IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 20 September 2022, accepted 14 October 2022, date of publication 21 October 2022, date of current version 22 November 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3216375

RESEARCH ARTICLE

Context-Aware Deep Learning Model for Detection of Roman Urdu Hate Speech on Social Media Platform

MUHAMMAD BILAL[1], ATIF KHAN[1], SALMAN JAN[2,3], AND SHAHRULNIZA MUSA[2,4]

[1]Department of Computer Science, Islamia College University Peshawar, Peshawar 25120, Pakistan
[2]Malaysian Institute of Information Technology, Universiti Kuala Lumpur, Kuala Lumpur 50250, Malaysia
[3]Department of Computer Science, Bacha Khan University Charsadda, Charsadda 24420, Pakistan
[4]UniKL–LR University Joint ICT Laboratory (KLR-JIL), Universiti Kuala Lumpur–La Rochelle University, Kuala Lumpur 50250, Malaysia

Corresponding author: Salman Jan (salman.jan@unikl.edu.my)

ABSTRACT Over the last two decades, social media platforms have grown dramatically. Twitter and Facebook are the two most popular social media platforms, with millions of active users posting billions of messages daily. These platforms allow users to have freedom of expression. However, some users exploit this facility by disseminating hate speeches. Manual detection and censorship of such hate speeches are impractical; thus, an automatic detection mechanism is required to detect and counter hate speeches in a real-time environment. Most research in hate speech detection has been carried out in the English language. Still, minimal work has been explored in other languages, mainly Urdu written in Roman Urdu script. A few research have attempted machine learning, and deep learning models for Roman Urdu hate speech detection; however, due to a scarcity of Roman Urdu resources, and a large corpus with defined annotation rules, a robust hate speech detection model is still required. With this motivation, this study contributes in the following manner: we developed annotation guidelines for Roman Urdu Hate Speech. Second, we constructed a new Roman Urdu Hate Speech Dataset (RU-HSD-30K) that was annotated by a team of experts using the annotation rules. To the best of our knowledge, the Bi-LSTM model with an attention layer for Roman-Urdu Hate Speech Detection has not been explored. Therefore, we developed a context-aware Roman Urdu Hate Speech detection model based on Bi-LSTM with an attention layer and used custom word2vec for word embeddings. Finally, we examined the effect of lexical normalization of Roman Urdu words on the performance of the proposed model. Different traditional as well as deep learning models, including LSTM and CNN models, were used as baseline models. The performance of the models was assessed in terms of evaluation metrics like accuracy, precision, recall, and F1-score. The generalization of each model is also evaluated on a cross-domain dataset. Experimental results revealed that Bi-LSTM with attention outperformed the traditional machine learning models and other deep learning models with an accuracy score of 0.875 and an F-Score of 0.885. In addition, the results demonstrated that our suggested model (Bi-LSTM with Attention Layer) is more general than previous models when applied to unseen data. The results confirmed that lexical normalization of Roman Urdu words enhanced the performance of the suggested model.

INDEX TERMS Bi-LSTM, CNN, deep learning, hate speech, LSTM, natural language processing (NLP), Roman Urdu, social media.

I. INTRODUCTION

Over the past two decades, there has been a substantial increase in social media users. Twitter is a well-known social networking site. According to the report, Twitter has 330 million active users (J. Clement, 2020), and 200 billion tweets are posted annually. Likewise, Facebook has over 2.89 billion active users. Facebook users share an average of 4.75 billion items and 10 billion messages daily. These platforms enable users to openly share their ideas and perspectives on a

The associate editor coordinating the review of this manuscript and approving it for publication was Wei-Yen Hsu.

particular subject relating to current events, religion, politics, sports, the entertainment industry, etc.

On the other hand, some users abuse this service by promoting hatred, instigating violence, making words that incite conflict, and using inappropriate and abusive language. Some conduct online harassment based on gender, religion, race, or sexual orientation. In most terror incidents, the suspects have a long history of hate speech on social media, indicating that social media is a factor in their extremist activity. Facebook was used to live-stream a 2019 terrorist assault in Christchurch, New Zealand. Similarly, on April 14, 2017, a student at a Pakistani university was murdered by a mob of other students over anti-religious Facebook statements. A suicide bomb attack on a Shia Mosque during Friday prayers in Kandahar (Afghanistan) in October 2021 became a Twitter trend, resulting in nasty messages from Shia and Sunni faiths, Afghans, and Pakistanis, who blamed each other for the catastrophe. Such hate speech and abusive language incite violence and aggressiveness, making protecting human rights more challenging.

Furthermore, political campaigns against political leaders on social media employ highly derogatory language. Social media posts that hurt underrepresented groups based on their religious beliefs might provoke violence. Similarly, Muslims worldwide have an ardent affection for the Prophet Muhammad (SAW), which is an integral part of their religion. Therefore, anything shared on social media that promotes blasphemy against the Prophet Muhammad is intolerable to Muslims, and the scenario may result in violence. Moreover, the comments on social media about celebrities are vulgar. Likewise, online harassment of women is on the rise. According to one survey, 66% of adolescent girls claim to have been bullied on Facebook. Similarly, 73% of adults reported having experienced online harassment, and 40% had been personally targeted (Pew Research Center 2017).

The government enacted regulations such as the ''National Action Plan'' and the ''Prevention of Electronic Crimes Act, 2016'' to eradicate online hate speeches from electronic media. Advanced artificial intelligence (AI) tools to identify online hate speech must be developed to implement these regulations. Facebook's proactive detection rate for hate speech has increased 14 percent over the past year, from 80 percent to 94 percent (Facebook Community Standards Enforcement Report, 2020). However, this feature only works in English. They developed their technology to incorporate Spanish, Arabic, Indonesian, and Burmese, among other languages. However, little effort has been taken to combat hate speech written in Roman Urdu script, which is extensively used by social media users in Pakistan, Bangladesh, India, and Afghanistan. Thus, automatically detecting Roman Urdu hate speech on social media platforms is challenging. Numerous researchers have attempted to identify online hate speech in English, as stated in Section 2, and have identified the following significant obstacles:

1. Hate Speech Definition and Data Annotations
2. Understanding of User's Intention and Context
3. Robustness and Vulnerability of Model
4. Cross Domain Testing and Generalization
5. Data Biasness

1) Class Imbalance

However, there are specific challenges to Hate Speech Detection in Roman Urdu text, including:

1. Lack of annotated datasets.
2. Nonexistence of pre-trained embedding in Roman Urdu script.
3. Absence of guidelines.
4. Lack of an extensive benchmark dataset.
5. Lexical diversity in Roman Urdu terms.

To overcome the concerns mentioned earlier regarding the detection of Roman Urdu Hate Speech, we made the following contributions:

1. To develop annotation guidelines for Roman Urdu Hate Speech.
2. To construct a new Roman Urdu Hate Speech Dataset (RU-HSD-30K) annotated by a team of experts using the annotation guidelines developed for this study.
3. To develop a context-aware Roman Urdu Hate Speech detection model based on Bi-LSTM with attention layer and custom word2vec word embeddings and compare its efficacy to the state-of-the-art deep learning models.
4. To improve the performance of the proposed model by performing lexical normalization of Roman Urdu words.

II. RELATED WORK

This section describes previous studies undertaken on the detection of hate speech, which is summarized in Table 1. The literature is categorized as follows:

A. NORMALIZATION OF LEXICAL VARIATIONS

Roman Urdu is the name given to Urdu written with the English alphabet, depending on word pronunciation. Roman Urdu is not a standard language; hence its lexicon lacks standard spellings (words). Different individuals spell similar words differently. Therefore, it is difficult to adapt the numerous lexical variants of Roman Urdu terms to standard spelling, and there has been little research undertaken in this area. The authors in [1] devised a feature-based clustering technique that encodes roman Urdu words into phonetic representations and then clusters lexical variants of roman Urdu words with identical phonetics. The technique was evaluated using a manually annotated gold standard dataset and achieved a higher F-score than the baseline. Likewise, the authors of [2] proposed a method based on a phonetic algorithm to normalize lexical variations in the Roman Urdu text.

The authors of [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13] conducted a comparative study to assess how normalization techniques handle lexical variation in the text of social media posts in several languages, including English,

Japanese, Chinese, Bangla, Dutch, Finnish, Spanish, Arabic, Polish, and Roman Urdu. The different normalization techniques include the Rule-based approach, machine learning algorithms, phonetic algorithms, stemming and lemmatization, etc.

B. ANNOTATION OF DATASETS

Roman Urdu lacks annotated datasets but does have pre-trained embeddings. The viability of transfer learning was investigated by [14], which also examined the performance of five distinct multilingual embeddings, namely LASER, ELMo, BERT, XLM-RoBERTa, and FastText, for Roman Urdu text. This work asserted the development of RomUrEm, a pre-trained BERT model for Roman Urdu, but it is not accessible to scholars for further study and reuse.

The authors of [15] utilized an iterative strategy to develop annotation guidelines for their own constructed Hate Speech Roman Urdu 2020 corpus (HS-RU-20). The corpus was intended to be utilized for Hate Speech detection.

C. LEXICAL RULE-BASED APPROACHES

Researchers have proposed lexical rule-based approaches for detecting offensive and hate speech terms effectively. The work of [16] offered a lexicon-based methodology for identifying hate speech that detects subjectivity in sentences and constructs a vocabulary of hate-related words using a rule-based mechanism. A classifier is then trained on features collected from the lexicon and tested on the document to detect hate speech. Similarly, the authors of [17] presented VADER, a rule-based approach for sentiment analysis of content from social media. They designed and validated Gold Standard Sentiment Lexicon. They identified and analyzed rules concerning the conventional use of grammatical and syntactic aspects of text and compared the performance of lexical rule-based approaches to the baseline models.

The most significant shortcoming of lexical rule-based techniques is their inability to capture the context and domain of words in a document. In addition, rule-based approaches are lexicon-based; therefore, these methods are not robust against adversaries and subtleties in the text.

D. TRADITIONAL MACHINE LEARNING APPROACHES

Historically, lexical approaches have been deemed successful for detecting potentially offending phrases. Nonetheless, additional research revealed that, except for a few terms indicated by the Hatebase lexicon and identified by Human Coders, the lexical approaches gave erroneous findings in detecting hate speech. In contrast, machine learning algorithms performed significantly better than lexical techniques for hate speech detection [18].

The authors of [18] discovered that Logistic Regression and Linear SVM beat the other models for detecting hate speech in tweets by a significant margin. In addition, the study showed that biases in the dataset and the existence of overlapping phrases in both hate speech and offensive tweets led to misclassifications.

The research of [19] demonstrated that all proposed cutting-edge models are vulnerable to adversaries. They proved that adversarial training could not fix the issue fully and argued that character-level features are more resistant to attack than word-level features. They suggested Logistic Regression with character-level features for the development of a robust model for detecting hate speech.

The authors of [20] proposed a novel technique known as Multi-view SVM, which has the extra benefit of improved interpretability and outperformed the previous algorithms for detecting hate speech. Similarly, the work of [21] demonstrated that the bigram features, when combined with the support vector machine method, had the highest overall accuracy (79%) for automatic Hate Speech Detection.

The authors of [22] proposed the SVM- Radial Basis Function (RBF) method with character level FastText for Hate speech detection in Hindi-English code-mixed social media text. They demonstrated that character-level features of FastText provide more information than word and document-level features for classification.

The majority of hate speech detection efforts have been carried out in English. Detecting hate speech in Roman Urdu has received relatively little attention. The authors of [15] applied various machine learning algorithms for Roman Urdu Hate Speech detection and demonstrated that Logistic Regression outperformed other machine learning approaches and the deep learning approach (CNN) in distinguishing between neutral and hostile tweets. In addition, they asserted that a bag of words is an appropriate feature extraction method for detecting Roman-Urdu Hate Speech. Traditional Machine Learning models function effectively as long as the size of the dataset is restricted. However, the performance of conventional machine learning models declines for huge datasets.

E. DEEP LEARNING APPROACHES

Deep Learning outperforms other techniques if the data size is large. However, with small data sizes, traditional Machine Learning algorithms are preferred. The work by [23] proposed CNN with word2vec embedding for abusive content detection on social media sites and showed that deep learning models could outperform the classification results of the traditional SVM when the training dataset is imbalanced. The performance of the SVM can be dramatically improved through oversampling. This work assists researchers in selecting acceptable text classification algorithms for the identification of abusive content, including cases in which the training datasets exhibit class imbalance.

Similarly, the authors of [24] conducted numerous studies to detect cyberbullying on various social media platforms (SMPs). They utilized three datasets (Formspring with 12,000 posts, Twitter with 16,000 posts, and Wikipedia with 100,000 posts) and four Deep Neural Network (DNN) models, namely CNN, LSTM, Bidirectional LSTM (BiLSTM), and Bidirectional LSTM (BiLSTM) with Attention. All DNN models utilized the same fundamental structure as [25]. It was observed that the models favored non-bullying since the datasets were

completely unbalanced, with bullying constituting the minority class. Moreover, it was discovered that oversampling considerably enhanced performance. In conclusion, they determined that DNN models combined with Transfer Learning beat state-of-the-art outcomes on all three datasets. This work, however, encountered the issue of overfitting.

The authors of [26] utilized the BERT model for Abusive Language classification and demonstrated that BERT performs better than other state-of-the-art models when fine-tuned for the underlying problem. However, BERT can be fine-tuned for languages for which the BERT model has been pre-trained.

The authors of [27] proposed a multi-channel model with three variants of BERT (MC-BERT) for hate speech detection: English, Chinese, and multilingual BERTs. In addition, they investigated the use of translations as extra input by translating training and test sets into the languages required by various BERT models. They discovered that fine-tuning of the Pre-trained BERT model achieved state-of-the-art or comparable performance on three distinct datasets. Roman Urdu, however, lacks a pre-trained BERT model and is exceedingly difficult to translate due to the lexical variations of Roman Urdu words. In addition, dictionaries in both languages are required for translation and no Roman Urdu dictionary exists that includes the Standard Lexical form of Roman Urdu words.

The work of [14] introduced CNN-gram, a new deep learning architecture for detecting hate speech and abusive language in Roman Urdu, and compared its performance with seven existing baseline techniques on the RUHSOLD dataset. The proposed model showed more robustness as compared to the baselines. The work by [28] demonstrated that the performance of Neural-based approaches for the detection of hate speech is better than Classical ML techniques. Amongst Neural network architecture, BI-LSTM, with multiple embeddings, exhibited the best performance in detecting hate speech.

The authors [29] presented a transfer learning strategy for detecting hate speech based on an existing pre-trained language model known as BERT and assessed the proposed model using two publicly available datasets. Next, they developed a method for reducing the influence of bias in the training set during the fine-tuning of a pre-trained BERT-based model for the hate speech detection task.

The study by [30] demonstrated a pipeline for adapting the general-purpose RoBERT model to detect Vietnamese hate speech. They proposed a pipeline that greatly improved performance, attaining a new F1 score of 0.7221 for the Vietnamese Hate Speech Detection (HSD) campaign.

The work of [31] introduced HateXplain, the first benchmark dataset of hate speech spanning several facets of the topic. Using current state-of-the-art models, they noticed that even models that perform exceptionally well in classification do not score well on explainability criteria such as model plausibility and model faithfulness. Additionally, they discovered that models that employed human rationales for training are more effective at minimizing inadvertent bias towards target communities.

The research of [24], [25] was thoroughly analyzed by [32], which concluded that the findings provided by state-of-the-art systems indicate that supervised techniques attain nearly perfect performance, but only on particular datasets, the majority of which are in English. They examined the apparent disparity between available literature and real applications. They investigated the experimental methodology utilized in prior studies [24], [25] and their generalizability to additional datasets. Their findings revealed methodological flaws and a significant dataset bias. As a result, current state-of-the-art performance claims have become greatly exaggerated. They identified that the majority of the difficulties are due to data overfitting and sampling concerns.

According to a study [33], RNN outperformed other shallow and deep learning models in terms of accuracy on Dataset-1 and Dataset-2, with scores of 0.7871 and 0.9030, respectively. However, the datasets were distributed in an imbalanced manner in this study. In addition, the study contributes to the detection of English hate speech posted on social media during the COVID-19 era but was unable to detect Roman Urdu hate speech made on social media.

F. UNSUPERVISED LEARNING

Numerous researchers have proposed unsupervised learning approaches to the problem of hate speech detection. The authors of [34] proposed the Growing Hierarchical Self-Organizing Map (GHSOM), a method of unsupervised learning for cyberbullying detection on social media. This research retrieved hand-crafted features that were utilized to capture the semantic and syntactic traits of cyber bullies. The proposed strategy was effective in detecting cyberbullying on social media platforms. However, the approach was incapable of identifying sarcastic text. In addition, the work is restricted to the English language. The authors of [35] investigated a unique framework for detecting frequently discussed subjects/topics on Facebook that cause hate speech. They utilized graphs, sentiment and emotion analysis approaches to cluster and analyze posts on popular Facebook pages.

Consequently, the proposed framework can automatically detect pages that promote hate speech in comment sections on sensitive themes. According to the results, the proposed method achieved an accuracy of 0.74. However, this work is restricted to English hate speeches and uses English language and slurs within the context of American society. Language and other demographic factors influence hate speech. In Roman Urdu, insults and derogatory terms differ from their English counterparts. The authors of [39] presented an unsupervised approach for detecting German Hate Speech on Twitter. They used the skip-grams method to determine the context of the words and the k-mean clustering methodology to group them into meaningful categories, such as immigration, crime, and politics. The machine learning model was used to detect hate speech automatically. A complete qualitative and quantitative investigation of what constitutes

TABLE 1. Summary of the related work.

Method	Authors/ Year	Proposed Methodology	Features	Dataset	Language	Evaluation Score	Contribution	Limitations
Lexical Rule-based Approaches	[16]	Lexical Rule Based	Negative Polarity, Hate Verbs, and Theme-based grammatical pattern	Micro-blogging sites	English	F1: 0.70 Prec.:0.73 Recall: 0.73	Proposed a lexicon-based method for detecting hate speech that detects subjectivity in sentences and generates a vocabulary of hate-related words.	The approach uses the lexical rule-based method that does not consider the domain and context of terms in a document.
	[17]	VADER: A Lexical Rule-Based Model	Senti-WordNet VADER	Gold Standard Sentiment Lexicon	English	F 1: 0.96	1. Proposed VADER, a rule-based approach for sentiment analysis of social media text. 2. They created and validated the human-curated Gold Standard Sentiment Lexicon.	1. The suggested solution relies on a valence-aware dictionary or lexicon and cannot recognize the context of text messages to determine if it is hate speech. 2. Since the proposed rule-based strategy is lexicon-based, it lacks resistance to adversaries and textual subtleties.
Supervised Learning (Traditional Machine Learning)	[15]	Logistic Regression	Bag of Words (BOW)	HS-RU-20	Roman Urdu	Acc: 0.84 F1: 0.906 Prec.:0.84 Recall: 0.977	1. Contributed Roman Urdu 2020 Corpus of Hate Speech 2. For Roman Urdu hate speech identification, Logistic Regression with a bag of words feature extraction beat other machine learning methods as well as CNN (Deep Learning).	1. This study produced a small dataset of 5000 tweets, which was biased and needs class-balancing. 2. Logistic Regression with a bag of words feature extraction outperformed ML and Deep learning techniques. The findings might change, if the models are trained on a large dataset and word embedding techniques are used.
	[18]	Logistic Regression with L regularization	Bigram, unigram, and trigram features are weighted by their TF-IDF.	Davidson et al. (2017)	English	F1: 0.906 Prec.:0.9 Recall: 0.9	Proposed Logistic Regression with L regularization.	1. The work is confined to English language hate speech detection. 2. Logistic regression may perform poorly on a huge dataset.
	[19]	Logistic Regression	Char level features	Wikipedia	English	F 1: 0.75	1. Proposed character-level features and showed that models trained on character-level features are more resistant to attacks than those trained on word-level features.	1. This work is limited to the English language and discussed alternative ways for mitigating adversarial attacks. 2. The issues of Roman Urdu text are distinct, including the lexical variety of Roman Urdu words, the infeasibility of dictionaries for Roman Urdu words,
	[20]	Multi-view SVM	Word TF-IDF	Stormfront	English	Acc: 0.8033 F1: 0.8031	They proposed the Multi-view SVM technique with the added advantage of enhanced interpretability, which outperformed current systems.	This work is limited to the English language and did not consider the user intent and context when analyzing hate speech.
	[21]	SVM	TF-IDF Bi-gram	A dataset composed of 14509 tweets	English	Acc: 0.79 F1: 0.77 Prec.:0.77 Recall: 0.79	Bigram features combined with SVM performed best in automatic Hate Speech Detection with 79% accuracy.	1. The suggested ML model lacks real-time predictions and is unable to capture text message context. 2. The performance of classical models may decline as the dataset size grows.
	[22]	SVM- Radial Basis Funcrion (RBF)	FastText pre-trained word embedding + bilingual embedding	HASOC	Hindi-English	Acc: 0.8581 F1: 0.8580 Prec.:0.8586 Recall: 0.85	Demonstrated that FastText character level features provided more information for code mixed test classification than word and document level features.	The work is confined to Hindi and English languages
Supervised Learning (Deep Learning)	[36]	Deep Multi-tasking Model	Character n-gram	Waseem (2016)	English	F1: 0.74 Prec.:0.73 Recall: 0.78	They used Multi-Task Learning to recognize hate speech, which significantly improves the model's generalizability to new datasets.	Utilized Waseem's dataset, which had issues of user biases and imbalance, whereas, in Davidson's dataset, a significant proportion of positive classes are African American. Furthermore, the work is restricted to identifying hate speech in English.
	[23]	CNN	Word2Vec	D3	English	Recall: 0.93	1. Demonstrated that CNN outperforms SVM for abusive content identification when the training dataset is uneven. 2. Showed Oversampling improves SVM performance beyond the deep learning model	When the dataset contains millions of text messages, the SVM may not perform well on a balanced dataset.
	[24]	BiLSTM with Attention	GloVe embeddings, SSWE embeddings	Formspring	English	F1: 0.94 Prec.:0.94 Recall: 0.94	Showed that the Deep Learning model (BiLSTM) outperformed Machine Learning models for cyberbullying detection.	To balance the classes, they oversampled the entire dataset before to train-test split and before initiating the cross-validation procedure, which leads to overfitting.
	[26]	BERT	Pre-trained BERT	Davidson et al (2017).	English	Acc: 0.917 F1: 0.772	Showed that BERT performed better than other state-of-the-art models when fine-tuned for the underlying problem.	Fine-tuned BERT cannot be applied to other scripted languages, such as Roman Urdu, as there is no pre-trained BERT Model.
	[27]	Multi-Channel BERT (MC_BERT 4 HATE)	Pre-trained BERT	GermEval	Chinese	Acc: 0.80 F1: 0.764	1. Proposed a multi-channel model with three variants of BERT (MC-BERT) for hate speech detection. 2. Investigated the use of translations as extra input by translating training and test sets into the languages required by various BERT models.	1. Proposed work fine-tuned the pre-trained BERT and translated languages into the corresponding languages required by different BERT models. But, this work cannot be extended to Roman Urdu. 2. Roman Urdu, however, lacks a pre-trained BERT model and is exceedingly difficult to translate due to the lexical variations of Roman Urdu words.
	[14]	CNN-gram composed of four CNN layers.	RomUrEm (proposed)	RUHSOLD	Roman Urdu	Acc: 0.85 F1: 0.84 Prec.:0.84 Recall: 0.8	1. Developed annotated dataset of Roman Urdu tweets called RUHSOLD consisting of 10, 012 tweets. 2. They also proposed a novel deep learning architecture called CNN-gram composed of four CNN layers.	This work claimed the development of RomUrEm, a pre-trained BERT model for Roman Urdu, but practically it is not available for researchers to investigate its performance.
	[28]	BiLSTM, with an attention layer and two FF (Feed Forward)	• TF/IDF • GloVe • FasText • BERT	ETHOS	English	Acc: 0.77 F1: 0.768 Prec.:0.78 Recall: 0.77	1. Presented a protocol for creating a suitable textual dataset called 'ETHOS', based on YouTube and Reddit comments validated through the figure-eight crowdsourcing platform. 2. Demonstrated that performance of Neural-based approach (BiLSTM) is better than Classical ML techniques.	1. Only English comments were extracted from social media, so the HS detection dataset was not multi-purpose. 2. The corpus utilized in this study consists of shorter sentences. If segmentation is not utilized, the models trained on this dataset may be unable to detect HS in documents on a broader scale.
	[29]	BERT	pre-trained language model BERT	Davidson-dataset	English	F1 : 0.91 Prec.:0.91 Recall: 0.90	1. A loss-based fine-tuning technique is implemented to fine-tune the pre-trained BERT model using new re-weighted training data.	The work is confined to AAE/SAE languages, but it is not tested on other cross-domain datasets containing different languages/dialects.

TABLE 1. *(Continued.)* **Summary of the related work.**

Supervised Learning (Deep Learning)	[30]	RoBERTa language model	Pre-trained BERT for Vietnamese (RoBERTa)	HSD dataset	Vietnamese	F1: 0.721	Proposed a pipeline that substantially improves performance and obtained a new state-of-the-art level of performance in Vietnamese Hate Speech Detection (HSD).	This work is limited to hate speech detection in the Vietnamese language.
	[31]	BERT-HateXplain [LIME]	BERT pretrained	Twitter and Gab	English	Acc: 0.698 F1: 0.687	1. This work introduced HateXplain, a new benchmark dataset for hate speech detection. 2. Showed that the models, which utilize the human rationales for training, performed better in reducing unintended bias towards target communities.	The work is limited to the English language. It did not consider multilingual hate speech into account. The technique may be applied to other languages.
	[37]	BERT-HI BERT-EN	Word Embedding	Multilingual Dataset	English and Hindi	**Hindi:** Acc: 0.95 F1: 0.81 **English:**	Proposed a multilingual system that operates across multiple languages, and the proposed transfer learning techniques allow for the rapid addition and training of a new language.	1. The proposed system is evaluated in both Hindi and English. 2. The performance in other languages has not yet been evaluated
	[38]	Multi-task learning (MTL) Model	Pre-trained BERT (MarBERT).	OSACT-HS	Arabic	Acc: 0.98 F 1 :0.88	1. Evaluated a new pre-trained model, MarBERT, to classify both DA and MSA tweets. 3. Proposed a model to explore multi-corpus-based learning using Arabic LMs and MTL to improve the classification performance on Arabic offensive and hate speech detection.	The proposed model is based on the Arabic language. It cannot be applied to Roman Urdu because there is no pre-trained BERT model available for Roman Urdu.
	[33]	Deep Learning (RNN)	Stemmer, Bag-of-Words, TF-IDF, and Document matrix	Dataset-1: 2319 tweets Dataset-2: 20,000 tweets	English	Acc: 0.787 F1: 0.451 Acc: 0.90 F 1 :0.0.73	Showed that RNN performed better than other shallow and deep learning models in terms of accuracy on Dataset-1 and Dataset-2	1. Used unbalanced distribution of the datasets. 2. The work is confined to the detection of English hate speeches posted on social media during the COVID-19 period, but it cannot be extended to other languages such as Roman Urdu.
Unsupervised Learning	[34]	Growing Hierarchical Self Organizing Map (GHSOM)	Hand Crafted Features	FormSpring.me YouTube	English	Acc: 0.69 F1: 0.74 Prec.:0.60 Recall: 0.94	1. To detect cyberbullying on social media, an unsupervised learning technique based on a Growing Hierarchical Self-Organizing map was applied. 2. Extracted Handcrafted features that are capable of capturing semantic and syntactic characteristics of cyber bullies.	The approach was incapable of recognizing sarcastic messages. Additionally, the work is restricted to the English language.
	[35]	K-Mean Clustering	VADER JAMMIN TF-IDF	Facebook	English	Acc: 0.744	Proposed a novel unsupervised method based on graph analysis for identifying websites that may be disseminating hate speech.	1. The proposed model can be enhanced by Deep Learning systems with embeddings. 2. A further enhancement would be to account for the multimodal nature of tweets, such as text-image relationships.
	[39]	K-Mean Clustering	n-gram and k-means	50,000 hateful and 50,000 safe German Tweets	German	Acc: 0.84 F1: 0.84 Prec.:0.84	1. Used the skip-grams method to find the context of the words and employed the K-means clustering technique to group these words into meaningful categories.	This study is limited to contributing to hate speech detection of German tweets.
Hybrid Approaches	[25]	LSTM+GBDT (Gradient Boosted Decision Tree)	Random Embedding	Waseem and Hovy in 2016	English	F1: 0.93 Prec.:0.93 Recall: 0.93	Investigated the application of deep learning methods for hate speech identification and found that deep learning approaches outperform state-of-the-art char/word n-gram algorithms. The best method was LSTM + Random Embedding + GBDT.	The deep learning methods performed well only on the same dataset and do not generalize on the cross dataset. The reason is that it uses complete labeled data for feature extraction and embedding before splitting the dataset into train and test datasets and induces over-fitting which leads to an overestimated score.
	[40]	CNN, Gated Recurrent Units (GRU), CNN + GRU, BERT	Pre-trained word2vec model	Arabic dataset	Arabic	F1: 0.79	1. Developed a public dataset of 9316 Arabic hate tweets. 2. Investigated four models, CNN, gated recurrent units (GRU), CNN + GRU, and BERT, using a series of experiments on two datasets.	The work is based on Hate Speech detection on Arabic tweets and hence cannot be extended to other scripting languages such as Roman Urdu
	[41]	Bi-LSTM + CNN	Character Embedding based on FastText	TOCP (Chinese dataset)	Chinese	F1: 0.86 Prec.:0.85 Recall: 0.87	1. Developed TOCP, a larger dataset of Chinese profanity. 2. Showed that the most effective profanity detection algorithm consisted of two layers of BiLSTM preceded by CNN.	The proposed system is limited to Chinese profanity detection.
Meta-heuristic Approach	[42]	AntLion and MothFlame Optimization-Based Meta-heuristic approaches	BoW+TF + Word2Vec	Twitter Web Forum Twitter	English Spanish	**English** Acc: 0.73 F 1: 0. 71 **Spanish** Acc: 0.721 F 1: 0.70	This work proposed a meta-heuristic approach based on AntLion and MothFlame optimization algorithms for automatic hate speech detection(HSD)	Popular optimization algorithms such as genetic algorithm and particle swarm optimization should also be explored for HSD.

hate speech from a political communication perspective was also conducted. The outcome demonstrated that the proposed method yields an accuracy score of 84.21 and an F-score of 84.21. However, this study is confined to detecting hate speech in German tweets. In addition, the machine learning model utilized in this study may be prone to overfitting and may perform poorly on datasets from other domains.

G. METAHEURISTIC APPROACHES

A study of [42] suggested a meta-heuristic approach for automatic hate speech detection based on the AntLion and MothFlame Optimization algorithms. To extract features, they utilized Bag of Words (BoW), Term Frequency (TF), and document vector (Doc2Vec). The suggested model was evaluated on three datasets, with ALO and MFO exhibiting

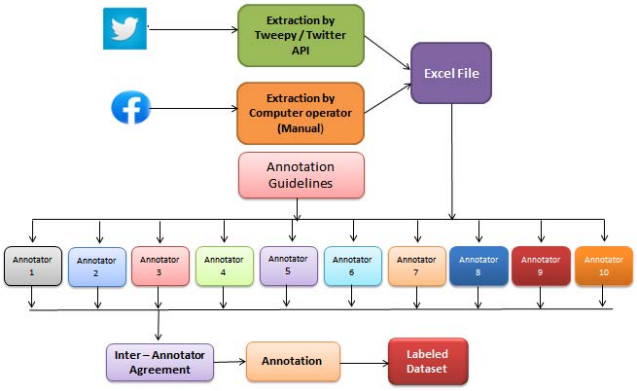

FIGURE 1. Dataset development.

the highest accuracy (0.921 and 0.90, respectively. This work is applied to datasets in English and Spanish; however, the technique might be extended to other languages, such as Roman Urdu.

The literature revealed that researchers have worked on Hate Speech and Offensive Language Detection, mainly in English and other regional languages worldwide. Many researchers chose their native languages to detect hate speech. However, relatively little research has been conducted on detecting Roman Urdu Hate Speech. As native Urdu speakers, we undertook this study to detect Roman Urdu hate speech on social media.

According to the literature, comprehensive datasets on Roman Urdu Hate Speech are limited, and there is no standard Roman Urdu dictionary. To the best of our knowledge, a deep learning model (Bi-LSTM with attention mechanism) has not been explored for detecting Roman-Urdu hate speech. Consequently, this study investigates the application of (Bi-LSTM with attention mechanism) in detecting Roman Urdu hate speech.

III. METHODOLOGY

This section describes our proposed method in depth as depicted in Figure 2. The proposed architecture encompasses the following five phases.

1. Roman Urdu Hate Speech Scraping
2. Dataset Development
3. Preprocessing
4. Training
5. Testing

A. ROMAN URDU HATE SPEECH SCRAPING

1) SELECTION OF VARIOUS TOPICS

The data selection was made with consideration for various topics, including sports, religion, national causes, politics, current events, sectarian posts, gender bias, showbiz, etc. The following features of Hate Speech were included in the dataset.

National origins;- Pakistan, India, Afghanistan
Religion:- Muslims, Hindus, Sikhs, Christian, Jews
Race: Pashtoon, Afghan, Panjabi, Baloch, Sindhi
Sectarian:- Sunni, Shia, Bralvi, Deobandi, Wahabi
Gender/Sexism: Male, Female, Transgender

2) RAW DATA EXTRACTED FROM FACEBOOK AND TWITTER

Twitter and Facebook, two popular social media networks, were chosen as data sources for the aim of data extraction. Twitter was scraped for data (tweets) using Tweepy, a Python package for accessing the Twitter API. However, scraping messages from Facebook was difficult since it does not permit any internet scraping application to extract its messages mechanically. As illustrated in Figure 1, a computer operator was employed for this reason to extract Facebook messages into an MS. Excel file manually.

3) SELECTION OF ROMAN URDU COMMENTS

The gathered data were manually filtered to include only Roman Urdu comments/tweets.

B. DATASET DEVELOPMENT

1) DEVELOPMENT OF ANNOTATION GUIDELINES

Annotation Guidelines/rules are developed to get consistent annotations of the dataset. If comprehensive guidelines are in place, the expert can readily annotate the data with relevant labels, resulting in high-quality data annotation. The maximum inter-annotator agreement is used to measure annotation quality.

The initial stage in establishing annotation rules is explicitly defining the classes/labels, i.e., distinguishing between hate speech and neutral or normal speech. As part of this study, we established annotation guidelines for Neutral and Hate speeches, which are provided in Table 2 and Table 3. These rules allowed the annotators/experts to distinguish between neutral and hateful speech and to label the data appropriately.

2) MANUAL ANNOTATION OF TEXT USING ANNOTATION GUIDELINES

Initially, we manually labeled the text messages as ''Hate'' and ''Neutral'' in the excel file as created in Section-III(1)(ii). Annotation Guidelines as discussed in Section-II(2)(i) were consulted while performing annotation. In this step, we developed a base annotated dataset which was yet to be validated by a team of experts.

3) VALIDATION BY EXPERTS

The corpus was shared with a team of ten experts, comprising representatives from various religions, genders, nations, and fields, to annotate data as Hate Speech and Neutral Speech. The team was given the Annotation Guidelines designed for this study.

4) FINALIZATION OF ANNOTATION BASED ON INTER ANNOTATOR AGREEMENT

After receiving dataset annotations from 10 annotators/experts, the final annotation choice was made based

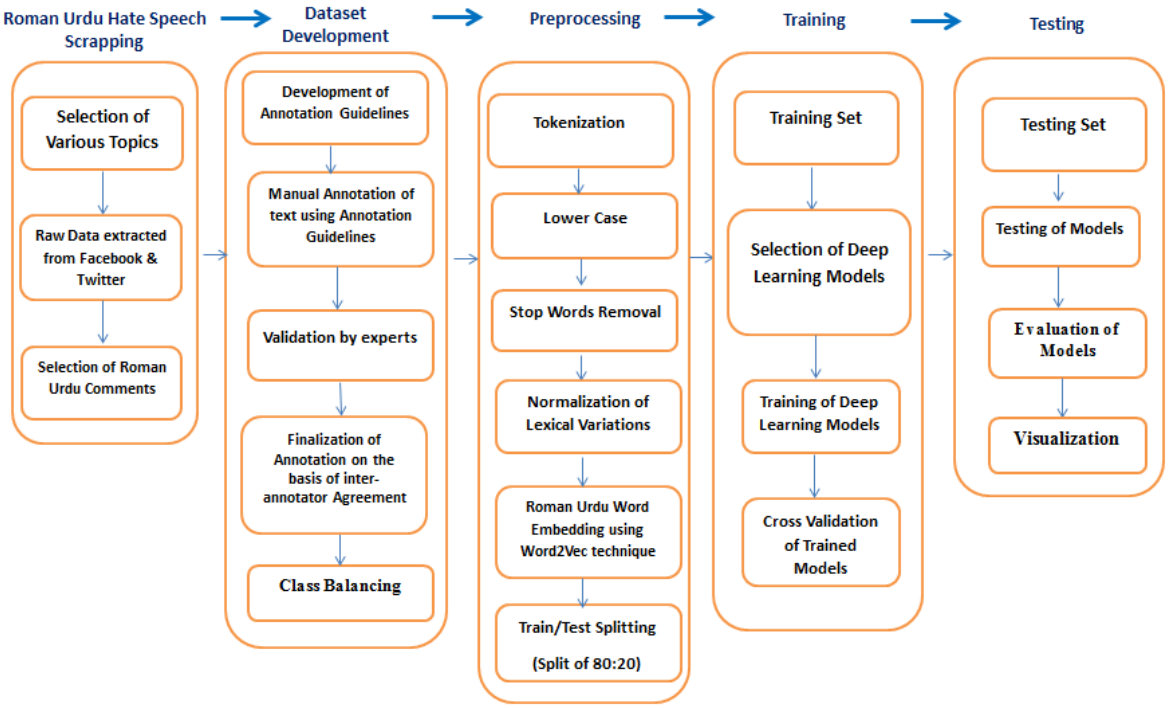

FIGURE 2. Proposed architecture for hate speech detection.

on the value of the highest inter-annotator agreement, as depicted in Figure 1. There are two measurements of inter-annotator agreement: Cen's kappa and Fleiss Kappa. Cohen Kappa is used to assess the degree of agreement between a pair of annotators, whereas Fliess' Kappa calculates the degree of agreement across a group of several annotators. In this study, there are ten annotators; consequently, Fliess' Kappa was utilized to calculate the Inter-Annotator agreement.

5) CLASS BALANCING

After data annotation was completed, the dataset was balanced by removing superfluous text messages to maintain an equal amount of both classes, i.e., Hate:15,000 and Neutral:15,000.

C. PREPROCESSING

Before feeding the dataset to the model, it was pre-processed and translated into an appropriate format for the model's optimal performance. The following steps were utilized for the dataset's preliminary processing.

1) TOKENIZATION

Each sentence of the text messages/tweets was tokenized using delimiters such as whitespace, tabs, and punctuation such as semicolon (;), colon (:), dot, the comma (,), etc.

2) LOWERCASE

To standardize the spelling, the entire text was transformed to lowercase letters to minimize case-sensitivity concerns with prediction.

3) STOP WORDS REMOVAL

Stop words are meaningless words in the corpus that add nothing to the efficiency of the model. To reduce the dataset's dimension, these terms are deleted. Typical examples of such terms include prepositions, articles, conjunctions, and so forth. In this research, we used Roman Urdu stop words available on the GitHub link: https://github.com/haseebelahi/roman-urdu-stopwords to filter them out from the text.

4) NORMALIZATION OF LEXICAL VARIATIONS

Roman Urdu is not an official language, but it is the name given to Urdu written in the Latin script, commonly known as the Roman script. There is no standard way to spell the words. People used several spellings of a word based on how it is pronounced. For example the Urdu Word [دہشتگردی] [Terrorism] is written as"deshatgardi","dehshatgardi","dahshatgardee", "dehshat gardi", "deshtgardee". Similarly (بے تر یغ) [Shamless] is written as "baghairat","bayghiarat", "by ghairat", "baygherat", "beghairat" etc. This is called lexical variation. Lexical variants of Roman Urdu words enhance the number of distinct words in the corpus and influence the performance of Natural Language Processing models. By normalizing these lexical variants to a standard form, it is possible to reduce sparsity and improve speed.

It is difficult to handle multiple lexical variants of Roman Urdu terms. Several normalizing strategies, such as Soundex and UrduPhones, manage such lexical variances. The Soundex algorithm encodes words based on their phonetic sounds. Then, combining words with similar sounds is feasible based on their phonetic codes. UrduPhone is a

TABLE 2. Annotation guidelines for neutral speech.

Neutral Speech
1. A statement that does not contain any hostility or inflame anger. Example: "Ap bohat khoob driving karty hain"
2. A statement that leaves pleasant feelings for an unbiased reader. Example: "bahir kitna dilkash mosam hy"
3. A statement contains some information, knowledge, or facts about a particular subject Example: "corona se bachny k liy hamen vaccine lagana chiahiy", "Chaudhry Aslam ka taluq KPK ki tehsil Manserah se tha", "Benazir Bhutto woh wahid khatoon hain jinhoon nay jadeed tarekh mein bazariya intakhabaat kisi Islami Mulk ki sarbarh-e-mumlikat honay ka azaaz hasil kia".
4. A conversation in lighter mode. Example: "yaar tu bhee poori cheez hy, kisi ko bhee nahi chorta", "yaar tujh se baat karnay se acha hy banda deewaron se baten karay"
5. A religious quote or sayings of Prophets, Saints or Gurus. Example: "Aamal ka daromedar neeyaton par hy", "ilm hasel karna har musalaman mard or orat par farz hy", "dunia me aman qaem karo"
6. A message showing love for others. Example: "me apny watan se mohabat krta hoon or us k liy jaan bhee de sakta hoon" , " me tum se mohabbat karta hoon"
7. A message sharing some news Example: "Omicron virus tezi se phael raha hy "
8. A message in which someone is praised. Example: "Imran Khan ik mukhles or nehayat Emandar PM hy "
9. A message expressing sorrow without targeting anyone. Example: "jo bhee howa bohat bura howa,baray

TABLE 3. Annotation guidelines for hate speech.

Hate Speech
1. A statement or message on social media that directly attack someone based on affiliation with a particular group like religion, nationality, race, gender, politics, etc Example: "tum pathan ho tumhen kia maloom ke aqal kia hoti hay", "mochi ke bachy jaldi se mera kam karo", "Punjabi baray hara*i hoty hain" .
2. A statement or message on social media that shows a clear intention to inflame harm, threats, or encourage hatred, by targeting individuals by indirectly referring to a group based on its characteristics. Example: Jo b Mumtaz Qadri k khilaf b*onkay usey qatal ker do (Kill anyone who ba*ks against Mumtaz Qadri).
3. A statement or message on social media that may harm a disable person. Example: In mazooron ko foot pathon pe nahi chorna chahiy inko yaha se dhakky mar k bahir nikalna chahiy.
4. A statement or message on social media that provokes revenge. Example: "Sepah e sahaba ne namoos e sehaba k liy kafi shio ko jahanum wasel kia"
5. A statement or message on social media that portrays negativity or hatred about a community. Example: "Afghani nimak haram qom hay" , "musalman saray dehshat gard log hain"
6. A statement or message on social media that misquotes from religious scripture having hostile or disrespectful content. Example: "Islam ghair muslim logon ke sath imteyazi sulooq ka hukum deta hai
7. Sarcasm messages that may trigger sentiments of inferiority. Example: "ye mahajar tou jesay is mulk k khas log hain" "boot ko izzat do",
8. A statement or message on social media that is against glory of religion including Blasphemous, sacrilegious and sectarian contents Example: "kafir kafir shia kafir", " ye sunni yazeed k bhai hain"
9. A statement or message on social media that contains Abusive language. Example: "Malala aik ghatia or khabees larki hy"
10. A statement or message on social media that contains incitement to terrorism. Examples: "Shaher k kony kony me aag laga dou jo rasty me ay jala dalo" "jo musalman gae kee qurbani karay ga usko bhee zabah kia jae"
11. A statement or message on social media that contains defamation towards public servants, institutions, army and police officers and provoke civil war and disturbance in the country or region. Example: "ye jo dehshat gardi hay is ke peachy wardi hy","police ka hy kaam ghunda gardi sar e aam","sarkari mulazim reshwat khor hotyn hain"
12. A statement or message in which someone is cursing a politician or any other individual Example: Lanat kho tum per jangir tareen" " Allah tala khan sab ko tabah o barbad karay"

phonetic encoding system devised specifically for the Roman Urdu script [29]. It is derived from Soundex but differs from it in two respects. First, Soundex is based on four (4) character codes, while UrduPhone is based on six characters code with more information to retain. Second, UrduPhones employs groups based on homophones, which are mapped differently in Soundex. UrduPhone encodes Roman Urdu text in accordance with its pronunciation. It tackles character-level differences that are expected to occur when Roman Script is used to write Urdu words.

In addition, lexical normalization is required to accommodate lexical variants in Roman Urdu terms and translate spelling variations into a single lexical form. There is no standard form of Roman Urdu words to which variants can be mapped. There is currently no standard lexicon or spelling for Roman Urdu terms. In light of this, we compiled a 4,000-word lexicon of Roman Urdu words. It facilitates the normalization of lexical variants using a supervised technique. The

TABLE 4. Mapping of Urdu alphabet with its corresponding alphabets in English.

English Alphabets | Associated letters in Urdu		
A → ا	B → ب	C → ک
D → د ڈ	E →,ا ی	F → ف
G → گ ,ج	H → ہ, ح	I → ا , ای
J → ج	K → ک	L→ ل
M → م	N → ن	O→ او
P → پ	Q → ق	R→ ڑ ,ر
S → س, ص, ث, ذ, ز	T → ت, ٹ	U →یو, او
V → و	W→ و	X → ز ,ذ, ایکس
Y→ ے, ی	Z → ز ,ذ, ض, ظ	

TABLE 5. Mapping of Urdu alphabet having multiple alphabets in English with its standard alphabets in English.

Urdu Alphabet	Corresponding Roman Urdu Alphabet	English Alphabet to be ignored	Example
'ک'	'K'	'C'	"Kameena", "Kirdaar"
'و'	'W'	'V'	''Wapis","Watan"
'ذ, ز, ظ, ض'	'Z'	'X','S'	"Plz", "Zulqarnain"

primary objective of creating a dictionary is to standardize Roman Urdu words, not to create a Roman Urdu-to-English dictionary. Roman Urdu is not a legitimate language, but rather a script in which the majority writes the Urdu language of Urdu speakers on social media utilizing electronic devices, such as computers, mobile phones, and tablets.

To standardize the Roman Urdu words, the following rules listed in Tables 4-7 were established.

RULE 1: → **MAPPING OF URDU ALPHABET WITH ITS CORRESPONDING ALPHABETS IN ENGLISH**

RULE 2: → ***IN ABOVE MAPPING, SOME CHARACTERS OF ENGLISH ALPHABETS SHALL BE IGNORED***

If there are numerous alphabets in the English language that make the sound of a single alphabet in Roman Urdu, such alphabets will be disregarded, and only the most frequently used alphabet by the users shall be employed. The intention is to eliminate spelling variances.

For example: **"ک."** can be written by 'K' and 'C'. Similarly: **"و"** can be written by ''W'' and ''V''.

RULE 3: → ***THE SPELLING OF ENGLISH WORDS, WHICH ARE MOST COMMONLY USED IN URDU LANGUAGE, SHALL BE THE SAME AS IN ENGLISH LANGUAGE.***

TABLE 6. Mapping of English words used commonly in Urdu.

English Words used commonly in Urdu	Its Roman equivalent Word
Aunty	Aunty
Cake	Cake
Camp	Camp
Cancel	Cancel
Computer	Computer
Control	Control
Geography	Geography

TABLE 7. Mapping of Urdu alphabet having no direct corresponding alphabets in English.

Urdu Alphabets	Corresponding English alphabet combination	Example Roman Urdu(Urdu) [English]
چ	Ch	Chaman (چمن) [Garden]
چھ	Chh	Chhoona(چھونا)[Touch]
خ	Kh	Khaawand (خاوند)[Husband], Khyber (خیبر)
ژ	Zh	Zhob (ژوب), Zhalabaari (ژالہ باری)[
ش	Sh	Shaadi (شادی)[marriage], Sher (شیر)[Lion]
غ	Gh	Ghamzada (غمزدہ)[Sad]
بھ	Bh	Bhabi (بھابی)[Sister in law], Bhaai(بھائی)[Brother]
پھ	Ph	Phool (پھول)[Flower]
جھ	Jh	Jhoola (جھولا) [swing]
تھ	Th	Thakaawat (تھکاوٹ) [Fatigue]
ٹھ	Th	Thokar (ٹھوکر)[Stumble]
ڈھ	Dh	Dhol (ڈھول) [drum], dheela (ڈھیلا)[loose]
آ	Aa	Aam(آم) (Mango]

However, Rule No. 2 shall not be applied to English words that are commonly used in Urdu. In such instances, English terms must be spelled exactly as they are in the English language.

RULE 4: → ***ALPHABETS IN URDU LANGUAGE WHICH HAVE NO DIRECT CORRESPONDING ALPHABETS IN ENGLISH THE LANGUAGE SHALL BE MAPPED TO COMBINATION OF ENGLISH ALPHABETS TO PRODUCE THE SAME SOUND.***

RULE 5: → **IF 'ای' [ALIF ZER YE] COMES AT THE START OF THE WORD, IT SHALL BE REPRESENTED BY 'EE'.**

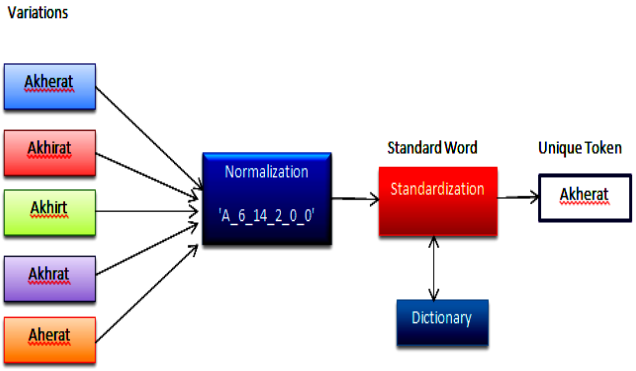

FIGURE 3. Normalization of lexical variation in roman urdu words.

EXAMPLE: 'EEMAAN'[BELIEVE], 'EENT'[BRICK], EENDHAN[FUEL] ETC.

RULE 6: → **IF 'ی' [ALIF ZER YE] COMES IN THE MIDDLE OF THE WORD, IT SHALL BE REPRESENTED BY 'EE'.**

EXAMPLE: 'ZAMEEN'[EARTH], 'KAMEENA'[MEAN], KEERA[INSECT] ETC.

RULE 7: → **IF 'ی' [ALIF ZER YE] COMES AT THE END OF THE WORD, IT SHALL BE REPRESENTED BY 'I'.**

EXAMPLE: 'DUSHMANI'[ANIMOSITY], 'ISLAMI'[ISLAMIC], PURAANI[OLD] ETC.

RULE 8: → **IF 'ی' [ALIF ZER YE] COMES AT THE END OF THE WORD, IT SHALL BE REPRESENTED BY 'I'.**

EXAMPLE: 'DUSHMANI'[ANIMOSITY], 'ISLAMI'[ISLAMIC], PURAANI[OLD] ETC.

RULE 8: → **IF 'ے' [ALIF ZABAR YE] COMES AT THE START OF THE WORD, IT SHALL BE REPRESENTED BY 'AI'.**

EXAMPLE: 'AIZAZ'[AIZAZ], 'AIB'[FAULT], AISH[LUXURY] ETCl.

RULE 9: → **IF 'ے' [ALIF ZABAR YE] COMES AT THE MIDDLE OF THE WORD, IT SHALL BE REPRESENTED BY 'AY'.**

EXAMPLE: 'BAYWAFA'[DISLOYAL], 'BAYSHARAM'[SHAMELESS], BAYWAQOOF [IDIOT] ETC.

RULE 10: → **IF 'ے' [ALIF ZABAR YE] COMES AT THE END OF THE WORD, IT SHALL BE REPRESENTED BY 'AY'.**

EXAMPLE: 'FAASLAY'[DISTANCES], 'HAMAARY'[OURS], RESHTAY [RELATIONSHIPS] ETC.

RULE 11: → **IF [ALIF ZABAR] COMES EVERYWHERE IN THE WORD, IT SHALL BE REPRESENTED BY 'AA'.**

```
'chakmaa': 'chakma,chkma',
'chillana': 'chalaien,chalain,chalan,chalana,chalanay,chalane,chalaney,chalayen,chalen,challan,chaln,chalna,c
'chilkaa': 'chalak,chalaki,chlak',
'chalees': 'chalees,chayalees',
'chalaang': 'challenge,challnge,chlngay',
'chamak': 'chamak,chamka',
'chamakdaar': 'chamakdar',
'choomna': 'chaman,choomen,chumny',
'chamcha': 'chamcha,chamche,chamchey,chamchi,chamcho,chmache,chmcha',
'chamra': 'chammar,chamra,chamri',
'chonch': 'chanch,chunacha,chunachay,chunache',
'chupana': 'chapion,chapne,chupanay,chupane,chupaney,chupney',
'chaplosi': 'chaploos,chaplosi',
'charbi': 'charbi,churbi',
'charcha': 'charcha,charche,church',
'charhhai': 'charhaho',
'charhna': 'charhana,charhanay,charhna',
'churail': 'charlie,churail',
'chawal': 'chawal,chawl,chowal,chwal,chwl',
'chooza': 'cheeez,cheez,cheezay,cheeze,cheezo,cheezz,cheiz,chez,cheza,cheze,chezo,chezy,chezz,chezzz,chiz,chi
'cheekhna': 'cheekhain,cheekhen,cheekhnay,chekahn,chekhen,chekhin',
'chotha': 'cheetah,chooteyah,chotha,chothey,chothy,chouthay,chouthe,chutiyh',
'chust': 'chest,chishti,chishtti,choostay,choosty,chushti,chust,chusti',
'chicken': 'chicken',
'chingaadar': 'chingadar',
'charaagh': 'chiragh',
'chamakna': 'chmkni',
'chodna': 'chodain,chodan,choden,chodna,chodnay,chodne,chodney,chodny,chodon,choudoon,chudanay,chudane,chudar
'choolha': 'choolhey',
'chipta': 'chpte,chupate,chupatey,chupattey,chupaty,chupta',
'christmas': 'christmas',
'chipkali': 'chupkali',
```

FIGURE 4. Clusters of lexical variation in roman urdu words.

EXAMPLE: 'FAASLAY'[DISTANCES], 'HAMAARY'[OURS], JUDAA [SEPARATE] ETC.

RULE 12: → **IF [ALIF ZER] COMES AT THE START OF A WORD, IT SHALL BE REPRESENTED BY 'I'.**

EXAMPLE: 'ILAAN'[ANNOUNCEMENT], 'ILAAQA'[AREA], IKHTELAAF[CONFLICT] ETC.

RULE 13: → **IF [ALIF ZER] COMES IN THE MIDDLE OR END OF A WORD, IT SHALL BE REPRESENTED BY 'E'.**

EXAMPLE: 'NEQAAB'[MASK], 'NEZAAKAT'[SENSITIVITY], NESHAAN [SIGN] ETC.

RULE 14: → **IF [ALIF PESH] COMES EVERYWHERE IN THE WORD, IT SHALL BE REPRESENTED BY 'U'.**

EXAMPLE: 'UTHAANA'[LIFT], 'UROOJ'[TOP], URAAN [FLIGHT], BUKHAAR[FEVER] ETC

RULE 15: → **IF [ALIF PESH WAO] COMES EVERYWHERE IN THE WORD, IT SHALL BE REPRESENTED BY 'OO'**

EXAMPLE: 'OOPAR[ABOVE], 'TOOFAAN'[STORM], FAALTOO [EXTRA] ETC.

RULE 16: → **IF [ZABAR] COMES ON EVERY ALPHABET, IT SHALL BE REPRESENTED BY 'A'**

EXAMPLE: 'OOPAR[ABOVE], 'ANDAR'[INSIDE], AQAL [WISDOM] ETC.

RULE 18: → **IF THERE IS ّ '[TASHDEED] COMES ABOVE AN ALPHABET, THE SAME ALPHABET SHALL BE WRITTEN TWO TIMES.**

EXAMPLE: 'MUKKA[PUNCH], 'TASHADDAD'[VIOLENCE], MUKAMMAL [COMPLETE] ETC. 'A_6_14_2_0_0': AAKHIRAT,AAKHRAT, AKHERAT, AKHIRAT,AKHIRT,AKHRAT

The algorithm is described as follows:

After establishing the Roman-Urdu Dictionary, numerous lexical variations of words are normalized. The procedure compares the phonetic codes of clusters with the phonetic codes of the corresponding standard terms in the dictionary and replaces lexical variations with their standard forms. The same module was applied to the entire corpus to replace variants of a term with its standard form, as depicted in Figures 3-4.

5) ROMAN URDU WORD EMBEDDING USING WORD2VEC TECHNIQUE

After pre-processing and normalization, extracting features is the next crucial step in analyzing raw data. The computer does not directly manipulate the raw data; instead, it converts the data into derived numerical values while maintaining the information included in the original data. There are various ways for feature extraction, including Bag of Words, TF/IDF (Term Frequency/Invert Document Frequency), with n-grams, and character gram.

The features were extracted by word2vec embedding. In this study, we employed custom Word2Vec embedding by passing our corpus to the Word2Vec function with a minimum count of 2 and a dimension size of 200. The Word2Vec embedding technique turns each corpus word based on its meaning and context into a 200-dimensional real-valued vector. In this manner, semantically related words are grouped in the vector space.

6) TRAIN / TEST SPLIT

For the experiment, the dataset is divided into two sets, i.e., Training Set and Testing Set.

In this research, we split our dataset into Training Set and Testing Set with a ratio of 80:20, respectively, by using the ''train_test_split'' function of the sklearn library in python.

D. TRAINING

1) TRAINING SET

It is the data set utilized for learning (by the model), i.e., to fit the parameters to the machine learning model. It is the data set used to train the model, and make it uncovers the hidden features/patterns in the data.

In each epoch, the neural network architecture is given the same training data multiple times, and the model continues to learn features from the data. The training set should contain a diverse group of inputs so that the model is trained in all scenarios and can predict any future unobserved data sample. We used 80 percent of the corpus for training data.

2) SELECTION OF DEEP LEARNING MODELS

In this research, the following Deep Learning models were applied to detect Romah Urdu hate speech.

- LSTM
- Bi-LSTM
- Bi-LSTM + Attention
- CNN

3) TRAINING OF DEEP LEARNING MODELS

At first, the training dataset was preprocessed, normalized, and embeddings were obtained with the custom word2vec embedding technique. Then, the aforementioned deep learning models were trained with embeddings (feature vectors) obtained from the training data as per the setting described in Section IV.

4) CROSS-VALIDATION OF TRAINED MODELS

Each model was trained and constructed at this stage using a validation split of 0.2. Here, we employed the simplest form of cross-validation, held-out cross-validation, which significantly decreases bias since the majority of the data is used for fitting. As most of the data are also utilized in the validation set, variance is also drastically reduced. The results of training during validation were recorded in ''history'' for purposes of visualization. The history object is used to maintain a record of metrics and loss values during the training procedure.

E. TESTING

1) TESTING SET

Section III (3)(vi) discussed that the dataset was divided into two parts. One part was used for training the models, and the other was used for testing the models. The testing set was also passed through preprocessing, normalization, and embedding steps.

2) TESTING OF MODELS

In this step, a testing dataset is supplied to each trained model for evaluation of models, and results in each case are recorded.

3) EVALUATION OF MODELS

In this step, the results of all deep learning models were evaluated by performing statistical analysis of test results. The following metrics were used for the evaluation of the models:

- Accuracy
- Precision
- Recall
- F-Score

4) VISUALIZATION

The performance of each model is interpreted using graphical representations. Python provides Matplotlib and Seaborn libraries for visualization purposes. We utilized the

Matplotlib package of Python to visualize the results of different models considered in this study.

IV. EXPERIMENTAL SETTINGS

We conducted our experiments using the Colab environment hosted by Google, which provides resources of Python 3 Google Compute Engine Backend (GPU) with 16 GB of RAM and 120 GB of storage. The DL algorithms were implemented in Keras-backed Tensor Flow in Python. We used various libraries of Python, including Keras, Pandas, NumPy, NLTK, JSON, Gensim, and Sklearn. We also used the ''UrduPhone'' library introduced by [43].

For Roman Urdu Hate speech detection, we compared our proposed context-aware deep learning model based on Bi-LSTM+attention to baseline traditional machine learning and DL models.

Additionally, we performed two experiments with each model. One investigation was conducted utilizing our provided dataset without normalization, while the other was conducted on a normalized dataset (the dataset is normalized by removing lexical variations in Roman Urdu words).

Using the procedure illustrated in Figure 1, a dataset including 30,000 text messages was produced. Out of these 30,000 text messages, 15,000 were marked as ''Hate,'' while the remaining 15,000 were marked as ''Neutral.'' The dataset was processed, tokenized, and normalized beforehand. Word2vec embedding was used to extract the features. The word2vec embedding method generates vectors for each word based on its n-dimensional context. The semantically related terms were distributed close to one another in the vector space. The tokenized data was passed to the Word2Vec function with a minimum count of 2 and dimension size of 200, which results in a vector of 200 dimensions being formed for each word.

The summary of experimental settings and Hyperparameters used in each experiment are provided in Table- 8.

A. TRADITIONAL MODELS

Initially, seven (07) traditional machine learning techniques were used as baselines to build hate speech detection models. For conventional machine learning, we defined a function named MeanEmbeddingVectorizer, which calculates the mean embedding of a word so that it could be used in traditional models. The dataset is split into the training set and testing set with a ratio of 80:20, respectively. The same experiment was performed on a normalized dataset, and the results in each case were recorded in Table 9.

B. DEEP LEARNING MODELS

1) LSTM MODEL

In LSTM, an initially embedding layer was added, followed by an LSTM layer of size 64 with an activation function as 'RELU' followed by a Dropout layer with a 0.2 value.

Thereafter a Dense layer of size 32 with activation function as 'relu' is placed. This was connected with a Drop of size 0.2, followed by another Dense Layer whose activation function was a sigmoid. Adam was used as the optimizer and Binary Cross Entropy as the loss function to compile the model. The model was compiled and fit our data.

TABLE 8. Traditional models with and without normalization.

Parameters	LSTM	Bi-LSTM	Bi-LSTM + Attention	CNN
Dataset Size	30,000	30,000	30,000	30,000
Train Test Split Ratio	80:20	80:20	80:20	80:20
Embedding Technique	Word2Vec	Word2Vec	Word2Vec	Word2Vec
Max Sentence Length	100	100	100	100
Embedding dimensions	200	200	200	200
Embedding Layer size	200	200	200	200
DL Model Layer Size	64	02 layers (32,16)	02 layers (32,16)	03 Conv layers with Size:
Filter Size	-	-	-	Filter size: 3,5,3
Activation Function	ReLu	ReLu	ReLu	ReLu
Dropout Layer	0.2	0.3	0.3	02 MaxPooling
Dense Layer size	32	-	Attention Layer	32
Activation Function	ReLu	-	-	ReLu
Padding	-	-	-	Same
Dropout Layer	0.2	-	-	-
Dense Layer size	1	1	1	1
Activation Function	Sigmoid	Sigmoid	Sigmoid	Sigmoid
Optimizer	Adam	Adam	Adam	Adam
Loss Function	Binary Cross Entropy	Binary Cross Entropy	Binary Cross Entropy	Binary Cross Entropy
Validation Split	0.2	0.2	0.2	0.2
Epochs	06	06	06	06

```
Epoch 1/6
81/81 [==============================] - 31s 338ms/step - loss: 4.8005 - acc: 0.9036 - val_loss: 0.3595 - val_acc: 0.8446
Epoch 2/6
81/81 [==============================] - 27s 333ms/step - loss: 1.3654 - acc: 0.9216 - val_loss: 1.0646 - val_acc: 0.6956
Epoch 3/6
81/81 [==============================] - 27s 333ms/step - loss: 0.2610 - acc: 0.9132 - val_loss: 0.3299 - val_acc: 0.8561
Epoch 4/6
81/81 [==============================] - 27s 333ms/step - loss: 0.1759 - acc: 0.9445 - val_loss: 0.3282 - val_acc: 0.8606
Epoch 5/6
81/81 [==============================] - 27s 339ms/step - loss: 0.1464 - acc: 0.9529 - val_loss: 0.3330 - val_acc: 0.8609
Epoch 6/6
81/81 [==============================] - 27s 328ms/step - loss: 0.1260 - acc: 0.9609 - val_loss: 0.3449 - val_acc: 0.8592
```

FIGURE 5. LSTM results in each epoch (before normalization of dataset).

TABLE 9. Summary of hyperparameter setting for DL models.

Model	Dataset	Accuracy	F Score	Precision	Recall
Logistic Regression	Simple	60%	75%	60%	100%
	Normalized	66%	74%	67%	84%
SVM	Simple	60%	75%	60%	100%
	Normalized	70%	76%	71%	82%
Linear SVM	Simple	60%	75%	60%	100%
	Normalized	68%	75%	68%	84%
XG Boost	Simple	63%	74%	65%	88%
	Normalized	71%	75%	74%	77%
Random Forest	Simple	64%	73%	67%	80%
	Normalized	72%	77%	74%	80%
Decision Tree	Simple	57%	63%	64%	62%
	Normalized	63%	67%	69%	67%
K-Nearest Neighbors (KNN)	Simple	58%	65%	65%	65%
	Normalized	68%	71%	73%	71%

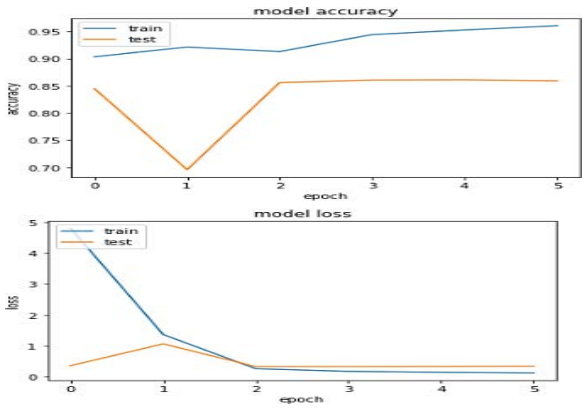

FIGURE 6. LSTM performance (before normalization of the dataset).

The same experiment was performed on a preprocessed non-normalized dataset (without lexical normalization) and a normalized dataset (with lexical normalization), and the results in each case are visualized by using the python library namely "matplotlib.pyplot" which is depicted in Figures 5-8:-

The trend of accuracy shown in Figure 6 demonstrates that when the dataset is not normalized, the training accuracy of the LSTM model grows with each training epoch, whereas the validation accuracy fluctuates. In subsequent epochs, the model appears to be overfitted. The model loss graph demonstrates that the training loss lowers with each epoch; however, the validation loss initially climbs, then declines, and then grows continually.

```
Epoch 1/6
81/81 [==============================] - 29s 329ms/step - loss: 0.7147 - acc: 0.6413 - val_loss: 0.6350 - val_acc: 0.6206
Epoch 2/6
81/81 [==============================] - 27s 329ms/step - loss: 0.5981 - acc: 0.6831 - val_loss: 0.5852 - val_acc: 0.7771
Epoch 3/6
81/81 [==============================] - 26s 323ms/step - loss: 0.5275 - acc: 0.7722 - val_loss: 0.4478 - val_acc: 0.8203
Epoch 4/6
81/81 [==============================] - 26s 323ms/step - loss: 0.4130 - acc: 0.8250 - val_loss: 0.3866 - val_acc: 0.8314
Epoch 5/6
81/81 [==============================] - 26s 325ms/step - loss: 0.3348 - acc: 0.8644 - val_loss: 0.3440 - val_acc: 0.8543
Epoch 6/6
81/81 [==============================] - 26s 325ms/step - loss: 0.2914 - acc: 0.8879 - val_loss: 0.3342 - val_acc: 0.8611
161/161 [==============================] - 4s 22ms/step - loss: 0.3342 - acc: 0.8611
```

FIGURE 7. LSTM results in each epoch (after normalization of dataset).

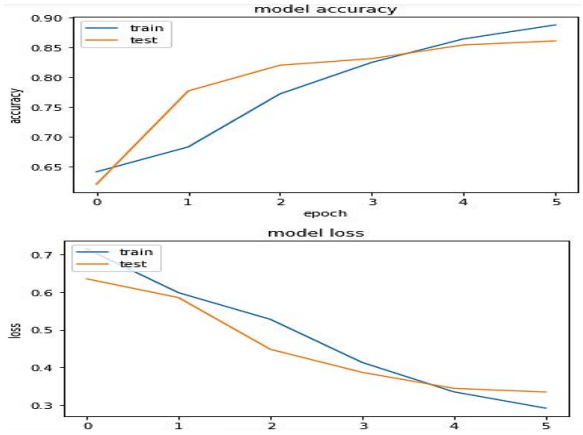

FIGURE 8. LSTM performance (after normalization of dataset).

The trend depicted in Figure 8 indicates that the training accuracy of the LSTM model, after normalization of the training dataset, increases with each training epoch, from around 0.65 to 0.89.

Likewise, the validation accuracy of the LSTM model ranges between 0.62 and 0.86. Similarly, the model loss diagram demonstrates that both the training loss and validation loss decrease steadily with each epoch. These results indicate that normalizing the dataset increased the generalization of the LSTM model.

2) BI-LSTM MODEL

In Bi-LSTM, an embedding layer was constructed. Afterward, a stack of two Bidirectional LSTM layers of 32 and 16 units was added, followed by a dropout of 0.3 after each BiLSTM layer. The final output was generated using a dense layer with a sigmoid function. Adam as the optimizer and Binary Cross Entropy as the loss function was used to train the model. The identical experiment was conducted on preprocessed non-normalized and normalized datasets, and the results are depicted in Figures 9 through 12.

The trend displayed in Figure 10 illustrates that the Bi-LSTM model's training accuracy increases with each training session even when the dataset is not normalized. However, training accuracy and validation accuracy are not consistent. Similarly, training loss diminishes after each epoch, whereas validation loss initially decreases and then increases continuously after a few epochs. These results indicate that Bi-LSTM overfits when applied to non-normalized data sets.

```
Epoch 1/6
81/81 [==============================] - 238s 3s/step - loss: 0.6012 - accuracy: 0.6750 - val_loss: 0.4573 - val_accuracy: 0.7886
Epoch 2/6
81/81 [==============================] - 223s 3s/step - loss: 0.4063 - accuracy: 0.8266 - val_loss: 0.3425 - val_accuracy: 0.8567
Epoch 3/6
81/81 [==============================] - 220s 3s/step - loss: 0.2559 - accuracy: 0.9070 - val_loss: 0.3215 - val_accuracy: 0.8736
Epoch 4/6
81/81 [==============================] - 221s 3s/step - loss: 0.1490 - accuracy: 0.9511 - val_loss: 0.3751 - val_accuracy: 0.8738
Epoch 5/6
81/81 [==============================] - 220s 3s/step - loss: 0.0976 - accuracy: 0.9708 - val_loss: 0.4158 - val_accuracy: 0.8707
Epoch 6/6
81/81 [==============================] - 223s 3s/step - loss: 0.0699 - accuracy: 0.9805 - val_loss: 0.4537 - val_accuracy: 0.8654
161/161 [==============================] - 21s 130ms/step - loss: 0.4537 - accuracy: 0.8654
```

FIGURE 9. Bi-LSTM results in each epoch (before normalization of dataset).

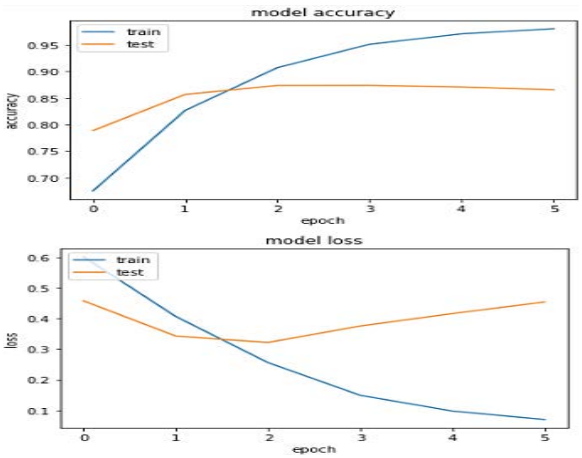

FIGURE 10. Bi-LSTM performance (before normalization of dataset).

```
Epoch 1/6
81/81 [==============================] - 144s 2s/step - loss: 0.5903 - accuracy: 0.6831 - val_loss: 0.4728 - val_accuracy: 0.7839
Epoch 2/6
81/81 [==============================] - 134s 2s/step - loss: 0.3939 - accuracy: 0.8267 - val_loss: 0.3931 - val_accuracy: 0.8289
Epoch 3/6
81/81 [==============================] - 134s 2s/step - loss: 0.3004 - accuracy: 0.8742 - val_loss: 0.3374 - val_accuracy: 0.8586
Epoch 4/6
81/81 [==============================] - 134s 2s/step - loss: 0.2284 - accuracy: 0.9102 - val_loss: 0.3620 - val_accuracy: 0.8576
Epoch 5/6
81/81 [==============================] - 134s 2s/step - loss: 0.1768 - accuracy: 0.9325 - val_loss: 0.3453 - val_accuracy: 0.8703
Epoch 6/6
```

FIGURE 11. Bi-LSTM results in each epoch (after normalization of dataset).

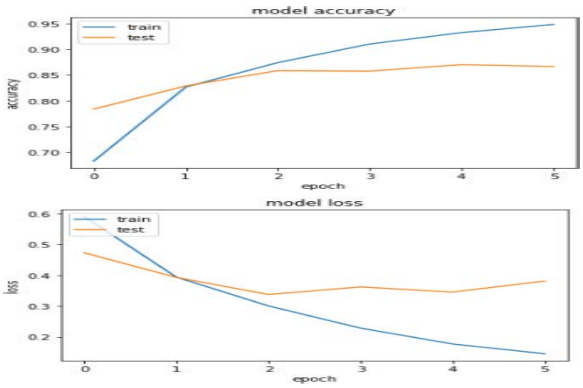

FIGURE 12. Bi-LSTM performance (after normalization of dataset).

Figure 12 demonstrates that, following the normalization of the dataset, both the training and validation accuracy of the Bi-LSTM model rise after each epoch, though at varying rates. This resulted in a slight improvement; nevertheless, after a few epochs, validation loss steadily increased. Consequently, the model does not reflect a good fit.

```
Epoch 1/6
81/81 [==============================] - 20s 134ms/step - loss: 0.6600 - accuracy: 0.5971 - val_loss: 0.6145 - val_accuracy: 0.6639
Epoch 2/6
81/81 [==============================] - 9s 109ms/step - loss: 0.5360 - accuracy: 0.7451 - val_loss: 0.4559 - val_accuracy: 0.7952
Epoch 3/6
81/81 [==============================] - 9s 109ms/step - loss: 0.4482 - accuracy: 0.7919 - val_loss: 0.3866 - val_accuracy: 0.8499
Epoch 4/6
81/81 [==============================] - 9s 109ms/step - loss: 0.3162 - accuracy: 0.8828 - val_loss: 0.3675 - val_accuracy: 0.8442
Epoch 5/6
81/81 [==============================] - 9s 110ms/step - loss: 0.2774 - accuracy: 0.9058 - val_loss: 0.3645 - val_accuracy: 0.8650
Epoch 6/6
81/81 [==============================] - 9s 108ms/step - loss: 0.1708 - accuracy: 0.9470 - val_loss: 0.3715 - val_accuracy: 0.8685
161/161 [==============================] - 6s 39ms/step - loss: 0.3715 - accuracy: 0.8685
```

FIGURE 13. Results of Bi-LSTM + attention layer in each epoch (before normalization of dataset).

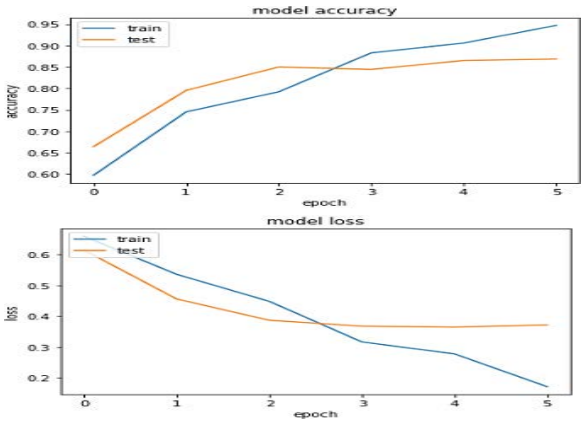

FIGURE 14. Performance of Bi-LSTM + attention layer (before normalization of dataset).

```
Epoch 1/6
81/81 [==============================] - 239s 3s/step - loss: 0.6231 - accuracy: 0.6472 - val_loss: 0.5107 - val_accuracy: 0.7649
Epoch 2/6
81/81 [==============================] - 220s 3s/step - loss: 0.5152 - accuracy: 0.7562 - val_loss: 0.5484 - val_accuracy: 0.7168
Epoch 3/6
81/81 [==============================] - 221s 3s/step - loss: 0.4132 - accuracy: 0.8313 - val_loss: 0.3896 - val_accuracy: 0.8413
Epoch 4/6
81/81 [==============================] - 220s 3s/step - loss: 0.3330 - accuracy: 0.8742 - val_loss: 0.4104 - val_accuracy: 0.8413
Epoch 5/6
81/81 [==============================] - 220s 3s/step - loss: 0.2625 - accuracy: 0.9045 - val_loss: 0.3474 - val_accuracy: 0.8654
Epoch 6/6
81/81 [==============================] - 228s 3s/step - loss: 0.2065 - accuracy: 0.9254 - val_loss: 0.3448 - val_accuracy: 0.8656
161/161 [==============================] - 6s 39ms/step loss: 0.3208 - accuracy: 0.8750
```

FIGURE 15. Results of Bi-LSTM + attention layer in each epoch (after normalization of dataset).

C. BI-LSTM + ATTENTION MODEL

In this configuration, an attention layer is added with Bi-LSTM. The rest of the procedure is the same as in Bi-LSTM. This experiment was also performed on a non-normalized and normalized dataset, and the results in each case are depicted in Figures 13 through 16.

Figure 14 demonstrates that the training and validation accuracy of the Bi-LSTM Model with the attention layer increases with each epoch until it achieves a maximum of 86 percent in the final epoch. Nevertheless, this accuracy is inferior to that of the same model trained on a normalized dataset.

Figures 15 and 16 show that the testing accuracy of Bi-LSTM + Attention Layer on a normalized dataset is 87.50 percent. Figure 16 demonstrates that, when applied to a normalized dataset, the Bi-LSTM model with the attention layer is more general than the previous deep learning models.

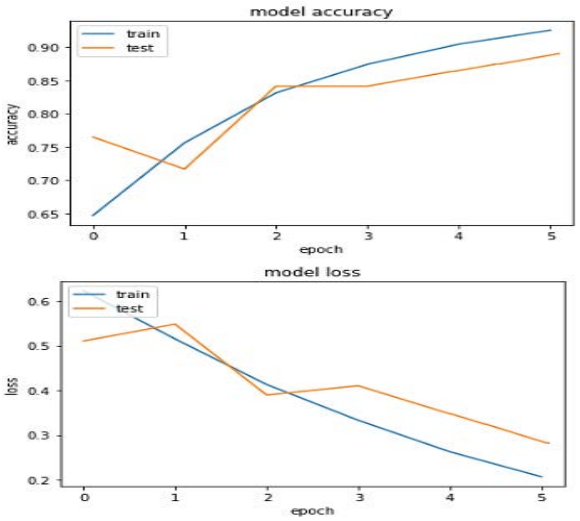

FIGURE 16. Performance of Bi-LSTM + attention layer (after normalization of dataset).

```
Epoch 1/6
643/643 [==============================] - 12s 17ms/step - loss: 0.3840 - accuracy: 0.8216 - val_loss: 0.2928 - val_accuracy: 0.8755
Epoch 2/6
643/643 [==============================] - 11s 17ms/step - loss: 0.1668 - accuracy: 0.9373 - val_loss: 0.3244 - val_accuracy: 0.8695
Epoch 3/6
643/643 [==============================] - 11s 17ms/step - loss: 0.0710 - accuracy: 0.9743 - val_loss: 0.4272 - val_accuracy: 0.8683
Epoch 4/6
643/643 [==============================] - 11s 17ms/step - loss: 0.0282 - accuracy: 0.9907 - val_loss: 0.6909 - val_accuracy: 0.8672
Epoch 5/6
643/643 [==============================] - 11s 17ms/step - loss: 0.0167 - accuracy: 0.9948 - val_loss: 0.7561 - val_accuracy: 0.8582
Epoch 6/6
643/643 [==============================] - 11s 17ms/step - loss: 0.0119 - accuracy: 0.9964 - val_loss: 0.8613 - val_accuracy: 0.8621
161/161 [==============================] - 1s 4ms/step - loss: 0.8613 - accuracy: 0.8621
```

FIGURE 17. Results of CNN in each epoch (before normalization of dataset).

D. CONVOLUTION NEURAL NETWORK (CNN) MODEL

The CNN architecture was developed by first utilizing an embedding layer with a size of 200, followed by a spatial dropout layer with a size of 0.3. Following this, a one-dimensional Convolutional Layer with size=64, filter size=3, and activation function=ReLU was added. After that, a Max Pooling layer was added. In the configuration, another Convolutional layer of size=32 with filter size=5, activation function as 'ReLU' and padding as 'same' was added, followed by a Max pooling layer connected to a third Convolutional layer of size=16 with filter size=3 and activation function as 'ReLU' and padding as 'same.' Then, a second Max pooling layer and a Flatten layer were added. Afterward, a Dense layer with the parameters size=32 and activation function='ReLU' was added. Another Dense layer with output=1 and sigmoid activation was added at the end.

This experiment was also performed on preprocessed normalized and non-normalized datasets and the results in each case are visualized as shown in Figures 17-20.

Figure 18 depicts the accuracy of a CNN model trained on a non-normalized dataset. The graph demonstrates that the training accuracy of the model steadily improves after each epoch, but the validation accuracy of the CNN model decreases from 0.87 to 0.862 when it was trained on a non-normalized dataset.

Similarly, when the CNN model is trained using a normalized dataset, its validation accuracy barely hits 0.866,

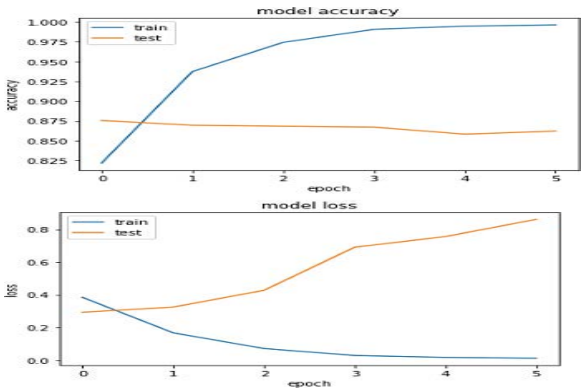

FIGURE 18. Performance of CNN (before normalization of dataset).

```
Epoch 1/3
643/643 [==============================] - 35s 54ms/step - loss: 0.4135 - accuracy: 0.8036 - val_loss: 0.3617 - val_accuracy: 0.8491
Epoch 2/3
643/643 [==============================] - 36s 56ms/step - loss: 0.2301 - accuracy: 0.9087 - val_loss: 0.3301 - val_accuracy: 0.8586
Epoch 3/3
643/643 [==============================] - 34s 53ms/step - loss: 0.1435 - accuracy: 0.9442 - val_loss: 0.3967 - val_accuracy: 0.8666
161/161 [==============================] - 1s 7ms/step - loss: 0.3967 - accuracy: 0.8666
```

FIGURE 19. Results of CNN in each epoch (after normalization of dataset).

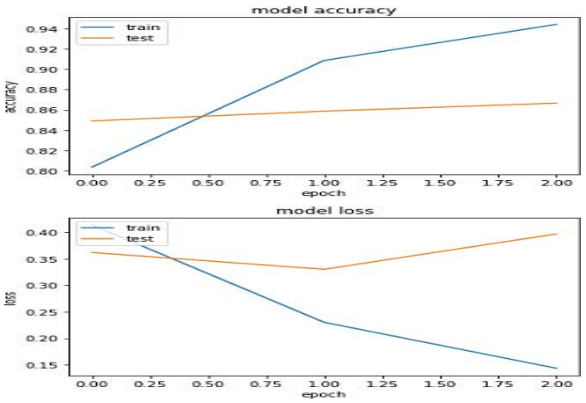

FIGURE 20. Performance of CNN (after normalization of dataset).

as shown in Figure 19. The CNN model's training loss lowers with each training epoch, beginning at 0.41 and reaching 0.14. The validation loss of CNN's model indicates that, at first, the loss decreased from 0.36 to 0.33 before jumping to 0.39. The training and validation patterns depicted in Figure 20 reveal that the CNN model on both normalized and unnormalized datasets suffers from severe overfitting.

V. RESULTS AND DISCUSSIONS

The testing dataset was used to evaluate both classic machine learning models and deep learning models. The performance of these trained models was tested using well-known metrics such as accuracy, precision, recall, and F-measure, and the results are presented in Table-X.

It is evident from the empirical results reported in Table 10 and illustrated in Figures 21 and 22 that Deep Learning models outperformed machine learning models. LSTM, Bi-LSTM, and CNN scored worse in accuracy and F-measure than Bi-LSTM with the attention layer. In addition, it was observed that Random Forest and XG Boost performed relatively better than other conventional machine

TABLE 10. Results of traditional and deep learning classification models using Word2Vec embedding on normalized dataset.

Classifier Model	Accuracy	F-Measure	Precision	Recall
Deep Learning Models				
LSTM	86.11%	87.18%	86.58%	88.40%
Bi-LSTM	87.03%	87.91%	89.54%	86.97%
Bi-LSTM + attention	**87.50%**	**88.48%**	**89.22%**	**88.36%**
CNN	86.66%	88.10%	89.82%	87.02%
Traditional Machine Learning Models				
Logistic Regression	66%	74%	67%	84%
SVM	68%	76%	67%	82%
Linear SVM	68%	75%	68%	84%
XG Boost	71%	75%	74%	77%
Random Forest	71%	77%	73%	80%
Decision Tree	62%	67%	69%	67%
K-Nearest Neighbors (KNN)	67%	71%	73%	71%

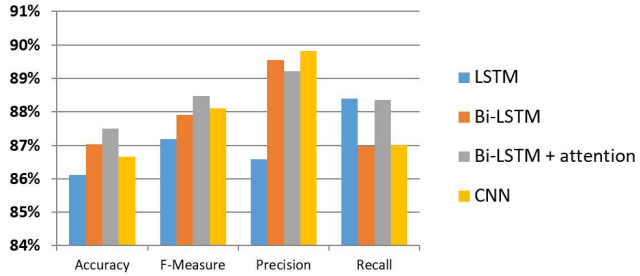

FIGURE 21. Comparison of deep learning models.

learning methods. Figure 16 demonstrates that Bi-LSTM with an attention layer not only performed well in terms of accuracy and F-score but is also more general than other models. Figure 8 demonstrates that after normalization, LSTM is also effective in generalization. As depicted in Figures 18 and 20, the CNN generalization was inadequate, resulting in substantial overfitting. LSTM and Bi-LSTM models were intended specifically for sequence and time series data, such as textual data used in Natural Language Processing (NLP), whereas CNN was developed for image processing.

Next, we reviewed the findings of the earlier work completed by Khan et al. [15]. We chose their outcomes according to two class categories, neutral and hostile. They implemented Naive Bayesian, Logical Regression, Random Forest, SVM, and CNN. These models were trained on 5000 Twitter messages. Their work utilized different feature extraction techniques, including Character-level features (CLF),

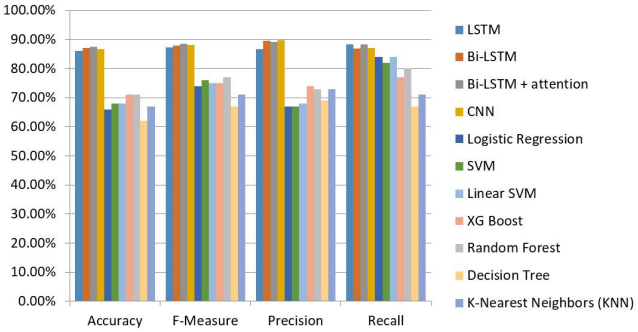

FIGURE 22. Comparison of deep learning vs. traditional models.

TABLE 11. Results of M.M.Khan et al 2021.

Classifier Model	Features	Accuracy	F-Score	Precision	Recall
Naïve Bayesian (NB)	CLF	80%	81.1%	75.1%	88.2%
	NGV	81.9%	83%	77.4%	87%
	WLTF	80%	82.1%	77.4%	87.4%
	CV	83%	88.3%	82.7%	94.7%
Logistic Regression	CLF	81%	84.1%	80.1%	88.6%
	NGV	80%	83.4%	81.1%	85.8%
	WLTF	81%	84.5%	79.7%	89.9%
	CV	84%	90.6%	84.4%	97.7%
Random Forest (RF)	CLF	81%	80.1%	72.5%	89.6%
	NGV	81%	88.6%	76.1%	88.6%
	WLTF	81%	82.2%	78.1%	86.7%
	CV	81%	81.7%	77.7%	86.2%
SVM	NGV	80%	80.9%	75.5%	87.1%
CNN	WE	80%	80.4%	72.7%	89.9%

CLF: Character-level features, NGV: N-Gram Vectorizer
WLTF: Word-Level Term Frequency, CV: Count Vectorizer
WE: Word Embedding

N-Gram Vectorizer (NGV), Word-Level Term Frequency (WLTF), Count Vectorizer (CV), and Word Embedding (WE).

VI. CONCLUSION AND FUTURE WORK

Detecting Roman-Urdu hate speech is a difficult challenge due to limited resources. Our proposed method demonstrates the viability of this novel HSD route. The following conclusions are derived from experimental findings.

Conclusively, the context-aware HSD model based on Bi-LSTM with attention layer outperformed machine learning and Deep learning models such as LSTM, Bi-LSTM, and CNN in terms of accuracy and F-measure. In addition, it was discovered that Random Forest and XG Boost performed relatively better than other conventional machine learning methods.

As demonstrated in Figure 14, Bi-LSTM with an attention layer not only performed well in terms of accuracy and F score, but the model is also more generalized than others.

According to Table-XI, previous research demonstrated that the Logistic Regression with Count Vectors model beats both traditional and deep learning models (i.e., CNN) in terms of accuracy, precision, recall, and f-measure. However, our experimental findings revealed that Deep Learning Models performed better than machine learning models, including Logistic Regression. The dataset's size might cause this boost in the performance of Deep Learning models. As a result, deep learning models perform favorably on massive datasets instead of small ones.

To analyze the effect of lexical normalization, Table 9 and Figures 5 to 20 reveal that the performance of all models, notably traditional models, improved dramatically after the dataset was normalized. However, normalization has little effect on deep learning models. Despite this, the objective of handling lexical variations through normalization and development of the Roman Urdu Dictionary was to improve performance and standardize the Roman Urdu language by providing standard spelling for each word of Roman Urdu, thereby enabling the models to operate quickly and effectively.

FUTURE DIRECTIONS

Future efforts must be conducted to improve the annotation quality of the Roman Urdu Hate Speech dataset for algorithms to provide the most accurate predictions regarding hate speech. Sentences containing sarcasm and implicit hate speech demand increased scrutiny. In addition, the magnitude of hateful content is dependent on the target. For instance, a slur spoken at a friend will not be viewed as offensive, but rather as amusing, yet the same phrase directed against an adversary may result in severe consequences. Similarly, if the target is religious holy figures, the sentence will have the highest level of hatred. In addition, if a person makes derogatory remarks about the race, nation, or gender to which he or she belongs, these remarks may be viewed as humorous and made in jest; otherwise, they would be regarded as poisonous. Future hate speech detection methods may incorporate these elements.

On the other hand, the impact of normalization could be improved by refining the phonetic encoding method, as we discovered during our experiments that UrduPhone has some limitations that cause unrelated words to cluster together, which degrades the performance, and the results of Deep Learning models were unimpressive. To address the issue of lexical variances in Roman Urdu words in the future, we will work on a Phonetic coding scheme.

REFERENCES

[1] A. Rafae, A. Qayyum, M. Moeenuddin, A. Karim, H. Sajjad, and F. Kamiran, "An unsupervised method for discovering lexical variations in Roman Urdu informal text," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, vol. 2015, pp. 823–828.

[2] Z. Sharf and S. U. Rahman, "Lexical normalization of Roman Urdu text," *Int. J. Comput. Sci. Netw. Secur.*, vol. 17, no. 12, pp. 213–221, 2017.

[3] B. Han and T. Baldwin, "Lexical normalisation of short text messages: Makn sens a# Twitter," in *Proc. 49th Annu. Meeting Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2011, pp. 368–378.

[4] B. Han, P. Cook, and T. Baldwin, "Lexical normalization for social media text," *ACM Trans. Intell. Syst. Technol.*, vol. 4, no. 1, pp. 1–27, 2013.

[5] D. Supranovich and V. Patsepnia, "IHS_RD: Lexical normalization for English tweets," in *Proc. Workshop Noisy User-Generated Text*, 2015, pp. 78–81.

[6] N. Kaji and M. Kitsuregawa, "Accurate word segmentation and POS tagging for Japanese microblogs: Corpus annotation and joint modeling with lexical normalization," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 99–109.

[7] A. Wang, M.-Y. Kan, D. Andrade, T. Onishi, and K. Ishikawa, "Chinese informal word normalization: An experimental study," in *Proc. 6th Int. Joint Conf. Natural Lang. Process.*, 2013, pp. 127–135.

[8] F. Alam, S. Habib, and M. Khan, "Text normalization system for Bangla," BRAC Univ., Dhaka, Bangladesh, Tech. Rep., 2008.

[9] S. Schulz, G. D. Pauw, O. D. Clercq, B. Desmet, V. Hoste, W. Daelemans, and L. Macken, "Multimodular text normalization of Dutch user-generated content," *ACM Trans. Intell. Syst. Technol.*, vol. 7, no. 4, pp. 1–22, Jul. 2016.

[10] T. Korenius, J. Laurikkala, K. Järvelin, and M. Juhola, "Stemming and lemmatization in the clustering of Finnish text documents," in *Proc. 13th ACM Conf. Inf. Knowl. Manage. (CIKM)*, 2004, pp. 625–633.

[11] P. Ruiz, M. Cuadros, and T. Etchegoyhen, "Lexical normalization of Spanish tweets with rule-based components and language models," in *Procesamiento del Lenguaje Natural*. 2014, p. 8.

[12] K. Darwish, W. Magdy, and A. Mourad, "Language processing for Arabic microblog retrieval," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2012, pp. 2427–2430.

[13] Ł. Brocki, K. Marasek, and D. Koržinek, "Multiple model text normalization for the Polish language," in *Proc. Int. Symp. Methodol. Intell. Syst.* Springer, 2012, pp. 143–148.

[14] H. Rizwan, M. H. Shakeel, and A. Karim, "Hate-speech and offensive language detection in Roman Urdu," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2020, pp. 2512–2522.

[15] M. M. Khan, K. Shahzad, and M. K. Malik, "Hate speech detection in Roman Urdu," *ACM Trans. Asian Low-Resource Lang. Inf. Process.*, vol. 20, no. 1, pp. 1–19, Apr. 2021.

[16] N. D. Gitari, Z. Zuping, 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.

[17] C. Hutto and E. Gilbert, "VADER: A parsimonious rule-based model for sentiment analysis of social media text," in *Proc. Int. AAAI Conf. Weblogs Social Media*, 2014, vol. 8, no. 1, pp. 216–225.

[18] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in *Proc. Int. AAAI Conf. Web Social Media*, 2017, vol. 11, no. 1, pp. 512–515.

[19] T. Gröndahl, L. Pajola, M. Juuti, M. Conti, and N. Asokan, "All you need is 'love' evading hate speech detection," in *Proc. 11th ACM Workshop Artif. Intell. Secur.*, 2018, pp. 2–12.

[20] S. MacAvaney, H.-R. Yao, E. Yang, K. Russell, N. Goharian, and O. Frieder, "Hate speech detection: Challenges and solutions," *PLoS ONE*, vol. 14, no. 8, Aug. 2019, Art. no. e0221152.

[21] S. Abro, S. Shaikh, Z. Hussain, Z. Ali, S. Khan, and G. Mujtaba, "Automatic hate speech detection using machine learning: A comparative study," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 8, pp. 1–8, 2020.

[22] K. Sreelakshmi, B. Premjith, and K. P. Soman, "Detection of hate speech text in Hindi-English code-mixed data," *Proc. Comput. Sci.*, vol. 171, pp. 737–744, Jan. 2020.

[23] H. Chen, S. McKeever, and S. J. Delany, "A comparison of classical versus deep learning techniques for abusive content detection on social media sites," in *Proc. Int. Conf. Social Inform.* Springer, 2018, pp. 117–133.

[24] S. Agrawal and A. Awekar, "Deep learning for detecting cyberbullying across multiple social media platforms," in *Proc. Eur. Conf. Inf. Retr.* Springer, 2018, pp. 141–153.

[25] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, "Deep learning for hate speech detection in tweets," in *Proc. 26th Int. Conf. World Wide Web Companion (WWW Companion)*, 2017, pp. 759–760.

[26] S. D. Swamy, A. Jamatia, and B. Gambäck, "Studying generalisability across abusive language detection datasets," in *Proc. 23rd Conf. Comput. Natural Lang. Learn. (CoNLL)*, 2019, pp. 940–950.

[27] H. Sohn and H. Lee, "MC-BERT4HATE: Hate speech detection using multi-channel BERT for different languages and translations," in *Proc. Int. Conf. Data Mining Workshops (ICDMW)*, Nov. 2019, pp. 551–559.

[28] I. Mollas, Z. Chrysopoulou, S. Karlos, and G. Tsoumakas, "ETHOS: An online hate speech detection dataset," 2020, *arXiv:2006.08328*.

[29] M. Mozafari, R. Farahbakhsh, and N. Crespi, "Hate speech detection and racial bias mitigation in social media based on BERT model," *PLoS ONE*, vol. 15, no. 8, Aug. 2020, Art. no. e0237861.
[30] Q. H. Pham, V. Anh Nguyen, L. B. Doan, N. N. Tran, and T. M. Thanh, "From universal language model to downstream task: Improving RoBERTa-based Vietnamese hate speech detection," in *Proc. 12th Int. Conf. Knowl. Syst. Eng. (KSE)*, Nov. 2020, pp. 37–42.
[31] B. Mathew, P. Saha, S. M. Yimam, C. Biemann, P. Goyal, and A. Mukherjee, "HateXplain: A benchmark dataset for explainable hate speech detection," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 17, pp. 14867–14875.
[32] A. Arango, J. Pérez, and B. Poblete, "Hate speech detection is not as easy as you may think: A closer look at model validation (extended version)," *Inf. Syst.*, vol. 105, Mar. 2022, Art. no. 101584.
[33] C. Baydogan, "Deep-Cov19-hate: A textual-based novel approach for automatic detection of hate speech in online social networks throughout COVID-19 with shallow and deep learning models," *Tehnički Vjesnik*, vol. 29, no. 1, pp. 149–156, 2022.
[34] M. Di Capua, E. Di Nardo, and A. Petrosino, "Unsupervised cyber bullying detection in social networks," in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, Dec. 2016, pp. 432–437.
[35] A. Rodriguez, C. Argueta, and Y.-L. Chen, "Automatic detection of hate speech on Facebook using sentiment and emotion analysis," in *Proc. Int. Conf. Artif. Intell. Inf. Commun. (ICAIIC)*, Feb. 2019, pp. 169–174.
[36] Z. Talat, J. Thorne, and J. Bingel, "Bridging the gaps: Multi task learning for domain transfer of hate speech detection," in *Online Harassment*. Springer, 2018, pp. 29–55.
[37] N. Vashistha and A. Zubiaga, "Online multilingual hate speech detection: Experimenting with Hindi and English social media," *Information*, vol. 12, no. 1, p. 5, Dec. 2020.
[38] W. Aldjanabi, A. Dahou, M. A. A. Al-qaness, M. A. Elaziz, A. M. Helmi, and R. Damaševičius, "Arabic offensive and hate speech detection using a cross-corpora multi-task learning model," *Informatics*, vol. 8, no. 4, p. 69, Oct. 2021.
[39] S. Jaki and T. De Smedt, "Right-wing German hate speech on Twitter: Analysis and automatic detection," 2019, *arXiv:1910.07518*.
[40] R. Alshalan and H. Al-Khalifa, "A deep learning approach for automatic hate speech detection in the Saudi twittersphere," *Appl. Sci.*, vol. 10, no. 23, p. 8614, Dec. 2020.
[41] H. Yang and C.-J. Lin, "TOCP: A dataset for Chinese profanity processing," in *Proc. 2nd Workshop Trolling, Aggression Cyberbullying*, 2020, pp. 6–12.
[42] C. Baydogan and B. Alatas, "Metaheuristic ant lion and moth flame optimization-based novel approach for automatic detection of hate speech in online social networks," *IEEE Access*, vol. 9, pp. 110047–110062, 2021.

MUHAMMAD BILAL received the M.Sc. degree in computer science from the University of Peshawar, Pakistan, in 2007, and the M.Phil. degree in computer science (specialization: database/data mining) from the Institute of Computer Science and IT (formerly called IBMS), The University of Agriculture Peshawar, in 2015. He is currently pursuing the Ph.D. degree in the discipline of computer science with the Islamia College University Peshawar. He published a research article with the title "Sentiment Analysis of Roman Urdu Opinion by Using Naïve Bayesian, Decision Tree and KNN Classification Techniques" in 2015 which laid base in the area as far as Roman Urdu script is concerned. He has recently published an article as the coauthor with title as "Machine Learning for Fake News Classification With Optimal Feature Selection" published in 2022 in *Soft Computing* journal. His current research interests include data mining, text mining, sentiment analysis and opinion mining, and machine learning.

ATIF KHAN received the M.Sc. degree in computer science from the University of Peshawar, Pakistan, in 2004, and the Ph.D. degree in computer science (text mining) from the Universiti Teknologi Malaysia (UTM), Johor Bahru, Malaysia, in 2016. Since 2016, he has been working as an Assistant Professor at the Islamia College University Peshawar, Khyber Pakhtunkhwa, Pakistan. His current research interests include data mining, text mining, sentiment analysis and opinion mining, recommender systems, and machine learning. He was a recipient of the Best Student Award and the Pro-Chancellor Award at UTM during his Ph.D., for his excellent contribution in the field of text mining. He is also serving as an Associate Editor for *ACM Transactions on Asian and Low-Resource Language Information Processing*. He is a technical committee member in many international conferences and a reviewer in many international conferences, journals.

SALMAN JAN received the master's degree in computer science from the University of Peshawar and the Ph.D. degree from the MIIT, Universiti Kuala Lumpur, Malaysia, in 2019. His Ph.D. thesis was on Android malware analysis and its integration with blockchain for preserving integrity of the behavioral logs and ultimately to secure the classification results of the behavioral logs through the employed deep learning models, including DCGAN, CNN, and FCNN.

He is currently working as a Senior Lecturer of cyber security and technological conversion at the Malaysian Institute of Information Technology, University of Kuala Lumpur. He has acquired skills and published in a number of well-reputed journals in areas of machine learning, deep learning, artificial intelligence, blockchain, the IoT, and security augmented and virtual reality. He also is actively working on international projects in the capacity of consultant and successfully completed four international projects. His publications and citations can be found at: https://scholar.google.com/citations?user=4n2MC74AAAAJ\&hl=en\&oi=ao.

SHAHRULNIZA MUSA received the Ph.D. degree in the field of communication network security from the Faculty of Electrical and Electronic Engineering, Loughborough University, U.K., in 2008. He has been the Vice-Chairman of the Security, Trust and Privacy Working Group and the Chairman of the IoT Security Sub-Working Group of Malaysian Technical Standards Forum Bhd (MTSFB), since 2018. He is currently a Full Professor at the Cybersecurity and Technological Convergence Section, Malaysian Institute of Information Technology, Universiti Kuala Lumpur (UniKL). His research interests include cybersecurity, the IoT security, blockchain, and big data. He has published more than 90 articles in academic peer-review publications in his research area. (SCOPUS ID: 52963995100).