Multimedia Systems (2022) 28:1925–1940
https://doi.org/10.1007/s00530-021-00784-8

SPECIAL ISSUE PAPER

Abusive language detection from social media comments using conventional machine learning and deep learning approaches

Muhammad Pervez Akhter[1] · Zheng Jiangbin[1] · Irfan Raza Naqvi[1] · Mohammed AbdelMajeed[2] · Tehseen Zia[3]

Received: 28 July 2020 / Accepted: 19 March 2021 / Published online: 1 April 2021

Abstract
With the increase in the culture of social media and netizen, every day, millions of comments are posted on the uploaded posts. The use of abusive language in user comments has been increased rapidly. Abusive language in online comments initiates cyber-bullying that targets individuals (celebrity, politician, and product) and a group of people (specific country, age, and religion). It is important to detect and analyze abusive language from online comments automatically. There have been several attempts in the literature to detect abusive language for English. In this study, we perform abusive language detection from Urdu and Roman Urdu comments using five diverse ML models (NB, SVM, IBK, Logistic, and JRip) and four DL models (CNN, LSTM, BLSTM, and CLSTM). We apply these models on a large dataset with ten thousands of Roman Urdu comments and a small dataset with more than two thousand comments of Urdu. Natural language constructs, English-like nature of Roman Urdu script, and Nastaleeq style of Urdu make it more challenging to process and classify the comments of both scripts using deep learning and machine learning approaches. From experiments, we find that the convolutional neural network outperforms the other models and achieves 96.2% and 91.4% accuracy on Urdu and Roman Urdu. Our results also reveal that the one-layer architectures of deep learning models give better results than two-layer architectures. Further, we compare the performance of deep learning models with five conventional machine learning models and conclude that deep learning models perform significantly better than machine learning models.

Keywords Cyberbullying · Natural language processing · Machine learning · Deep learning · Abusive language

1 Introduction

Abusive language is an expression that contains abusive or dirty words in conversation (oral or text). With the increase in the culture of social media and social networking sites, the use of abusive language has increased rapidly. People from all over the world with different religions, nationalities, races, skin-colors post videos, pictures, and messages on social media platforms such as Facebook, YouTube, and Twitter. Every day, millions of comments are posted on the uploaded posts. Abusive language in online comments initiates cyber-bullying that targets individuals (celebrity, politician, product, etc.) and a group of people (specific country, age, religion, etc.). Among the users of these social medial platforms, no eye-to-eye contact permits them to comment on the subject without any fear. So, the users whose behavior is against another group should be ban, discourage, or restrict automatically.

Manually it is almost impossible to detect and filter out abusive comments from massive online comments. Conventional machine learning (ML) and deep learning (DL) methods with natural language processing (NLP) have been widely used for this purpose. Lexicon-based-methods [1, 2], word and character n-grams [3–5], ensemble learning [6] are the most common methods of ML to detect offensive language. In few studies, to detect offensive language, DL models have been applied and compared with ML models like bidirectional long short-term memory (BLSTM) [4],

✉ Muhammad Pervez Akhter
pervezbcs@gmail.com

[1] School of Software and Microelectronics, Northwestern Polytechnical University, Xian 710072, People's Republic of China

[2] School of Computer Science and Technology, Northwestern Polytechnical University, Xian 710072, People's Republic of China

[3] Department of Computer Science, COMSATS University, Islamabad 44000, Pakistan

recurrent neural network (RNN), convolutional neural network (CNN) [5, 7], and long short-term memory (LSTM) [8].

In the past, automatic detection of abusive language for resource-rich languages like English has been the focus of researchers. In recent years, due to the advancement in technology and the availability of language resources, only a few studies have been performed to detect abusive language from text of resource-poor languages like Japanese [3, 9], Germen [10], Portuguese [6], Indonesian [11], Danish [12], and Arabic [13]. Urdu is the national language of Pakistan, with more than 300 million speakers worldwide [14]. Urdu and Roman Urdu are two writing scripts of the Urdu language. Urdu script is written in the Nastaleeq style that is very complex. Lack of language resources, complex morphology, and unique features of Urdu are the main challenges for DL and ML methods to process Urdu text [15]. To the best of our knowledge, no study explores the potential of DL and conventional ML models to detect abusive language from user comments of both Roman Urdu and Urdu.

Text classification using DL models has several advantages over conventional ML models. The main challenge for ML models is that their performance is dependent on feature selection methods like information gain, gain ratio, etc. Several studies conclude that there is no universal feature selection method that works well with all types of ML models [14, 16, 17]. Deep learning models are independent of explicit feature selection methods. Hidden layers within a DL model extract the useful features automatically. It is difficult to observe which features are chosen for a specific task [18]. Further, DL models are considered robust when dealing with high-dimensional feature space [19]. LSTM and BLSTM are good at capturing long-term dependencies among the input words of a comment using its memory cells. In this way, both models can capture semantic and rich information from the whole text for better classification [8, 12, 20].

The writing styles of Urdu and Hindi are different, but the Roman scripts of Urdu and the Romanagari script of Hindi are almost identical [21]. The Roman script is written in English characters. The Roman script is popular on social media as it is easier than Urdu and Hindi to write using an English keyboard on mobile phones, laptops, and tablets. Therefore, the Roman script is easily readable, writable, and understandable in Pakistan, India, and many other world regions [22]. In recent years, as compared to Urdu, Roman Urdu has a broader scope and the focus of many recent research studies was to process, analyze, and classify Roman Urdu. The only focus of these studies was to carry out sentiment analysis tasks [23, 24]. No study investigates the unethical behavior from the Roman text of online users. Therefore, it is vital to detect abusive comments of Urdu and Roman Urdu script from social media platforms.

Social media applications are popular sources to design annotated datasets to investigate various subjects like sentiment analysis and abusive language detection. Alakrot created a dataset from YouTube comments to detect abusive language from the Arabic language [13]. YouTube [13, 25], Twitter [7, 26], Facebook [4], news blogs [3, 9] are popular social media platforms to collect and annotate the datasets in various languages.

To the best of our knowledge, it is the first study that investigates abusive language detection using conventional ML and DL models for the Urdu language. Specifically, this study has the following contributions.

1. We explore the performance of DL models to detect abusive language from online comments.
2. We empirically investigate the performance of these models on two popular scripts of the Urdu language: Urdu and Roman Urdu.
3. We also explore the architectures of RNN models having one-layer and two-layers of hidden units.
4. We also compare the performance of DL models with ML models to detect abusive language.

The organization of this paper is as follows: Sect. 2 discusses the important features of both Urdu and Roman Urdu scripts, related literature is discussed in Sect. 3, datasets and the models are described in Sect. 4, results are discussed in Sect. 5, and the conclusion is given Sect. 6.

2 Urdu and roman Urdu scripts

The text of the Urdu language over the internet is available in two scripts: Urdu and Roman Urdu. Nastaleeq style of Urdu is a rich morphological script with many unique features like free word order, right-to-left direction, context-sensitive, and no capitalization [15]. Roman Urdu is more challenging because of its similar English style but no restriction on grammar, dictionary, and word order. Because of the lack of linguistic resources, complex morphological style, and unique characteristics, the Urdu language is an important research language in South Asia [27]. A few important features that create differences among both scripts are discussed below:

- *Alphabets:* the Urdu language has 38 alphabets, as shown in Fig. 1, while Roman Urdu uses the alphabets of the English language.
- *Font style:* Urdu is written in the Nastaleeq font style that is a very complex and rich morphological style. For example, a sentence "I am a student." can be written in Urdu as "میں ایک طالب علم ہوں۔" Roman Urdu can be written using English fonts. The same sentence can be writ-

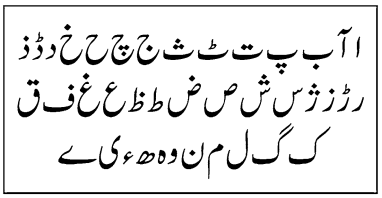

Fig. 1 Alphabets of Urdu language in Nastaleeq style

ten in Roman script as "main ek talib e ilam hon" or "Main ik talib-e-ilam hon".

- *Writing direction:* the direction of writing Urdu script is from Right-to-Left while Roman Urdu uses Left-to-Right direction. For example, the sentence "That is my school." can be written in Urdu as "یہ میرا سکول ہے" . Here "یہ" is the first word and "ہے" is the last word in the sentence. The same sentence can be written in Roman Urdu as "wo mera sakool ha". Here the direction is the same as in English sentence. "wo" is the first word and "ha" is the last word in the sentence.
- *Capitalization:* like English, Roman Urdu uses both upper and lower case letters, but the rules are no strict as English. As compared to English and Roman Urdu, all the letters are written in the same case, and there is no capital or small letters in Urdu script.
- *Diacritics:* Urdu script uses a few special characters called diacritics. Diacritics help incorrect pronunciation and change the meaning of the word when applied above or below a word. For example, in "گنا" (means sugarcane) and "گنا" (means number of times) word is the same, but due to the diacritics, their meanings are different.
- *Free word order:* both Urdu and Roman Urdu are called free word order languages. Word order in a sentence may be different but the meaning would be the same. For example, "he bought this book from the market" can be written in Urdu in two ways: "اس نے بازار سے یہ کتاب خریدی۔", "اس نے یہ کتاب بازار سے خریدی۔" Both sentences have the same meaning with different word orders.

A few important features of Urdu and Roman Urdu scripts are given in Table 1. Roman Urdu script is more flexible in writing as English characters can be typed using an English keyboard. Reading and understanding of Roman Urdu script are challenging because the Roman script does not follow any standard dictionary and grammar-free wiring style. Compared to Roman Urdu, Urdu script follows a standard dictionary, grammar rules, and writing characters that make Urdu text more challenging than Roman Urdu.

Table 1 Important features of Urdu and Roman Urdu scripts

Features	Roman Urdu	Urdu
Alphabets	26 as the English language	38
Grammars	No	Yes
Dictionary	No	Yes
Style	English	Nastaleeq
Word order	No	Yes
Typing	Easy	Not Easy
Reading	Easy	Not Easy
Writing and Understanding	Not Easy	Not Easy

3 Related work

Recently, the interest of the computational linguistic community is increasing to study unethical behavior of users on social media networks like Facebook [4], YouTube [2, 13, 25], Twitter [10, 28, 29], and other web blogs [6, 9, 30]. Individuals and groups from all over the world post their images, videos, and messages on social media and receive hundreds of positive or negative comments or feedback on the posted objects. These comments usually include hate words or abusive language because of the difference among the users in race, culture, religion, or nationality [1, 31]. These abusive or hated words initiate cyberbullying among users.

Manually, it is almost impossible to detect and remove abusive language comments from social media. Several methods of conventional ML and DL have been considered to detect abusive language comments of various languages. [1, 6] detected hate words from web blogs using a lexicon-based approach. Burnap applied a support vector machine and random forest to detect hate speech from Twitter [31]. The performance of a pattern-based approach was compared with k-NN, random forest, and support vector machine to detect sarcasm from tweets [29]. Lee detected abusive comments with the help of abusive and non-abusive lists of words [26]. A programmatic approach was applied by Watanabe to detect hate speech from online comments [28]. In all the above-cited studies, proposed methods were compared with a few ML models while there is a lack of a study that fully explore the power of various ML models for abusive language detection task.

The number of features and the quality of these features selected by various feature selection methods from the dataset can affect the performance of a classifier. In several research studies, character n-grams and word n-grams show superior performance than a Bag-of-Word (BoW) [28, 29, 31]. Word n-gram was an effective method to detect abusive languages from Twitter [32], YouTube [2, 13], blogs, and emails [6, 9, 33]. In a few studies, character n-grams improved the performance of ML models from Twitter text

[5, 34]. In the past, in the above-cited studies, either character n-grams or word n-grams methods have been used, but both methods have not been compared yet.

[11, 26] compared the performance of the proposed list-based approach with word and character n-gram approaches and showed that their proposed approach is better than the n-gram approach. [11] shows that combined n-grams (unigram + bigrams) help Naïve Bayes (NB) to detect abusive comments better than other models. Individual n-gram, unigram, was effective in sentiment analysis tasks for the Roman Urdu on YouTube comments [35].

Until now, resource-rich languages such as English focused on research studies to detect abusive language from social media. Lack of language resources such as annotated datasets of resource-poor languages like Urdu is the main language processing obstacle. It creates a gap between resource-poor and resource-rich languages. Various DL and ML models have been applied to detect abusive language comments of English, but only a few studies investigated this abusive language detection for resource-poor languages. To detect abusive language from the Portuguese, the classifier ensemble learning technique was applied to classify abusive text collected from web pages [6]. In the study of Ibrohim, three ML models decision tree (DT), NB, and SVM were used to classify abusive and non-abusive text of the Indonesian language from social media [11]. Schneider applied ML models to classify abusive tweets of the German language [10]. Similarly, n-grams techniques were applied to extract features from Arabic comments, and then ML models were used to classify these comments into abusive or non-abusive labels [13, 36].

In a few studies, the performance of DL models has been compared with ML models to detect abusive language. Sigurbergsson applied logistic regression and Bidirectional Long Short-Term Memory (BLSTM) models to detect abusive language from the Danish language and shows that BLSTM model outperforms the other to classify comments collected from Reddit, Facebook, and Twitter [4]. To detect hate speech from tweets in English, LSTM, and CNN models of DL were compared with SVM, logistic regression, and gradient boosted decision tree models of ML models [8]. Lee concluded that DL models Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) performed better than ML models NB, SVM, logistic regression, and random forest to classify abusive tweets of the English language [7]. In the study of [8], applied a ML model SVM and two DL models CNN and Long Short-Term Memory (LSTM) to detect hate tweets in English [37]. These studies cannot be considered well because of four reasons. First, one or two models of DL have been compared with multiple ML models. Second, the potential of DL models has not been explored by comparing multiple models. Third, these studies used resource-rich language (English) tweets but resource-poor languages have not been considered. Forth, other social media applications like YouTube, Facebook, or Blogs have not been considered and only Twitter datasets were used.

Social media platforms have been used to collect views, opinions, and trends of people from different world regions about some company, product, advertisements, images, videos, etc. In many research studies, authors collect comments from these social media platforms and annotate these comments for sentiment analysis or analyze user behavior. [36] collected sixteen thousand comments from multiple YouTube videos to detect abusive language. A small dataset of 1250 Portuguese comments was collected and annotated from Brazilian websites to study online users' abusive behavior [6]. A Twitter dataset of 2,500 tweets was designed by collecting abusive and non-abusive tweets of the Indonesian language. In this study, we also use collected comments of the Urdu and Roman Urdu scripts from YouTube and applied ML and DL models to classify these comments into abusive or non-abusive labels. A summary of the work discussed is given in Table 2.

3.1 Research gaps of the study

From the summarized literature in Table 2 and discussion in Sect. 3, the identified research gaps in abusive language detection for Urdu and Roman Urdu are given below:

Dataset preparation: the English language has several research studies as compare to resource-poor languages. It is because of the lack of language resources for resource-poor languages like Urdu. Because of the unavailability of the abusive text dataset, no study investigates the automatic detection of abusive language from Urdu text using deep learning models.
Deep learning models: most of the research studies are based on ML models (one to four). None of the studies explores the performance of various deep learning models to classify comments into abusive and non-abusive labels. In this study, we apply four DL models (CNN, LSTM, BLSTM, and CLSTM) and empirically compare their performance.
Machine learning vs. deep learning models: in some studies, ML models have been compared with one DL model. This type of comparison does not fully explore the potential of DL models. In this study, we have compared the performance of four DL models with five ML models.
Urdu and Roman Urdu language processing: Although several studies used ML techniques to process and classify Urdu or Roman Urdu but, to the best of our knowledge, none of the studies use both scripts to evaluates DL and Ml models for abusive language detection.

Table 2 Comparison of past studies about abusive language detection from social media comments

Reference	Language	Platform	Classification models
[13]	Arabic	YouTube	SVM
[6]	English, Portuguese	Twitter, Blogs	NB and SVM
[26]	English	Twitter, Articles	Unsupervised learning
[38]	English	Twitter	SVM (linear, polynomial, radial
[11]	Indonesian	Twitter	NB, SVM, RF
[12]	Danish, English	Twitter, Reddit, Facebook	LR, BLSTM
[10]	German	Twitter	CNN
[32]	English	Twitter	SVM
[9]	Japanese	Blogs	SVM
[7]	English	Twitter	NB, LR, SVM, RF, Gradient Boosted Trees, CNN, RNN
[30]	English	News Group	Decision Table NB (DTNB)
[5]	English	Twitter	Logistic Regression, SVM, CNN
[2]	English	YouTube	NB, SVM
[34]	English	Twitter	LR, Graph Convolutional Network
[37]	English	Twitter	SVM, BLSTM, CNN

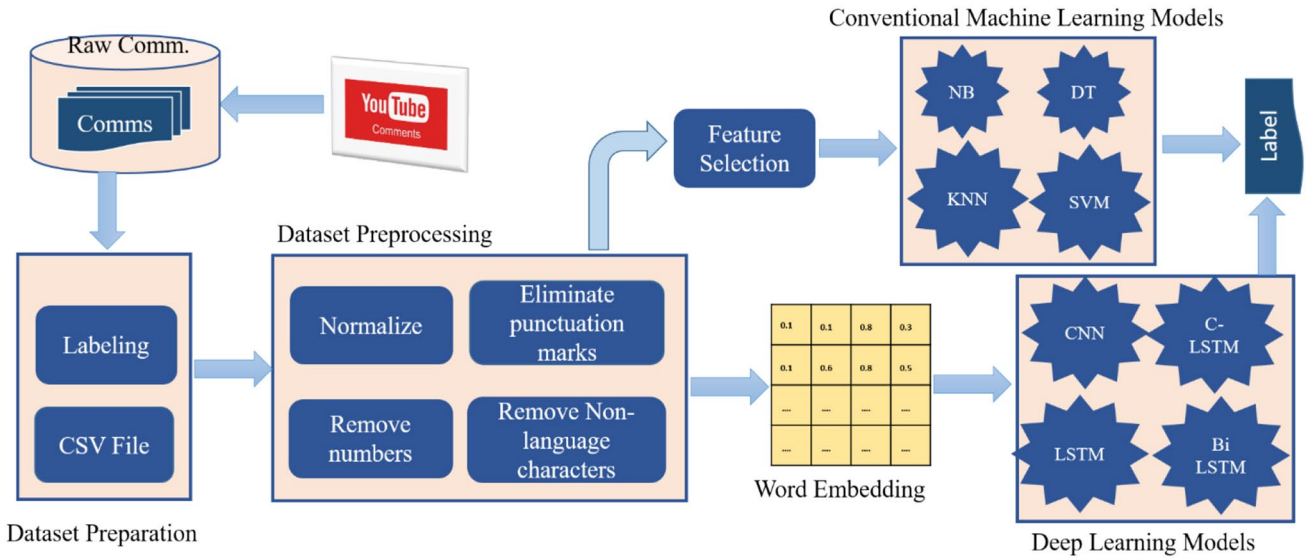

Fig. 2 The overall process to detect abusive language from YouTube comments

4 Materials and methods

This section presents the datasets, dataset preprocessing methods, DL and ML models, and performance evaluation measures used in this study. The overall process used to detect abusive language from YouTube comments is shown in Fig. 2. Urdu comments were collected from YouTube videos as collected and annotated for Arabic [13] and English [25, 39]. These collected raw comments are manually labeled into abusive or non-abusive. The final dataset is in the comma-separated values (CSV) file. Before applying automatic ML or DL models on the dataset, we preprocess and convert the dataset into the readable format by these models. In preprocessing, we normalize each comment, remove digits, eliminate punctuation marks, and remove any other language characters, for example, characters of Roman Urdu or English. For ML models, after cleaning the dataset, we use the term frequency-inverse document frequency (TFIDF) to convert each comment into a vector. Feature selection methods use these vectors to select valuable features or words that help the ML classifiers to achieve maximum accuracy on the dataset. In this study,

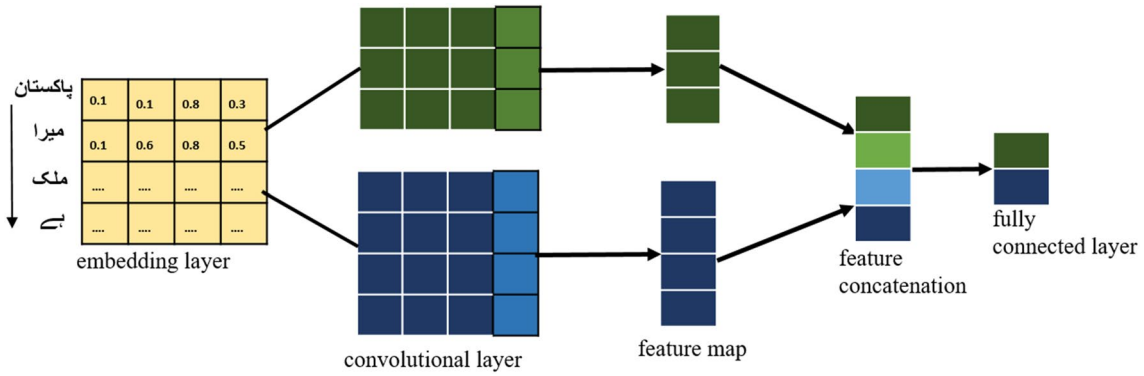

Fig. 3 Single-layer multi-filter convolutional neural network

we use Bag-of-Word (BoW) approach for feature generation. Input to the DL models is the word embedding layer that is also a real-valued matrix to present a comment. The nodes in the convolutional layer of CNN and the units of LSTM extract the useful feature for the output layer to predict the final label of the input comment.

4.1 Convolutional neural network (CNN)

CNN has shown good performance in image processing as well as text processing. CNN model uses a one-dimensional structure in convolutional operation to extract features from text data. A recent study concludes that CNN model outperforms the other DL and ML models to classify long text documents [15, 40]. Figure 2 shows the CNN architecture used in this study. This model reads text comments from the embedding layer. The convolutional layer applies multiple filters of different sizes on the input text. Filters of different sizes extract variable-length features from text [41]. The pooling layer explores short as well as long-term relations among words in the text [41, 42].

If x is a comment with L words and d length of word vector in the embedding layer, then input comment can be represented as $x \in R^{L \times d}$. A comment of length n can be expressed as Eq. 1 where + is the concatenation operation.

$$x_{1:n} = x_1 + x_2 + ... + x_n \tag{1}$$

A filter f in the convolutional layer of length k can be represented as $n \in R^{k \times d}$. A filter is passed over a window of words of the input comment x from j position to $(j + k - 1)$ position at a time. The window w_i can be represented as given in Eq. 2.

$$w_i = [x_i + x_{j+1} + \cdots + x_{j+k-1}] \tag{2}$$

From a window w of words $x_{i:i+h-1}$, the convolutional filter creates a new context local feature c_i as given in Fig. 3. Where f is a non-linear function, is an element-wise multiplication, and $b \in R$ is a biased term.

$$c_i = f(w.x_{i:i+h-1} + \mathrm{b}) \tag{3}$$

A feature map c represents a set of features obtained from the convolutional operation as given in Eq. 4.

$$c = [c_1, c_2, ..., c_{n-h+1}] \tag{4}$$

Pooling operation is performed over feature map c after the convolutional operation to select top k features to accurately predict the input comment. The pooling layer also helps the model to avoid overfitting. We applied max-pooling operation over the feature map after the convolutional operation as given in Eq. 5. c' is the feature map after max-pooling.

$$c' = \max\{c_1, c_2, ..., c_{n-h+1}\} \tag{5}$$

Softmax is a fully connected layer that gives the probability distribution over labels (abusive or non-abusive) as below:

$$y_j = w_j y_{j-1} + b_j \tag{6}$$

y_{j-1}, w_j, and b_j are the output vector, transition matrix, and the bias factor of the softmax layer, respectively. The probability distribution over the comment labels is calculated as in Eq. 7.

$$P(i|t;\varnothing) = \frac{\exp(y_{ij})}{\sum_{k=1}^{n} \exp(y_{ij})} \tag{7}$$

Dropout is applied to a fully connected layer to avoid overfitting the model due to many hyperparameters, hidden units, and the connections among them.

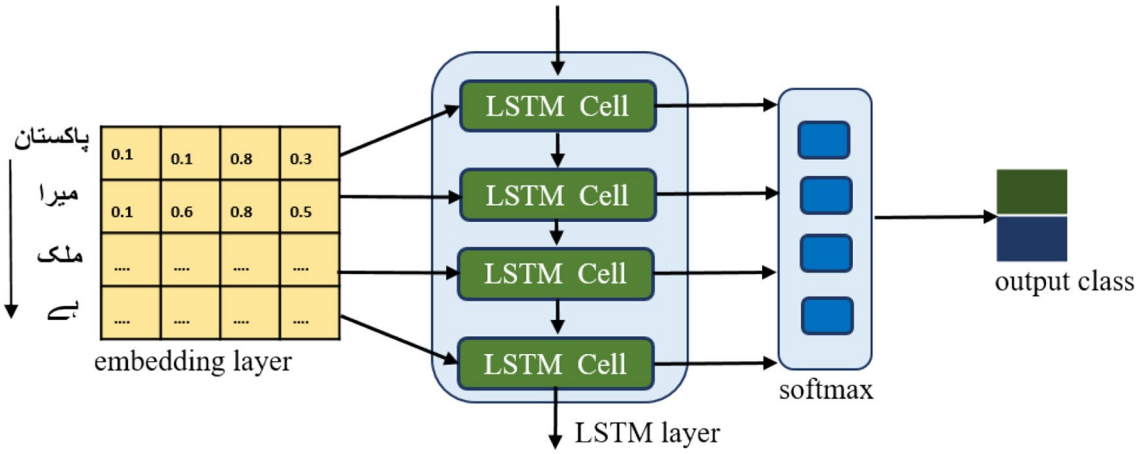

Fig. 4 The architecture of long short-term memory model

4.2 Long short-term memory (LSTM)

A major problem of the CNN model is to capture long-term dependencies among the input words of a comment. LSTM model overcomes the problem of CNN by using its memory cells. These memory cells help the model to capture long-term dependencies among the words by maintaining its state for a long period. Therefore, the LSTM model can capture semantic and richer information from the whole text. The LSTM model used in this study is shown in Fig. 4.

LSTM model takes input from the embedding layer. The hidden layer consists of multiple cells or units. A cell is made of three gates forget, input, and output. Input and output gates are used for input and output operations, while the forget gate is used to main the cell state for some time. With the help of these gates, LSTM overcomes the vanishing gradient problem. The operations performed by LSTM models can be calculated using Eq. 8 to Eq. 13.

$$i_t = \sigma(x_t U_i + h_{t-1} W_i + b_i) \tag{8}$$

$$f_t = \sigma(x_t U_f + h_{t-1} W_f + b_f) \tag{9}$$

$$o_t = \sigma(x_t U_o + h_{t-1} W_o + b_o) \tag{10}$$

$$q_t = \tanh(x_t U_q + h_{t-1} W_q + b_q) \tag{11}$$

$$p_t = f_t * p_{t-1} + i_t * q_t \tag{12}$$

$$h_t = o_t * \tanh(p_t) \tag{13}$$

Input i_t, output o_t and forget f_t values are generated from the input x_t, bias term b and proceeding hidden state h_{t-1} using a sigmoid function. At step t, temporary result q_t is calculated over the input x_t and preceding hidden state h_{t-1} using a non-linear *tanh* function. Then q_t is with history p_{t-1} by input gate i_t and forgot gate f_t respectively to get an updated history p_t. Finally, the output gate o_t use updated history p_t to get the final hidden state that is the output to the softmax layer for the final outcome.

4.3 Bidirectional long short-term memory (BLSTM)

As discussed in the previous section that LSTM can only use past information to process the current word vector. LSTM cannot use future information during processing, and it is the major shortcoming of the LSTM model. BLSTM is considered more effective than LSTM to process current information because it takes advantage of both the past and future information (both directions) to processing current information. BLSTM architecture is shown in Fig. 5. BLSTM architecture uses an additional hidden layer as compare to LSTM architecture. Forward layer and backward layers capture the future and the past information to process the current information. Forward and backward layers are connected to an output layer, and this output layer is connected to the final output layer.

4.4 Convolutional long short-term memory (CLSTM)

CLSTM is a hybrid model of LSTM and CNN models that combines the advantages of both models to learn contextual semantic features. The architecture of CLSTM used in this study is shown in Fig. 6. Input to CLSTM model is a word embedding layer that is the same as other DL models. First, CNN extract features from the input comment using convolutional and max-pooling operations and then the selected features are input to the LSTM model. LSTM model learns the dependencies among the selected features. The final output is the output of the fully connected layer (softmax). For n LSTM units, the output of LSTM is a vector

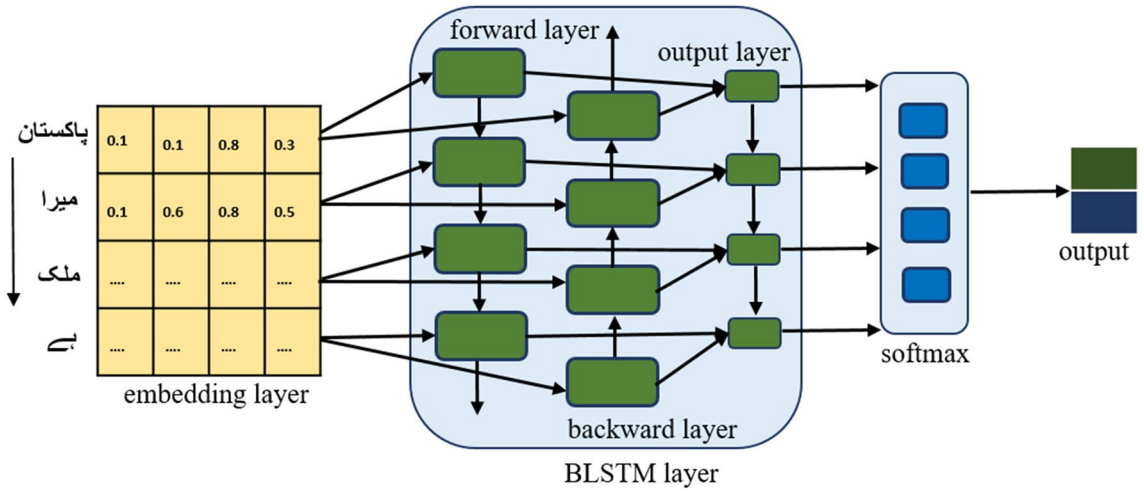

Fig. 5 The architecture of the BLSTM model

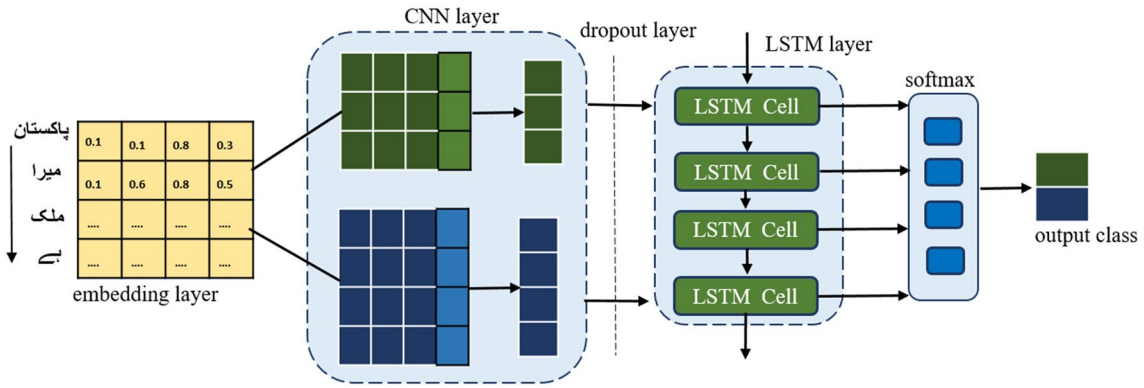

Fig. 6 The architecture of CLSTM Model

$h = (h_1, h_2, \ldots, h_n)$. The fully connected layer takes this vector and generates the final output d as given in Eq. 8.

$$d_i = \sigma(w_i * h_i + b_i) \tag{14}$$

σ, w_i, b are the non-linear activation function, weight of the *ith* node, and the bias term.

4.5 Machine learning models

We have also applied ML models to detect abusive language and compared the performance of these models with DL models.

- *Naïve Bayes (NB):* it uses conditional probability and Bayes theorem. It is easy and simple to build for large datasets.
- *K-Nearest Neighbors (k-NN):* final class of an example is based on the class information of the k-nearest neighbors. Value of k and similarity measure are two important factors that can affect its performance.
- *Support Vector Machine (SVM):* it learns the n-dimensional plan from data to classify the examples into one of the classes.
- *Logistic:* it is based on multinomial logistic regression with a ridge estimator to measure the relationship among the dependent and independent variables.
- JRip: is a rule-based model that works in two steps. First, it learns rules from all the positive examples, and then it applies Repeated Incremental Pruning to Produce Error Reduction (RIPPER) to prune the learned rules. It is efficient on a large, noisy dataset for the classification task

Table 3 Statistics of Urdu and Roman Urdu datasets

Properties	Urdu	Roman Urdu
Total no. of comments	2171	10000
No. of classes	02	02
No. of abusive comments	1109	5000
No. of non-abusive comments	1062	5000
Minimum length of a comment	02	01
Maximum length of a comment	37	511
Training–testing ratio	80–20	80–20
No. of comments in Training	432	8000
No. of comments in Testing	1739	2000

4.6 Datasets

Multiple datasets are publically available for resource-rich languages like English (see Table 2). For Roman Urdu, an annotated dataset is publically available at GitHub[1] in the CSV file. This dataset is balanced, and it contained one hundred thousand text comments. We used ten thousand comments from this dataset. Five thousand are abusive, and the other five thousand are non-abusive. For Urdu, we used Urdu Offensive Dataset (UOD). It is the first dataset that is publically available after a recent study[43]. This dataset contained 2171 comments collected from YouTube videos, including entertainment, politics, sports, showbiz, and religious topics. Statistics of both datasets are given in Table 3.

4.7 Performance evaluation measures

To evaluate the performance of a model and compare its performance with other models, we use four performance measures accuracy, true-positive rate (TPR), false-positive rate (FPR), and F-measure, which are the common performance metrics for the classification task. Accuracy is not a good performance metric when the datasets are imbalanced [15, 44]. We are using the accuracy metric because both datasets are balanced. Accuracy is the proportion of the correctly classified comments from the dataset and can be defined as:

$$\text{Accuracy} = \frac{T_p + T_N}{T_p + T_N + F_p + F_N} \tag{15}$$

where the values of T_p, T_N, F_p, and F_N can be taken from the confusion matrix [15, 32] and can be defined as below:

- True Positive (T_p): refers to the number of abusive comments in the dataset that are correctly classified as abusive by the model
- True Negative (T_N): refers to the number of non-abusive comments that are correctly predicted as non-abusive by the model
- False Positive (F_p): refers to the number of non-abusive comments that have been predicted as abusive by the model
- False Negative (F_N): refers to the number of abusive comments that have been predicted as non-abusive by the model

True-positive rate is the rate of correctly predicted abusive comments from the dataset, while the false-positive rate is the number of incorrectly predicted abusive comments. TPR and FPR can be calculated as follows:

$$\text{TPR} = \frac{T_p}{T_p + F_N} \tag{16}$$

$$\text{TPR} = \frac{F_p}{F_p + T_N} \tag{17}$$

Because of the dataset characteristics like size, noise, imbalance level, a classification model either increases the ratio of positive instances correctly detected (i.e., recall) or the accuracy of the positive predictions (i.e., precision). It is a well-known precision-recall trade-off situation. Therefore, we use harmonic means of precision and recall, known as F-measure, and can be calculated as.

$$F - \text{measure} = \frac{2 \times T_p}{2 \times T_p + F_p + F_N} \tag{18}$$

5 Results and discussion

In this section, we compare the performance of our models to detect abusive language for Urdu and Roman Urdu datasets. Experimental results are shown graphically for better comparison and understanding. First, we find the optimal values of hyperparameters for DL models. Second, DL models have been compared on both datasets. Third, LSTM and BLSTM models have been compared. The performance of DL and ML models is given at the end of this section.

5.1 Hyperparameters settings

DL models have several parameters known as hyperparameters, and optimizing these parameters is always data-dependent [39]. For example, for the CNN model, number

[1] https://github.com/shaheerakr/roman-urdu-abusive-comment-detector

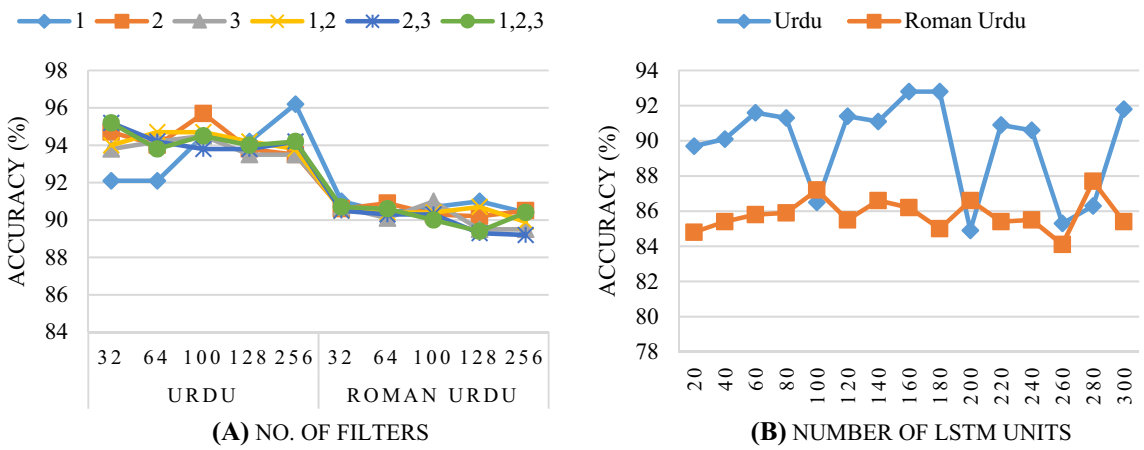

Fig. 7 No. of Filters vs. filter size for CNN in (**a**) and no. of LSTM units of CLSTM model in (**b**)

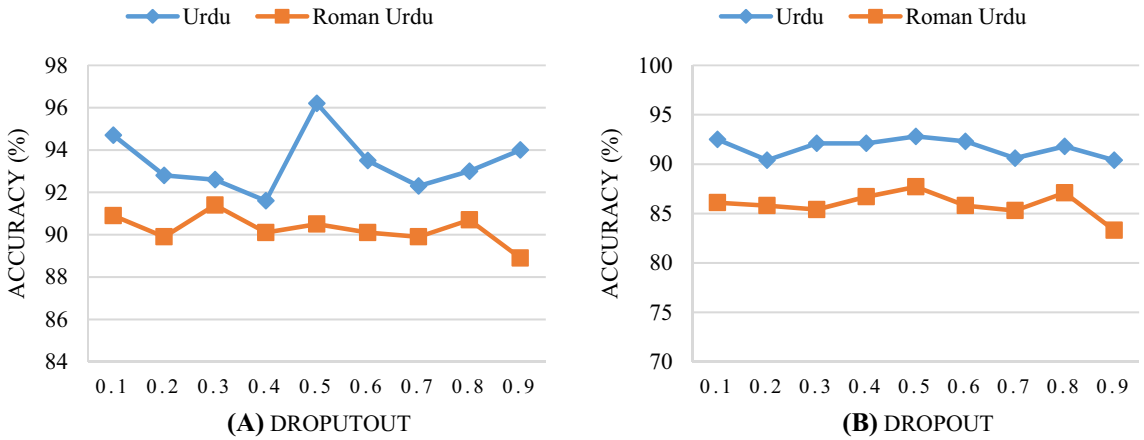

Fig. 8 Dropout values of CNN and LSTM models

of filters, filter size, word embedding size, number of epochs, dropout, etc. These parameters have a significant effect on the performance of a model. These parameters must be fine-tuned to achieve a reliable and robust model. Several studies have fine-tuned these parameters of CNN [41], LSTM [20], CLSTM [42], and BLSTM [45]. Fine-tuning means to find out the best values of these parameters on which the model gain maximum performance on a given dataset. Although fine-tuning these parameters is a resource exhausting process, we perform a grid search on 80% of both datasets to fine-tuning the parameters of our models. To find out the optimal value of a parameter, such as a filter size, other parameters of the model are kept unchanged. Also, we use the efficient Adam optimization algorithm with 0.001 learning rate to train the model.

The architecture of the CNN model, in this study, has multiple filters of variable size in the convolutional layer. To find out the number of filters and the size of these filters for both dataset, we performed multiple experiments using several filters (32, 64, 100, 128, and 256) of the same filter size (1, 2, and 3) and variable filter size [(1,2), (2,3), and (1,2,3)]. Figure 7a shows that 256 filter of size one achieves the maximum accuracy on the Urdu dataset. For Roman Urdu, there is no significant difference in the performance of these filters, but 64 filters of size two have better performance than others. Same size filters perform better than variable size filters on both datasets. It is because, from the abusive words, single and double words are the prominent words in Urdu and Roman Urdu, respectively. We have also found out the number of LSTM units for the LSTM model on both datasets, and results are shown in Fig. 7b. For Urdu, 160 and 180 equally perform better than others do. The model achieved high accuracy with 180 LSTM units on Roman Urdu.

Dropout helps a model to prevent noise and overfitting. If the dataset is noisy or has fewer examples, then the model can face the problem of overfitting. This issue can be resolved by increasing the size of the dataset or removing the

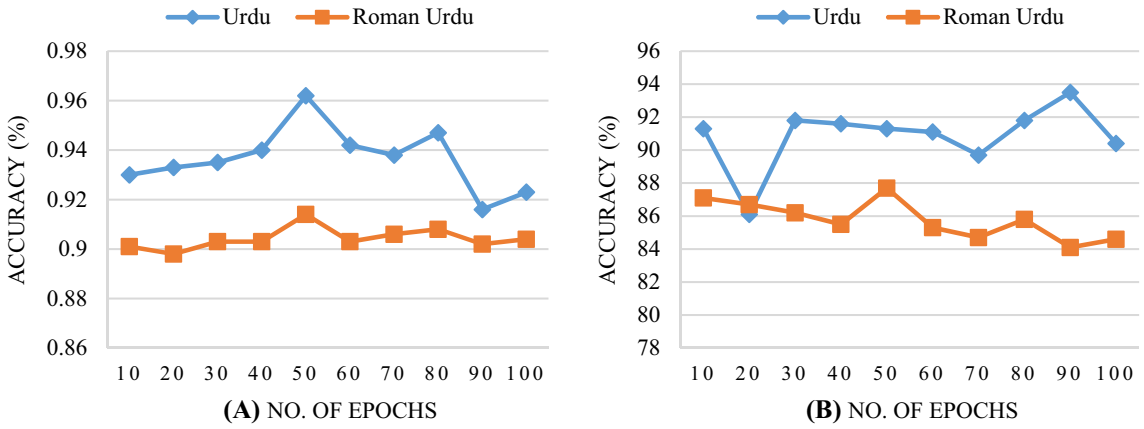

Fig. 9 Performance of CNN and LSTM on number of epochs

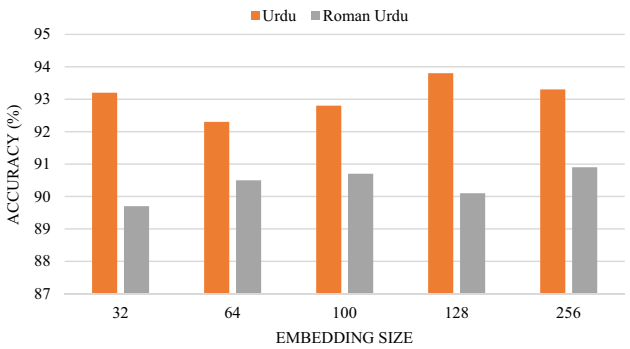

Fig. 10 The accuracy is achieved by the CNN model using the different lengths of word embedding

Table 4 Optimized parameters of CNN and LSTM models on both datasets

Properties	CNN		LSTM	
	Urdu	Roman Urdu	Urdu	Roman Urdu
Batch Size	32	32	32	32
Embedding Size	128	256	160	280
No. of Epochs	50	50	90	50
Dropout	0.5	0.3	0.5	0.5
LSTM Units	–	–	160	280
No. of Filters	256	64	–	–
Filter Size	01	02	–	–
Activation Function	ReLu	ReLu	Sigmoid	Sigmoid

number of hidden units used for computing features. Dropout inactivates the inactive units in the hidden layer that do not participate in the next iteration. The optimized values of the dropout parameter of CNN are 0.5 and 0.3 for Urdu and Roman Urdu, respectively, as shown in Fig. 8a. For the LSTM model, in Fig. 8b, on dropout values 0.5, the model achieves high accuracy for both datasets.

Choosing the appropriate number of epochs for a model is important to learn complex and small datasets. An inappropriate number of epochs can lead the model to the overfitting problem. We tested our models from 10 to 100 epochs, and the results of CNN and LSTM models are shown in Fig. 9a, b, respectively. CNN model achieves the best accuracy values with 50 epochs on both datasets. Because of the complex morphological script of Urdu, LSTM achieves high accuracy on 90 epochs. Roman Urdu script has the same writing style and characters as English. It is comparatively easy than Urdu. Therefore, LSTM achieves high accuracy with fewer epochs (50 epochs) on Roman Urdu than Urdu.

Word embedding represents the vocabulary as real-value vectors, and the size of the real-value vector can affect the model performance. It helps the model to assess the word relevance based on the distance between the two embedding vectors. We have taken the length of the embedding vector as a hyperparameter. In training, the performance of the CNN model is examined using five-word embedding (32, 64, 100, 128, and 256), and the results are shown in Fig. 10. Experimental results show that embedding size 128 and 256 help the model to achieve high performance on Urdu and Roman Urdu datasets. The first layer is also an embedding layer in the LSTM model. We have taken the embedding size in the embedding layer equal to the number of LSTM units that are chosen in training as a hyperparameter.

After hyperparameter tuning, the optimized parameters of both models are shown in Table 4. We set the initial values of these parameters the same as specified by [41] for CNN and [20] for LSTM. We tested three activation functions: ReLu, Sigmoid, and Tanh. Activation function ReLu in CNN shows better performance on both datasets while Sigmoid in LSTM was better than others. In all the experiments, we use

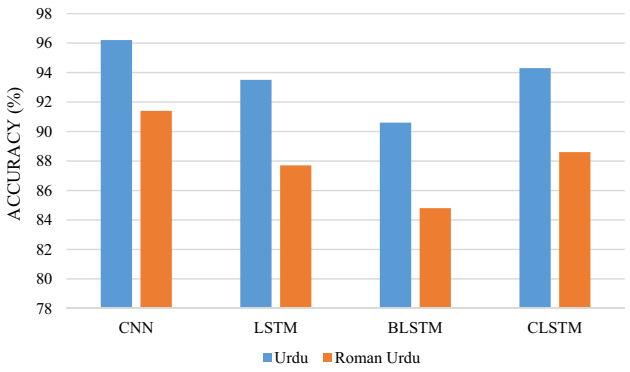

Fig. 11 Maximum performance achieved by DL models to detect abusive language

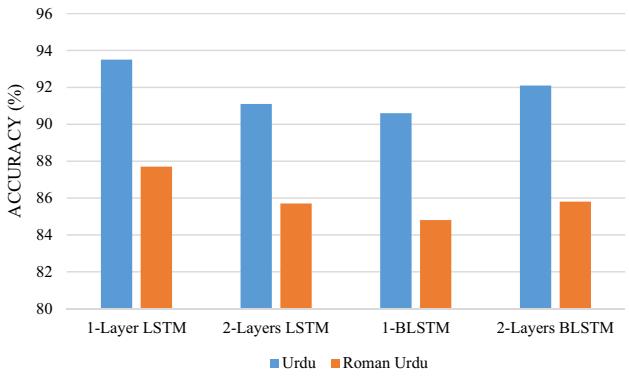

Fig. 12 Performance comparison of one-layer and two-layer architectures of LSTM and BLSTM

mini-batch size to 32 because a small batch size improves performance and needs less system memory and time [46]. BLSTM model consists of two layers of LSTM units and CLSTM is the hybrid model of CNN and LSTM. Therefore, we use the same tuned parameters of LSTM and CNN for BLSTM and CLSTM models.

5.2 Comparison of deep learning models

After tuning the hyperparameters of our models, we perform final experiments on both datasets, and the maximum accuracy achieved by each model is reported in Fig. 11. Results demonstrate that CNN outperforms the other models and achieves 96.2% and 91.4% accuracy values on both Urdu and Roman Urdu datasets. It gets the advantage of its unique architecture and the best feature extraction using convolutional and max-pooling operations. RNN-based LSTM and BLSTM models have not shown good performance on both datasets. BLSTM shows the worse performance and cannot take advantage of the previous and the next words' information to process current information. Similarly, LSTM also takes no benefit of the stored past information to perform better in current information processing. Our results endorse the results of [37], where CNN also outperforms the BLSTM and SVM. Although LSTM and BLSTM fail to process and detect abusive language very well but the hybrid model CLSTM performs better than LSTM and BLSTM. It is because CLSTM has all the features of CNN and LSTM architectures. It seems that the performance achieved by CNN architecture is decreased by the LSTM model in the CLSTM model. Therefore, CLSTM performs 1.9% and 2.8% lower than CNN's performance on Urdu and Roman Urdu.

5.3 One-layer vs. two-layers RNN models

It is shown in [47, 48] that LSTM models perform better with two hidden layers instead of a single hidden layer. Features extracted from the first layer of a model are fed into the second layer of the same model where these features are further refined and help improve the model's performance. We also explore our LSTM and BLSTM models at a deeper level and perform experiments using two hidden layers. The results of these experiments are shown in Fig. 12. Experimental results show that the LSTM model does not take advantage of its two-layer architecture and its performance goes lower than one-layer architecture. Two-layer LSTM achieves 2.4% and 2% less accuracy on Urdu and Roman Urdu than one-layer architecture. However, it is not the same in BLSTM where two-layer architecture achieves 1.5% and 1% higher accuracy on Urdu and Roman Urdu than one-layer BLSTM. From the results, we conclude that adding hidden layers in BLSTM helps to increase efficiency but it is not true for the LSTM model. Here in the second hidden layer, BLSTM performs well to capture semantic and long-term relations among the words in both directions that were missed to capture in the first hidden layer.

5.4 Deep learning vs. machine learning models

This section compares the performance of DL models with five popular ML models: NB, IBK, SVM, Logistics, and JRip. We have performed all the experiments of machine learning models in WEKA. WEKA is an open-source data mining tool that provides a GUI environment to conduct machine learning experiments [49]. In this study, we set default parameters for each classifier and did not fine-tune these parameters. We use Bag-of-Word (BoW) for feature selection because of its simplicity and, most often, in text processing. Figure 13 shows the maximum accuracy values

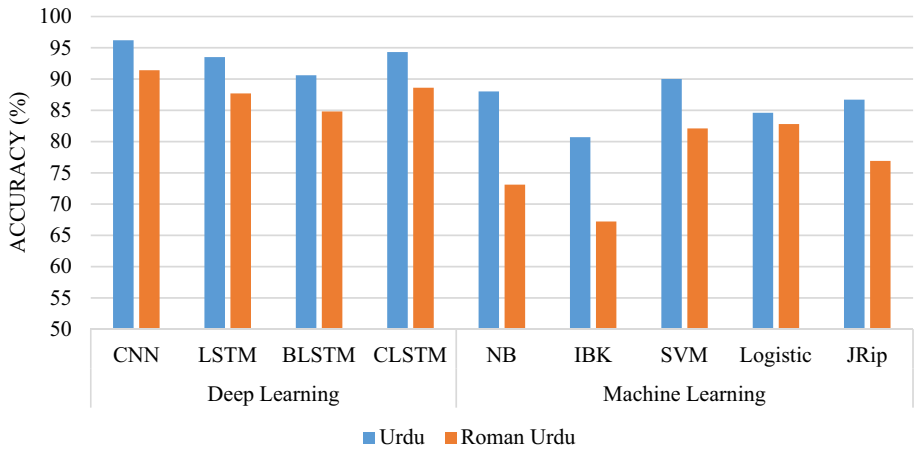

Fig. 13 Performance of DL models compared to machine learning models

Table 5 Classification results of ML and DL models using TPR, FPR, and *F*-measure

	Urdu			Roman Urdu		
	TPR	FPR	*F*-Measure	TPR	FPR	*F*-Measure
CNN	**96.1**	**3.9**	**96.2**	**91.4**	**8.6**	**91.6**
1-Layer LSTM	93.5	6.5	93.5	87.6	12.4	88.2
1-Layer BLSTM	90.6	9.4	90.6	84.8	15.2	84.9
2-Layer LSTM	91.1	8.9	91.1	85.7	14.3	85.7
2-Layer BLSTM	92.1	7.9	92.1	85.8	14.2	86.2
CLSTM	94.2	5.8	94.3	88.1	11.9	88.6
NB	88.1	11.9	88.0	73.2	26.8	73.1
IBK	80.7	19.4	80.7	69.3	30.7	67.2
SVM	***90.1***	***9.9***	***90.1***	82.2	17.8	82.3
Logistic	84.5	15.5	84.6	***82.9***	***17.2***	***82.8***
JRip	86.7	13.1	86.7	76.9	23.1	76.2

achieved by each classifier. It has proven in a few studies to detect abusive language for English that CNN did not perform better than ML models like SVM, Logistic regression [5, 39]. We report that DL models achieve high accuracy and show superior performance than ML models on Urdu and Roman Urdu. It was concluded by [39] that SVM performs good at the imbalanced dataset and less accurate on the more balanced dataset. Therefore, our results also report that SVM shows better performance on partial imbalanced dataset Urdu and achieves 90% accuracy. Whereas it decreases its performance on a balanced dataset of Roman Urdu. The logistic model achieves high accuracy than other ML models on Roman Urdu dataset with 82.8% accuracy. The accuracy achieved by IBK is the lowest than all other models for both datasets. We attempted a range of values of K from one to ten and the best results with 75.2% and 67.2% accuracy values are achieved with $K=2$.

Further, we compare the classification results of ML and DL models using three other performance measures TPR, FPR, and F-measure. The values of these performance measures on both datasets are expressed in Table 5. The analysis shows that the rate of correctly predicted abusive comments is higher among the other DL and ML models on both Urdu and Roman Urdu datasets. CNN achieves 96.1% TPR, 3.9% FPR, and 96.2% F-measure scores on the Urdu dataset. Similarly, on the Roman Urdu dataset, CNN achieves 91.4% TPR, 8.6% FPR, and 91.6% F-measure values. SVM again outperforms the other four ML models by achieving 90.1% TPR, 9.9% FPR, and 90.1% F-measure values in Urdu. On a balanced dataset of Roman Urdu abusive comments, Logistic slightly performs better than SVM. It achieves 0.7% and 0.5% higher TPR and F-measure scores, respectively, to recognize abusive and non-abusive comments from the dataset of Roman Urdu comments. The number of filters in the hidden layer of CNN successfully extract high-performing contextual features from the input text, which leads the CNN to show the best performance over other ML and DL models on both Urdu and Roman Urdu text.

After analysis of all the experimental results shown in Figs. 7, 8, 9, 10, 11, 12, 13, we can summarize our findings as follows:

- CNN outperforms the other models on both datasets, and it endorses the findings of [7] where CNN showed superior performance than NB, SVM, Logistic models.
- In the convolution layer, multiple filters of the same size perform better than multiple filters of different sizes because multiple filters of different sizes could not exploit the features of different n-grams in the short text [42], but it was not the same for the long text of Urdu [15]. It is because the abusive part of the text mostly consists of a single word in Urdu and double words in Roman Urdu.
- RNN based models LSTM and BLSTM do not perform well. BLSTM shows a worst performance than other DL models
- The hybrid model of CNN and LSTM performs better than RNN based models, and it endorses the findings of [42] to classify English language text.
- One-layer architecture of LSTM performs better than two-layer architecture, and the two-layer architecture of BLSTM increases performance than one-layer architecture.
- DL models show good performance than ML models that disagree with conclusions of [39] for English and [50] for Lithuanian languages, where DL models proved to be less accurate than ML.
- From all of the figures, all the models show high performance on the Urdu dataset because of its small vocabulary size and less number of examples. It is contradicting the findings of [50] that DL models demonstrate good results on small datasets.
- The performance of all the models remains low on the Roman Urdu dataset because of its large vocabulary size, a high number of examples, a freestyle of writing, spelling variations of the same words, etc.
- SVM performs better than other ML models on the small dataset that does not endorse the Indonesian language findings [11] where NB outperforms the others.

6 Conclusion

Automatic detection of abusive language using ML and DL methods is an open research field. In this study, the first time, we have explored DL models to detect abusive language from Urdu and Roman Urdu text. Urdu and Roman Urdu scripts both have complex and unique features. To the best of our knowledge, it is the first study that considers both scripts of Urdu. We empirically compared the performance of four DL models with five ML models. We conclude that DL models are more effective than ML models to detect abusive comments for Urdu and Roman Urdu. We have found that the CNN model outperforms the other DL and ML models on both datasets. The hybrid model CLSTM performs better than LSTM and BLSTM models. LSTM and IBK show the worst performance from the DL and ML models. One-layer architecture of LSTM performs better than two hidden layers architecture, but it is not the same for BLSTM. We have also found that the Roman Urdu script is more challenging than Urdu for automatic processing.

For future work, we aim to design a large dataset from other social media applications such as Facebook and Twitter to detect abusive language and hate speech. Because Urdu and Hindi are almost the same languages. A study to design a multilingual dataset of Urdu and Hindi and explore automatic classification methods to process the proposed dataset would significantly contribute to researchers. Further, another research direction is to explore hybrid models of DL and ML to detect abusive language [51].

Acknowledgements We acknowledge Shabana Allah Ditta for result analysis and linguistic support for the preparation of this manuscript.

Authors' contributions MPA performed the experiments and wrote the whole manuscript. IN and MA revised the manuscript. ZJ provided supervision. TZ performed results analysis. All the authors read and approved the final manuscript.

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

Availability of data and materials Both of the datasets are publically available on GitHub. Roman Urdu dataset is available at https://github.com/shaheerakr/roman-urdu-abusive-comment-detector and Urdu Offensive Dataset (UOD) is also available at https://github.com/perve zbcs/Urdu-Abusive-Dataset

Declarations

Competing interests The authors declare that they have no competing interests.

References

1. Gitari, N.D., Zuping, Z., Damien, H., Long, J.: A lexicon-based approach for hate speech detection. Int. J. Multimed. Ubiquitous Eng. **10**, 215–230 (2015). https://doi.org/10.14257/ijmue.2015.10.4.21
2. Chen, Y., Zhou, Y., Zhu, S., Xu, H.: Detecting offensive language in social media to protect adolescent online safety. Proc. - 2012 ASE/IEEE Int. Conf. Privacy, Secur. Risk Trust 2012 ASE/IEEE Int. Conf. Soc. Comput. Soc. 2012. 71–80 (2012). Doi: https://doi.org/10.1109/SocialCom-PASSAT.2012.55
3. Ptaszynski, M., Lempa, P., Masui, F., Kimura, Y., Rzepka, R., Araki, K., Wroczynski, M., Leliwa, G.: Brute-force sentence pattern extortion from harmful messages for cyberbullying detection. J. Assoc. Inf. Syst. **20**, 1075–1127 (2019). https://doi.org/10.17705/1jais.00562

4. Ingi Sigurbergsson, G., Derczynski, L.: Offensive Language and Hate Speech Detection for Danish. arXiv e-prints. arXiv:1908.04531. (2019)
5. Park, J.H., Fung, P.: One-step and Two-step Classification for Abusive Language Detection on Twitter. CoRR. abs/1706.0, (2017)
6. Pelle, R., Alcântara, C., Moreira, V.P.: A classifier ensemble for offensive text detection. Presented at the (2018)
7. Lee, Y., Yoon, S., Jung, K.: Comparative Studies of Detecting Abusive Language on Twitter. CoRR. abs/1808.1, (2018)
8. Badjatiya, P., Gupta, S., Gupta, M., Varma, V.: Deep learning for hate speech detection in tweets. In: International World Wide Web Conference Committee. pp. 759–760. , Perth, Australia (2019)
9. Ishisaka, T., Yamamoto, K.: Detecting nasty comments from BBS posts. PACLIC 24-Proc. 24th Pacific Asia Conf. Lang. Inf. Comput. 645–652 (2010)
10. Schneider, J.M., Roller, R., Bourgonje, P., Hegele, S., Rehm, G.: Towards the Automatic Classification of Offensive Language and Related Phenomena in German Tweets. (2018)
11 Ibrohim, M.O., Budi, I.: A dataset and preliminaries study for abusive language detection in indonesian social media. ProcediaComput. Sci. **135**, 222–229 (2018). https://doi.org/10.1016/j.procs.2018.08.169
12. Sigurbergsson, G.I., Derczynski, L.: Offensive Language and Hate Speech Detection for Danish. 1–13 (2019)
13 Alakrot, A., Murray, L., Nikolov, N.S.: Towards accurate detection of offensive language in online communication in arabic. ProcediaComput. Sci. **142**, 315–320 (2018). https://doi.org/10.1016/j.procs.2018.10.491
14. Tehseen, Z., Akhter, M.P., Abbas, Q.: Comparative study of feature selection approaches for urdu text categorization. Malaysian J. Comput. Sci. **28**, 93–109 (2015)
15. Akhter, M.P., Jiangbin, Z., Naqvi, I.R., Abdelmajeed, M., Mehmood, A., Sadiq, M.T.: Document-Level text classification using single-layer multisize filters convolutional neural network. IEEE Access. **8**, 42689–42707 (2020). https://doi.org/10.1109/ACCESS.2020.2976744
16. Deng, X., Li, Y., Weng, J., Zhang, J.: Feature selection for text classification: A review. Multimed. Tools Appl. (2018). https://doi.org/10.1007/s11042-018-6083-5
17 Akhter, M.P., Jiangbin, Z., Naqvi, I.R., Abdelmajeed, M., Sadiq, M.T.: Automatic detection of offensive language for Urdu and Roman Urdu. IEEE Access. (2020). https://doi.org/10.1109/ACCESS.2020.2994950
18. Akhter, M.P., Jiangbin, Z., Naqvi, I.R., Abdelmajeed, M., Fayyaz, M.: Exploring deep learning approaches for Urdu text classification in product manufacturing. Enterp. Inf. Syst. **00**, 1–26 (2020). https://doi.org/10.1080/17517575.2020.1755455
19. Lu, H.-Y., Zhang, M., Liu, Y.-Q., Ma, S.-P.: Convolution neural network feature importance analysis and feature selection enhanced model. Ruan Jian XueBao/Journal Softw. **28**, 2879–2890 (2017). https://doi.org/10.13328/j.cnki.jos.005349
20. Jain, G., Sharma, M., Agarwal, B.: Optimizing semantic LSTM for spam detection. Int. J. Inf. Technol. (2018). https://doi.org/10.1007/s41870-018-0157-5
21. Riaz, K.: Comparison of Hindi and Urdu in computational context. Int. J. Comput. Linguist. Nat. Lang. Process. **01**, 92–97 (2012)
22. Bilal, M., Israr, H., Shahid, M., Khan, A.: Sentiment classification of Roman-Urdu opinions using Naïve Bayesian, Decision Tree and KNN classification techniques. J. King. Saud. Univ. Comput. Inf. Sci. **28**, 330–344 (2016). https://doi.org/10.1016/j.jksuci.2015.11.003
23 Mehmood, K., Essam, D., Shafi, K., Malik, M.K.: Sentiment analysis for a resource poor language—Roman Urdu. ACM. Trans. Asian. Low.-Resour. Lang. Inf. Process. **19**, 1–5 (2019). https://doi.org/10.1145/3329709
24. Noor, F., Bakhtyar, M., Baber, J.: Sentiment Analysis in E-commerce Using SVM on Roman Urdu Text, https://www.scopus.com/inward/record.uri?eid=2-s2.0-85070641982&doi=10.1007%2F978-3-030-23943-5_16&partnerID=40&md5=4347fd8557834a4c814b0ae5f0ca6831, (2019)
25. Dinakar, K., Jones, B., Havasi, C., Lieberman, H., Picard, R.: Common sense reasoning for detection, prevention, and mitigation of cyberbullying. ACM Trans. Interact. Intell. Syst. (2012). https://doi.org/10.1145/2362394.2362400
26. Lee, H.S., Lee, H.R., Park, J.U., Han, Y.S.: An abusive text detection system based on enhanced abusive and non-abusive word lists. Decis. Support Syst. **113**, 22–31 (2018). https://doi.org/10.1016/j.dss.2018.06.009
27. Daud, A., Khan, W., Che, D.: Urdu language processing: a survey. Artif. Intell. Rev. **47**, 279–311 (2017). https://doi.org/10.1007/s10462-016-9482-x
28. Watanabe, H., Bouazizi, M., Ohtsuki, T.: Hate speech on Twitter: a pragmatic approach to collect hateful and offensive expressions and perform hate speech detection. IEEE Access. **6**, 13825–13835 (2018). https://doi.org/10.1109/ACCESS.2018.2806394
29. Bouazizi, M., Otsuki, T.: A pattern-based approach for sarcasm detection on Twitter. IEEE Access. **4**, 5477–5488 (2016). https://doi.org/10.1109/ACCESS.2016.2594194
30 Razavi, A.H., Inkpen, D., Uritsky, S., Matwin, S.: Offensive language detection using multi-level classification. Lect. Notes Comput. Sci. **6085 LNAI**, 16–27 (2010). https://doi.org/10.1007/978-3-642-13059-5_5
31. Burnap, P., Williams, M.L.: Cyber hate speech on twitter: an application of machine classification and statistical modeling for policy and decision making. Policy Internet **7**, 223–242 (2015). https://doi.org/10.1002/poi3.85
32. 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. pp. 668–671. Association for Computational Linguistics, Minneapolis, Minnesota, USA (2019)
33. Ptaszynski, M., Dybala, P., Matsuba, T., Masui, F., Rzepka, R., Araki, K., Momouchi, Y.: In the service of online order: tackling cyberbulling with machine learning and affect analysis. Int. J. Comput. Linguist. Res. **1**, 135–154 (2010)
34. Mishra, P., Tredici, M. Del, Yannakoudakis, H., Shutova, E.: Abusive language detection with graph convolutional networks. CoRR. abs/1904.0, (2019)
35. Mehmood, K., Essam, D., Shafi, K.: Sentiment Analysis System for Roman Urdu BT-Intelligent Computing. In: Arai, K., Kapoor, S., Bhatia, R. (eds.) Advances in Intelligent Systems and Computing, pp. 29–42. Springer International Publishing, Cham (2019)
36 Alakrot, A., Murray, L., Nikolov, N.S.: Dataset construction for the detection of anti-social behaviour in online communication in Arabic. ProcediaComput. Sci. **142**, 174–181 (2018). https://doi.org/10.1016/j.procs.2018.10.473
37. Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., Kumar, R.: Predicting the type and target of offensive posts in social media. arXiv preprint arXiv (2019). https://doi.org/10.18653/v1/n19-1144
38 Burnap, P., Williams, M.L.: Us and them: identifying cyber hate on Twitter across multiple protected characteristics. EPJ Data Sci. (2016). https://doi.org/10.1140/epjds/s13688-016-0072-6
39. Chen, H., McKeever, S., Delany, S.J.: A Comparison of Classical Versus Deep Learning Techniques for Abusive Content Detection on Social Media Sites BT-Social Informatics. Presented at the (2018)
40 Akhter, M.P., Jiangbin, Z., Naqvi, I.R., Abdelmajeed, M., Fayyaz, M.: Exploring deep learning approaches for Urdu text classification in product manufacturing. Enterp. Inf. Syst. (2020). https://doi.org/10.1080/17517575.2020.1755455

41. Kim, Y.: Convolutional Neural Networks for Sentence Classification. CoRR. abs/1408.5, (2014)
42. Zhou, C., Sun, C., Liu, Z., Lau, F.C.M.: A {C-LSTM} Neural Network for Text Classification. CoRR. abs/1511.0, (2015)
43. Akhter, M.P., Jiangbin, Z., Sadiq, M.T.: Automatic detection of offensive language for Urdu and Roman Urdu. IEEE Access. **8**, 1–14 (2020)
44 Maldonado, S., López, J.: Dealing with high-dimensional class-imbalanced datasets: Embedded feature selection for SVM classification. Appl. Soft. Comput. **67**, 94–105 (2018). https://doi.org/10.1016/j.asoc.2018.02.051
45. Yang, S., Sun, Q., Zhou, H., Gong, Z.: A multi-layer neural network model integrating BiLSTM and CNN for Chinese sentiment recognition. Presented at the (2018)
46. Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. CoRR. abs/1609.0, (2016)
47. Zia, T., Zahid, U.: Long short-term memory recurrent neural network architectures for Urdu acoustic modeling. Int. J. Speech Technol. **22**, 21–30 (2019). https://doi.org/10.1007/s10772-018-09573-7
48 Rao, G., Huang, W., Feng, Z., Cong, Q.: LSTM with sentence representations for document-level sentiment classification. Neurocomputing **308**, 49–57 (2018). https://doi.org/10.1016/j.neucom.2018.04.045
49. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: TheWEKA data mining software: an update. SIGKDDExplor. Newsl. **11**, 10–18 (2009). https://doi.org/10.1145/1656274.1656278
50 Kapočiūtė-Dzikienė, J., Damaševičius, R., Woźniak, M.: Sentiment analysis of Lithuanian texts using traditional and deep learning approaches. Computers. (2019). https://doi.org/10.3390/computers8010004
51. Tripathy, A., Anand, A., Rath, S.K.: Document-level sentiment classification using hybrid machine learning approach. Knowl. Inf. Syst. **53**, 805–831 (2017). https://doi.org/10.1007/s10115-017-1055-z