

Received April 28, 2020, accepted May 11, 2020, date of publication May 15, 2020, date of current version May 28, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2994950

Automatic Detection of Offensive Language for Urdu and Roman Urdu

MUHAMMAD PERVEZ AKHTER¹, ZHENG JIANGBIN¹, IRFAN RAZA NAQVI¹, MOHAMMED ABDELMAJEED², AND MUHAMMAD TARIQ SADIQ³

¹School of Software and Microelectronics, Northwestern Polytechnical University, Xian 710072, China

²School of Computer Science and Technology, Northwestern Polytechnical University, Xian 710072, China

³School of Automation, Northwestern Polytechnical University, Xian 710072, China

Corresponding author: Muhammad Pervez Akhter (pervezbcs@gmail.com)


This work was supported in part by the Research and Development Plan of Shaanxi Province under Grant 2017ZDXM-GY-094 and Grant 2015KTZDGY04-01, and in part by the National Natural Science Foundation of China under Grant 61972321.

ABSTRACT In recent years, unethical behavior in the cyber-environment has been revealed. The presence of offensive language on social media platforms and automatic detection of such language is becoming a major challenge in modern society. The complexity of natural language constructs makes this task even more challenging. Until now, most of the research has focused on resource-rich languages like English. Roman Urdu and Urdu are two scripts of writing the Urdu language on social media. The Roman script uses the English language characters while the Urdu script uses Urdu language characters. Urdu and Hindi languages are similar with the only difference in their writing script but the Roman scripts of both languages are similar. This study is about the detection of offensive language from the user's comments presented in a resource-poor language Urdu. We propose the first offensive dataset of Urdu containing user-generated comments from social media. We use individual and combined n-grams techniques to extract features at character-level and word-level. We apply seventeen classifiers from seven machine learning techniques to detect offensive language from both Urdu and Roman Urdu text comments. Experiments show that the regression-based models using character n-grams show superior performance to process the Urdu language. Character-level tri-gram outperforms the other word and character n-grams. LogitBoost and SimpleLogistic outperform the other models and achieve 99.2% and 95.9% values of F-measure on Roman Urdu and Urdu datasets respectively. Our designed dataset is publically available on GitHub for future research.

INDEX TERMS Social media, offensive language detection, natural language Processing, machine learning, text processing.

I. INTRODUCTION

Cyberbullying using offensive language on the Internet has become a major problem among all age groups. Automatic detection of offensive language from social media applications, websites and blogs is a difficult but an important task. Social media platforms (like Twitter, YouTube, and Facebook) provide a common place to communicate and share user opinion about various topics like news, videos, and personalities. In the modern age, ease in the availability and popularity of Internet, laptops, tablets and cellphones, cyberbullying can take place anytime and anywhere which turning cyberbullying into a serious problem. There is no

The associate editor coordinating the review of this manuscript and approving it for publication was Shiping Wen .

eye-to-eye contact among users, which enables a user to present his opinion without any fear.

Social media applications and websites provide a central point of communication among the people of the world. People who are parted from each other based on geographic, religion, skin color, and culture (like division of Indian Sub-continent into India, Pakistan) often attack each other using offensive language [1], [2]. Users usually prefer and feel comfortable to use their native language than English to write their opinion, feedback or comments about online products, videos, articles [3]. Comments with offensive language words should not be visible to other users because it causes cyberbullying. Therefore, it is important to design an automatic system to detect, stop or ban offensive language before it is published online.

YouTube is a popular video website that contains multi-purpose videos. It is the most trafficked website after Google. YouTube has billions of hours of videos watched every day and 1.9 billion monthly active users. Recently, information processing, opinion mining, and behavior analysis from YouTube comments are popular research areas [4], [5]. India is the second source of YouTube traffic with 8.3% contribution and 2.45 million active users. T-Series channel of India is the number one YouTube channel with 2.98 billion views per month.¹ 73% of internet users of Pakistan use YouTube every month. ARY Digital is the number one Pakistani YouTube channel with 5,820,924,305 video views and 10,500,000 subscribers.² Since the division of the subcontinent, relationships between Pakistan and India are not good (because of multiple wars in various disputed areas). Both nations mostly understand their national languages (Hindi and Urdu) and criticize each other using offensive language on various topics (e.g., politics and entertainment) on YouTube. Urdu and Hindi are similar with the only difference being in their writing style [6], [7]. The Roman script is the common script that is easily readable, understandable and writable for both languages [8] (see section 2 for detail discussion). That is why automatic detection of offensive comments of Urdu and Roman Urdu is important and has a broader scope.

In recent years, machine learning techniques have been widely used for natural language processing (NLP) especially detection of offensive language and hate speeches from online user comments. For the Arabic language, [9], [10] used n-gram features and machine learning models to detect offensive language from YouTube comments. Ibrohim used machine learning models with n-gram features to detect abusive text from Indonesian social media [11]. For German offensive text detection, [12] used convolutional networks to detect offensive text from the Twitter message. [1] use LSTM and logistic regression to detect offensive comments written in Danish and English language. This paper investigates the performance of different machine learning techniques for Urdu and Roman Urdu text.

There are two main steps in supervised classification: feature extraction and classification. There are several feature extraction and classification techniques. The conventional features for offensive language detection are based on a blacklist [13], lexicon-based [14], pattern matching [15] and n-grams methods [16]. In past several studies used n-gram method for feature selection. N-gram features are based on a sequence of characters or words in the text. Several studies reported that n-gram models outperform the other models [9], [11], [13], [16]. N-gram approach has several applications like spelling correction, next word prediction and text translation.

To the best of our knowledge, offensive language detection from Urdu text comments has not been performed because Urdu is known as a resource-poor language; there is no

TABLE 1. A comparison of Roman Urdu and Urdu language text.

Features	Roman Urdu	Urdu
Alphabets characters	26 as the English	38
Font style	English	Nastaleeq
Grammars	No	Yes
Dictionary	No	Yes
Word order	No	Yes
Easy to type	Yes	No
Easy to read and understand	Yes	No

standard dataset publically available for offensive text detection. In this study, we design and annotate a dataset of offensive text comments written and make it publically available for future research. Individual character or word n-grams have been used in past studies to extract useful words from the offensive text but no research effort investigates the effectiveness of combined n-grams. In this study, we comparatively investigate the performance of both individual and combined character and word n-grams. We also compare seventeen classifiers from seven machine learning techniques for classification of offensive comments of both Urdu and Roman Urdu. Rest of the paper is organized as follows. Similarities and differences between Urdu and Roman Urdu are given in Section II. Related work is discussed in Section III. Methods and techniques used in the study are briefly described in Section IV. Experimental results, discussion and summary are given in Section V. The conclusion is given at the end in section VI.

II. URDU AND ROMAN URDU SCRIPT

Urdu is the national language of Pakistan and the official language in six states of India. Urdu has more than 300 million speakers all over the world. Urdu is written in Nastaleeq style that is a very complex and rich morphological script [6]. Urdu has many unique features like no capitalization, right-to-left, diacritics, context-sensitive, free word order [17]. In the past, researchers neglected Urdu because of its complex morphology, unique characteristics and the lack of linguistic resources [7].

Hindi is the national language of India. Hindi and Urdu languages are almost the same with the distinction of their writing script [6]. Roman Urdu is written in Roman script (i.e., with English alphabets). It is easy to write on computers, tablets, and cell phones with an English keyboard. Romanagari script of Hindi language is also written in Roman script. The Roman script of both languages is the same and easily readable and understandable by billions of people from India, Pakistan and other regions of the world [8]. Therefore, automatic detection of offensive language from the user's comments written in Urdu and Roman Urdu script is very vital.

A comparison of both scripts is given in Table 1 that shows that Roman Urdu is more flexibility than Urdu in reading,

¹ <https://www.businessofapps.com/data/youtube-statistics/>

² <https://www.socialbakers.com/statistics/youtube/channels/pakistan>

writing and understanding because it uses the alphabets and characters of English language and a person with little knowledge of English can read the text of Roman Urdu. There is no standard dictionary of Roman Urdu to know about a word is either a valid or invalid. Similarly, there are no grammatical rules of writing a sentence. As compare to Roman Urdu, Urdu script has its alphabets, dictionary and grammar that makes Urdu a difficult script for writing, reading than Roman Urdu. For example an English language sentence: “That is my school” can be written in Roman Urdu using different ways: “wo mera school ha” or “vo mira skool ha” but in Urdu it can only be written as: “ ”.

Several studies to detect offensive contents are in English and a few other languages like Arabic [2], German [12], Indonesian [11]. To the best of our knowledge, this is the first work to detect offensive language from Urdu text.

III. RELATED WORK

Recently, an increasing amount of attention of computational linguistic community has been given to detect offensive language and hate speeches from several online social media applications like YouTube [9], [18], [19], Twitter [2], [12], [15], Facebook [1] and blogs [20]–[22]. People from all over the world share their comments about the uploaded images, videos and products on social media platforms. Because of the difference in nationality, culture, religion, and race, user comments usually include offensive or hate words that cause cyberbullying among the users [14], [23]. Therefore, it is important to detect and remove offensive comments automatically. Various features of a language and the complexity of natural language constructs make this a more challenging task.

Automatic detection of offensive language from social media has become a trending topic of research in recent years. Several machine learning methods have been applied to the text of various languages. Lexicon-based approach was used to detect hate speech from websites such as blogs and forums [14], [22]. Bouazizi applied a pattern-based approach to detect sarcasm from Twitter posts and also compared the performance of the proposed method with Random Forest, SVM, k-NN and, maximum entropy classifier [15]. In the study of Lee, they detected abusive text by designing two lists of abusive words and non-abusive words [13]. Burnap used supervised classifiers Random forest and SVM to detect hate speech on Twitter [23]. Watanabe used a programmatic approach to performs hate speech detection from tweets [2]. All these studies employed only a couple of machine learning classifiers to compare the performance with proposed approaches but the power of machine learning classifiers have not been fully explored. Therefore, it is required to comparatively analyze the performance of various machine learning classifiers to detect offensive language from the text.

Performance of a classifier heavily depends on the number of features and the quality of the features selected by feature selection approaches. Several studies show that n-gram approaches at the character and word level are very effective

to detect offensive language than Bag of Word (BoW) [2], [15], [23]. [16] explored word n-grams to detect offensive language from tweets. [9], [19] employed word n-grams to detect offensive language from YouTube comments [20], [22], [24] also used word n-grams to detect offensive language from the comments collected from blogs and emails. Similarly, for character n-grams, [25], [26] used character n-grams to detect offensive language from tweets. [1] also used BoW and character n-grams to detect offensive and hate speech from tweets and comments collected from Twitter, Reddit and Facebook. All these studies use either word n-grams or character n-grams but the power of both techniques has not been explored yet. [11], [13] use a character n-grams and word n-gram approaches to detect abusive text by creating a list of abusive words. [11] shows that uni+bi-grams features performed best with NaiveBayes. For Roman Urdu sentiment analysis task, the uni-gram approach showed the best performance on YouTube comments [27]. In this study, we applied both character n-gram and word n-grams to detect offensive language from text comments of Urdu and Roman Urdu.

Until now, most of the research has focused on resource-rich languages like English while resource-poor languages could not gain the attention of the researchers because of the lack of language resources like annotated datasets. Recently, various machine learning techniques have been used to detect offensive and hate speech detection from social media text of different languages other than English. In the study of [22], a classifier ensemble techniques were used to detect offensive text from the web pages of the Portuguese language. For the Arabic language, [9], [10] used n-gram methods to detect offensive language from online comments. Ibrohim uses naïve bayes, decision tree and support vector machine on n-gram features to detect abusive text taken from Indonesian social media [11]. Sigurbegsson applied machine learning models to detect offensive language from the Danish language [1] and Schneider also used various models of machine learning to classify German language tweets to detect offensive text [12]. In this study, we design a dataset of comments of a resource-poor language Urdu from YouTube videos and apply and compare machine learning models to automatically detect abusive comments.

From the last decade, social media platforms are the popular sources to collect public opinion, views, and trends about some person, product, video etc. [11] collected a Twitter dataset of Indonesian language tweets for abusive language detection. After cleaning the dataset and removing the duplicates, 2,500 tweets were used in final experiments. [10] designed a dataset of 16,000 comments from YouTube to detect offensive language. [22] used 1,250 comments of Portuguese language collected from Brazilian websites to detect offensive text. In this study, we collected comments of Urdu language from YouTube and design a dataset to detect offensive language. A summary of the work reviewed in this section is given in Table 2.

TABLE 2. Comparison of past studies about offensive language detection from social media comments

Reference	Language	Platform	Feature extraction methods	Classification models
[9]	Arabic	YouTube	Word n-grams	SVM
[22]	English, Portuguese	Twitter, Blogs	hateword2vec, hatedoc2vec, Unigram	NB and SVM
[13]	English	Twitter, Articles	Abusive and non-abusive word list	Unsupervised learning
[36]	English	Twitter	Word n-grams, hate or non-hate words list	SVM (linear, polynomial, radial
[11]	Indonesian	Twitter	Word and char. n-grams	NB, SVM, RF
[1]	Danish, English	Twitter, Reddit, Facebook	BoW, char. n-grams	LR, BiLSTM
[12]	German	Twitter	Twitter and Wikipedia embedding,	CNN
[16]	English	Twitter	Word n-grams (1-8)	SVM
[20]	Japanese	Blogs	Word n-grams (1-5)	SVM
[37]	English	Twitter	BoW, Word n-grams (1-3) and Char. n-grams (3-8)	NB, LR, SVM, RF, Gradient Boosted Trees, CNN, RNN
[21]	English	News Group	Complement NB, Multinomial Updateable NB,	Decision Table NB (DTNB)
[25]	English	Twitter	Char. n-gram, BoW	Logistic Regression, SVM, CNN
[19]	English	YouTube	Lexical Syntactic Feature, Word n-grams (2, 3, 5), BoW	NB, SVM
[26]	English	Twitter	Char. n-grams (1-4)	LR, Graph Convolutional Network
[41]	English	Twitter	Word Unigram	SVM, BiLSTM, CNN

A. RESEARCH GAPS OF THE STUDY

Based on the literature discussed above and given in Table 2, we recognized the following gaps in offensive language detection from user comments on social media.

- *Dataset preparation:* It can be seen from Table 2 that the English language has several research studies as compared to other resource-poor languages because of the available language resources like datasets. Urdu is also a resource-poor language and to the best of our knowledge, there is no public annotated dataset of Urdu that can be used for offensive language detection. In this study, we collect comments and annotate them manually to design a dataset from YouTube videos.
- *Feature selection:* From Table 2, it can be seen that the n-gram approach is effective and popular in the detection of offensive language. All the studies used either word n-grams or character n-gram methods. None of the studies compares the performance of character n-gram with word n-gram to detect offensive language. Moreover, none of the studies explores the effects of these n-gram approaches after combining them.
- *Classification models:* previous research studies used few of the machine learning classifiers (one to four). None of the studies explores the performance of various machine learning classifiers to classify text into offensive and non-offensive. In this study, we have found that regression-based classifiers achieved better performance than other popular classifiers.

- *Urdu and Roman Urdu scripts:* Although several studies process and compare several feature selection methods [28] and classifiers [17] for Urdu text document classification and Roman Urdu text classification [27]. However, to the best of our knowledge, it is the first study that explores, evaluates and compares machine learning methods to process and detect offensive language from both Urdu and Roman Urdu text.

IV. MATERIAL AND METHODS

In this section, we explain the proposed combined n-gram approach used to detect offensive language from Roman Urdu and Urdu text. First, we collect comments from YouTube videos and manually annotate these comments into offensive or non-offensive to design Urdu Offensive Dataset (UOD). Second, we clean and tokenize both Urdu and Roman Urdu datasets. Third, we extract six types of character n-grams features and six types of word n-grams (uni-gram, bi-gram, tri-gram, uni+bi-gram, bi+tri-gram, and uni+bi+tri-gram). Forth, using generated n-grams in the previous step, we classify the comments into offensive and non-offensive comments using seventeen supervised classifiers that belong to seven machine learning techniques. Last, we evaluate and compare the performance of these classifiers. All the n-grams and classifiers used in this study are shown in Figure 1.

A. OFFENSIVE LANGUAGE DATASETS

There are many datasets are available to detect offensive text of resource-rich languages like English as shown in Table 2.

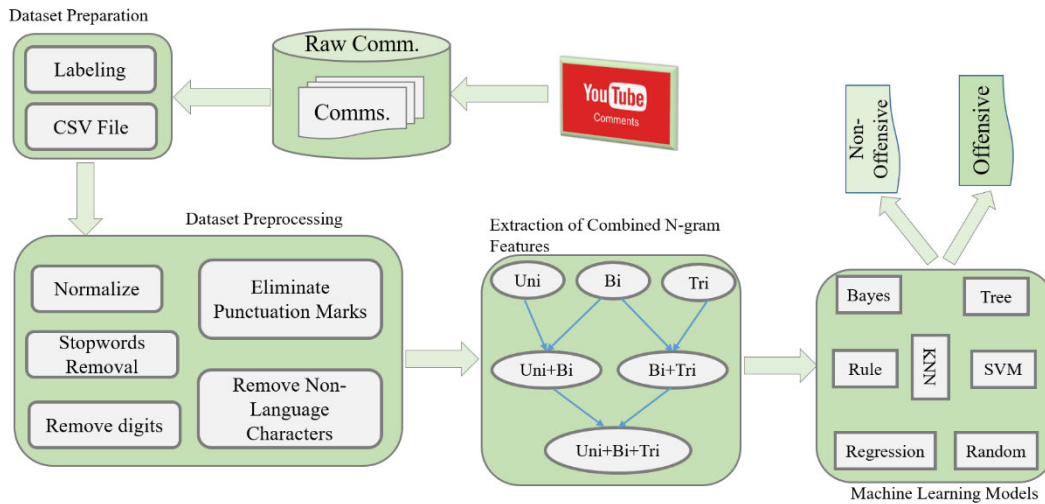


FIGURE 1. The proposed methodology to detect offensive language that shows the number of steps used for dataset design and preparation, preprocessing, feature selection and classification.

TABLE 3. Datasets of Roman Urdu and Urdu language text.

Dataset	#comments	#classes	#offensive	#non-offensive
Urdu	2,171	02	1,109	1,062
Roman Urdu	10,000	02	5,000	5,000

In this study, we used two datasets of Roman Urdu and Urdu languages. For Roman Urdu, we use a dataset that is publically available at GitHub. For Urdu language, we design a dataset from YouTube videos because there is no publically available dataset. The detail discussion of both datasets is given in this section and a summary is given in Table 3.

1) ROMAN URDU DATASET

A Roman Urdu dataset is publically available at GitHub.³ This dataset contains 1,47,000 user comments collected from multiple videos from YouTube. This dataset is available in comma-separated file (CSV) format. Each comment is labelled either offensive or non-offensive. In this study, we use the subset of this dataset and randomly extracted ten thousand comments from this dataset.

2) URDU OFFENSIVE DATASET (UOD)

Because there is no standard dataset of Urdu that can be used for offensive language detection. Therefore, we collect 2,171 comments and design a dataset of Urdu language from YouTube videos. All the comments are manually collected from political, entertainment, sports, and religion videos uploaded by India and Pakistan. Three annotators that are graduate students and are local speakers of the Urdu language annotate Dataset. We provided them with a set of comments and a question “a comment either is offensive or non-offensive?” We clean the dataset by removing

non-Urdu words and characters, URLs, numbers and special characters from each comment. Our designed dataset and a subset of Roman Urdu dataset used in this study are publically available in CSV formats on GitHub.⁴

B. WORD N-GRAM AND CHARACTER N-GRAM

Features play an important role in the classification task by classifiers and are extracted from the text under analysis. n-gram features consist of a contiguous sequence of n words or characters. For natural language processing, n-gram is a popular and useful technique that is used to assign a probability value to a word or a sequence of words from the text. Classifiers use the assigned probability value to classify the text. Followings are the popular n-grams:

- Uni-gram: a feature made of single word or character (i.e. n = 1)
- Bi-gram: a feature based on two contiguous words in the text (i.e. n = 2)
- Tri-grams: it is based on three contiguous words or characters (i.e. n = 3)

The number of n-grams from a sentence can be calculated as given below:

$$Ngrams = X - (N - 1) \tag{1}$$

where X is the number of words (or characters) in a sentence and N is the number of contiguous words (or characters). Examples of extracting n-grams from a sentence of Roman Urdu and Urdu are given in Table 4. A sentence has five words. There are five uni-grams, four bi-grams and three tri-grams.

Character n-grams are another feature type that represents a text as a sequence of characters. Character n-gram is different from word n-gram where n is the number of contiguous

³ <https://github.com/shaheerakr/roman-urdu-abusive-comment-detector>

⁴ <https://github.com/pervezbcs/Urdu-Abusive-Dataset>

TABLE 4. Examples of designing n-grams from Roman Urdu and Urdu sentence.

N-grams	Roman Urdu	Urdu
Sentence	Wo ek chotiya banda ha	وہ ایک چوتیا بندہ ہے
Unigram	'wo', 'ek', 'chotia', 'banda', 'ha'	'وہ', 'ایک', 'چوتیا', 'بندہ', 'ہے'
Bigram	'wo ek', 'ek chotiya', 'chotiya bnda', 'banda ha'	'وہ ایک', 'ایک چوتیا', 'چوتیا بندہ', 'بندہ ہے'
Trigram	'wo ek chotiya', 'ek chotiya banda', 'chotiya banda ha'	'وہ ایک چوتیا', 'ایک چوتیا بندہ', 'چوتیا بندہ ہے'

TABLE 5. No. of character-level and word-level n-grams extracted from both datasets.

N-gram/Type	Word		Char	
	Roman Urdu	Urdu	Roman Urdu	Urdu
Uni-gram	3316	2705	28	50
Bi-gram	4179	2430	665	864
Tri-gram	3277	17774	2320	1281
Uni+Bi-gram	4179	1921	691	911
Bi+Tri-gram	2196	2449	1227	863
Uni+Bi+Tri-gram	1496	1853	1186	1259

characters instead of words. In this study, we employed uni-gram, bi-gram, and tri-gram to extract features from the comments. We also combined n-grams to extract complex features like uni+bi-gram, bi+tri-gram, and uni+bi+tri-grams. Here '+' character show the group of one n-gram with another [11], [29]. For example, uni+bi-gram features means all the n-grams of length one and two (see Table 4). The number of extracted character n-grams and word n-grams from both datasets are given in Table 5.

C. MACHINE LEARNING TECHNIQUES

In this section, we give a brief introduction of machine learning techniques used in this study to detect offensive language from Urdu and Roman Urdu scripts. We used seventeen classifiers from seven machine learning techniques. We used a popular data mining tool WEKA [30] for the experiments. For a detail description of these classifiers, please see the online documentation of WEKA. However, here we describe these classifiers.

1) BAYESIAN MODELS

These models are based on Bayes theorem and conditional probability. Bayes models are simple, useful and easy to build for large datasets. Bayes theorem is as follows:

$$P(C|D) = \frac{P(D|C) * P(C)}{P(D)} \quad (2)$$

where C and D are two events and $P(D) \neq 0$. $P(D)$ and $P(C)$ are the prior probabilities of observing D and C without regard to each other. $P(D|C)$ is the probability of observing event D given that C is true. We used two Bayes models: Naïve Bayes (NB) and Bayes Network (BayesNet).

- *BayesNet*: it models a directed acyclic graph (DAG) with the conditional probability distribution of each node of

that graph. The arc between the nodes represents the probabilistic dependency among them. Nodes represent the attributes of the dataset. BayesNet calculates and gives the posterior probability distribution of the classification node given the values of the other nodes. In learning, BayesNet performs two tasks: learning the graphical structure and then learning the parameters of that structure.

- *Naïve Bayes (NB)*: it has a simple structure than BayesNet where the classification node is the parent node of all the other nodes. In learning, NB assumes that all the features of a dataset are independent of each other. Because of this assumption, NB is efficient and easy to construct.

2) NEAREST NEIGHBORS

Assigns a label to an instance based on the labels of its k-nearest neighbors. Its performance is based on the value of k and the similarity measure that is used to predict the class of an instance. For a large dataset, it is computationally expensive. In this study, we used Instance-Based Learning (IBk) model with Euclidean Distance to measure the similarity among the k instances.

3) TREE

These models construct a tree from the given data. Nodes of a tree represent attributes or features of an example with its importance to classify it. Leave nodes of the tree represent classes in the data. Easy to interpret but complex and time-consuming for a high dimensional dataset. We used three tree-based models: Hoeffding Tree, J48, and Reduced Error Pruning Tree (REPTree).

- *Hoeffding Tree*: use Hoeffding bound to calculate a certain level of confidence score and to decide how many examples are needed to achieve that confidence.
- *J48*: is a decision tree-based model that is an extension of the ID3 algorithm. It constructs a decision tree from the training data and prunes it.
- *REPTree*: produces a fast decision tree using information gain and prune the produced tree using reduce error pruning. It can produce multiple trees and choose the fine one.

4) RANDOM

Random algorithms construct a tree by considering k randomly chosen attributes at each node. Random behavior of

a model helps to reduce both errors due to bias and error due to variance [31]. We used Random Tree and Random Forest models that are described below:

- *Random Tree*: is like a decision tree but it does not use all the features of a dataset to construct a tree. It randomly selects some features to construct a decision tree for the classification task.
- *Random Forest*: It consists of multiple decision trees. Each tree is constructed using a subset of features. Each tree is used for the classification task but the final classification is performed using aggregating the classification results (like using majority voting) of all the trees.

5) REGRESSION

Use a statistical process to measure the relationship between a dependent variable and one or more independent variables. We use three regression-based models called linear multinomial logistic regression with a ridge estimator (Logistic), additive logistic regression (LogitBoost), and regression model with simple regression function (SimpleLogistic).

- *Logistic*: it builds a model using multinomial logistic regression with ridge estimator. It replaces missing attributes and transforms nominal attributes into numeric attributes. It can also handle weighted and non-weighted instances.
- *LogitBoost*: is based on AdaBoost procedure that trains the model on weighted samples. It assigns higher weights to misclassified samples. After performing a sequence of steps, the final classifier is the linear combination of classifiers at each stage [32].
- *SimpleLogistic*: by using LogitBoost algorithm, it fits a multinomial logistic regression model. In training or learning the dataset, in each iteration, it adds one simple linear regression model per class into the logistic regression model. It stops adding linear regression models when cross-validation error no longer decreases.

6) SUPPORT VECTOR MACHINE

Learns n-dimensional hyperplane that separates examples into classes. It can classify both linear and non-linear data. High memory and poor interpretability are its drawbacks [33]. Kernel function in SVM is used to analyze the patterns in data. We apply four mostly used kernels: polynomial, radial, sigmoid, and linear kernels.

- *Polynomial Kernel*: For degree “d,” the polynomial kernel can be defined as:

$$K(x_i, x_j) = \{x_i^T x_j + c\}^d \quad (3)$$

where x_i and x_j are the input space vector and x_i^T is the transpose of x_i . c is a parameter used for the trade-off between the highest order and lowest order polynomial.

- *Radial Basis Function (RBF) Kernel*: is a real-valued function, whose value depends upon the distance from the origin. RBF kernel can be defined as follows:

$$K(x_i, x_j) = \exp(-\gamma(x_i - x_j)^2) \text{ for } \gamma > 0 \quad (4)$$

where the value of γ can be used as $1/2\sigma^2$ where σ^2 is the variance of input data.

- *Sigmoid Kernel*: The sigmoid kernel function can be defined as:

$$K(x_i, x_j) = \tanh(ax_i^T x_j + b) \quad (5)$$

$a > 0$ is the scaling parameter for the input data, and b is the shifting parameter that controls the threshold of mapping.

- *Linear Kernel*: can be represented as below where x_i, x_j , and x_i^T are same as defined in the polynomial kernel.

$$K(x_i, x_j) = x_i^T x_j \quad (6)$$

7) RULE-BASED

These models implement a propositional rule learner. For each label, exactly one rule is defined. A rule is to build by trying every possible value of each attribute and select the condition with the highest information gain. In this study, OneR and JRip are used.

- *OneR*: is a simple rule-based classifier that constructs one rule for each predictor in rules learning. The rule with minimum error is selected for the final classification.
- *JRip*: is based on propositional rule learner RIPPER (Repeated Incremental Pruning to Produce Error Reduction) that incrementally learns rules and then optimize these rules. First, it constructs rules for all positive instances and then prunes them. It is efficient on a large noisy dataset for classification task [18].

D. PERFORMANCE EVALUATION MEASURES

To measure the classification performance, we used the most common performance measures used for classification task: F-measure [34], [35]. F-measure can be calculated using True Positive (TP), False Positive (FP), False Negative (FN) of a confusion matrix [17]. TP is the number of comments correctly predicted as the positive class (offensive). FP is the number of comments predicted wrongly as the positive class when it was not. FN is the number of comments predicted wrongly as the negative class (non-offensive) when it was not. F-measure is the harmonic mean of precision and recall values. F-measure can be calculated as:

$$F - \text{measure} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (7)$$

where precision and recall can be calculated as given below:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (8)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (9)$$

V. RESULTS AND DISCUSSION

In this section, the experimental results are discussed. During experiments, we investigated the following questions:

- Which n-gram technique (word or character) outperforms others?
- Are individual n-grams better than combined n-grams?
- Which machine learning technique is the best for offensive language detection?
- In each technique, which classifier is the best to classify offensive and non-offensive comments?

Because the datasets are not divided into training, testing or validation sets, we used ten-fold cross-validation to train and test the machine learning models [17], [36]. All the experiments have been performed using open source and publically available software WEKA.

A. WORD N-GRAMS

In this section, we evaluate the performance of our models on six types of word n-grams. For Roman Urdu, as shown in Figure 2, it can be seen that uni-gram is the best n-gram than other individual or combined n-grams because most of the classifiers achieved maximum performance using uni-gram. It is because the single word or uni-gram are prominent in offensive comments such as “harami”, “kutta”, and “chotiya”. It also endorsed the findings of [2]. Regression-based classifier SimpleLogistic and SVM linear outperform the other models on uni-gram as concluded in [16] and achieved 94.2% F-measure value. For combined word n-grams, after uni-gram, uni+bi+tri-gram shows better performance than the other four n-grams. SVM polynomial shows better result than other models on uni+bi+tri n-grams and achieves 92% score of F-measure. OneR shows worse than all other models as it constructs one rule only from the dataset as compare to JRip that incrementally learns rules and optimizes that rule [18].

To detect offensive language from Urdu dataset, again uni-gram is more accurate and superior over other n-grams. Tree-based REPTree model outperforms the other models on uni-gram. Because of information gain and reduce error pruning techniques to build and prune the tree, REPTree achieves the highest value 94.7% of F-measure. SVM sigmoid outperforms better with 85.2% F-measure on bi-gram and SVM polynomial performs superior with 74.1% score on tri-gram features. For combined n-grams, again REPTree achieves high performance on both uni+ bi-gram and uni+bi+tri-gram features and achieves 94.5% and 94.4% F-measure value respectively. Hoeffding tree failed to achieve a certain level of confidence score and performed the worse than other classifiers except better than OneR classifier. As compare to other techniques, overall SVM models show better performance than other techniques using all the n-grams except uni-gram to detect offensive language from Urdu comments.

If we analyze the results of word n-grams as shown in Figure 2 and Figure 3, we conclude that uni-gram is the best feature from six types of n-grams (individual and combined) to detect the offensive language in the text of both Roman Urdu and Urdu datasets and it endorses the finding of [27]. From all seventeen classifiers, SimpleLogistic and SVM

linear outperform the other models on Roman Urdu while REPTree outperformed on Urdu. Experimental results also show that individual feature of n-gram performs better than combined features of different length of words or n-grams. For both of the datasets, both Hoeffding Tree and OneR perform worse than all the other models because of the complex and morphological features of Urdu and the free writing style of Roman Urdu. Therefore, OneR failed to construct an optimum rule and Hoeffding Tree failed to calculate a certain level of confidence score to classify offensive and non-offensive comments from both datasets.

B. CHARACTER N-GRAMS

For character n-grams, F-measure scores achieved by each classifier on Roman Urdu are shown in Figure 4. Regression-based model LogitBoost outperforms the other models and achieves 99.2% value of F-measure on the tri-gram feature. LogitBoost also shows maximum performance on all the individual or combined n-grams except uni-gram. SVM radial shows better performance than other models on uni-gram and achieved 77.3% F-measure that is 21.9% less than LogitBoost performance. On combined n-grams, LogitBoost again outperforms the others. It achieves 96.0%, 98.6% and 98.4% values of F-measure on uni+bi-gram, bi+tri-gram and uni+bi+tri-gram feature respectively. Hoeffding tree and OneR perform the worse than the other models the same as in the case of word n-grams as shown and discussed in the previous section.

For Urdu dataset, regression-based model SimpleLogistic outperforms the others models on tri-gram feature and achieves 95.9% F-measure score. SVM radial and polynomial show better results than other models on uni-gram and bi-gram as it is shown in Figure 5. For combined n-grams, again, uni+bi+tri-gram feature outperforms than others and SVM polynomial achieved 95.5% values of F-measure. Random Tree and OneR models perform worse than other models on all the features.

After the analysis of both Figure 4 and Figure 5, for character n-grams, we conclude that character tri-gram is the best feature and very helpful in the detection of offensive language for both Roman Urdu and Urdu datasets. It is because one or two characters do not design meaningful words of Urdu and Roman Urdu script. These words are usually known as stopwords. The study of [17] concludes that stopwords of Urdu decrease the performance of classifiers. The performance of regression-based models LogitBoost and SimpleLogistic are outstanding at character n-grams features except for uni-gram. Again, OneR performs worse than other models. For word n-grams, uni-gram approach outperforms the others on both datasets. For character n-grams, tri-gram outperforms the others on both datasets.

C. WORD N-GRAMS VS. CHARACTER N-GRAMS

If we analyze the performance of both words n-grams and character n-grams on Urdu and Roman Urdu datasets, we conclude that individual n-grams are better than combined

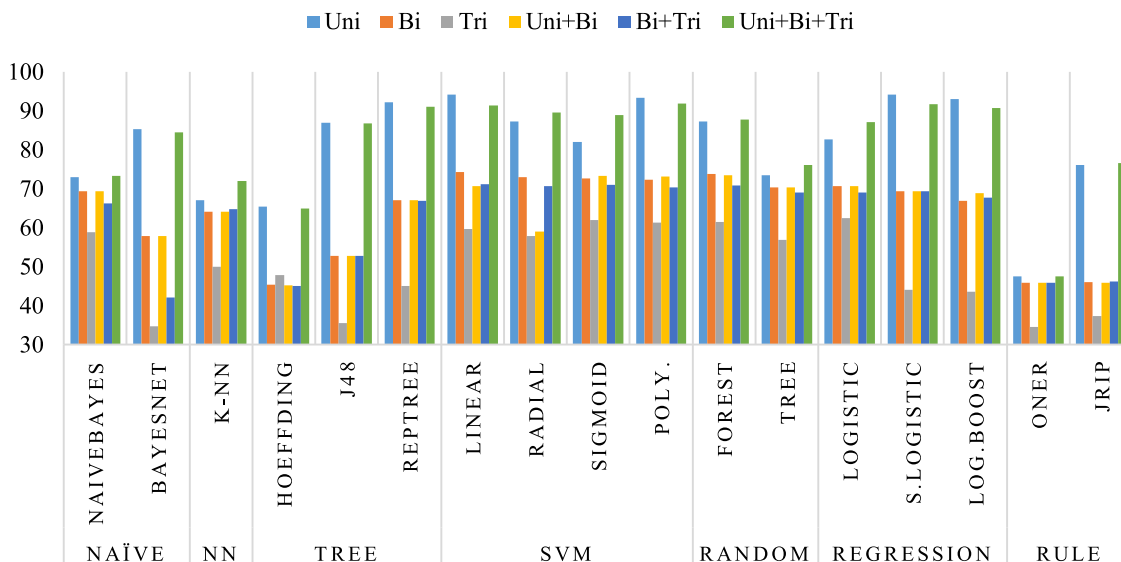


FIGURE 2. F-measure values of word-level n-grams on Roman Urdu dataset.

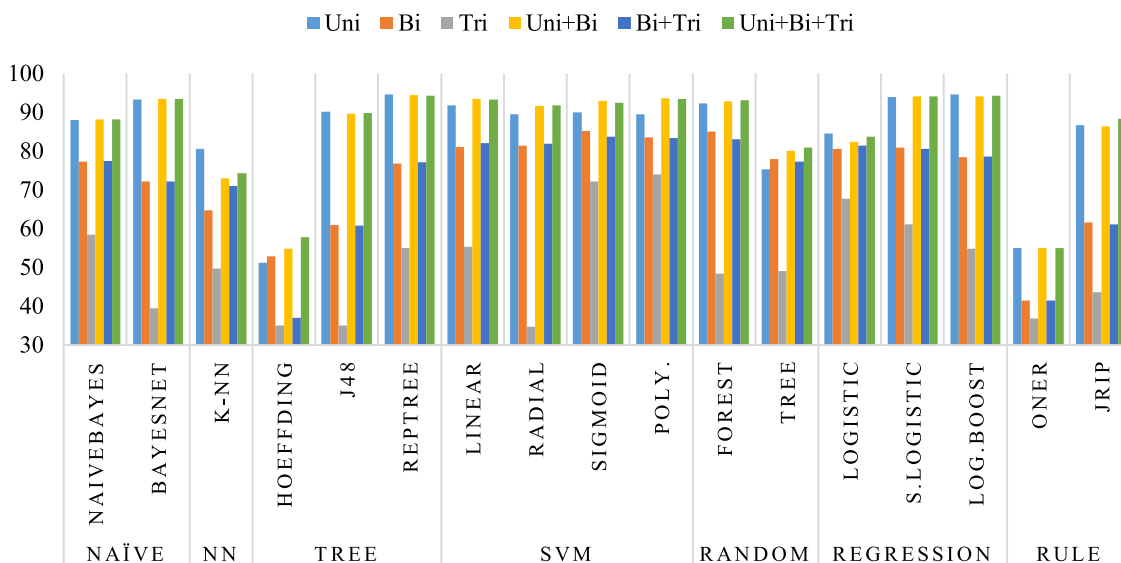


FIGURE 3. F-measure values of word-level n-grams on Urdu dataset.

n-grams. It can also be concluded that character n-gram shows better performance than word n-grams to detect offensive language and it endorsed the findings of [1], [37] on offensive text detection on English and Danish languages. For character n-grams, tri-gram is the most effective n-gram than other character n-grams (individual or combined). Character tri-gram achieved 95.9% and 99.2% F-measure values on Urdu and Roman Urdu respectively. Learning the complex morphology of Urdu and the insufficient number of samples in Urdu dataset are affecting the performance of machine learning models than Roman Urdu. For word n-grams, the performance of uni-gram is the best n-gram than other word n-grams. Word uni-gram achieved 94.2% F-measure on Roman Urdu and 94.7% values of F-measure on Urdu. We have also compared the combined n-grams of word

n-grams or character n-grams. For combined word n-grams, uni+bi+tri-gram and uni+bi-gram perform better than other word n-grams as they achieved 92% and 94.5% F-measure values on Roman Urdu and Urdu datasets respectively. At combined character n-grams, bi+tri-gram and uni+bi+tri-gram features achieved 98.6% and 95.5% F-measure scores and showed the best performance on Roman Urdu and Urdu respectively.

D. PERFORMANCE COMPARISON OF MACHINE LEARNING MODELS

After the analysis of n-gram methods, in this section, we compare the overall performance of machine learning models to classify comments into offensive or non-offensive comments. Table 6 shows the maximum F-measure scores of each model

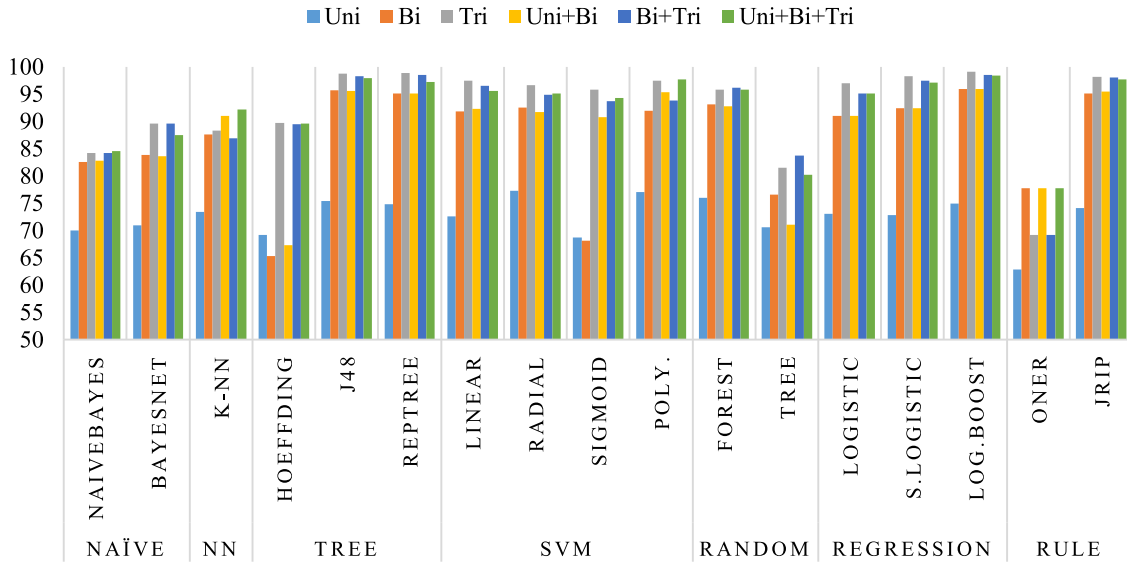


FIGURE 4. F-measure values of character n-grams on Roman Urdu dataset.

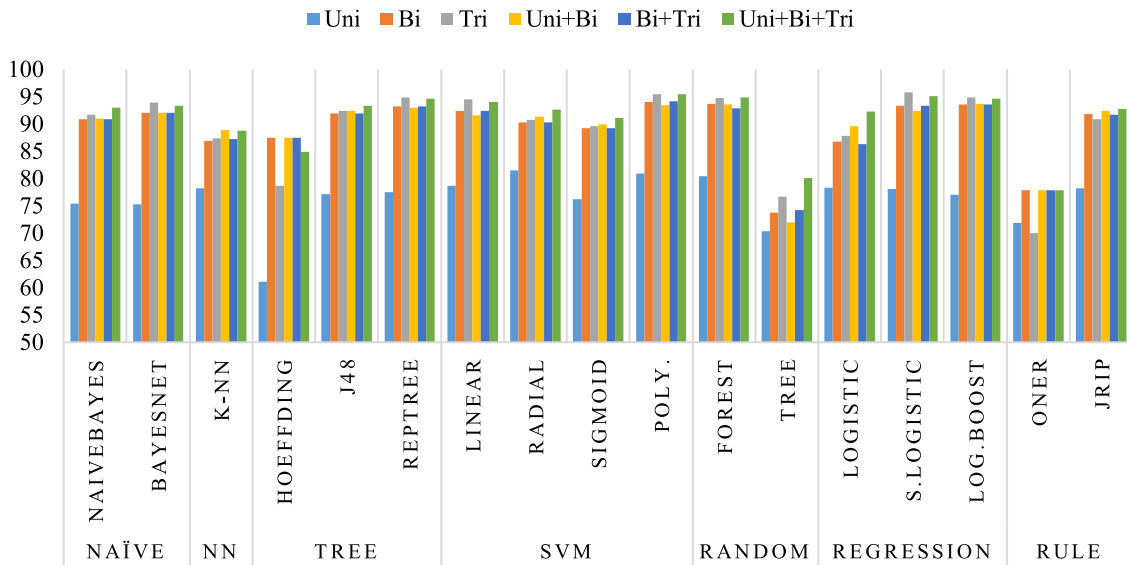


FIGURE 5. F-measure values of character-level n-grams on Urdu dataset.

achieved on both character n-grams and word n-grams. All the models except k-NN and SVM show the best performance with the default parameters as given in the WEKA. k-NN with $k = 11$ and $k = 6$ achieves the best performance on Roman Urdu and Urdu respectively. SVM radial with $g = 0.0$, sigmoid with $g = 0.1$, and polynomial with $d = 2$ and $g = 0.1$ achieve the best F-measure scores on Urdu. For Roman Urdu, SVM radial and sigmoid shows high performance with $g = 0.0$ but SVM polynomial with $d = 2$ and $g = 0.3$ shows the best performance.

From the analysis of values in Table 6, we conclude that regression-based models are the most effective than other models on both datasets with character tri-gram features.

LogitBoost achieved 99.2% F-measure on Roman Urdu while SimpleLogistic outperforms the others on Urdu dataset and achieved 95.9% score of F-measure. Overall, character n-grams perform better than word n-grams with all the machine learning models except Random Tree and SVM sigmoid. Character tri-gram shows superior performance with most of the models on both datasets. Combined n-gram features at both word n-grams and character n-grams do not perform well with most of the classifiers.

For Bayes theorem models, BayesNet achieves better performance than NB on character bi-gram and tri-gram features on both Roman Urdu and Urdu datasets respectively. Our finding endorsed the findings of [17] about BayesNet in

TABLE 6. Summary of results achieved by seven machine learning techniques. Maximum values of F-measure achieved by a classifier on type of n-gram both datasets.

Technique	Classifier	Roman Urdu		Urdu			
		F-measure	N-grams	Type	F-measure	N-grams	Type
Bayes	NB	84.6	U+B+T	Char	93.0	U+B+T	Char
	BayesNet	89.6	B, B+T	Char	94.0	T	Char
NN	k-NN	92.2	U+B+T	Char	88.9	U+B	Char
Tree	Hoeffding Tree	89.7	T	Char	87.5	B, U+B, B+T	Char
	J48	98.8	T	Char	93.4	U+B+T	Char
	REPTree	98.9	T	Char	94.9	T	Char
SVM	Linear	97.5	T	Char	94.6	T	Char
	Radial	96.7	T	Char	92.7	U+B+T	Char
	Sigmoid	95.8	T	Char	93.1	U+B	Word
	Polynomial	97.7	U+B+T	Char	95.5	U+B+T	Char
Random	Random Forest	96.2	B+T	Char	94.9	U+B+T	Char
	Random Tree	83.8	B+T	Char	80.9	U+B+T	Word
Regression	Logistic	97.0	T	Char	92.4	U+B+T	Char
	SimpleLogistic	98.3	T	Char	95.9	T	Char
	LogitBoost	99.2	T	Char	94.9	T	Char
Rule	OneR	77.8	B, U+B, U+B+T	Char	77.9	B, U+B, B+T, U+B+T	Char
	JRip	98.2	T	Char	92.8	U+B+T	Char

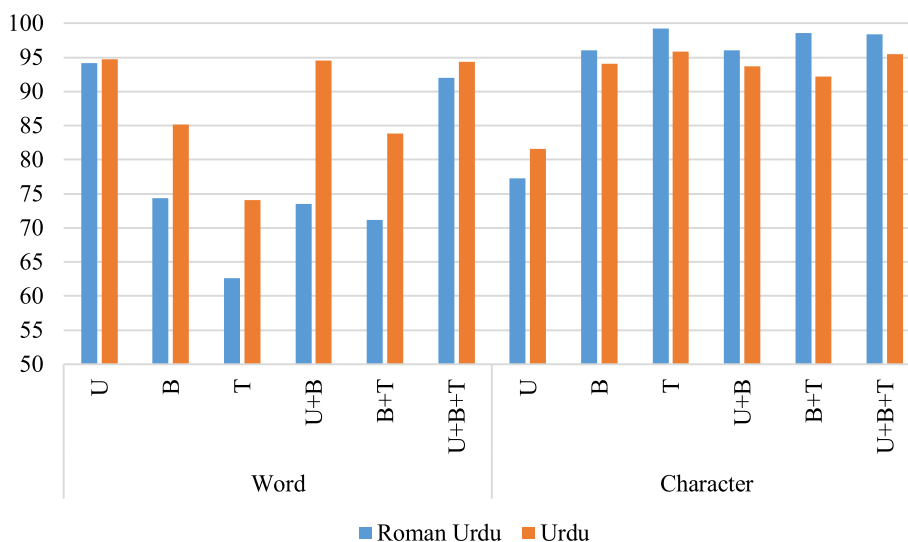


FIGURE 6. Performance comparison of word-level and character-level n-grams on both datasets.

TABLE 7. Time (in seconds) taken to build the model.

Dataset	BayesNet	k-NN	REPTree	SVM Poly.	RandomForest	Simple Logistic	LogitBoost	JRip
Time	1.09	0.01	29.89	70.84	80.01	145.55	742.11	194.66

the classification of Urdu text documents. k-NN shows the highest performance with $k = 6$ and $k = 11$ using combined character n-grams on both Roman Urdu and Urdu datasets respectively. For tree-based models, REPTree outperforms the j48 and Hoeffding Tree with character tri-gram on both datasets. SVM with polynomial kernel shows better performance than linear, radial and sigmoid kernels using combined

character n-gram on both datasets. From random classifiers, Random Forest gives better results than Random Tree on both datasets using combined character n-grams. In regression-based classifiers, SimpleLogistic and LogitBoost outperform the other sixteen classifiers using character tri-grams on Urdu and Roman Urdu. For the rule-based technique, JRip is better to classify offensive language than OneR. JRip achieves

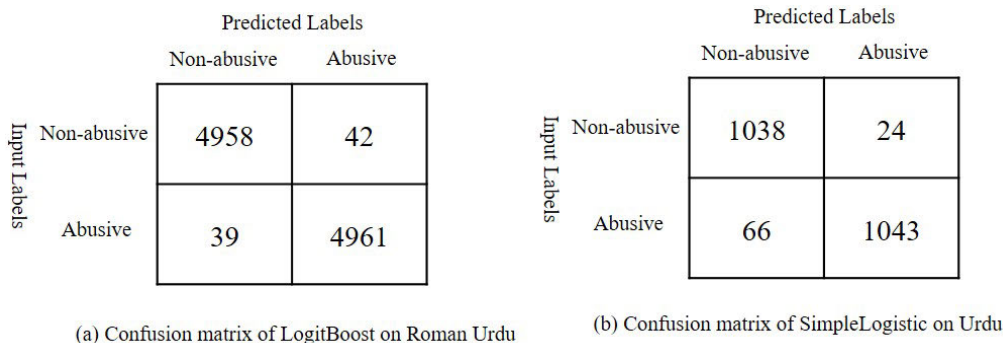


FIGURE 7. Confusion matrices of LogitBoost and SimpleLogistic models.

98.2% value of F-measure on Roman Urdu and 92.8% value on Urdu dataset that is 20.4% and 14.9% higher than OneR.

In short, LogitBoost and Simple Logistic outperform the other classifiers using character tri-gram on both datasets. Confusion matrices of both LogitBoost and Simple Logistic models are shown in Figure 7. Confusion matrix of LogitBoost, in Figure 7 (a) shows that the classification ratio is 99.19% and the misclassification ratio is less than 1%. For SimpleLogistic model in Figure 7 (b), the misclassification ratio is 4.15% only. From the confusion matrix, we conclude that both regression-based models are the effective classifiers from sixteen machine learning classifiers to detect offensive language from the comments of both scripts.

We also compare the efficiency of those models from each machine learning technique who achieved high F-measure values in Table 6. Although regression-based models show superior performance, these models take longer time to build the model than others do. From Table 7, it can be seen that LogitBoost takes 742.11 seconds to build the model and to achieve 99.2% F-measure. Similarly, the time of SimpleLogistic to build the model is 145.55 seconds that is much less than LogitBoost. The most efficient model is the k-NN model that takes 0.01 second to build the model and achieved 92.2% F-measure that is 7% less than LogitBoost.

VI. CONCLUSION

In this work, we performed automatic detection of offensive language from YouTube comments of Roman Urdu and Urdu. Our major contribution is to provide the first dataset of the Urdu language to detect offensive language automatically from the text. We explored the performance of seventeen models from seven machine learning techniques to process and detect offensive language from both Urdu and Roman Urdu datasets. We have also compared the effectiveness of individual as well as combined character n-grams and word n-grams to extract useful features from text to help the models in classification. After the analysis of results from different aspects, we conclude that character n-grams outperform the word n-grams. Character tri-gram is the most effective n-gram feature than other five types of character and word n-grams for both Urdu and Roman Urdu datasets. We also

found that combined n-grams do not perform better than individual n-grams. From the seven classification techniques of machine learning, regression-based technique outperforms the other six techniques but these models take longer time to build the model. LogitBoost shows superior performance on Roman Urdu using character tri-gram and achieved 99.2% score of F-measure. SimpleLogistic outperforms the others classifiers using character tri-gram on Urdu dataset and achieved 95.8% F-measure value. k-NN takes less time to build the model but its performance is not as good as many other models.

For future work, we aimed to apply neural network based models like fully convolutional neural networks [38] and character-level convolutional neural networks [39], [40] approaches for the detection of offensive language for Urdu and Roman Urdu. We have also aimed to design a multilingual text dataset of both languages from other popular social platforms. Our dataset is publically available to reproduce results and to do work in this direction.

REFERENCES

- [1] G. I. Sigurbjergsson and L. Derczynski, "Offensive language and hate speech detection for Danish," Aug. 2019, *arXiv:1908.04531*. [Online]. Available: <https://arxiv.org/abs/1908.04531>
- [2] H. Watanabe, M. Bouazizi, and T. Ohtsuki, "Hate speech on Twitter: A pragmatic approach to collect hateful and offensive expressions and perform hate speech detection," *IEEE Access*, vol. 6, pp. 13825–13835, 2018.
- [3] F. Noor, M. Bakhtyar, and J. Baber, "Sentiment analysis in E-commerce using SVM on roman urdu text," in *Emerging Technologies in Computing*, vol. 285. Cham, Switzerland: Springer, 2019, pp. 213–222.
- [4] A. Severyn, A. Moschitti, O. Uryupina, B. Plank, and K. Filippova, "Multilingual opinion mining on YouTube," *Inf. Process. Manage.*, vol. 52, no. 1, pp. 46–60, Jan. 2016.
- [5] H. T. Nguyen and M. Le Nguyen, "Multilingual opinion mining on YouTube—A convolutional N-gram BiLSTM word embedding," *Inf. Process. Manage.*, vol. 54, no. 3, pp. 451–462, May 2018.
- [6] K. Riaz, "Comparison of Hindi and Urdu in computational context," *Int. J. Comput. Linguist. Nat. Lang. Process.*, vol. 1, no. 3, pp. 92–97, 2012.
- [7] A. Daud, W. Khan, and D. Che, "Urdu language processing: A survey," *Artif. Intell. Rev.*, vol. 47, no. 3, pp. 279–311, Mar. 2017.
- [8] M. Bilal, H. Israr, M. Shahid, and A. Khan, "Sentiment classification of roman-urdu opinions using Naïve Bayesian, decision tree and KNN classification techniques," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 28, no. 3, pp. 330–344, Jul. 2016.
- [9] A. Alakrot, L. Murray, and N. S. Nikolov, "Towards accurate detection of offensive language in online communication in Arabic," *Procedia Comput. Sci.*, vol. 142, pp. 315–320, Jan. 2018.

- [10] A. Alakrot, L. Murray, and N. S. Nikolov, "Dataset construction for the detection of anti-social behaviour in online communication in Arabic," *Procedia Comput. Sci.*, vol. 142, pp. 174–181, Jan. 2018.
- [11] M. O. Ibrohim and I. Budi, "A dataset and preliminaries study for abusive language detection in Indonesian social media," *Procedia Comput. Sci.*, vol. 135, pp. 222–229, Jan. 2018.
- [12] J. M. Schneider, R. Roller, P. Bourgonje, S. Hegele, and G. Rehm, "Towards the automatic classification of offensive language and related phenomena in German tweets," in *Proc. 14th Conf. Natural Lang. Process. (Konvens)*, 2018, p. 95.
- [13] H.-S. Lee, H.-R. Lee, J.-U. Park, and Y.-S. Han, "An abusive text detection system based on enhanced abusive and non-abusive word lists," *Decis. Support Syst.*, vol. 113, pp. 22–31, Sep. 2018.
- [14] N. D. Gitari, Z. Zhang, H. Damien, and J. Long, "A lexicon-based approach for hate speech detection," *Int. J. Multimedia Ubiquitous Eng.*, vol. 10, no. 4, pp. 215–230, Apr. 2015.
- [15] M. Bouazizi and T. Otsuki Ohtsuki, "A pattern-based approach for sarcasm detection on Twitter," *IEEE Access*, vol. 4, pp. 5477–5488, 2016.
- [16] P. Rani and A. K. Ojha, "KMI-coling at SemEval-2019 task 6: Exploring N-grams for offensive language detection," in *Proc. 13th Int. Workshop Semantic Eval.*, 2019, pp. 668–671.
- [17] M. P. Akhter, Z. Jiangbin, I. R. Naqvi, M. Abdelmajeed, A. Mehmood, and M. T. Sadiq, "Document-level text classification using single-layer multisize filters convolutional neural network," *IEEE Access*, vol. 8, pp. 42689–42707, 2020.
- [18] K. Dinakar, B. Jones, C. Havasi, H. Lieberman, and R. Picard, "Common sense reasoning for detection, prevention, and mitigation of cyberbullying," *ACM Trans. Interact. Intell. Syst.*, vol. 2, no. 3, pp. 1–30, Sep. 2012.
- [19] Y. Chen, Y. Zhou, S. Zhu, and H. Xu, "Detecting offensive language in social media to protect adolescent online safety," in *Proc. Int. Conf. Privacy, Secur., Risk Trust Int. Confernece Social Comput.*, Sep. 2012, pp. 71–80.
- [20] T. Ishisaka and K. Yamamoto, "Detecting nasty comments from BBS posts," in *Proc. 24th Pacific Asia Conf. Lang. Inf. Comput. (PACLIC)*, 2010, pp. 645–652.
- [21] A. H. Razavi, D. Inkpen, S. Uritsky, and S. Matwin, "Offensive language detection using multi-level classification," in *Advances in Artificial Intelligence (Lecture Notes in Computer Science: Lecture Notes Artificial Intelligence: Lecture Notes Bioinformatics)*, vol. 6085, A. Farzindar and V. Kesselj, Eds. Berlin, Germany: Springer-Verlag, 2010, pp. 16–27.
- [22] R. Pelle, C. Alcántara, and V. P. Moreira, "A classifier ensemble for offensive text detection," in *Proc. 24th Brazilian Symp. Multimedia Web*, 2018, pp. 237–243.
- [23] P. Burnap and M. L. Williams, "Cyber hate speech on Twitter: An application of machine classification and statistical modeling for policy and decision making," *Policy Internet*, vol. 7, no. 2, pp. 223–242, Jun. 2015.
- [24] M. Ptaszynski, "In the service of online order: Tackling cyber-bullying with machine learning and affect analysis," *Int. J. Comput. Linguist. Res.*, vol. 1, pp. 135–154, Jul. 2015.
- [25] J. H. Park and P. Fung, "One-step and two-step classification for abusive language detection on Twitter," *CoRR*, vol. abs/1706.0, pp. 41–45, Aug. 2017. [Online]. Available: <https://www.aclweb.org/anthology/W17-3006/>
- [26] P. Mishra, M. Del Tredici, H. Yannakoudakis, and E. Shutova, "Abusive language detection with graph convolutional networks," *CoRR*, vol. abs/1904.0, pp. 2145–2150, Jun. 2019. [Online]. Available: <https://www.aclweb.org/anthology/N19-1221/>
- [27] K. Mehmood, D. Essam, and K. Shafi, "Sentiment analysis system for roman urdu BT—Intelligent computing," in *Proc. Adv. Intell. Syst. Comput.*, 2019, pp. 29–42.
- [28] Z. Tehseen, M. P. Akhter, and Q. Abbas, "Comparative study of feature selection approaches for urdu text categorization," *Malaysian J. Comput. Sci.*, vol. 28, no. 2, pp. 93–109, 2015.
- [29] K. Mehmood, D. Essam, K. Shafi, and M. K. Malik, "Sentiment analysis for a resource poor language—Roman urdu," *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, vol. 19, no. 1, pp. 1–15, Aug. 2019.
- [30] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *ACM SIGKDD Explorations Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009.
- [31] A. Pérez, P. Larrañaga, and I. Inza, "Bayesian classifiers based on kernel density estimation: Flexible classifiers," *Int. J. Approx. Reasoning*, vol. 50, no. 2, pp. 341–362, Feb. 2009.
- [32] B. Ziolk, S. Manandhar, R. C. Wilson, and M. Ziolk, "Logitboost weka classifier speech segmentation," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jun. 2008, pp. 1297–1300.
- [33] T. Zia, M. P. Akhter, and Q. Abbas, "Comparative study of feature selection approaches for Urdu text categorization," *Malaysian J. Comput. Sci.*, vol. 28, no. 2, pp. 93–109, 2015.
- [34] A. Tripathy, A. Anand, and S. K. Rath, "Document-level sentiment classification using hybrid machine learning approach," *Knowl. Inf. Syst.*, vol. 53, no. 3, pp. 805–831, Dec. 2017.
- [35] Q. A. Al-Radaideh and M. A. Al-Aburat, "An arabic text categorization approach using term weighting and multiple reducts," *Soft Comput.*, vol. 23, no. 14, pp. 5849–5863, Jul. 2019.
- [36] P. Burnap and M. L. Williams, "Us and them: Identifying cyber hate on Twitter across multiple protected characteristics," *EPJ Data Sci.*, vol. 5, no. 1, p. 11, Dec. 2016.
- [37] Y. Lee, S. Yoon, and K. Jung, "Comparative studies of detecting abusive language on Twitter," *CoRR*, vol. abs/1808.1, pp. 101–106, Oct. 2018. [Online]. Available: <https://www.aclweb.org/anthology/W18-5113/>
- [38] M. Dong, S. Wen, Z. Zeng, Z. Yan, and T. Huang, "Sparse fully convolutional network for face labeling," *Neurocomputing*, vol. 331, pp. 465–472, Feb. 2019.
- [39] S. Ghasemi and A. H. Jadidinejad, "Persian text classification via character-level convolutional neural networks," in *Proc. 8th Conf. AI Robot., 10th RoboCup Iranopen Int. Symp. (IRANOPEN)*, Apr. 2018, pp. 1–6.
- [40] M. Sato, R. Orihara, Y. Sei, Y. Tahara, and A. Ohsuga, "Text classification and transfer learning based on character-level deep convolutional neural networks," in *Proc. Int. Conf. Agents Artif. Intell.*, 2018, pp. 62–81.
- [41] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar, "Predicting the type and target of offensive posts in social media," 2019, *arXiv:1902.09666*. [Online]. Available: <https://arxiv.org/abs/1902.09666>



MUHAMMAD PERVEZ AKHTER received B.S. and M.S. degrees in computer science from University of Sargodha, Sargodha, Pakistan in 2009 and 2013 respectively. He is currently pursuing Ph.D. degree in Software Engineering at School of Software and Microelectronics, Northwestern Polytechnical University, Xi'an, P.R. China. He has eight years of teaching experience. He has published several research articles in high quality journals. His research interests include text processing, data mining, and image processing and computer vision.



ZHENG JIANGBIN received Ph.D. degree from IT Academy, P.R. China. He has presided over the research work of scientific research projects such as the National Natural Science Foundation, the 863 Program, and provincial and ministerial funds, and has published more than 100 articles.

From 2008, he was working as professor at Northwestern Polytechnical University, Xi'an, P.R. China. He is currently acting as a dean at School of Software and Microelectronics, Northwestern Polytechnical University, Xi'an, P.R. China. His research interests are image processing and computer vision, the IoTs, big data processing, and embedded computing technology.



IRFAN RAZA NAQVI received his bachelor degree in B.S Computer Science from Bahaudin Zakariya University, Multan, Pakistan in 2012. He then commenced his master degree in Software Engineering from Air University, Pakistan. Currently he is a Ph.D. student at Northwestern Polytechnical University, Xi'an, P.R. China. His current research interests include security and privacy concerns in the context of Internet of Things (IoT), machine learning and big data.



MOHAMMED ABDELMAJEED received the B.Sc. degree from Omdurman Islamic University, Khartoum, Sudan, in 2009, and the M.Sc. degree from Alneelain University, Khartoum, in 2015. He is currently pursuing the Ph.D. degree in computer science with Northwestern Polytechnical University, Xi'an, China. He has four years of teaching experience. His research areas are text processing and computer vision.



MUHAMMAD TARIQ SADIQ is Ph.D scholar at Northwestern Polytechnical University, Xi'an, P.R. China and working as an Assistant Professor at Electrical Engineering Department of The University of Lahore. He has received his B.Sc with (Hons.) and M.Sc. degrees both in Electrical Engineering from Comsats Institute of Information Technology, Lahore, Pakistan and Blekinge Institute of Technology, Sweden, in the year 2009 and 2011 respectively. Previously he was an Assistant Professor at the Sharif College of Engineering & Technology (SCET) which is affiliated with the University of Engineering & Technology Lahore and Lecturer at the University of South Asia. He was also Project Manager at SCET to manage final year student's projects, Patron of IEEE-SCET Student Branch and a lifetime member of Pakistan Engineering Council, Islamabad, Pakistan. His research interests include biomedical signal analysis and classification.

• • •