
SVM	0.94	0.95	0.94	0.94
Multinomial Naive Bayes	0.92	0.92	0.92	0.92
Logistic Regression	0.93	0.94	0.92	0.93

TABLE V
TEST RESULTS FOR CNN

ML Model	epoch1	epoch2	epoch3	epoch4	epoch5	Final
CNN	0.92	0.92	0.93	0.93	0.93	0.93

VI. Discussion

Considering we employed both Machine learning and Deep Neural Networks, it is important to note that they use different Vectorization techniques. Machine learning models use Count and TF-IDF Vectorizers, while CNN uses GLOVE. The most generally used vectorizers for Machine learning algorithms are the count vectorizer and the TF-IDF vectorizer, and we choose to investigate both to discover the optimal one for our needs. We chose GLOVE for CNN since it has been shown to outperform other Word embeddings such as Word2Vec, Bert, etc [10] [11]. Using 5-fold cross-validation, we discovered training accuracy for our models. Following that, we also determined the test accuracy using distinct test data we had. Rather than only showing the test accuracy, we discovered both accuracies to establish the results in a better way. It was also discovered that using more techniques like stemming and lemmatization during Data Pre-processing for Machine Learning models had a detrimental impact on their performance.

VII. Conclusion

We proposed a solution for the automatic detection of hate tweets using both machine learning and Deep Neural Networks. The tweets were classified into one of two categories using different machine learning classifiers like KNN, SVM, Logistic Regression, Multinomial Naive Bayes and Convolutional Neural Network. We used 5 fold cross-validation on our training data and testing data is used for accuracy. We compared all the model's accuracy and Logistic Regression got the highest accuracy of 95 percent using count vectorizer whereas SVM got the accuracy of 94 percent in both Count and TF-IDF features. In future, a dataset with a larger number of tweets could be used because it contains more vocabulary and substantial material, It's challenging to find real-world datasets with accurately specified labels. As a result, we may look into semi-supervised and unsupervised algorithms for detecting mean tweets.

VIII. Acknowledgements (Team Member Contributions)

I would like to say thanks to Dr. Thiago Eustaquio Alves de Oliveira, Professor of Pattern Recognition Computer Science, for his advice and giving us an opportunity to prepare a project on the Twitter Hate Speech Recognition.

1) Ajay Ramasubramanian: Worked on selecting the dataset, literature review, Implementation of Machine Learning Models, project presentation, final report.

2) Karthik Akshay Kamalesh: Worked on selecting the dataset, literature review, Implementation of preprocessing steps and CNN, Final report and formatting in latex.

References

[1] N. R. Fatahillah, P. Suryati, and C. Haryawan, "Implementation of naive bayes classifier algorithm on social media (twitter) to the teaching of Indonesian hate speech," 2017 International Conference on Sustainable Information Engineering and Technology (SIET), 2017.

[2] B. Pariyani, K. Shah, M. Shah, T. Vyas, and S. Degadwala, "Hate speech detection in Twitter using Natural Language Processing," 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), 2021.

[3] Badjatiya, Pinkesh, et al. "Deep Learning for Hate Speech Detection in Tweets." Proceedings of the 26th International Conference on World Wide Web Companion - WWW '17 Companion, 2017, https://doi.org/10.1145/3041021.3054223.

[4] Shimaa M Abd El-Salam, Mohamed M Ezz, Somaya Hashem, Wafaa Elakel, Rabab Salama, Hesham ElMakhzangy, and Mahmoud ElHefnawi. Performance of machine learning approaches on predict ion of esophageal varices for egyptian chronic hepat it is c pat ients. Informat ics in Medicine Unlocked, 17:100267, 2019.

[5] Fabio Del Vigna12, Andrea Cimino23, Felice Dell'Orlet ta, Marinella Pet rocchi, and Maurizio Tesconi. Hate me, hate me not: Hate speech detection on facebook. In Proceedings of the First Italian Conference on Cybersecurity (ITASEC17), pages 86–95, 2017.

[6] H. Şahi, Y. Kılıç and R. B. Sağlam, "Automated Detection of Hate Speech towards Woman on Twitter," 2018 3rd International Conference on Computer Science and Engineering (UBMK), 2018, pp. 533-536, doi: 10.1109/UBMK.2018.8566304.

[7] 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," in IEEE Access, vol. 6, pp. 13825-13835, 2018, doi: 10.1109/ACCESS.2018.2806394.

[8] S. Masud et al., "Hate is the New Infodemic: A Topic-aware Modeling of Hate Speech Diffusion on Twitter," 2021 IEEE 37th International Conference on Data Engineering (ICDE), 2021, pp. 504-515, doi: 10.1109/ICDE51399.2021.00050.

[9] N. D. Srivastava, Sakshi and Y. Sharma, "Combating Online Hate: A Comparative Study on Identification of Hate Speech and Offensive Content in Social Media Text," 2020 IEEE Recent Advances in Intelligent Computational Systems (RAICS), 2020, pp. 47-52, doi: 10.1109/RAICS51191.2020.9332469.

[10] A. Zouzou and I. E. Azami, "Text sentiment analysis with CNN GRU model using GloVe," 2021 Fifth International Conference On Intelligent Computing in Data Sciences (ICDS), 2021, pp. 1-5, doi: 10.1109/ICDS53782.2021.9626715.

[11] E. Ayodele et al., "Grasp Classification With Weft Knit Data Glove Using a Convolutional Neural Network," in IEEE Sensors Journal, vol. 21, no. 9, pp. 10824-10833, 1 May1, 2021, doi: 10.1109/JSEN.2021.3059028.