Detecting Hate Speech on Twitter Network Using Ensemble Machine Learning

Raymond T Mutanga[1], Nalindren Naicker[2], Oludayo O Olugbara[3]
Department of Information Technology, Durban University of Technology, Durban, South Africa[1, 3]
Department of Information Systems, Durban University of Technology, Durban, South Africa[2]

***Abstract*—Twitter is habitually exploited now-a-days to propagate torrents of hate speeches, misogynistic, and misandry tweets that are written in slang. Machine learning methods have been explored in manifold studies to address the inherent challenges of hate speech detection in online spaces. Nevertheless, language has subtleties that can make it stiff for machines to adequately comprehend and disambiguate the semantics of words that are heavily dependent on the usage context. Deep learning methods have demonstrated promising results for automatic hate speech detection, but they require a significant volume of training data. Classical machine learning methods suffer from the innate problem of high variance that in turn affects the performance of hate speech detection systems. This study presents a voting ensemble machine learning method that harnesses the strengths of logistic regression, decision trees, and support vector machines for the automatic detection of hate speech in tweets. The method was evaluated against ten widely used machine learning methods on two standard tweet data sets using the famous performance evaluation metrics to achieve an improved average F1-score of 94.2%.**

Keywords*—Classical learning; deep learning; ensemble learning; hate speech; social media; twitter network; voting ensemble

I. Introduction

Twitter is a popular microblogging social networking service platform invented for the central purpose of connecting geographically dispersed people to seamlessly collaborate, communicate, microblog, network, socialise and share information. It is recently used for fostering business entities as a way of reaching out to a throng of clients and retaining them. However, despite its popularity and usefulness, there is a rapid rise in its usage for propagating hateful speeches and aiding torrents of invectives against innocent people. The level of anonymity of the accounts granted by social media networking platforms has made them havens for promoting hateful, discriminating, and vulgar speeches. Considering that Twitter generates a high volume of tweets daily, hate speech propagation should be curbed to avoid people deactivating their accounts and quitting the network platform. Human annotators are currently employed by Twitter and Facebook to delete nocuous tweets perceived to be hateful in curtailing the excessiveness of hate speech propaganda on social media platforms. In addition, the public is requested to report nocuous tweets to the service providers. However, these manual methods are laborious, sentimental, and susceptible to a subjective human judgement of what truly constitutes hate speech [1].

The repercussions of hateful tweets, limitation of legislation, and ineffectiveness of human annotators have created the necessity to apply machine learning methods for automatic hate speech detection. Classical and deep machine learning methods can be employed to automatically detect hate speech in text documents. The classical machine learning methods mostly use the vector-based representation of handcrafted features, which is time-consuming to craft and is typically incomplete [2]. Moreover, the vector space model often fails to effectively capture the semantic and syntactic representations of text. Deep learning methods generally allow for more accurate prediction through auto-generation of suitable feature representations. Recurrent neural networks (RNN) are deep learning methods that can preserve the sequence information over time. The contextual information can be considered in the task of object classification using deep learning methods [3]. However, deep learning requires a large chunk of data to obtain accurate results. Furthermore, the end-to-end mechanism through which deep learning methods make decisions may not be suitable for text processing in the discipline of natural language processing because of the lack of interpretability. This is particularly pertinent to hate speech detection, where a manual appeal process is needed for a system that censors the speech of a person [4].

Research studies in machine learning have evolved to ensemble learning methods that agglutinate multiple learning methods to improve the performance of a detection system. This allows for harnessing the strengths of multiple learning methods and optimisation of classical machine learning methods in an object classification task. In general, ensemble learning methods can be classified appositely into four main categories of bagging, boosting, stacking, and voting [5]. The predictions from many decision trees are combined in a bagging ensemble learning method. Boosting involves correcting the performances of prior classifiers and adding them sequentially to the ensemble. Since every classifier is obliged to fix the errors in the predecessors, boosting is sensitive to outliers which are considered a disadvantage. Learning how to best combine the predictions from several inducers is achieved through a stacking meta-learning method. Like all meta-model ensemble methods, stacking is simply not feasible in many real-world situations because of a lot of reasons [6]. Predicting a class with the most votes by adding the votes of crisp class labels is called a voting ensemble that works by combining the predictions from multiple classifiers. The majority vote in the task of classification is predicted by

summing the prediction for each label, which makes it suited for complex multiclass problems [7].

Different ensemble machine learning methods have been effectively applied to diverse application domains such as speech emotion recognition [8, 9], product image classification [10], and lung cancer prediction [11]. However, it is more challenging to process highly unstructured text documents with the orthodox machine learning methods that are well developed for numerical data processing. Consequently, a voting ensemble machine learning method that agglutinates logistic regression, support vector machines, and decision trees is proposed in this study for hate speech detection in tweets. Logistic regression has shown positive results on binary text classification because of its ability to be easily tuned to accommodate new data. Support vector machines are widely used for many types of classification problems because of their ability to work in high dimensional spaces to address the overfitting logjam. Decision trees have shown promising results in dealing with highly unstructured data because they do not require data scaling.

In general, tweets are short messages, and their meanings are often rife with idioms, onomatopoeias, homophones, phonemes, and acronyms [12]. Hence, the work reported in this paper agglutinates the strengths of logistic regression, support vector machines, and decision trees in a voting ensemble learning method for hate speech detection in tweets. It is envisaged that support vector machines will bring stability to the voting ensemble because it is not influenced by outliers in a data set. The process of carefully choosing and configuring the parameters for an ensemble learning method is still an open area. The parameter configuration in the proposed voting ensemble learning was carefully fine-tuned for optimal performance. This research study is aimed at enhancing the performance of hate speech detection systems using the method of voting ensemble learning and testing its performance against numerous baseline methods.

This paper is compactly structured as follows. In Section II, the related literature on hate speech detection is briefly reviewed. In Section III, the materials and methods of the study are discussed. In Section IV, the experimental results and discussion are explicated. The concluding statements are delineated in Section V of this paper.

II. Related Literature

Hate speech detection is an automated task of determining whether a given piece of text content contains hateful utterances or not. It is a difficult problem in the fields of natural language processing (NLP) and artificial intelligence (AI) for which the classical or deep learning methods experimented. The classical machine learning methods heavily depend on a complex process of feature engineering where features from an input text are rigorously extracted. Deep learning methods eliminate the need for feature engineering by automatically learning features from the input text [7]. There is ongoing research to increase the accuracy of text classification methods owing to the unstructured and complex nature of NLP problems. The review of related literature is planned under the themes of classical learning, deep learning, and ensemble learning as explicated in this section.

A. Classical Learning

The classical machine learning approach uses the established vector-based model such as n-grams and bag of words for text representation, while support vector machine (SVM), decision tree (DT), and logistic regression (LR) are traditionally deployed for text classification. The SVM was originally designed for binary classification tasks [7], but its usage has long been extended to a multiclass classification problem by breaking a given classification problem into several binary sub problems. The binary classification method divides n-dimensional space features into distinct regions that correspond to two specified output classes [13]. Its performance is attributed to the ability to model nonlinear decision boundaries and it is robust against overfitting [14]. DT can achieve a good performance in several classification tasks while producing easily interpretable decisions. The knowledge learned by a DT during the training session is represented in a hierarchical structure that allows for easy comprehension and interpretation by non-experts. LR method uses a probability function or a sigmoid cost function whose output is limited to values between 0 and 1 to make it well suited for binary classification problems. Davidson et al. [15] used a crowd-sourced hate speech lexicon to collect and label tweets containing hate speech. They trained six classical learning methods to distinguish three classes of hate speech as contained in their data set. Their best result was an F1-score of 90.00%.

B. Deep Learning

Deep learning methods learn through a series of interconnected network layers wherein each layer receives input from a prior layer and provides input to a subsequent layer [2]. The raw data in a deep learning text classification task are vectorised to produce the desired input sequence [14]. The size of the input layer is defined by the number of inputs. The additional layers improve the learning capability to obtain a stable output. The output layer provides a result in the form of probabilities of the output classes and has the same number of neurons as the output classes [16]. The long-short term memory (LSTM) can model an ordered sequential input such as textual data [17]. The LSTM was specifically developed to address the vanishing gradient problem faced by the vanilla version of recurrent neural network (RNN) [14] and it has been used in many classification tasks [1, 16, 18, 19]. It has been proven to work well with text data, but it requires a large amount of data for training and validation [17]. Convolution neural network (CNN) uses the pooling technique to minimise the outputs of network layers, but it is prone to high dimensionality in a text processing task. Mutanga et al. [20] explored the use of a transformer method to detect hate speech to obtain the best accuracy of 92.00% and F1-score of 75.00% using DistilBERT.

C. Ensemble Learning

It is promising to harness the strengths of different machine learning methods through the framework of ensemble learning for improving the performance of hate speech detection systems. Popular ensemble learning methods include bagging, boosting, and stacking. Bagging minimises variance by combining the verdicts from different decision trees [21,

22]. It has led to the development of many other decision tree-based ensemble learning methods. The idea behind the bagging ensemble is to create numerous subsets of data from the training sample picked arbitrarily with replacement. Each of the subsets created is used to train its decision trees, resulting in an ensemble of different models. However, the bagging approach does not necessarily lead to improved performance. It can result in performance declination, for example, when a model already has low variance. In addition, empirical evidence has suggested that bagging can push an unstable method towards an optimal performance [23-25]. Conversely, it may lead to a declination in the performance of stable methods. Models are sequentially added to an ensemble in boosting, where each model rectifies the error made by the prior method in the sequence [26, 27]. However, one apparent hiccup of boosting is that it is highly responsive to outliers because each method is required to address errors in the predecessor method. The stacking ensembles are generally used to learn how to best combine predictions from multiple inducers. Stacking ensembles, like all meta-model ensemble learning methods, are not feasible in many real-world applications because they can be expensive to train, deploy and maintain.

There are relatively few studies conducted on hate speech detection using ensemble machine learning methods. In their work, MacAvaney et al. [4] evaluated the efficacy of support vector machines, bidirectional encoder representations from transformers, and an ensemble of neural networks for detecting hate speech. They trained their model on four hate speech data sets to achieve the best F1-score of 91.18% obtained using an ensemble of neural networks on a hate speech tweet data set. Ahluwalia et al. [19] used an ensemble learning method of LR, SVM, random forest (RF), and gradient boosting machine (GBM) to detect English hate speech against women. They trained their model on a data set of binary classes and a data set of multiple classes to achieve the best accuracy of 65.10% for binary classification and an F1-score of 40.60% for multiclass classification. The said works have employed ensemble learning methods for hate speech detection, but it should be noted that none of them has combined logistic regression, decision trees, and support vector machines in an ensemble architecture despite the efficacy shown by the algorithms when used solitarily [4, 15, 19]. The contribution of this paper is the development of a new robust voting ensemble method that harnesses the capabilities of LR, DT, and SVM [14, 15] to address overfitting, accommodate new data, and allow for interpretability of a hate speech detection system.

The review of the related literature has generally indicated that relatively few studies have focused on using ensemble learning for hate speech detection in online spaces. Most of these few studies have reported performance results that require further improvement. The current method based on voting ensemble learning gave the state-of-the-art results of 94.20% accuracy and an F1-score of 94.21% surpassing the results of earlier studies that used the same data set. The results have reflected an improvement over the F1-score of 90.00% reported in [15] and the highest benchmarked accuracy result of 92.00% reported in [20].

III. Materials and Methods

The materials and methods used in this study are lucidly presented in this section based on experimental data sets with baseline methods, and essential steps of the proposed voting ensemble method.

A. Experimental Materials

The publicly available data sets of hate speech offensive (HSO) language and Kaggle were used for experimentation in this study. The HSO data set comprised of 11310 tweets that were labelled as ‘Hate’ or ‘Neutral’ as made available on the GitHub repository [15]. The Kaggle data set is made up of 8778 neutral tweets and 1155 hate tweets. The data set was grossly imbalanced, and it was important to measure the performance of machine learning methods on a smaller data set. Consequently, the data set was reduced programmatically to 2300 tweets to test the performance of the experimental methods on a smaller data set. The balanced version of the data set consisted of 1150 hate tweets and 1150 neutral tweets that were used for experimentation in this study.

The baseline experimental methods and the proposed voting ensemble method were all implemented using the Python programming language. The Keras library was used to implement the deep learning methods, while the scikit-learn Python library was used to implement the baseline classical and ensemble learning methods. Specifically, sklearn.tree, sklearn.linear_model, and SVM submodules were used to implement DT, LR, and SVM respectively. All the baseline ensemble learning methods were implemented using the sklearn.ensemble submodule. The Keras library was used to implement the CNN and LSTM deep learning methods. Several experiments were faithfully conducted on a computer machine running Windows 10 operating system with configuration of Intel (R) Core (TM) i5-8250U CPU @ 1.60GHz (8 CPUs), 1.8GHz, 8 GB RAM, and 500 Gigabytes of a hard disk drive.

B. Proposed Method

The proposed voting ensemble method comprises the phases of pre-processing, feature representation, and feature classification. The essential steps of the pre-processing include the removal of special characters and punctuations, normalisation of hashtags, lowercasing of the characters of the input text, removal of short words, and text tokenisation. The feature representations were based on the widely used bag of words and word embedding. They were applied after pre-processing to convert the raw tweets data into a useful form amenable to machine learning processing. The bag of words representation converts a text document into a fixed-length vector of occurrence of words in the input text and it was used to implement the classical learning methods. The regularity presented by specific keywords has provided a solid foundation for a bag of words representation to focus on specific words in a data set [28]. Since hate speech is generally expressed through largely homogenous words, it is envisaged that a bag of words representation can effectively capture and represent the vocabulary of known hate words such as black, white, Indian, Jews, foreigners, strangers, enemies, and so on.

Word embedding is a more promising text vocabulary representation that is used by deep learning methods to encode meanings of words into a real value vector such that highly similar words are closer in the vector space. It is a foundation for sentence embedding that presents a huge advantage over the bag of words vector model. It can capture word context, syntactic and semantic relationships with words in a text document. Moreover, it eliminates the sparse representation hiccup often associated with the bag of words representation. The word embedding approach follows the distributional hypothesis, where semantically similar words are found in the same context [2]. The word embedding layer for text classification is usually the first data processing layer of a deep learning model and word embedding methods have been demonstrated to perform well in different NLP tasks [29-31]. In this study, word embedding was implemented using the Keras embedding layer of deep learning because of its ability to capture contextual words and syntactic similarities to enhance the interpretation of tweet meanings.

The basic idea behind the proposed method of this study lies in the selection of an optimal bias-variance trade-off. The presence of high variance can lead to the problem of overfitting, while high bias may result in underfitting. Due to the nature of tweets, variance is likely to occur, particularly in fora that focus on a specific type of hate speech. The Islamophobic for instance may express hate speech in largely similar terms that are difficult to detect using a learning method. The proposed voting ensemble method aggregates the decisions from three classical inducers, which are LR, DT, and SVM to obtain accurate classification decisions. Fig. 1 shows the architecture to illustrate the steps of the proposed voting ensemble learning method.

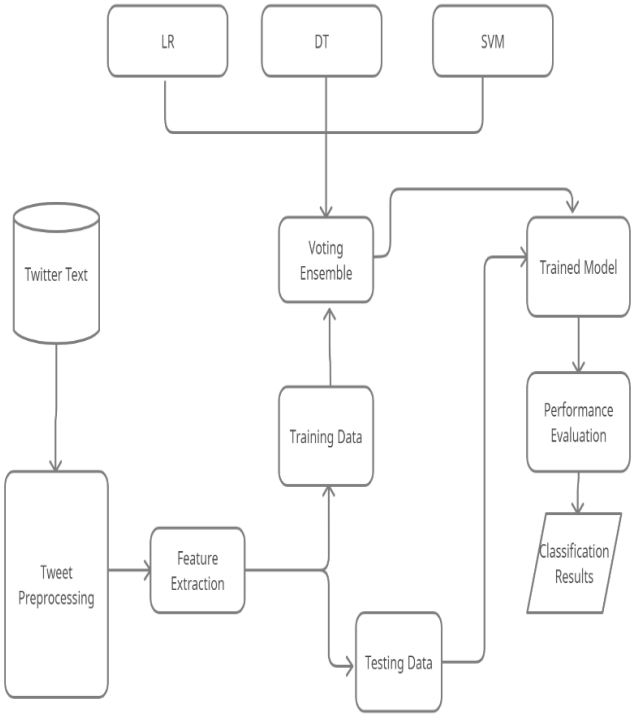

Fig. 1. The Architecture of the Proposed Voting Ensemble Learning Method.

The base inducer of DT is used when the dependent variable is qualitative as in the episode of a text classification task. DT is highly interpretable, fast to train, and works well with decision boundaries [14]. The inclusion of the DT method in the proposed ensemble is based on its appropriateness when dealing with categorical data such as distinguishing hate tweets from innocuous tweets. Earlier studies have investigated the use of DT methods in hate speech detection tasks and recorded satisfactory performance [32]. The important parameters in DT to perform the grid search cross-validation technique are max_depth and random_state. The max_depth parameter determines the depth of a tree. The deeper the tree, the more splits it has, and it captures more information about the data. In our experiments, the max_depth value for optimal searching was 10 trees. The depth parameter is also used as a regularisation scheme to prevent overfitting. This step is crucial in our study because tweets are generally regarded as noisy and highly dimensional. The random_state parameter that controls the random choices for the training sample was set at 42.

The LR inducer attempts to find a probability-based relationship between the independent variable and class label in each data set. It aims to create a probability function that uses features as inputs and returns the probability of that instance belonging to a given class [33, 34]. The LR does not require scaling of input features and it requires comparatively fewer computation resources [14]. The regularisation parameter (L2), and 'squared magnitude' of coefficient as a penalty to the loss function were used for optimisation. The 'fit_intercept' parameter was set to 'True' to incorporate the intercept value to the LR method. The 'Solver' parameter that defines the method to be used in the optimisation problem was set to 'sag' which is compatible with the L2 penalty.

The optimisation of the SVM inducer employed the grid search cross-validation scheme to come up with the best parameters for model fitting. The optimal value for parameter C that defines the tolerance threshold for misclassification was set at 0.1. Moreover, the linear kernel that works on the assumption that input data is linear was applied. Thereafter, the auto deprecated gamma setting, which is the recommended default value was used in conjunction with the linear kernel. The optimal value of the degree parameter was set at 3. The learning rate for the proposed ensemble learner was specified in the Python program before training. The low learning rate specified in Table I was used for preventing the ensemble model from converging to an undesirable optimum [35]. The tolerance setting is a stopping technique that stops the iteration process once the specified value is reached and it affects the training time of a model [36]. The parameters for the inducers and voting ensemble (VE) method are succinctly summarised in Table I.

The results computed by the voting ensemble learning method can be based on either hard or soft voting. The class probability score for each classification method that the current sample belongs to, is considered soft voting [34]. At that point, soft voting criteria determine the class with the highest probability by averaging the individual values of the inducers [37]. Hard voting involves summing the votes for crisp class labels from the other inducers and predicting the class with the

most votes. The class label Y can be decided by the majority voting of each classifier C as in the following example.

$$Y = \text{mode}(c1(x), c2(x), \ldots, cm(x)) \quad (1)$$

If the predictions from c1, c2, and c3 are 'hate', 'neutral', and 'hate', respectively, the final prediction will be 'hate' according to the principle of majority voting. Consequently,

$$Y = mode[hate, neutral, hate] = hate \quad (2)$$

The hard voting scheme is suited for predicting distinct class labels, while soft voting is appropriate for predicting continuous values. This study was based on tweets labelled under distinct categories. Hence, it implies that hard voting is more desirable for this study as compared to soft voting.

TABLE I. CONFIGURATION SETTINGS FOR THE PROPOSED VOTING ENSEMBLE METHOD

Parameter	*DT*	*LR*	*SVM*	*VE*
max_ depth	10			10
learning_ rate				0.4
n_ estimators				3
random_ state	42		None	
C		0.1	0.1	1.0
cache_ size			200	200
degree			3	3
gamma			auto_ deprecated	
kernel			linear	linear
max_ iterations		100	−1	−1
shrinking			True	True
tolerance (tol)		0.0001	0.001	0.001
penalty		L2		
fit_ intercept		True		
solver		sag		

IV. RESULTS AND DISCUSSION

This section presents a discussion of the comparative results of the proposed voting ensemble learning method against ten widely used machine learning methods. The baseline methods are AdaBoost, AdaBoost-DT, Bagging, Bagging-SVM, CNN, DT, LR, LSTM, RF, and SVM. The experimental data sets of Kaggle and HSO were each split into training and testing data in the ratios of 80:20 and 70:30. Although the proposed voting ensemble learning method is comprised of LR, SVM, and DT inducers, each inducer was implemented separately to establish a comparison with the proposed voting ensemble learning method. In addition, other widely used machine learning methods were evaluated against the proposed method. The performances of the learning methods were analysed and discussed in terms of four functional metrics of accuracy, precision, recall, and F1-score. In addition, the performances of the learning methods were evaluated and discussed in terms of non-functional metrics of kappa, hamming loss, Jaccard similarity, and execution time.

A. Accuracy Results

This section presents the analysis of the accuracy of the experimental results of the proposed voting ensemble method along with several baseline methods. The accuracy scores calculated for the two experimental data sets are listed in Table II. It can be observed that accuracy scores computed by the proposed voting ensemble learning method are consistently higher than the scores computed by other learning methods across the two experimental data sets. The proposed ensemble learning method recorded the highest average accuracy score of 94.212% across both data sets, irrespective of the test split. It is worth mentioning that the voting ensemble learning method had the highest accuracy scores across the two data sets under the different train and test splits. Expectedly, all methods performed better with the larger HSO data set as compared to the smaller Kaggle data set, with the proposed voting ensemble learning method giving the highest accuracy score of 96.739% under the bigger data set using the 80:20 train test split. This trend is attributable to the fact that bigger data sets allow methods to learn data patterns more comprehensively during training, thereby impacting overall performance, particularly in the case of deep learning methods, which generally require large data sets to perform well.

TABLE II. ACCURACY SCORES OF LEARNING METHODS USING DIFFERENT TRAIN-TEST SPLITS

Data set	**Kaggle**		**HSO**		
Train: test split	**80:20**	**70:30**	**80:20**	**70:30**	**Average**
AdaBoost	87.887	87.883	91.304	90.870	89.486
AdaBoost-DT	90.539	90.448	93.043	89.855	90.972
Bagging	90.318	90.566	92.174	89.275	90.583
Bagging-SVM	91.468	90.654	95.217	94.493	92.958
CNN	88.240	88.031	95.000	94.493	91.441
Decision Tree	90.097	90.301	91.739	89.275	90.353
Logistic Regression	91.689	91.509	95.435	94.493	93.281
LSTM	91.202	90.890	95.435	94.638	93.041
Random Forest	89.434	90.065	93.478	93.478	91.614
Support Vector Machine	89.788	88.709	92.391	92.464	90.838
Voting Ensemble	**92.042**	**91.834**	**96.739**	**96.232**	**94.212**

Fig. 2 shows the plot of average accuracy scores computed by the learning methods across the experimental data sets to visually illustrate the extent to which one learning method gives better accuracy than another. This result implies that the voting ensemble method can detect all the correct cases better than any other method, while AdaBoost performed worst in this case. The voting ensemble method is therefore the most useful when all classes are equally important while AdaBoost is not useful in this scenario.

B. Precision Results

This section presents the precision scores computed by the proposed voting ensemble learning method against the baseline learning methods explored in this study. It can be observed in Table III that the voting ensemble learning method

outperformed other learning methods across the two experimental data sets. The proposed voting ensemble learning method recorded the highest average precision score of 93.779%. The LSTM performed relatively well, scoring the second-highest precision score of 93.457%. The exceptional performance of the LSTM may be linked to its ability to capture long-term dependencies. This property makes it suitable for text classification tasks such as hate speech detection, where the semantics of a tweet can be derived from the arrangement of words in the tweeted document.

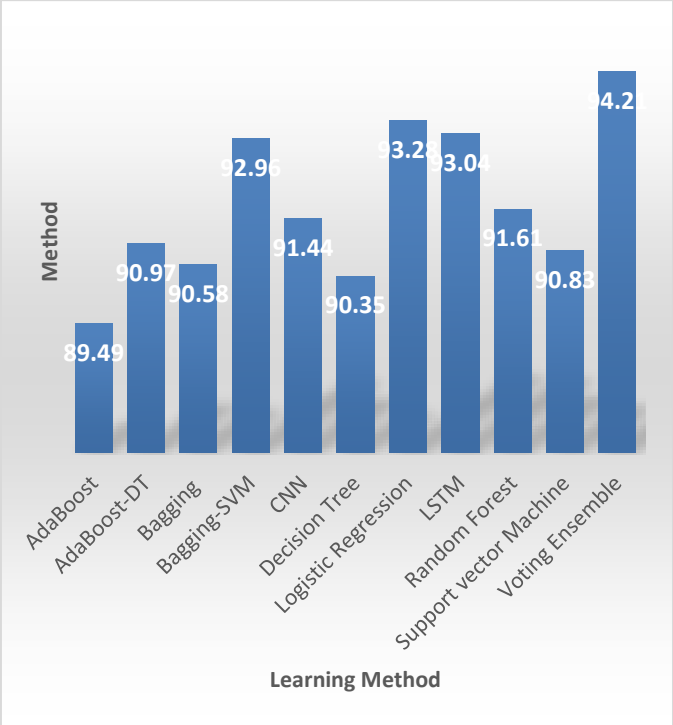

Fig. 2. The Average Accuracy of Learning Methods.

TABLE III. PRECISION SCORES OF LEARNING METHODS USING DIFFERENT TRAIN-TEST SPLITS

Data set	Kaggle		HSO		
Train: Test Split	**80:20**	**70:30:00**	**80:20**	**70:30**	**Average**
AdaBoost	88.093	88.011	91.686	91.182	89.743
AdaBoost-DT	90.523	90.436	93.096	89.970	91.006
Bagging	90.306	90.562	92.495	89.550	90.728
Bagging-SVM	91.519	90.723	95.248	94.489	92.995
CNN	88.495	88.315	95.042	94.674	91.631
Decision Tree	90.083	90.288	91.790	89.342	90.376
Logistic Regression	91.680	91.508	95.539	94.538	93.316
LSTM	91.207	90.919	95.477	96.224	93.457
Random Forest	89.418	90.053	93.508	93.478	91.614
Support Vector Machine	89.804	88.702	92.488	92.475	90.868
Voting Ensemble	**92.030**	**91.830**	**96.747**	**94.507**	**93.779**

The least average precision score of 89.743% was recorded by the AdaBoost method with the default parameter setting. The combination of AdaBoost with another classifier in ensemble learning improves the system performance. The AdaBoost method with DT as base learner outperformed the default AdaBoost because it gave an average precision of 91.006%, which is higher than 89.743% recorded by the default AdaBoost method. Most methods performed better with the 80:20 train test split as compared to the 70:30 split. However, only RF and DT performed better with a 70:30 split. The precision computed by DT and RF fell when the training data set was increased by 10% on the larger HSO data set. The drop-in performance may be the result of overfitting because both DT and RF are susceptible to overfitting [17]. Table III shows the precision scores of all the learning methods experimentally compared in this study.

Fig. 3 shows the plot of the average precision scores computed by the learning methods across the experimental data sets to visually illustrate the extent to which one method gives better precision than another. This result implies that the voting ensemble method can correctly detect hate speeches from the predicted class of hate speeches better than any other method, while AdaBoost performed worst in this case. The voting ensemble method is therefore the most useful when the cost of false positives is high while AdaBoost is not useful in this scenario.

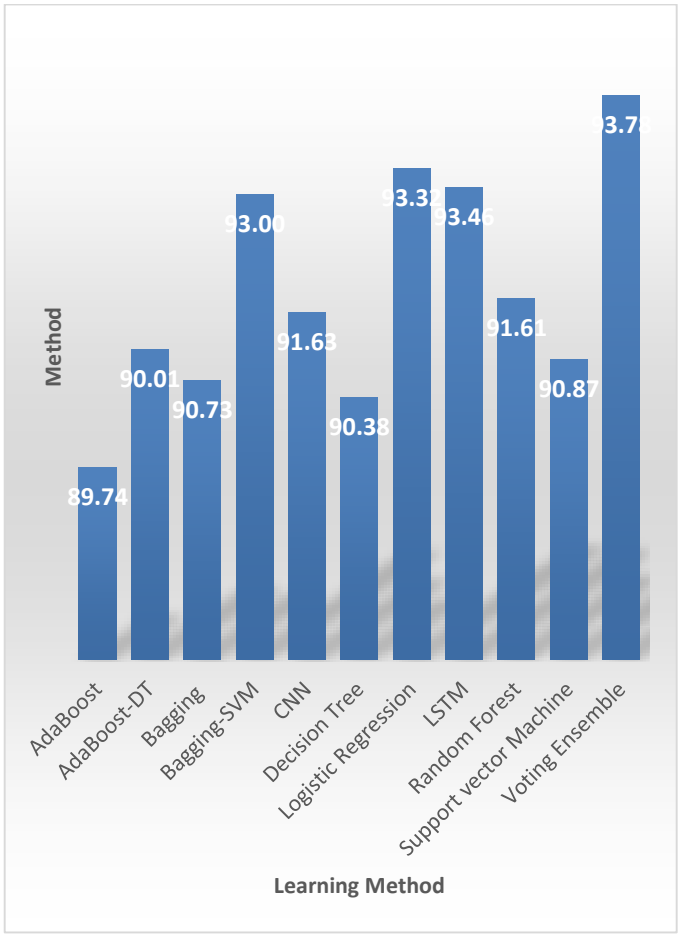

Fig. 3. The Precision of Learning Methods.

C. Recall Results

This section presents an evaluation of the learning methods investigated in this study based on the recall metric. Results from Table IV show that the proposed voting ensemble learning method gave an average recall value of 94.210%, which is superior to that of baseline learning methods used in this study. In addition, it can be noted that default meta classifiers of Bagging and AdaBoost were outperformed by their ensemble variants, which used different learning methods as base learners. The recall score for the default AdaBoost is 89.408%, while the recall score for the AdaBoost-DT is 90.959%. In addition, the recall score for the bagging method is 90.552%, while bagging-SVM had a recall score of 92.937%. It is obvious from these results that combining meta classifiers with different learning methods can lead to improved performance as shown in Table IV.

TABLE IV. RECALL SCORES OF LEARNING METHODS USING DIFFERENT TRAIN-TEST SPLITS

Data set	**Kaggle**		**HSO**		
Train: Test split	**80:20**	**70:30**	**80:20**	**70:30**	**Average**
AdaBoost	87.767	87.801	91.304	90.761	89.408
AdaBoost-DT	90.549	90.457	93.043	89.788	90.959
Bagging	90.312	90.553	92.174	89.170	90.552
Bagging-SVM	91.414	90.599	95.217	94.519	92.937
CNN	88.384	88.160	95.000	94.534	91.520
Decision Tree	90.119	90.310	91.739	89.223	90.348
Logistic Regression	91.680	91.496	95.435	94.550	93.290
LSTM	91.177	90.854	95.435	94.691	93.039
Random Forest	89.447	90.068	93.478	93.508	91.625
Support Vector Machine	89.749	88.695	92.391	92.444	90.820
Voting Ensemble	**92.040**	**91.823**	**96.739**	**96.237**	**94.210**

Fig. 4 shows the plot of average recall scores computed by the learning methods across the experimental data sets to visually illustrate the extent to which one learning method gives a better recall than another. This result implies that the voting ensemble method can correctly detect cases of hate speeches from all the actual classes of hate speeches better than any other learning method, while AdaBoost performed worst in this case. The voting ensemble method is therefore the most useful when the cost of false negatives is high while AdaBoost is not useful in this scenario.

D. F1-score Results

This section presents the results of the overall F1-score for the learning methods explored in this study. Table V shows that the proposed voting ensemble method consistently outperformed the baseline learning methods investigated by recording the highest average F1-score of 94.208%. The solitary bagging ensemble learning method recorded an average F1-score of 90.564%, while the bagging-SVM ensemble method recorded an average F1-score of 92.948%. Furthermore, the mean score of 89.897% of the average F1-scores for both AdaBoost and Decision tree learning methods is inferior to the average F1-score of 90.692% for AdaBoost–DT ensemble learning method. The analysis of the F1-score for LSTM and CNN deep learning methods has shown that LSTM consistently outperforms CNN. It can be perceived in Table V that LSTM recorded an average F1-score of 93.035%, while CNN recorded an average F1-score of 91.439%. This difference in performance may come from the capability of LSTM to capture long-term dependencies that are necessary when extracting word meanings in a text. The superior performance of the ensemble learning methods as compared to any solitary methods, including deep learning has illustrated that agglutinating multiple learning methods through ensemble learning is highly promising for reducing the error rate of the final learner in a hate speech detection system.

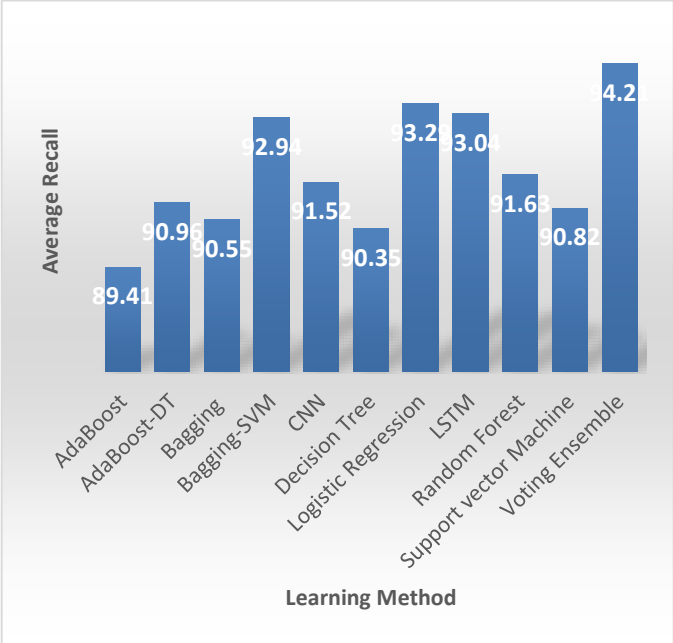

Fig. 4. The Average Recall of Learning Methods.

TABLE V. F-MEASURE SCORES OF LEARNING METHODS USING DIFFERENT TRAIN-TEST SPLITS

Data Set	**Kaggle**		**HSO**		
Train: Test Split	**80:20**	**70:30**	**80:20**	**70:30**	**Average**
AdaBoost	87.835	87.848	91.284	90.831	89.449
AdaBoost-DT	90.533	90.443	93.041	89.830	90.962
Bagging	90.309	90.557	92.159	89.232	90.564
Bagging-SVM	91.449	90.635	95.217	94.492	92.948
CNN	88.238	88.026	94.999	94.492	91.439
Decision Tree	90.092	90.296	91.737	89.255	90.345
Logistic Regression	91.680	91.501	95.432	94.493	93.277
LSTM	91.190	90.877	95.434	94.638	93.035
Random Forest	89.427	90.059	93.477	93.477	91.610
Support Vector Machine	89.771	88.699	92.387	92.456	90.828
Voting Ensemble	**92.035**	**91.826**	**96.739**	**96.230**	**94.208**

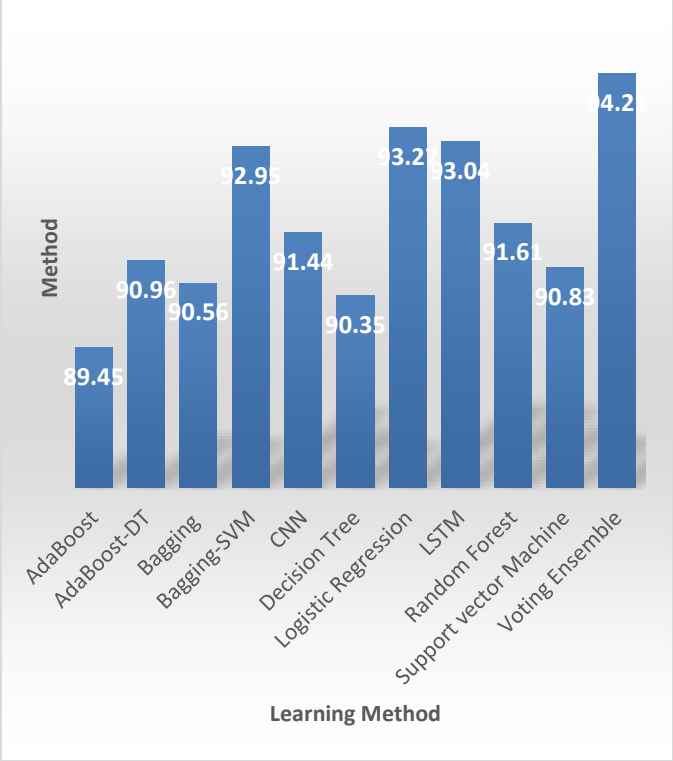

Fig. 5. The Average F-Measure Score of Learning Methods.

Fig. 5 shows the plot of average F1-scores computed by the learning methods across the experimental data sets to visually illustrate the extent to which one method gives a better F1-score than another. This result implies that the proposed voting ensemble method can better detect incorrectly classified cases better than any other learning method, while AdaBoost performed worst in this case. The voting ensemble is therefore the most useful when the classes are imbalanced while AdaBoost is not useful in this scenario.

Results from the functional metrics used in this study indicate that the voting ensemble outperformed the benchmark algorithms used in the study. It is worth noting that the individual performance of the meta classifiers was inferior to that of the proposed voting ensemble model. This superior performance of the proposed model may be attributed to the minimal overfitting, model extensibility, and interpretability features from each of the base learners [14, 15]. These results confirm the literature position that ensemble learning outperformed individual classifier algorithms in hate speech detection [38, 39].

E. Non-functional Results

This section presents the evaluation of all the learning methods investigated using the non-functional metrics of Hamming loss, Jaccard, Kappa, and execution time. The proposed voting ensemble method recorded the best Hamming loss, Jaccard, and Kappa scores as shown in Table VI. This result shows that the proposed ensemble learning method can maximise predictive capability while concomitantly minimising misclassification errors better than any of the baseline learning methods investigated. However, the proposed voting ensemble learning method recorded the second-highest training time of 0.095 hours, which is a tradeoff decision to consider between efficiency versus accuracy. The long execution time taken by the proposed ensemble method was because each inducer was trained separately, and the final aggregated decision was achieved through the principle of majority voting.

It can be observed from Table VI that SVM recorded the lowest Kappa score indicating a low level of inter-annotator agreement. This may be the result of minimal parameter tuning applied to the SVM method. The learning method recorded the worst Hamming loss of 10% to suggest a poor selection of parameters for the method. It is interesting to observe that ensemble learning methods such as bagging-SVM and voting ensemble took more time to train than the deep learning methods. This implies that although they perform better, ensemble learning methods are computationally expensive. However, the benefits of an improved performance can outweigh the need for increased resources in critical applications like hate speech detection.

TABLE VI. NON-FUNCTIONAL PERFORMANCE OF LEARNING METHODS

Method	Hamming loss	Jaccard	Kappa	Execution Time
AdaBoost	8.696	91.304	82.609	0.013
AdaBoost-DT	7.174	92.826	85.652	0.010
Bagging	6.739	93.261	86.522	0.064
Bagging-SVM	4.565	95.435	90.870	0.353
CNN	5.000	95.000	90.000	0.001
Decision Tree	6.087	93.913	87.826	0.009
Logistic Regression	4.565	95.435	90.870	0.003
LSTM	4.565	95.435	90.870	0.082
Random Forest	6.522	93.478	86.957	0.076
Support Vector Machine	10.000	90.000	80.000	0.011
Voting Ensemble	**3.261**	**96.739**	**93.478**	**0.095**

V. CONCLUSION

The primary contribution of this study is the construction and validation of a voting ensemble learning method to improve the automatic detection of hate speech in tweets. This is challenging open research because of the anaphoric, synonymy, and polysemy nature of the slang of tweets that make the interpretation of hate speech ambiguous, difficult, and controversial. The voting ensemble learning method has been demonstrated in this study to yield the best performance when compared to other learning methods.

However, one apparent curb of this study is the bag of words representation of features that suffers from the anaphoric, synonymy, and polysemy nature of words. In addition, a bag of words representation presents the inability to capture important information about interdependencies that exist among words. Moreover, word embedding representation can fall short in making machines draw adequate inferences from certain classes of sentences.

In the future, a knowledge-based method with sentence embedding will be introduced for tweet hate speech detection and compared the results against those of the existing word embedding learning methods. This envisioned novel method will circumvent the intrinsic curbs of the bag of words and word embedding representations. It will significantly increase the confidence level of social media prosecutors to genuinely regulate whether a given tweet is of hate speech or not.

REFERENCES

[1] K. Pitsilis, H. Ramampiaro, and H. Langseth, "Effective hate-speech detection in Twitter data using recurrent neural networks," Applied Intelligence, vol. 48, no. 12, pp. 4730-4742, 2018.

[2] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," IEEE Computational Intelligence Magazine, vol. 13, no. 3, pp. 55-75, 2018.

[3] S. Sohangir, D. Wang, A. Pomeranets, and T. M. Khoshgoftaar, "Big data: Deep learning for financial sentiment analysis," Journal of Big Data, vol. 5, no. 1, p. 3, 2018.

[4] S. MacAvaney, H.-R. Yao, E. Yang, K. Russell, N. Goharian, and O. Frieder, "Hate speech detection: Challenges and solutions," PloS one, vol. 14, no. 8, 2019.

[5] R. Polikar, "Ensemble learning," in Ensemble Machine Learning: Springer, pp. 1-34, 2012.

[6] M. Graczyk, T. Lasota, B. Trawiński, and K. Trawiński, "Comparison of bagging, boosting and stacking ensembles applied to real estate appraisal," in Asian Conference on Intelligent Information and Database Systems, Springer, pp. 340-350, 2010.

[7] U. Abubakar, S. A. Bashir, M. B. Abdullahi, and O. S. Adebayo, "Comparative study of various machine learning algorithms for tweet classification," i-manager's Journal on Computer Science, vol. 6, no. 4, p. 12, 2019.

[8] K. Zvarevashe and O. O. Olugbara, "Recognition of cross-language acoustic emotional valence using stacked ensemble learning," Algorithms, vol. 13, no. 10, p. 246, 2020.

[9] K. Zvarevashe and O. Olugbara, "Ensemble learning of hybrid acoustic features for speech emotion recognition," Algorithms, vol. 13, no. 3, p. 70, 2020.

[10] S. Oyewole and O. Olugbara, "Product image classification using Eigen Colour feature with ensemble machine learning," Egyptian Informatics Journal, vol. 19, no. 2, pp. 83-100, 2018.

[11] E. Adetiba and O. O. Olugbara, "Lung cancer prediction using neural network ensemble with histogram of oriented gradient genomic features," The Scientific World Journal, vol. 2, 2015.

[12] A. Modupe, O. O. Olugbara, and S. O. Ojo, "Filtering of mobile short messaging service communication using latent Dirichlet allocation with social network analysis," in Transactions on Engineering Technologies: Springer, pp. 671-686, 2014,

[13] C. J. Burges, "A tutorial on support vector machines for pattern recognition," Data Mining and Knowledge Discovery, vol. 2, no. 2, pp. 121-167, 1998.

[14] K. Kowsari, K. J. Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," Information, vol. 10, no. 4, p. 150, 2019.

[15] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in Eleventh International AAAI Conference on Web and Social Media, 2017.

[16] L. Gao and R. Huang, "Detecting online hate speech using context aware models," arXiv preprint arXiv:1710.07395, 2017.

[17] Q. Liu, F. Zhou, R. Hang, and X. Yuan, "Bidirectional-convolutional LSTM based spectral-spatial feature learning for hyperspectral image classification," Remote Sensing, vol. 9, no. 12, p. 1330, 2017.

[18] I. Kwok and Y. Wang, "Locate the hate: Detecting tweets against blacks," in Twenty-Seventh AAAI Conference on Artificial Intelligence, 2013.

[19] R. Ahluwalia, H. Soni, E. Callow, A. Nascimento, and M. De Cock, "Detecting hate speech against women in English tweets," EVALITA Evaluation of NLP and Speech Tools for Italian, vol. 12, pp. 194-199, 2018.

[20] R. T. Mutanga, N. Naicker, and O. O. Olugbara, "Hate speech detection using transformer methods," International Journal of Advanced Computer Science and Applications, vol. 11, no. 9, 2020.

[21] I. Siloko and C. Ishiekwene, "Boosting and bagging in kernel density estimation," The Nigerian Journal of Science and Environment, vol. 14, no. 1, pp. 32-37, 2016.

[22] A. Kadiyala and A. Kumar, "Applications of python to evaluate the performance of bagging methods," Environmental Progress & Sustainable Energy, vol. 37, no. 5, pp. 1555-1559, 2018.

[23] F. Li, J. Fan, L. Wang, H. Zhang, and R. Duan, "A method based on manifold learning and Bagging for text classification," in 2011 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), 2011.

[24] L. Xinqin, S. Tianyun, L. Ping, and Z. Wen, "Application of bagging ensemble classifier based on genetic algorithm in the text classification of railway fault hazards," in 2019 2nd International Conference on Artificial Intelligence and Big Data (ICAIBD), 2019: IEEE, pp. 286-290.

[25] H. ALSaif and T. Alotaibi, "Arabic text classification using feature-reduction techniques for detecting violence on social media," Work, vol. 10, no. 4, 2019.

[26] J. Prusa, T. M. Khoshgoftaar, and D. J. Dittman, "Using ensemble learners to improve classifier performance on tweet sentiment data," in 2015 IEEE International Conference on Information Reuse and Integration, 2015.

[27] D.-S. Cao, Q.-S. Xu, Y.-Z. Liang, L.-X. Zhang, and H.-D. Li, "The boosting: A new idea of building models," Chemometrics and Intelligent Laboratory Systems, vol. 100, no. 1, pp. 1-11, 2010.

[28] K. Wang, Y. Wang, Q. Zhao, D. Meng, X. Liao, and Z. Xu, "SPLBoost: An improved robust boosting algorithm based on self-paced learning," IEEE Transactions on Cybernetics, 2019.

[29] E. Cambria, S. Poria, A. Gelbukh, and M. Thelwall, "Sentiment analysis is a big suitcase," IEEE Intelligent Systems, vol. 32, no. 6, pp. 74-80, 2017.

[30] R. Socher, C. C. Lin, C. Manning, and A. Y. Ng, "Parsing natural scenes and natural language with recursive neural networks," in Proceedings of the 28th International Conference on Machine Learning (ICML-11), pp. 129-136, 2011.

[31] P. D. Turney and P. Pantel, "From frequency to meaning: Vector space models of semantics," Journal of Artificial Intelligence research, vol. 37, pp. 141-188, 2010.

[32] P. Fortuna and S. Nunes, "A survey on automatic detection of hate speech in text," ACM Computing Surveys (CSUR), vol. 51, no. 4, pp. 1-30, 2018.

[33] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," Journal of Machine Learning Research, vol. 9, no. Aug, pp. 1871-1874, 2008.

[34] A. Genkin, D. D. Lewis, and D. Madigan, "Large-scale Bayesian logistic regression for text categorization," Technometrics, vol. 49, no. 3, pp. 291-304, 2007.

[35] A. Vaswani et al., "Attention is all you need," in Advances in Neural Information Processing Systems, 2017, pp. 5998-6008.

[36] S. Liu, "A survey on fault-tolerance in distributed optimization and machine learning," arXiv preprint arXiv:2106.08545, 2021.

[37] D. Agnihotri, K. Verma, P. Tripathi, and B. K. Singh, "Soft voting technique to improve the performance of global filter based feature selection in text corpus," Applied Intelligence, vol. 49, no. 4, pp. 1597-1619, 2019.

[38] S. A. Kokatnoor and B. Krishnan, "Twitter hate speech detection using stacked weighted ensemble (SWE) model," in 2020 Fifth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), 2020.

[39] M. K. A. Aljero and N. Dimililer, "A novel stacked ensemble for hate speech recognition," Applied Sciences, vol. 11, no. 24, p. 11684, 2021.