applied sciences

MDPI

Article

Threatening URDU Language Detection from Tweets Using Machine Learning

Aneela Mehmood [1], Muhammad Shoaib Farooq [1], Ansar Naseem [1], Furqan Rustam [2], Mónica Gracia Villar [3,4,5,*], Carmen Lili Rodríguez [3,6,7] and Imran Ashraf [8,*]

1 Department of Computer Science, University of Management and Technology, Lahore 54000, Pakistan
2 School of Computer Science, University College Dublin, D04 V1W8 Dublin, Ireland
3 Faculty of Social Science and Humanities, Universidad Europea del Atlántico, Isabel Torres 21, 39011 Santander, Spain
4 Department of Project Management, Universidad Internacional Iberoamericana, Arecibo, PR 00613, USA
5 Department of Extension, Universidade Internacional do Cuanza, Cuito EN250, Bié, Angola
6 Department of Project Management, Universidad Internacional Iberoamericana, Campeche 24560, Mexico
7 Fundación Universitaria Internacional de Colombia, Bogotá 111311, Colombia
8 Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 38541, Korea
* Correspondence: monica.gracia@uneatlantico.es (M.G.V.); imranashraf@ynu.ac.kr (I.A.)

Abstract: Technology's expansion has contributed to the rise in popularity of social media platforms. Twitter is one of the leading social media platforms that people use to share their opinions. Such opinions, sometimes, may contain threatening text, deliberately or non-deliberately, which can be disturbing for other users. Consequently, the detection of threatening content on social media is an important task. Contrary to high-resource languages like English, Dutch, and others that have several such approaches, the low-resource Urdu language does not have such a luxury. Therefore, this study presents an intelligent threatening language detection for the Urdu language. A stacking model is proposed that uses an extra tree (ET) classifier and Bayes theorem-based Bernoulli Naive Bayes (BNB) as the based learners while logistic regression (LR) is employed as the meta learner. A performance analysis is carried out by deploying a support vector classifier, ET, LR, BNB, fully connected network, convolutional neural network, long short-term memory, and gated recurrent unit. Experimental results indicate that the stacked model performs better than both machine learning and deep learning models. With 74.01% accuracy, 70.84% precision, 75.65% recall, and 73.99% F1 score, the model outperforms the existing benchmark study.

Keywords: threatening language detection; Urdu text classification; machine learning; stacking

Citation: Mehmood, A.; Farooq, M.S.; Naseem, A.; Rustam, F.; Villar, M.G.; Rodríguez, C.L.; Ashraf, I. Threatening URDU Language Detection from Tweets Using Machine Learning. *Appl. Sci.* **2022**, *12*, 10342. https://doi.org/10.3390/app122010342

Academic Editors: Valentino Santucci and Paolo Mengoni

Received: 22 September 2022
Accepted: 11 October 2022
Published: 14 October 2022

1. Introduction

Online social networks have expanded to play a key role in our daily lives because of the development of the Internet and communication technology. The number of social media users doubled just from 2017 to 2021, from 2.73 billion to 4.26 billion, and is expected to grow to 5.85 billion by 2027 [1]. According to [2], the number of Twitter users is approximately 486 million and over 1.4 trillion tweets are posted each year. Twitter is a social media platform for sharing short texts, called tweets, with a 280-character limit per tweet. Twitter provides freedom of speech, and users are allowed to express their views and opinions freely. Nonetheless, some Twitter users have manipulated information to commit online crimes such as malware distribution and phishing [3,4]. In addition, the controversial content raises more complex concerns, such as sexual misconduct and self-harm incitement. Threats against victim groups, gender-based violence, and physical violence may also be instigated [5].

Wide and unrestricted access to social media platforms like Twitter has raised serious concerns. Cyberbullying, online racism, use of abusive language, and social misconduct

have become prevalent on social media platforms. In this regard, several works are carried out for cyber-threat detection. For example, a cascaded convolutional neural network (CNN) is presented in [6] for the detection and classification of cyber-threats on Twitter. Similarly, a comprehensive overview of ransomware is presented in [7]. However, to address this issue on a larger scale on social media networks such as Twitter, more research efforts are needed. In addition, research works on offensive text detection, cyberbullying, and abusive language are carried out for high-resource languages like English.

Automatic threatening language detection techniques on Twitter have extended outside the English language as well [8,9]. Automated detection of threatening language is studied in other languages, and particularly Bengali, German, Dutch, Italian, Indonesian, and Arabic were the subjects of many research works [5,10,11]. To automatically identify threatening languages, these researchers looked into linguistic expertise and resources. Very little work is done for threatening language detection for low-resource languages like Urdu. Amjad et al.'s work [12] is the only significant effort for automated threatening language identification in Urdu [12]. To fill this research gap in the Urdu language, this study endeavors to build a stacking model for threatening Urdu language detection. In this regard, this study makes the following contributions:

- For threatening language detection, we conduct a series of experiments employing machine learning algorithms such as the extra tree (ET) classifier, Bernoulli Naive Bayes (BNB), support vector classifier (SVC), and logistic regression (LR). In addition, deep learning models CNN, fully connected network (FCN), gated recurrent unit (GRU), and long short-term memory (LSTM) are utilized.
- One of the main contributions in this study is the stacking of features for machine learning approaches. We propose a stacked feature vector based on preprocessed tweets and verbs extracted from preprocessed tweets. A stacking classifier is developed for threatening text detection for the Urdu language. The proposed models use ET and BNB as the base learners while LR is used as the meta-learner.
- For machine-learning term frequency-inverse document frequency (TF-IDF) and bag of words (BoW) are employed for the construction of features, and embedding has been employed as feature vector deep neural networks.
- To evaluate the performance of the proposed approach, accuracy, precision, recall, F1 score, and receiver operating curve (ROC) area under the curve (AUC) is used. In addition, a performance comparison is carried out with the benchmark study.

The remainder of the paper is organized as follows. Section 2 describes the works related to threatening text detection. Section 3 includes details for benchmark datasets and proposed methodology. In Section 4, we provide analysis and discussion of results. The last section provides the conclusion and future directions.

2. Literature Review

This section discusses existing studies for threatening language detection for the Urdu language. To the best of the authors' knowledge, there is only one significant study for threatening language detection in the Urdu language. Amjad et al. [12] have contributed an Urdu corpus b for threatening Urdu language detection. In addition, the authors performed experiments using machine learning-based algorithms such as LR, multi-layer perceptron (MLP), AdaBoost, and deep learning methods for the detection of threatening language.

Many existing studies focus on different languages like English, Arabic, Spanish, and others. In the English language, many studies focused on inappropriate and abusive language detection. For example, ref. [13] present an automatic flame detection, which can be rants, taunts, and squalid phrases in the English language. The authors used a multi-level classification paradigm where complement Naive Bayes (NB) is utilized. For the second level, multinomial updatable NB is selected, while the last level chose a decision table and NB-based hybrid classifier.

The study [14] proposed the use of lexical syntactic features for offensive language detection on social media platforms. The study specifically focused on name-calling

harassment by studying the user's writing style and structure. Results demonstrate that the proposed approach can obtain 98.24% for offensive sentence detection and performs better than existing methods. Similarly, ref. [15] focused on detecting different kinds of offensive content on social media platforms. The data is annotated using a three-layer scheme. Experiments are performed using a support vector machine (SVM), bi-directional LSTM, and CNN. A 0.80 macro F1 is reported using the CNN model.

Xiang et al. [16] proposed a semi-supervised approach for offensive language detection on Twitter. Statistical topic modeling is used to exploit linguistic regularities for this purpose. Results show a 75.1% true positive rate using the LR model. Similarly, hateful and antagonistic content is identified in [17] employing the K-means algorithm. The study focuses on text containing more than one characteristic like race and sexual orientation. Machine learning models using the BoW approach provided an average precision of 87.83% for six classes of hateful content. Adono et al. [18] employed NB and SVM with hateword2vec, and hatedoc2vec with n-grams for aggressive text detection from tweets. In addition, the synthetic minority oversampling technique (SMOTE) is applied for data oversampling. An F1 score of 42.85% is obtained using SMOTE for the testing dataset. The study [19] uses graph convolutional networks (GCN) to obtain deep properties by modeling follower–following relationships. Using LR with GCN, a precision of 86.23% is obtained while recall and F1 scores are 84.73% and 85.42%, respectively.

Besides the English language, abusive content detection from other languages is also studied by several researchers. For example, ref. [20] proposes a method to detect nasty comments from Japanese posts from bulletin board systems. The study employs SVM with an n-gram model and obtains a 72.25% F1 score.

The study [21] explored n-grams up to 8-grams to detect the offensive language in English. Three systems are developed for three sub-tasks in offensive text detection. The proposed approach achieves 79.76%, 87.91%, and 44.37% accuracy scores, respectively.

Polignano et al. [22] used ALBERTO with BERT tokens for hate speech detection in Italian detection. The model is based on Italian language understanding and is trained with BERT and 200M Italian tweets. Results show an average F1 score of 0.8410 for the model.

The hate speech dataset and detection system for the Turkish language are presented in [23]. Tweets collected for experiments are from two different domains. The study proposed BERTurk, which is a transformer architecture. Experimental results using five-fold cross-validation indicate an accuracy of 77%. A comparative summary of the discussed works is presented in Table 1.

The low-resource Urdu language has limited contributions regarding threatening language detection. To our knowledge, just one previous study of the Urdu language has been done, where Amjad et al. [12] contributed the dataset for Urdu threatening text detection and a detection model. However, the authors just selected the preprocessed text as a feature. Another recent study is by Mithun et al. [24], which used extreme gradient boosting, mBERT, and dehateber-mono-arabic for threatening text detection for the Urdu language. The best F1 score of 0.54574 is the obtained dehateber-mono-arabic model. This study adopts the stacking of features and algorithms. In the feature selection layer, we select two types of contents from tweets; computed features on preprocessed tweets and extracted verbs from preprocessed tweets. Finally, on these features, the feature vectors are computed and stacked.

Table 1. Comparative overview of the discussed research works.

Ref.	Language	Features	Methods	Results
[12]	Urdu	Word, char n-grams, fastText embedding	LR, MLP, AdaBoost, RF, SVM, CNN and LSTM	Accuracy: 72.50%
[13]	English	Char n-grams (1–4)	LR	Accuracy: 96.72%
[14]	English	BOW, char n-grams	LR, SVM, CNN	Offensive sentence precision: 98.24%, offensive user precision: 77.9%
[15]	English	BOW, char n-grams (2,3,5)	SVM, NB	F1 macro: 0.80
[16]	English	Latent Dirichlet Allocation	LR	True positive rate: 75.1%
[17]	English	Word n-gram (3–8), char n-gram (1–3)	CNN, RNN, NB, RF, SVM	Precision: 87.83%
[18]	English	Hateword2ved, hatedoc2vec, n-grams	NB, SVM	F1 score: 42.85%
[19]	English	follower-following relationship	LR+GCN	86.23%
[20]	Japanese	Char n-grams	SVM	F1 score: 72.25%
[21]	English	unigram	Svm, BILSTM, CNN	F1 score: 55.31%
[22]	Italian	BERT tokens	ALBERTO	F1 score: 84.10%
[23]	Turkish	BERT	uncased BERTurk	Accuracy: 77%
[25]	English	Abusive and non-abusive word list	k-means	-
[24]	Urdu	BERT	XGBoost, LightGBM, mBERT, dehateber-mono-arabic	F1 score: 54.57%

3. Materials and Methods

In this section, layered architecture for threatening and non-threatening tweets is discussed. Figure 1 shows the four-layer architecture followed in this study. Each layer has different functions to perform and the output of the conceding layer serves as the input of the preceding layer.

The first layer is the input layer and is used for data acquisition. The second layer consists of the preparation of tweets where steps like data cleaning and feature extraction are used. The third layer implies the development of intelligent threatening tweet identifiers where ensemble-based approaches, bagging, and stacking are discussed. The BNB and ET are utilized for bagging. For stacking, ET and BNB are utilized as base learners while LR is used as the meta learner. In addition, deep learning models are employed for experiments including LSTM, GRU, CNN, and FCN. Finally, the last layer signifies the application and deploys an automatic threatening language classifier.

3.1. Input Layer

The dataset is collected from the study [12], in which the Tweepy library was used to collect tweets. The process of data collection begins with the collection of Urdu language and annotation. The collected dataset comprises 3564 tweets where 1782 threatening and 1782 non-threatening tweets have been assembled. The summary of the dataset is provided in Table 2 and sample is shown in Figure 2.

3.2. Data Preparation Layer

The collected data might contain noise and the second layer directs two tasks related to noise removal. Firstly, the dataset has been cleaned by using different text cleaning techniques like special character removal, white space removal, etc., as indicated in Figure 1. Secondly, verbs have been extracted from the cleaned text. Finally, the feature vectors have been computed using TF-IDF and BOW using word and character level n-grams.

3.2.1. Features Extraction

Features are especially important for machine learning problems in text classification. To conduct this study, we employed verbs as features and cleaned text features. First, we computed features on the raw cleaned text and then from the cleaned text, verbs are extracted as features. Lastly, these two feature vectors are combined to construct one feature vector. This feature vector is fed to the model for training and prediction of threatening and non-threatening tweets.

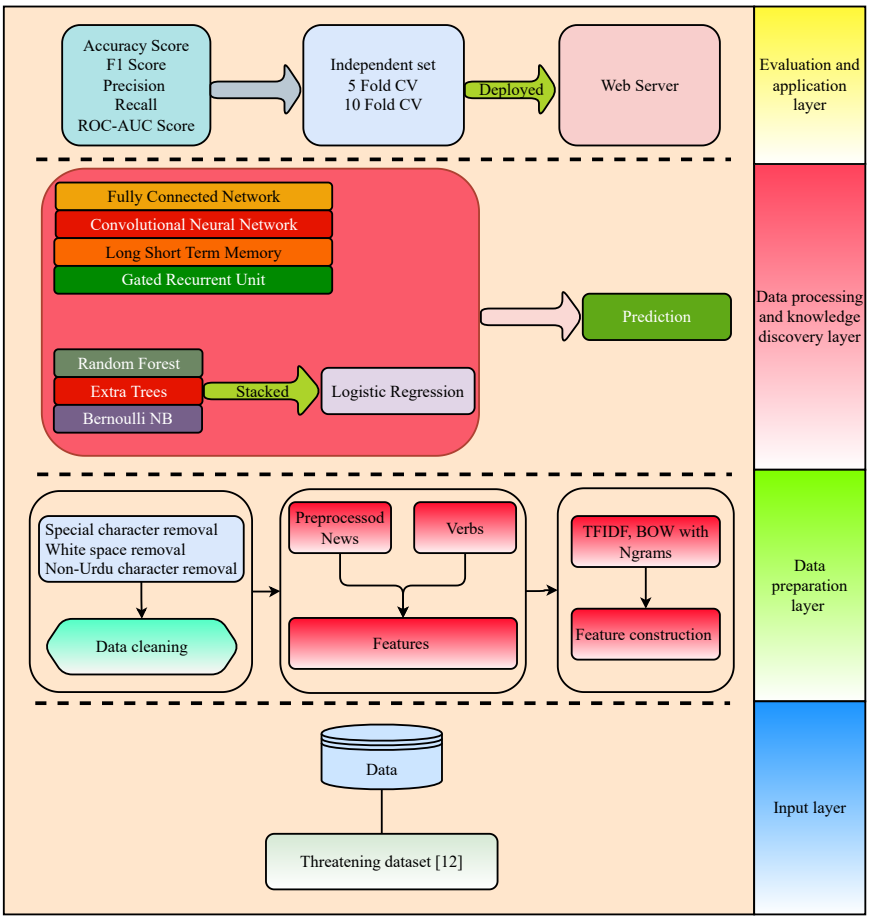


Figure 1. Workflow of the adopted methodology.

Table 2. summary stats of dataset.

Tweet	Vocabulary (Words)	Vocabulary (Char)	Corpus
Threatening	30,518	134,408	1782
Non-threatening	31,135	140,225	1782
Total	61,653	274,633	3564

Tweet	Class
بھارت کی ایٹمی حملے کی دھمکی	**Threatening**
بکواس مت کرو	**Threatening**
کشمیر پاکستان کی شہباز شریف	**Non-threatening**
شاباش میرے بلوچ بھا نیو	**Non-threatening**

Figure 2. Examples from the dataset containing tweets from the threatening vs. non-threatening classes.

3.2.2. Feature Vector Approaches

For feature extraction from the preprocessed text, this study uses TF-IDF and BoW approaches. A brief explanation of these features is provided here for completeness.

Term frequency (TF) describes how frequently a term/feature occurs. In a tweet, a feature's frequency corresponds to its importance. Every feature in the tweet corpus has been transformed by term frequency into a matrix with the number of tweets in the rows and the number of unique terms in the columns. The frequency of a feature is shown by the number of samples that share that feature overall. The inverse document frequency (IDF), which reduces the weight of a feature if the feature's occurrences have distributed among all documents, has been used to determine the weight of a feature.

BoW counts the existence of each feature in one document and in the whole corpus and constructs the feature vector with a fixed length.

Word-based and character-based n-grams are used to extract features from the preprocessed corpus that helps the model distinguish between threatening and non-threatening tweets. In this study, we used character level n-grams ranging from unigram (n = 1) to tetragram (n = 4). To train the model on word-level n-grams, we used unigram (n = 1) and bigram (n = 2) approaches. This attribute is used because it provides structural details and helps to make the text more pertinent.

3.3. Data Processing and Knowledge Discovery

Now, the feature vector attained from the data preparation layer is ready to be fed to machine-learning classifiers. The third layer symbolizes the approaches employed for threatening language detection using various machine-learning classifiers. In this methodology, two ensemble-based approaches and one Bayes thorium-based approach have been employed. Moreover, several deep learning models are also trained. A brief explanation of each method is given in subsequent sections.

3.3.1. Extra Tree

ET is a member of the bagging family of algorithms; it functions identically to the random forest, with two primary distinctions. As part of its training, the model receives threatening and non-threatening tweets along with a label. With tweets of equal size in each subset and a variety of sub-datasets based on threatening and non-threatening tweets, the model produces several sub-datasets. Using a fixed number of split nodes, decision trees are generated. Every model of a weak learner is given a test tweet. The class prediction with the most votes is used to determine the test tweet.

3.3.2. Bernoulli Naive Bayes

In the BNB model, each tweet is represented by a vector x of binary values for each feature, indicating which words in tweets appear and do not appear in the document. Each word in tweets is also represented by a Boolean value (0 or 1), which indicates both its presence and absence. Since the features are distinct binary variables and the number of occurrences of the keyword in a tweet is irrelevant in BNB, Bernoulli's method has the advantage of displaying the absence of a word in the tweet. It has frequently been used to classify tokens or keywords. Formally, the Bernoulli trials are expressed as [26]. Let $p(x_i|w_j)$ represent the maximum likelihood approximation that a particular token occurs in a class and is computed as follows:

$$p(x_i|w_j) = \frac{df(x_i, y) + 1}{d(fy) + 2} \tag{1}$$

where $df(x, y)$ represents the number of training set tweets that contain the keyword and belong to the specified class. $Df(y)$ is the number of class-related training dataset tweets. Wj, +1, and +2 are smoothness Laplace parameters.

3.3.3. Support Vector Classifier

SVC is a state-of-the-art classification algorithm used to detect threatening language detection for the Urdu language. In this approach, SVC finds which tweets in vector form from both classes exist on the line of the decision boundary. In SVC, many hyperplanes are created and the optimal hyperplane is selected that maximizes the distance between threatening and non-threatening tweets. Next, the hyperplane is selected with maximum distance, and when a tweet is passed from test data to this hyperplane, the non-threatening class is assigned if the predictor computes the negative number and the threatening class is assigned if the model computes the positive number.

$$y = b + \omega_1 * x_1 + \omega_2 * x_2 +, ..., \tag{2}$$

while b is the slope, ω_1 and afterward are the weights identified in the coefficient, and x_1 and hereafter are the input variables.

After we have created the hyperplane, we can utilize it to generate predictions. The hypothesis function h is formulated as below:

$$h(x_i) = \begin{cases} +1 & \text{if } \omega.x + b \geq 0 \\ -1 & \text{if } \omega.x + b < 0 \end{cases} \tag{3}$$

3.3.4. Logistic Regression

LR is a supervised machine-learning algorithm and comprises two parts. In the first part, the weights are multiplied with the feature vector of threatening and non-threatening tweets in addition to bias. After multiplication, each tweet generates a number, which is not a probability space. To convert it into probability space, a sigmoid activation function has been applied. The result ranges from 0 to 1, where a number greater than 0.5 is assigned to the threatening class and less than 0.5 is assigned to the non-threatening class. Z represents the sigmoid activation function and Y symbolizes the linear part, which comprises weights multiplication with the feature vector.

$$Z = \frac{1}{1 + e^{-z}} \tag{4}$$

where

$$Y = (w_1 x_1 + w_2 x_2 +, ..., + w_n x_n + b) \tag{5}$$

3.3.5. Stacking

Stacking is an ensemble technique used to improve model performance when a single model might not perform adequately. Using the stacking method, models like ET, LR, SVC, and BNB provide predictions for each tweet. The output of these models is used to construct a new trainable feature vector, and the generated feature vector is provided to the meta learner, where LR is employed as the meta learner. LR is trained on newly generated predictions from individual classifiers and provides the final prediction. Hyper-parameters settings for machine learning models shown in Table 3.

Table 3. Hyper-parameters list for machine learning classifiers.

Classifier	Hyper-Parameter
ET	N_estimators = 100, min_samples_split = 2, max_feature = 'sqrt'
Bernoulli NB	Alpha = 1.0, binarize= 0.0, fit_prior = True, class_prior = None
LR	tol = 0.0001, C = 1.0,solver = 'lbfgs'
SVC	C = 1.0,kernel = 'rbf',degree = 3,gamma = 'scale'
Stacking	Base_estimator = ET and BNB, final_estimator = LR

3.3.6. Deep Learning Architectures

This part motivates explaining the architecture of deep learning models. This study employed many neural network architectures including CNN, FCN, LSTM, and GRU. The performance of the models is optimized, and a randomized search approach is used to find the optimal parameters.

Each neuron in the layer before is connected to every neuron in the layer following it in an FCN. The goal of FCN is to acquire function f, which is defined as $y = f(\alpha, x)$. To obtain the best performance, the right parameters need to be applied to the input x. Finding the optimal set of parameters enables $y = f(x)$ mapping to provide the most accurate estimation of f. For threatening language detection, FCN is deployed using the architecture given in Table 4.

Table 4. Layered-wise details for a fully connected network.

Layer	Weights
200 ReLU units with dense layer	(46 + 1) * 200 = 9400
100 ReLU units with dense layer	(200 + 1) * 100 = 20,100
50 ReLU units with dense layer	(100 + 1) * 50 = 5050
20 ReLU units with dense layer	(50 + 1) * 20 = 1020
Output layer with single sigmoid neuron	(20 + 1) * 1 = 21

The CNN model for prediction of threatening language is composed of two convo-max pool sections parted by a dropout layer, an embedding layer, a global average layer, a feature extraction layer, and lastly an output layer comprised of the single sigmoid unit as depicted in Table 5. The embedding layer is used to translate each tweet sample x with a length of 32 to provide the $X \in R(\eta \times \xi)$ tensor. The first convo-max pool has a filter size of three with 46 1D convolution neurons in addition to ReLU non-linearity with 1D max pool operation. The max pooling takes an average of each of the 32 feature mappings from previous layers, while the pooling layer flattens the output of prior layers in a one-dimensional array of 18 values.

Table 5. Layered-wise details for CNN.

Type of Layer	Weights
Embedding layer with 100 neurons	1,000,000
Con1D with 32 filters	(300 + 1) * 32 = 9632
Max pooling1D	0
flatten	0
Dense = 128	(736 + 1) * 128 = 94,336
Output layer with sigmoid	(128 + 1) * 1 = 129

In the LSTM model, each record was processed as a single-member sequence of 10,000-dimensional vectors to 100 LSTM units. Two dense layers with ten (10) and one neuron, respectively, were attached with LSTM outputs to make predictions for a two-class problem. The first dense layer used ReLU activation while the classification layer with a single unit used Sigmoid activation to make predictions. A dropout layer of 0.5 was introduced between LSTM output with *tanh*. The LSTM model was trained on the dataset for 18 epochs. Architectural details of LSTM are given in Table 6.

Table 6. Layered-wise details for LSTM.

Type of Layer	Weights
Embedding = 10,000	100 * (99 + 1) = 10,000
Dropout = 0.5	No weights
Dense = 10 with relu	(100 + 1) * 10 = 1010
LSTM = 100	(443 + 1) * 100 = 44,400
Dense = 1 with sigmoid (output layer)	(100 + 1) * 1

In the GRU model, each record was processed as a single-member sequence of 100,000-dimensional vectors to 32 GRU units. Two dense layers with ten (10) and one neuron, respectively, were attached with GRU outputs to make predictions for a two-class problem. The first dense layer used ReLU activation while the classification layer with a single unit used sigmoid activation to make predictions. A drop-out layer of 0.5 was introduced between GRU output. GRU model was trained using 10 epochs. Details of GRU models are given in Table 7.

Table 7. Layered-wise details for GRU.

Type of Layer	Weights
Embedding = 10,000	100 * (99 + 1) = 10,000
Dropout = 0.5	nothing
Dense = 10 with relu	(100 + 1) * 10 = 1010
Bidirectional = 64	(401 + 1) * 64 = 25,728
Dense = 1 with sigmoid (output layer)	(10 + 1) * 1 = 11

3.4. Application and Evaluation Layer

This layer comprises the deployment of a trained classifier for threatening text detection in the Urdu language. The last layer consists of evaluation metrics that have been utilized to check the performance of classifiers using different testing techniques. The predictor with the best results will be deployed on the web server.

4. Experiments and Results

To evaluate the performance of a threatening and non-threatening tweets predictor, metrics such as accuracy score, precision, recall, and the ROC AUC curve are employed. The accuracy score indicates the proportion of correctly identified threatening and non-threatening tweets based on a comparison of all tweets. The precision specifies the predicted positive class tweets, whereas recall describes the proportion of the total positive tweets that are anticipated to be positive. The following equations are used for these metrics.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{6}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{7}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{8}$$

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{9}$$

4.1. Results Using BoW Features

Several experiments are performed to detect threatening and non-threatening tweets. This study used word n-grams and character n-grams for features ranging from 1 to 4 n-

grams. For features, verbs from cleaned text and preprocessed text are employed. For word level, BoW technique results are presented with stacked features (cleaned text + verbs).

Table 8 shows the results using uni-gram features. The best accuracy of 73.10% is achieved by stacking. Similarly, the best values for precision, recall, and F1 scores are also from the stacking model. For bi-gram, BNB outperforms other classifiers regarding accuracy, recall, and F1 with 73.64%, 75.97%, and 73.59%, respectively. The highest precision of 70.47% using bi-gram is obtained by the stacking model. The highest F1 score for uni-gram and bi-gram are 72.73 and 73.59, respectively.

Table 9 presents the results of all machine learning models using tri-gram to tetra-gram features. It is observed that increasing the features leads to better performance. For tri-grams, the best results are achieved with LR which has 73.08% each for accuracy and F1 score. For tetra-gram, SVC achieves the highest accuracy of 73.36% and an F1 score of 73.31%. Figure 3 shows the AUC of machine learning models using BoW features. ROC-ACU curve shows that the stacking model using BoW with word level produces better results.

Table 8. Threatening language detection using uni-gram BoW features.

N-Grams	Features	Model	Accuracy	Precision	Recall	F1
(1,1)	9062	ET	71.40	70.10	71.98	71.40
		BNB	71.87	67.48	73.97	71.81
		SVC	70.32	75.52	67.31	70.32
		LR	70.1	69.48	71.17	70.21
		Stack	73.10	70.75	74.70	72.73
(1,2)	53,239	ET	72.43	75.14	71.28	72.41
		BNB	73.64	69.16	75.97	73.59
		SVC	72.32	70.34	73.43	72.17
		LR	72.21	71.48	74.45	72.28
		Stack	73.18	70.47	74.51	73.16

Table 9. Threatening language detection using N-gram range for BoW features.

N-Grams	Features	Model	Accuracy	Precision	Recall	F1
(1,3)	12,931	ET	71.87	70.47	72.5	71.86
		BNB	71.21	67.30	67.29	71.18
		SVC	73.08	75.71	71.94	73.06
		LR	73.08	74.01	72,66	73.08
		Stacking	71.30	68.79	72.44	71.29
(1,4)	48,753	ET	72.52	71.59	72.95	72.52
		BNB	71.78	66.36	74.42	71.69
		SVC	73.36	77.76	71.48	73.31
		LR	72.82	74.39	72.10	72.80
		Stacking	72.90	69.35	74.65	72.86

4.2. Experimental Results Using TF-IDF

Experiments are performed using TF-IDF features with the employed machine learning models. Results for uni-gram and bi-gram with TF-IDF features are presented in Table 10. Results indicate that the stacking model performs superbly with both uni-gram and bi-gram features and obtains the highest accuracy of 73.64% and 74.01%, respectively. Stacking

models also obtain the highest values for precision and F1 score. Results also reveal that the performance of models with TF-IDF features is better than BoW features.

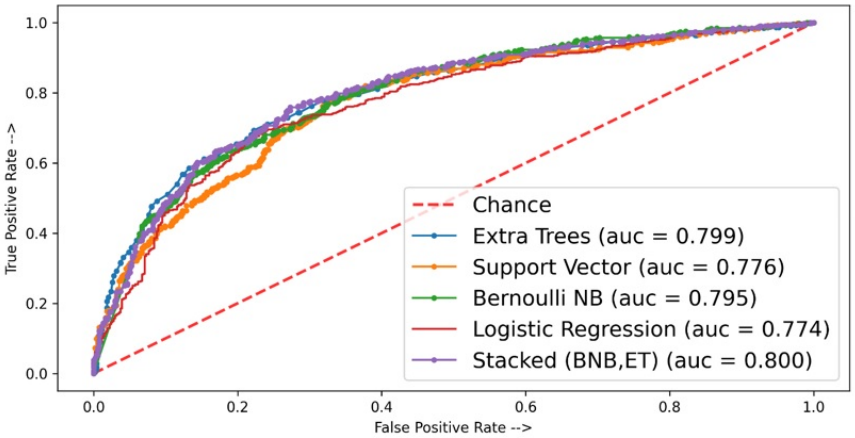


Figure 3. ROC using BoW features.

Table 10. Threatening language detection using TF-IDF features.

N-Grams	Features	Model	Accuracy	Precision	Recall	F1
(1,1)	9062	ET	71.40	68.97	72.50	71.38
		BNB	71.87	67.48	73.98	71.81
		SVC	70.93	76.45	68.86	70.85
		LR	70.65	70.28	70.81	70.65
		Stacking	73.64	71.40	74.75	73.63
(1,2)	53,239	ET	73.18	76.82	71.60	73.15
		BNB	73.64	59.15	75.98	73.59
		SVC	72.21	72.32	73.40	71.40
		LR	73.12	72.12	74.02	71.04
		Stacking	74.01	70.84	75.65	73.99

ROC-AUC of models with TF-IDF features is given in Figure 4. It shows that by using bi-gram features with a stacking model, the best results are obtained for threatening language detection.

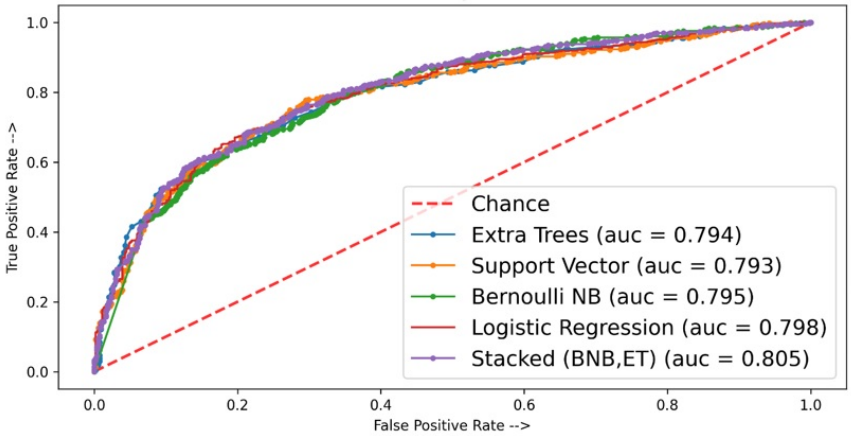
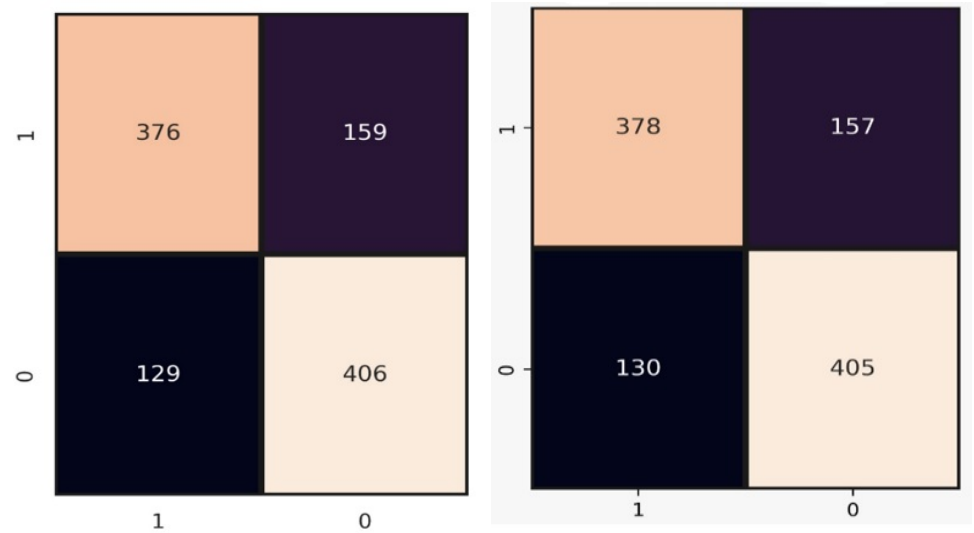


Figure 4. ROC using TF-IDF features.

Figure 5 shows the confusion matrix generated by a stacked-based approach using BoW and TF-IDF features. The stacking model correctly predicts 376 threatening tweets and 406 non-threatening tweets. Incorrect predictions are 288 for the stacking model using BoW features. Figure 5b shows the correct and incorrect predictions when TF-IDF features are used with the stacking approach. It correctly predicts 378 threatening tweets and 405 non-threatening tweets.

4.3. Results Using Cross-Validation

Cross-validation is another testing technique employed in this study. In independent set testing, just one split is done. However, sometimes the split might divide the biased data into train and test, which leads to underfitting or overfitting. To overcome this problem, cross-validation testing has been employed. Cross-validation is robust testing, and different folds are used for validation. In this study, testing is carried out using 5-folds and 10-folds. Five-folds correspond to the division of data into five folds, where the first fold is used for testing while the remaining four folds are used for training. In the next iteration, the second fold is used for testing, while the first and last three folds are used for training. This iteration continues until all five folds are used for testing. Finally, the average accuracy is computed for the individual accuracy from each fold.



(**a**) (**b**)

Figure 5. Confusion matrices, (**a**) BoW-based results, and (**b**) TF-IDF-based results.

Results given in Table 11 indicate that SVC achieves the best precision and recall scores of 78.20% each. The best accuracy is obtained by the stacking model, i.e., 73.50%, while the highest F1 score of 73.80% is achieved by the ET classifier. ROC-AUC curve for 5-fold cross-validation is shown in Figure 6. It shows that the performance of the proposed stacking model is superior to other machine learning models employed in this study.

Similar to 5-fold cross-validation, 10-fold cross-validation splits the data into 10 folds, where one fold is used for testing while the remaining nine folds are used for training. Ten iterations are needed for 10-fold cross-validation. In the end, the average accuracy is obtained using the accuracy for the individual fold.

The results given in Table 12 show that the best performance in terms of accuracy is provided by the proposed stacking model, which obtains 73.90% accuracy using 10-fold cross-validation. Similarly, the best F1 score of 73.90% is also obtained by the stacking model. On the other hand, the best precision and recall are obtained by the SVC. Figure 7 shows the AUC-ROC curve using 10-fold cross-validation. Results indicate that the proposed stacking model outperforms all other models with an AUC of 0.806.

Table 11. Results of 5-fold cross-validation.

Model	SVC	ET	Stacking	BNB	LR
Accuracy	71.90	73.00	73.50	72.00	71.50
Precision	78.20	73.30	72.90	68.50	75.00
Recall	78.20	73.30	72.90	68.50	75.00
F1 score	71.75	73.80	73.50	72.00	71.50

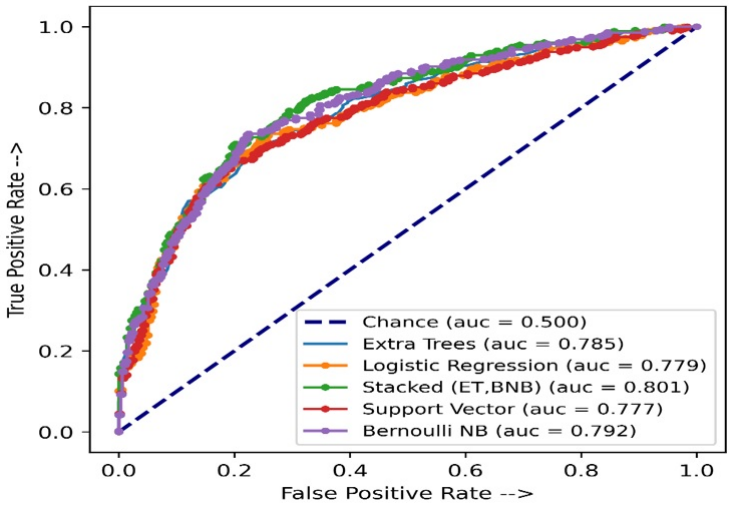


Figure 6. ROC-AUC using 5-fold cross-validation.

Table 12. Results of 10-fold cross-validation.

Model	SVC	ExT	Stacking	BNB	LR
Accuracy	72.50	73.00	73.90	72.70	72.60
Precision	78.60	73.90	73.50	69.00	75.40
Recall	78.60	73.90	73.50	69.00	75.40
F1 score	72.30	73.00	73.90	72.60	72.50

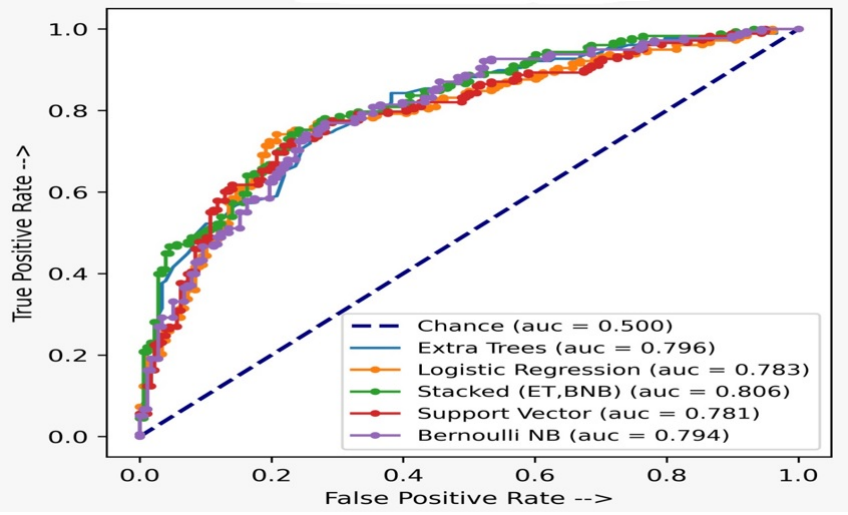


Figure 7. ROC-AUC using 10-fold cross-validation.

4.4. Results Using Deep Learning Models

This study also employs deep learning models for threatening text in the Urdu language, which include LSTM, GRU, CNN, and FCN. The experimental results for these models are provided in Table 13. The results suggest that the performance of deep learning models is poor as compared to machine learning models. The best accuracy of 65.40% is obtained by the LSTM model. Similarly, LSTM also shows the best F1 score of 65.40% and the best recall of 67.90%. The performance of CNN is the best regarding precision, which is 68.10% for threatening text detection. Deep learning models tend to show better performance when trained using a large dataset. However, the dataset used in this study is comparatively small and insufficient for deep learning models to obtain a good fit.

Table 13. Results of deep learning models.

Model	LSTM	GRU	CNN	FCN
Accuracy	65.40	65.10	65.20	53.60
Precision	62.40	61.90	68.10	51.70
Recall	67.90	67.70	65.70	55.30
F1 score	65.40	65.10	65.10	53.60
Standard deviation	0.07	0.07	0.007	0.01
ROC	0.73	0.73	0.72	0.73

Figure 8 shows the ROC-AUC curve for deep learning models. It confirms that the performance of LSTM is superior to other models. It has an AUC of 0.7304, while GRU and CNN perform marginally poor performance as compared to LSTM. FCN shows the worst performance with an AUC of 0.5449 only.

Figure 9 shows the confusion matrices for deep learning models.

LSTM has 700 correct predictions with 344 correct predictions for threatening tweets and 356 correct predictions for non-threatening tweets. FCN shows poor performance and has only 574 correct predictions. The performance of CNN and GRU is marginally poor than the LSTM. CNN has 698 correct predictions, 375 for threatening and 323 for non-threatening text, while GRU has a total of 697 correct predictions.

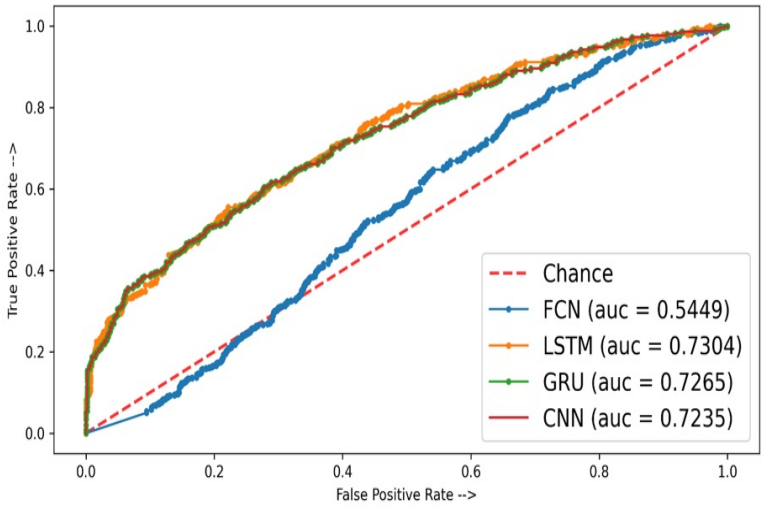


Figure 8. ROC-AUC for deep learning models.

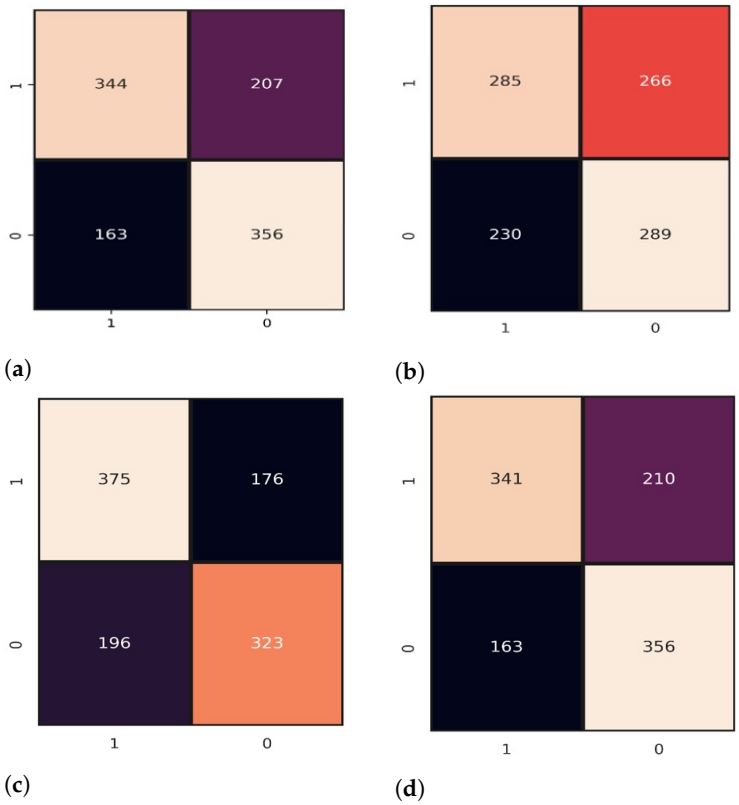


Figure 9. Confusion matrices for deep learning models, (**a**) long short-term memory model, (**b**) fully connected network, (**c**) convolutional neural network, and (**d**) gated recurrent unit.

4.5. Comparison with Base Study

This study uses the study from Amjad et al. [12] as the base study and uses it for performance comparison. Similarly, another study [24] on threatening language detection is also considered for comparison. Results given in Table 14 indicate that the proposed approach shows better performance than the base study regarding F1 score and accuracy.

Table 14. Performance comparison with base study.

Reference	Precision	Recall	F1 Score	Accuracy
Amjad et al. [12]	72.33	73.28	72.74	72.50
Mithun et al. [24]	-	-	54.57	-
Proposed study	70.84	75.65	73.99	74.01

The proposed study has outperformed the best existing studies, where TF-IDF with a stacked-based feature vector has attained better scores for precision, recall, F1-score, and accuracy. It has obtained 70.84, 75.65, 73.99, and 74.01 scores for precision, recall, F1, and accuracy, respectively.

The results attained from the proposed method are significant for Urdu-threatening language detection. The attained confusion matrix consists of four quadrants as shown in Figure 5. The employed evaluation measures precision and indicates the number of times the model predicts Urdu threatening tweets from threatening and non-threatening tweets. The recall corresponds to the number of times the model predicts threatening tweets from given tweets. The F1 score consists of a combination of precision and recall. The ROC-AUC curve states the area under the curve, and more area under the curve leads to improvement in the model. The results demonstrate that using verbs as features helps the model obtain a better relationship between the samples and labels, which enhances the performance of the

models. As a result, the performance of the proposed approach is better than the base and recent studies.

4.6. Discussion

Threatening text detection has become an important research area owing to the wide use of social media platforms like Twitter. Often, the use of abusive, offensive, and racist language offends different individuals, groups, and communities. Timely detection of threatening text can reduce the chances of this event. As a result, research on offensive and threatening text detection is carried out in different languages, particularly English, German, Danish, Arabic, etc. Unlike these high-resource languages, low-resource languages like Urdu do not have this luxury, and the work for threatening language detection is very limited. Only a few works are available for threatening text detection in the Urdu language, such as [12,24]. In this study, we enhance the performance of this task by utilizing a benchmark dataset compiled by Amjad et al. [12].

Contrary to existing studies, which utilize the features from preprocessed text alone, this study uses verbs from the preprocessed text as features. These features are combined with raw features from preprocessed text to make a single feature set on which machine learning models are trained. First, preprocessed text was converted into a feature vector, and then, using TF-IDF and BoW, verbs were extracted from the preprocessed text and converted into a feature vector. Besides using standalone machine learning models, a stacking model is proposed that uses ET and BNB as the based learners while LR is used as the meta learner. Moreover, deep learning models CNN, LSTM, GRU, and FCN are also utilized. Experimental results suggest that the performance of machine learning models is better than deep learning models. Due to small-sized datasets, deep learning models do not obtain a good fit and thus perform poorly. The proposed stacking model outperforms using bi-gram features with TF-IDF. It performs better than the benchmark study as well.

5. Conclusions and Future Work

Detecting threatening text detection on social media platforms is an important task and has been widely studied recently. Predominantly, the research focused on high-resource languages like English, and Urdu, being a low-resource language, is ignored. This study aims at providing automatic detection of threatening text in the Urdu language. Performance is improved following two contributions, the stacking model and the use of two feature vectors. Stacking uses ET and BNB as base learners and LR as the meta learner. Individual feature vectors are incapable of producing better results, so we stacked the verbs feature vector and the preprocessed feature vector to create a single feature vector. Experiments are performed using uni-gram, bi-gram, tri-gram, and tetra-gram features using TF-IDF and BoW. TF-IDF-based features with word-level bi-gram demonstrated the best performance when used with the proposed stacking model. The stacking method outperformed other methods with an F1 score of 75.65% and an accuracy of 73.99%. In the future, we intend to increase the dataset size to improve the performance of deep learning models. We also intend to employ embedding techniques such as Word2Vec, Doc2Vec, FastText, and BERT.

Author Contributions: Conceptualization, A.M. and M.S.F.; data curation, A.N.; formal analysis, A.M. and C.L.R.; funding acquisition, M.G.V.; investigation, A.N. and F.R.; methodology, M.S.F.; project administration, M.S.F., A.N. and M.G.V.; resources, M.G.V.; software, F.R. and C.L.R.; supervision, I.A.; validation, C.L.R. and I.A.; visualization, F.R.; writing—original draft, A.M.; writing—review and editing, I.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the European University of the Atlantic.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Dataset is avaialble at given link: https://github.com/MaazAmjad/Threatening_Dataset/blob/main/dataset.xlsx (accessed on 10 July 2022).

Conflicts of Interest: The authors declare no conflict of interests.

References

1. Statista. Number of Social Media Users Worldwide from 2018 to 2027. Available online: https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/ (accessed on 10 July 2022).
2. Chaffey, D.G. Social Media Statistics Research Summary. 2022. Available online: https://www.smartinsights.com/social-media-marketing/social-media-strategy/new-global-social-media-research/ (accessed on 10 July 2022).
3. Schmidt, A.; Wiegand, M. A survey on hate speech detection using natural language processing. In Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media, Valencia, Spain, 11–17 April 2017; pp. 1–10.
4. Wang, X.; Liu, Y.; Sun, C.J.; Wang, B.; Wang, X. Predicting polarities of tweets by composing word embeddings with long short-term memory. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, Beijing, China, 26–31 July 2015; Volume 1; pp. 1343–1353.
5. Del Vigna, F.; Cimino, A.; Dell'Orletta, F.; Petrocchi, M.; Tesconi, M. Hate me, hate me not: Hate speech detection on facebook. In Proceedings of the First Italian Conference on Cybersecurity (ITASEC17), Venice, Italy, 17–20 January 2017; pp. 86–95.
6. Behzadan, V.; Aguirre, C.; Bose, A.; Hsu, W. Corpus and deep learning classifier for collection of cyber threat indicators in twitter stream. In Proceedings of the 2018 IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 10–13 December 2018; pp. 5002–5007.
7. Kok, S.; Abdullah, A.; Jhanjhi, N.; Supramaniam, M. Ransomware, threat and detection techniques: A review. *Int. J. Comput. Sci. Netw. Secur* **2019**, *19*, 136.
8. Dionísio, N.; Alves, F.; Ferreira, P.M.; Bessani, A. Cyberthreat detection from twitter using deep neural networks. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019; pp. 1–8.
9. Malmasi, S.; Zampieri, M. Detecting hate speech in social media. *arXiv* **2017**, arXiv:1712.06427.
10. Oostdijk, N.; Halteren, H.v. N-gram-based recognition of threatening tweets. *International Conference on Intelligent Text Processing and Computational Linguistics*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 183–196.
11. Alakrot, A.; Murray, L.; Nikolov, N.S. Towards accurate detection of offensive language in online communication in arabic. *Procedia Comput. Sci.* **2018**, *142*, 315–320. [CrossRef]
12. Amjad, M.; Ashraf, N.; Zhila, A.; Sidorov, G.; Zubiaga, A.; Gelbukh, A. Threatening language detection and target identification in Urdu tweets. *IEEE Access* **2021**, *9*, 128302–128313. [CrossRef]
13. Razavi, A.H.; Inkpen, D.; Uritsky, S.; Matwin, S. Offensive language detection using multi-level classification. In *Canadian Conference on Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 16–27.
14. Chen, Y.; Zhou, Y.; Zhu, S.; Xu, H. Detecting offensive language in social media to protect adolescent online safety. In Proceedings of the 2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing, Amsterdam, The Netherlands, 3–5 September 2012; pp. 71–80.
15. Zampieri, M.; Malmasi, S.; Nakov, P.; Rosenthal, S.; Farra, N.; Kumar, R. Predicting the type and target of offensive posts in social media. *arXiv* **2019**, arXiv:1902.09666.
16. Xiang, G.; Fan, B.; Wang, L.; Hong, J.; Rose, C. Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. In Proceedings of the 21st ACM International Conference on Information and Knowledge Management, Maui, HI, USA, 29 October–2 November 2012; pp. 1980–1984.
17. Burnap, P.; Williams, M.L. Us and them: Identifying cyber hate on Twitter across multiple protected characteristics. *EPJ Data Sci.* **2016**, *5*, 1–15. [CrossRef] [PubMed]
18. Gómez-Adorno, H.; Enguix, G.B.; Sierra, G.; Sánchez, O.; Quezada, D. A Machine Learning Approach for Detecting Aggressive Tweets in Spanish. *IberEval@ SEPLN* **2018**, 102–107. Available online: http://ceur-ws.org/Vol-2150/MEX-A3T_paper2.pdf (accessed on 10 July 2022).
19. Mishra, P.; Del Tredici, M.; Yannakoudakis, H.; Shutova, E. Abusive language detection with graph convolutional networks. *arXiv* **2019**, arXiv:1904.04073.
20. Ishisaka, T.; Yamamoto, K. Detecting nasty comments from BBS posts. In Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation, Tohoku, Japan, 4–7 November 2010; pp. 645–652.
21. Rani, P.; Ojha, A.K. KMI-coling at SemEval-2019 task 6: Exploring N-grams for offensive language detection. In Proceedings of the 13th International Workshop on Semantic Evaluation, Minneapolis, MN, USA, 6–7 June 2019; pp. 668–671.
22. Polignano, M.; Basile, P.; De Gemmis, M.; Semeraro, G. Hate Speech Detection through AlBERTo Italian Language Understanding Model. In *NL4AI@ AI* IA*, Rende, Italy, 19–22 November 2019; pp. 1–13. Available online: http://ceur-ws.org/Vol-2521/paper-06.pdf (accessed on 20 July 2022).
23. Beyhan, F.; Çarık, B.; Arın, I.; Terzioğlu, A.; Yanikoglu, B.; Yeniterzi, R. A Turkish hate speech dataset and detection system. In Proceedings of the Language Resources and Evaluation Conference, Marseille, France, 20–25 June 2022; pp. 4177–4185.
24. Das, M.; Banerjee, S.; Saha, P. Abusive and threatening language detection in urdu using boosting based and bert based models: A comparative approach. *arXiv* **2021**, arXiv:2111.14830.

25. Pelle, R.; Alcântara, C.; Moreira, V.P. A classifier ensemble for offensive text detection. In Proceedings of the 24th Brazilian Symposium on Multimedia and the Web, Salvador, Brazil, 16–19 October 2018; pp. 237–243.
26. Jaleel, H.Q.; Stephan, J.J.; Naji, S.A. Textual Dataset Classification Using Supervised Machine Learning Techniques. *Eng. Technol. J.* **2022**, *40*, 527–538. [CrossRef]