Multimedia Tools and Applications (2024) 83:48021–48048
https://doi.org/10.1007/s11042-023-16598-x

HateDetector: Multilingual technique for the analysis and detection of online hate speech in social networks

Anjum[1] · Rahul Katarya[1]

Received: 12 December 2022 / Revised: 24 April 2023 / Accepted: 21 August 2023 /
Published online: 1 November 2023

Abstract
With the proliferation of social media platforms that provide anonymity, easy access, and the establishment of online communities and discussion, hate speech identification and monitoring has become a major concern for society, individuals and policymakers, which can be interpreted as hate speech. Many researchers attempted to detect hate speeches in multiple languages from social media, but the research was limited due to high complexity and minimum accuracy. A novel 'HateDetector: Multilingual Hate Speech Detection Technique' has been proposed to overcome these issues. In this technique, Bidirectional Encoder Representations from Transformers (BERT) with Multi-Layer Perceptron (MLP) is developed to identify the nature of the tweets by performing the process of code conversion and similarity check that results in good vector values representing the tweet nature. Additionally, the exact sentiment or nature of a tweet, whether hate or non-hate, is identified using the Profanity Check Technique (PCT), composed of ReLu activation function with a logistic regression classifier that classifies the resultant vectors and its respective emoji to neutral or hate speech. This technique performs all analyses of a tweet. It also auto-detects and easily finds hate speech, even from poorly written and complex text. According to the experiment's findings, the proposed technique performed exceptionally well, with a classification accuracy of 97.9%. Our proposed technique was able to compete with other state-of-the-art models.

Keywords BERT · Multilingual encoder-decode (MBart) · Online Hate speech (OHS) · Log-likelihood test · Logistic Regression · ReLu

✉ Anjum
anjum_2792@yahoo.com

Rahul Katarya
rahuldtu@gmail.com

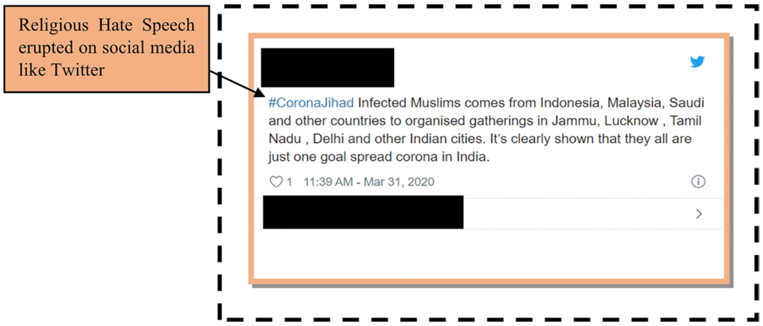
1 Big Data Analytics & Web Intelligence Laboratory, Department of Computer Science and Engineering, Delhi Technological University, New Delhi, India

1 Introduction

The vast majority of people across the globe are increasingly utilizing social media platforms such as Facebook, WhatsApp, Instagram, and Twitter. These platforms are user-friendly, easy to use, and give individuals the opportunity to have their voices heard. Now, people may readily share their thoughts and information from any location and at any time. Without meaningful restrictions or procedures, anybody can make detrimental and untrue comments in abusive or offensive language against anybody intending to spoil one's image and status in the community. Hate speech is the use of offensive and discriminatory language directed towards someone because of their origin, religion, race, sexual orientation, socioeconomic class, or any other attribute. When such things happen on social media to create content, blogs, or to exploit someone is called Online Hate Speech (OHS) [1]. The enormous number of posts, comments, and messages on these websites make it extremely difficult to monitor the information that is shared on them, despite the fact that they serve as a platform for individuals to express and exchange their ideas and viewpoints [2]. Moreover, many people tend to use aggressive, unwanted, and hateful language when discussing certain backgrounds, cultures, and other aspects [3]. This finally leads to inhumanity which may affect the social media users mentally, as also shown in Fig. 1.

People of all ages use social media, and they speak a variety of languages. These languages need to be translated into a common language, such as English, which is commonly referred to as "code." This is because English is a lingua franca language; more specifically, it is a declarative programming language that is vast, flexible, and standard. As a result, the term "multilingualism" refers to the ability of an individual to speak many languages and the presence of diverse language groups in the same geographic area [5]. Accordingly, in contrast to the general belief, most of the population uses multi-language or two languages at a minimum [6]. The minority population of the World uses a single language which is said to be monolingual. Multilingualism uses several languages as well as code-mixing [7]. Code-mixing is the mashing up of words, sentences, and phrases from two different grammatical systems within the same speech [8] (e.g., Tamil + English = Tanglish or Hindi + English = Hinglish).

The code-mixing of Hindi, which is the most widely spoken language in South Asia, with English is known as "Hinglish," a portmanteau derived from the two language's names. Hinglish differs significantly from its parent languages in syntax, phonetics, grammar, and even punctuation. The accent and sentiments are Hindi, while the vocabulary



[redacted]

#CoronaJihad Infected Muslims comes from Indonesia, Malaysia, Saudi and other countries to organised gatherings in Jammu, Lucknow , Tamil Nadu , Delhi and other Indian cities. It's clearly shown that they all are just one goal spread corona in India.

1 11:39 AM - Mar 31, 2020

[redacted]

Fig. 1 Hate Speech content on Twitter [4]

is made up of several English (Roman) transliterations of Hindi words, as well as a few English terminologies was the mixing of words, phrases, and sentences from two distinct grammatical (sub)systems within the same speech event [9]. With the wide-reaching popularity of social media platforms, code-mixing has emerged as one of the significant linguistic phenomena among multilingual communities that switched languages [10]. Thus, to detect hate speech, the existing presentations used a meta-learning approach based on metric-based and optimization-based (MAML and Proto-MAML) methods [9], Sentiment reversal analysis [11], Small-sized Transformer model [12], used several layers of classifiers, RGWE method for sentiment analysis [13], Deep convolution neural network [14], Lateral semantics analysis [15] and so on. The above-said methods quickly adapt and generalize new languages with labeled data points and obtain their objective. However, improvement is needed to analyze the data set given out as output because many techniques use low-resource languages and not high-resource languages, many classifiers to bring out its performance, and some concentrate only on some parts of hate speech. The major contribution in this paper is given as follows:

1. We have proposed a novel HateDetector that uses BERT-based word embedding with MLP in Multilingual Hate Speech Detection.
2. A ‘Profanity Check Technique’ is proposed to extract the emotion of the tweet for a better classification of hate speech.
3. The proposed model is trained and tested on three datasets against various models, which yields high performance under different experimentation settings.

The flow of this paper is as follows. Section 2 of this study briefly describes the literature review of various past studies related to detecting and analyzing OHS. Section 3 presents a detailed methodology of our proposed approach for the detection and analysis of OHS. Section 4 of this article describes the experimental results obtained after training the proposed model. Section 5 concludes our research work and presents the scope of future developments possible.

2 Literature survey

Classifying and detecting hate speech has recently gained considerable attention, and many researchers have proposed various approaches using ML and Deep Learning models. In this section, we have explored the various research papers for solving the problem of OHS. Some of the proposed models for OHS detection have been reviewed with its drawback.

2.1 Deep learning-based convolution neural network model

In standard machine learning, features are manually extracted, but in deep learning, this step is skipped in favour of directly incorporating data into a deep learning algorithm like a Convolutional Neural Network (CNN), which then makes additional predictions about the item. CNN is a subclass of DNN (deep neural networks). CNN is mainly used in the area of analyzing visual imagery. The three layers of an image are converted into a vector of suitable size, and then a DNN is trained on it. Their other applications include video understanding, speech recognition, and natural language processing. CNN was utilized by the author [16] to locate racist and sexist statements. The proposed model was tested by tenfold cross-validation and gave a 78.3% f-score. The author [17] employed text features, i.e., surface-level features, linguistic features, and sentiment features in deep learning classifiers, and then implemented an ensemble-based novel approach. The author found an accuracy of 0.918 with the novel approach. The authors provided the CNN-based web browser plugin to visualize the online hostility on Twitter and Facebook [18]. By monitoring Tweets before they are shared among other Twitter users, the authors have constructed the Deep Convolutional Neural Network (DCNN), an automated approach to filter postings related to hate speech [19]. A database of 500 million tweets per day was difficult to process though it utilized GloVe embedding vector to filter the tweets with the help of a convolution operation. However, it cannot achieve better accuracy in the current dataset extension as it grows bigger daily. The authors [15] have presented a model utilizing Long–Short-Term Memory (LSTM), Ada Boost, and BERT. The pre-trained data set was given to BERT, and the classifier outcomes were given to AdaBoost. This model was used to identify the hate speech in the Bengal language along with Benglish content. Using L-Boost efficacy, it attained 95.11% accuracy. However, they did this only for one language with many classifiers.

2.2 Transformer methods

The transformer [20] is the latest innovation that has taken the natural language processing domain by storm. The transformer can account for long-term dependencies, but unlike LSTMs, transformers do not process data sequentially as done in the case of RNN's and LSTMs. Instead, to account for the position of each word is added to its embedding. The transformer was first introduced for machine translations (Sequence to Sequence Model), and thus it has two components, an encoder and a decoder. Though only the encoder is relevant in text classification tasks such as Hate speech detection. In an encoder, the inputs are first fed into a self-attention layer which generates an embedding taking into account all other words in a sentence and depicts the relevance of each word for a particular word. The embeddings obtained from the self-attention layer are fed into a neural network. This process is repeated many times, i.e., Many layers of self-attention and neural networks are stacked to form the encoder. The decoder of a transformer is very similar to the encoder except for an Encoder-Decoder attention layer, which is added to find the inputs relevant to a particular output [20].

In the context of hate speech detection, embedding obtained from the pre-trained model such as BERT (Bidirectional Encoder Representations from Transformers) has been widely used. BERT is a transformer trained using the masked LLM technique. Masked LM technique [21] requires 15% of the words in the sentence to be masked, and the transformer then attempts to predict these words from context during the training process. In the paper [22], the authors showed the efficacy of fine-tuning the BERT in the context of hate speech detection. Comparing pre-trained models for hate speech detection explores and compares various multilingual transformers such as mBART (multilingual encoder-decode). The research authors [23] contend that to obtain cutting-edge accuracy for the multi-class classification problem of online hate speech, transformers must be utilised instead of classic machine learning, RNN-based deep learning, or even attention-based RNN models. They propose a streamlined version of BERT, called DistilBERT, which has half the number of parameters with no loss in performance. On comparison and experimentation with various LSTM and BERT-based models, DistilBERT outperforms all the models given on various metrics. In our proposed work, we have introduced a novel technique that uses novel features of mBART, a sequence-to-sequence pre-trained model on large multilingual corpora.

In the paper [12], the author presented a simple transformer model on Twitter-based hate speech. They attained a 0.8426 F1 score and by providing more training data, they achieved a 0.8504 F1 score. However, it can be improved by the fore-coming techniques and other transformer models to extend the evaluation of Hate speech detection.

2.3 Sentiment analysis approach

The author [9] has presented a Meta-learning-based approach to analyze the issues of offensive content and few-shot hate speech in low-resource languages by only evaluating a few-labeled data items in a specific target language at high speed. This research has identified hate speech and offensive language as two different tasks and collected the data into two different datasets, some datasets across eight languages for hate speech and others across six languages for offensive languages. However, the number of languages was restricted, and high-resource languages were not considered. In the paper [24], the author has investigated Sentiment reversal which was used to find the relation between textual information and the sentiment diffusion pattern of Twitter messages. This investigation was done on the iterative algorithm Sentidiff, which yields 5.09% and 8.38% of PR-AUC on the classification of Twitter sentiment tasks. However, this model holds only the sentiment analysis where no other part of Twitter data handling was done. The authors [25] have investigated hate speech and offensive language detection by considering hate speech and similar content. Hate speech and offensive language were related to sentiment and emotion analysis, where the sentiment analysis was evaluated using negative opinions and emotions by pretending or experiencing the anger addressed in the speech. Finally concluded that hate speech and offensive language targeted a specific group or person. However, improvements can be made to identify the given speeches in the content. In the paper [26], the authors have introduced the RGWE method based on the concept of sentiment analysis which was used to find the optimal concept for the sentiment of words under different contexts with sentiment representation of words. However, only the noun concept was done by Microsoft concept graph, whereas verbs, adverbs and so on were not found in the paper. The author [27] have presented a model using Latent Semantic Analysis (LSA) to detect hate speech in social networks, which uses ten binary data sets with different categories. It used 2–4 ngrams to find character and 1 to 5 ngrams to find words and an advanced machine language model, CAT boost, to attain its objective of detecting hate speech, but it did not work for complex and overlapping hate speech. In Table 1, the systematic literature review of the hate speech paper has presented in which the first two column shows the reference and objective of that reference. Next, the classifiers used in the experiments, the dataset reference, and the results and limitations were discussed.

From Table 1, We can determine that the authors have included numerous feature extraction approaches, and by using those features, they trained different ML classification algorithms. Authors have devised various novel models for hate speech detection and have worked on fine-tuning its parameters.

2.4 Limitations of the existing research

1. The research work has been limited to spotting hate in the English Language and few pieces of research in Arabic, Indonesian, Italian, Turkish, Swedish, Albanian Language, and hate content in the rest of the languages like Hindi and Hinglish goes unfiltered.

Table 1 Systematic Literature Review of latest hate speech detection techniques

Ref	Objective	Classifier Used	Features Used	Dataset	Results and Limitations
[28]	Deep multitask framework has proposed to combat the online hate speech	CNN, GRU, LSTM	Unigrams, Bigrams, Trigrams, TF-IDF	Hatebase.org	• A deep multi-task learning framework has been suggested to exploit the information from these various connected tasks • The framework can be extended to include user metadata, user posting patterns that promote harmful speech, and the likelihood that any hazardous language will spread
[19]	Proposed a Deep convolution neural network	DT, LR, NB, SVM, GB, KNN	Self-extraction of contextual features	T. Davidson from GitHub	• The proposed DCNN model makes use of the tweet text and the GloVe embedding vector to convolutionally extract the semantics of the tweets •To detect hate speech patterns, a more comprehensive dictionary could be developed
[29]	Proposed a neural architecture for classifying multilingual hate speech on Twitter	Fasttext Ccrawl, LSTM, GRU	Word embedding, Ngrams, Social network–specific features, Emoji embedding	Data was collected from Zeerak Waseem and Dirk Hovy Dataset from HateSpeechData website by Evalita	• The author has experimented with English, Italian, and German language • They have first discovered a recurrent neural architecture that is fairly reliable and effective across several languages

Table 1 (continued)

Ref	Objective	Classifier Used	Features Used	Dataset	Results and Limitations
[30]	Proposed technique for Classifying texts on whether they are racist or not	SVM	BOW, Bigrams, POS tags	Hate speech data was gattered from Yahoo Directories of (3 million words)	• The text classification Using using SVM, problems like overfitting can be prevented • Proposed techniques save time and labour • Training and testing was performed on comparatively small dataset
[31]	Detection of intent in a Twitter Post	Random Forest, Naive Bayes, Decision Tree	Lexicon Features, Cohen's Kappa coefficient, NLP	Data collected from Tumblr (3228 text posts)	• They discovered numerous factors that could result in unjustified bias • As a result of their focus on a small number of annotators, the number of posts examined in this paper is rather small
[32]	Study of Twitter data for racial bias	Logistic regression	BOW, TF-IDF	Blodgett et al. 2016 (59.2 million tweets)	• Classifiers trained with each of the five datasets exhibited racial bias • Distinct algorithms and global features may produce variable outcomes • Only one facet of racial bias was studied in this study
[33]	Proposed a Novel algorithm to detect hate speech messages of Twitter	Random Forest	Kappa coefficient	Data collected using Twitter APIs (20,880 tweets)	• Their performance prediction contributed to the regulation of consistent nodes, which resulted in a smaller tree that was not overfitted • The dataset is confined to attacks in London

Table 1 (continued)

Ref	Objective	Classifier Used	Features Used	Dataset	Results and Limitations
[34]	Identifying hate speech in posts using RNN	RNN	Character Embeddings, GRU	HatEval (19,800 tweets)	• The approach presented in this paper was able to manage posts with large noises • Several variants were assessed to establish the ideal model
[35]	Detection of hate speech using CNN deep learning model	CNN	Character 4-g, Word2vec, N-grams, BOW	Waseem (6656 tweets)	• Various types of hate speech have been classified • Various features such as Bert and combination techniques such as LSTMs or transformers may be incorporated in the future

2. In order to furnish research in the field of OHS, a multimodal and multilingual dataset should be developed.
3. One of the constraints is the lack of publicly available data towards the progress of online hate speech detection.
4. Very few give high-accurate performance profanity check techniques have been introduced.
5. Lastly, to reduce the complexity of words caused by incoming datasets while performing detection is not explored.

From the existing reviewed proposals, it was viewed that they used many classifiers, concentrated only on a single analysis, hate speech detection was done only for one language, and required improved performance in those models. Thus, a novel technique must be implemented to overcome the abovementioned issues. Our research has tried to devise a novel technique using Bidirectional Encoder Representations from Transformers with multi-layer Perceptron to identify the nature of the tweets by performing the process of code conversion and similarity check that results in good vector values representing the tweet nature. Additionally, we exacted the sentiment or nature of the tweet, whether it is positive or hateful, is identified using PCT that is composed of ReLu activation function with a logistic regression classifier that classifies the resultant vectors and its respective emoji's. In next Sect. 3, we discussed the proposed methodology of research work.

3 Proposed methodology for HateDetector

The "HateDetector: Multilingual Hate Speech Detection" model consists of several steps that help in detecting and analyzing hate speech. Firstly, the model uses a Bidirectional Encoder Representation of Transformer (BERT) to pre-process the tweets from Twitter media. BERT is a powerful natural language processing model that can understand the context of the text and produce high-quality embeddings for text. The word embeddings produced by BERT are then given to the mBART model, which converts the code-mixed tweet to English. This conversion helps in ensuring that the model can detect hate speech in multiple languages and code-mixed languages. After the tweet has been converted to English, it is given to Bag of Words (BOW) with N-gram. BOW is a technique that splits the given sentence into words and weighs each word using numbers by mapping the sentence provided using vector equations. This technique helps in detecting and identifying the frequency of the occurrence of specific words that may be associated with hate speech. The output of BOW is then fed to the similarity checker to check the spelling and grammar of the tweet. This step ensures that the model can detect and analyze hate speech that may contain incorrect spelling or grammar. Finally, the model employs the Profanity Check Technique (PCT) with Rectified Linear Unit (ReLu) activation function to recognize the emotion of the sentence. PCT is a technique that uses a set of profane words to identify the emotion associated with a sentence. The output of the PCT layer is then given to a sigmoid function that works between the zeros and ones. The sigmoid function provides binary outcomes that indicate whether the sentence is positive or negative. Then finally the logistic regression layer transmits the final output by analyzing the nature of the sentence, whether it is positive or negative, and identifying any hate speech in the sentence. The proposed model provides an effective solution to the challenges faced by Twitter in managing and detecting hate speech in the open space. The model employs various techniques such as BERT, Multilayer Perceptron-based word embedding, and Profanity Check Technique (PCT) to detect and analyze hate speech in multiple languages, code-mixed language, and identify the emotions associated with the sentence. The model's ability to detect and analyze hate speech in various languages and code-mixed language makes it a valuable tool in promoting a safe and healthy social media environment.

Figure 2 shows the process flow of the proposed model. Initially, a code-mixed tweet is given to the BERT with Multilayer Perceptron (MLP), which consists of several encoders

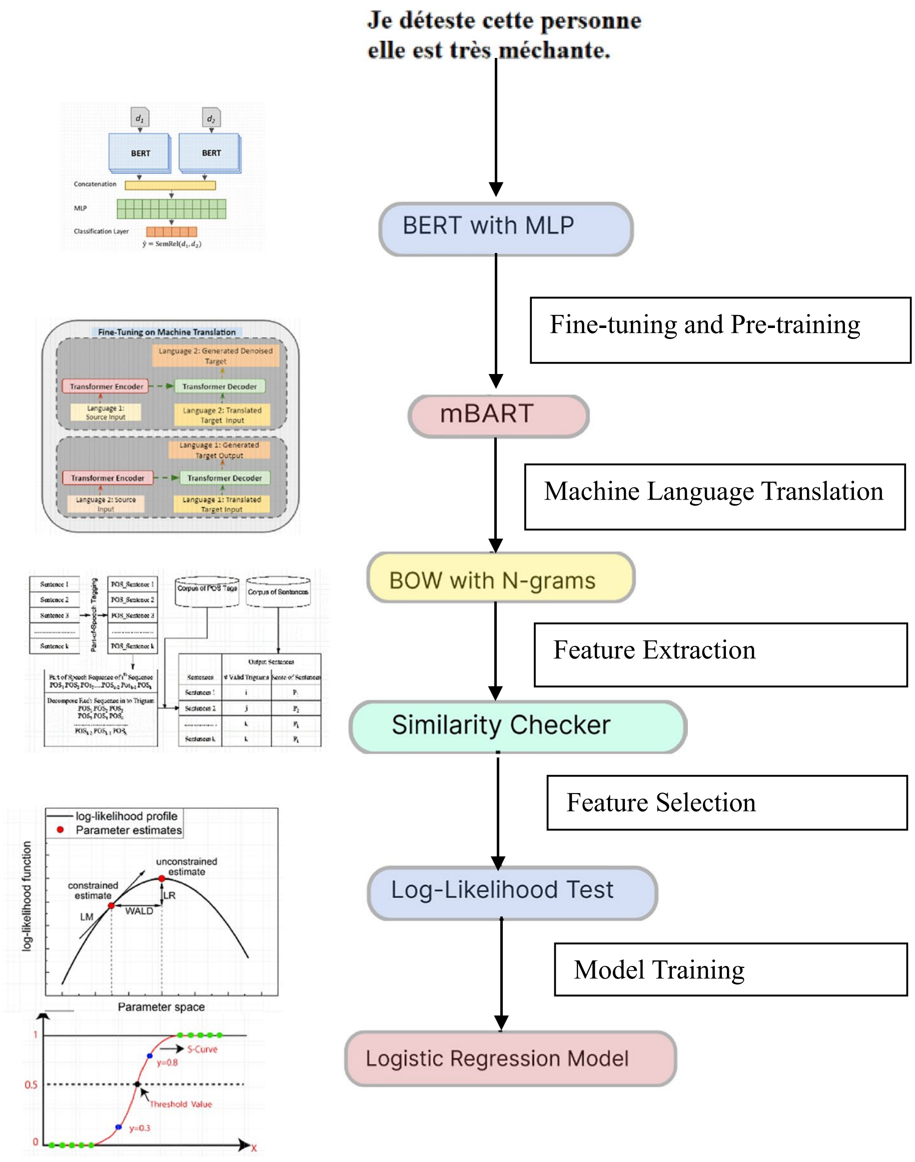


Fig. 2 Flow chart of the proposed technique

and decoders to pre-process the mixed tweets and remove all the unnecessary stopping words and tags. All the hidden layers are used to compress both text and image datasets. The pre-processed data is entered into the mBART (multilingual encoder-decode), which de-noise the key-phrased dataset again. The BOW with n-gram maps the vectors in the dimensional feature space and splits according to the gram. A similarity checker is used to gamble the vocabulary that collects the data, analyses the meaning confirms it with the trained data available and investigates the speech. The logarithmic value of the log-likelihood test performs the process of sensing the Positive sentiments of the words. However, it could not reach the correct sense of the sentence, so PCT with activation function and classifier is performed with the sigmoidal flow of zeros and once, which precisely identifies the nature of the sentence and detects the hate speech. In the next section, we discussed our proposed methodology in a detailed fashion.

3.1 Multilingual hate speech detection technique

A multilingual hate speech detection that holds up a BERT with multi-layer perceptron-based word embedding. It detects hate speech of tweets with non-linguistic concepts. This is done with the help of several tests and functions performed within the multilingual hate speech detection technique, which we will discuss in the below sections. The classification performance is divided into a trained, test, and validation set. This training set is used as a reference for the pre-processed dataset, the test set is to optimize the proposal's approach, and the validation set is to evaluate the proposed technique.

3.1.1 Bidirectional encoder representation of transformer with MLP

The proposed model consists of BERT [21] with a 12-layer transformer network that pre-trains the data set and fine-tunes the tweet. The pre-trained tokens are provided to each layer, a token from each layer is used as a token for the next layer, and this word embedding uses a neural network. Multi-layer Perceptron embedded within the BERT is a feed-forward artificial neural network with one input layer and 768 hidden layers and one output layer that processes the input data by backpropagation.

In Fig. 3, the data set assigned for the task is given into the input layer of the BERT, where the input data is fine-tuned and pre-trained in the hidden layers composed within

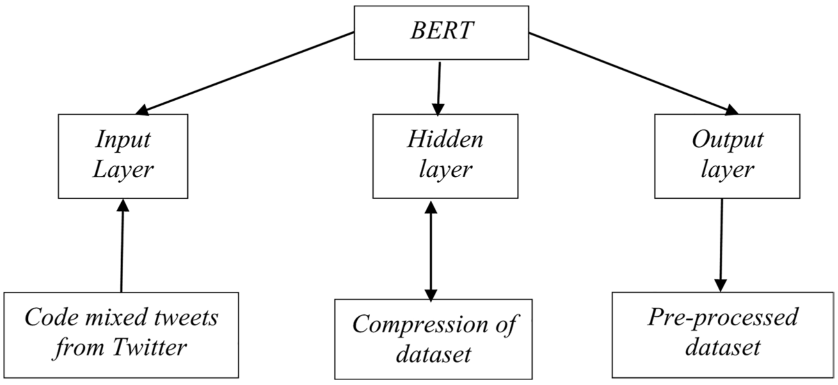


Fig. 3 Pre-processing of tweets in Twitter datasets by BERT

the BERT, and the key phrases are given out through the output layer. Here, the input layer sends the data set, the upcoming hidden layer compresses the tweets fed into it, and the pre-processed key phrases are given out in the output layer. The BERT holds many pre-trained tasks that helps to perform well. Algorithm 1 shows the pre-processing flow where the tweet is cleansed by converting the lower to upper case and removing the stopping notations, URLs, tags. The tweet is split into each character, and the BERT model starts reading each character compared with the pre-trained data set.

```
Input: Tweet text
Output: Cleaned tweet tex
    1. Convert all characters to lowercase
    2. Remove URLs and mentions (words that start with '@')
    3. Remove hashtags but keep the words after them
    4. Replace contractions (e.g., "can't" -> "cannot")
    5. Replace slang and informal words with their standard equivalents (e.g., "u" -> "you")
    6. Remove non-alphabetic characters and punctuation, except for exclamation and question
       marks
    7. Tokenize the text into individual words
    8. Remove stop words (common words that don't carry much meaning, such as "the" and
       "and")
    9. Perform stemming or lemmatization to reduce words to their base form
    10. Combine the remaining words back into a single string and return the cleaned text.
```

Algorithm 1 Pre-processing of Tweets for the Hate speech detection

In the pre-processing section, the given data set is processed before it is given as input to the mBART. The dataset consists of tweets which further consist of Emojis to express emotions, tags, stopping notations, uppercase letters, and URLs are cleansed initially from each tweet by the above algorithm 1. Whether the text contains hate or not, these embellishments add nothing to the text. The next step of pre-processing the tweet is tokenization and lemmatization, where another specific symbol replaces the sensitive data without giving up the security level and the tokens are lemmatized and finally, the output is stemmed and key-phrased where plurals are removed and forms as a key-phrase sentence that is stored and used when it is needed.

In Fig. 4, the complete pre-processing technique of the Twitter data set is shown with an illustration. In a sense, how a tweet is converted into Tokens and key phrases into a sentence. Here, a tweet from the Twitter page is used. It is first tokenized into a single word, then lemmatized, which means the keyword is arranged in the pre-processed data set, shortcuts of words are enlarged, then key-phrased into a sentence. A brief illustration of the implementation of BERT with MLP is given below.

1. Preprocess the input data: We have tokenized the input into individual words or subwords, converted it into numerical sequences, and padded it to a fixed length. It involves breaking the input text into individual words or subwords. We have used a tokenizer

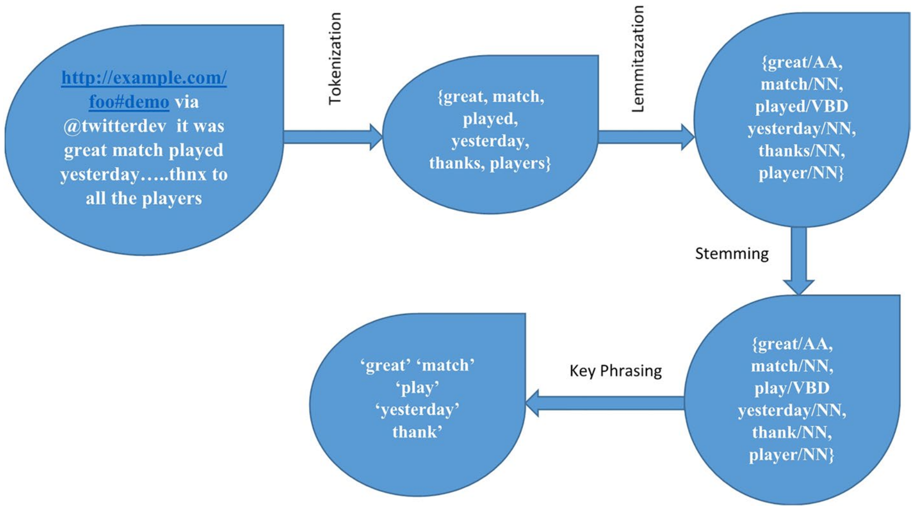

http://example.com/foo#demo via @twitterdev it was great match played yesterday…..thnx to all the players
Tokenization
{great, match, played, yesterday, thanks, players}
Lemmitazation
{great/AA, match/NN, played/VBD yesterday/NN, thanks/NN, player/NN}
Stemming
{great/AA, match/NN, play/VBD yesterday/NN, thank/NN, player/NN}
Key Phrasing
'great' 'match' 'play' 'yesterday' thank'

Fig. 4 Systematic representation of Preprocessing of the Twitter Dataset

library such as Tokenizer from the Transformers library in Python. We have created an instance of the tokenizer class and used its encode method to tokenize the input text into a list of tokens.

2. Load the pre-trained BERT encoder: Then we used TensorFlow to load the pre-trained BERT model.
3. Extract features using BERT encoder: In this step, we have passed the preprocessed input sequences through the BERT encoder to obtain the output features for each token. BERT outputs a sequence of vectors of dimension d, where d is the hidden size of the BERT model.
4. Feed features into an MLP classifier: We flatten the sequence of output features to obtain a fixed-size feature vector for the entire input sequence and feed this vector into an MLP classifier. The MLP consists of one or more hidden layers, each with a nonlinear activation function such as ReLU, and a final output layer with the appropriate number of units for the task-specific prediction.
5. Train and evaluate the BERT-MLP model: We have trained the model on the labeled training data using an optimization algorithm such as Adam and evaluated its performance on a held-out validation set. Then model is fine-tuned by adjusting the hyperparameters such as learning rate, number of hidden layers, and batch size.

3.1.2 mBART: Multilingual bidirectional encoder representation of transformer

The pre-processed and pre-trained output from BERT is given as an input to mBART [36] that denoises the multilingual text by encoding it sequence by sequence, Hinglish is converted into English since English is the code for the NLP, which fine-tune any of the supervised or unsupervised pair of languages without any specific modification in task with one trained set of parameters for all languages even if it is complex.

The process flow of mBART is shown in Fig. 5, an autoencoder converting the output from BERT to the reconstructed code output by compressing the key phrases into their

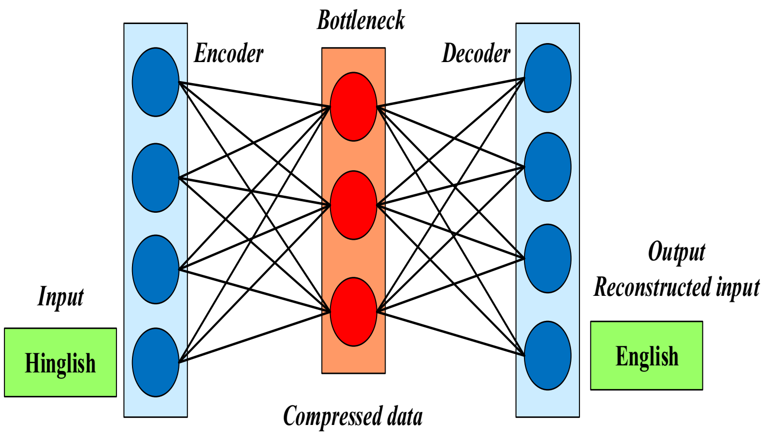


Fig. 5 Process flow of mBART [37] for proposed technique

original language and then converting it into the code language understandable by the machine.

Example String: "Je déteste cette personne, elle est très méchante." This is a French sentence that translates to "I hate this person, she is very mean." in English. Here we have used mBART model to detect whether this sentence contains hate speech or not. So, first, we have performed the preprocessing on the above text to convert it into a format that can be fed to the mBART model. This involves tokenizing the text, converting it into numerical sequences, and padding the sequences to a fixed length. In this research, we have tokenized the text into individual words, converted each word into its corresponding numerical index in the mBART vocabulary, and pad the sequence to a fixed length of 50 tokens.

Tokenized text: [’Je’, ’déteste’, ’cette’, ’personne’, ’,’, ’elle’, ’est’, ’très’, ’méchante’, ’.’].

Numerical sequence: Next, we converted the tokenized sequence into a numerical sequence by mapping each subword to its corresponding numerical index in the mBART vocabulary. This gives us the following numerical sequence: [27, 30,888, 618, 4480, 18, 299, 51, 529, 31416].

Since mBART processes sequences of fixed length, we need to pad the numerical sequence to a fixed length. In our experiment, we used the padding of 10. Then we can pad the sequence with the special padding token, which has a numerical index of 1, to get the following padded sequence: [27, 30,888, 618, 4480, 18, 299, 51, 529, 31,416, 1]. Now, we fed the padded numerical sequence into the mBART encoder network to get the fixed-length representation of the input sequence. The encoder network applies a series of self-attention and feedforward layers to the input sequence to compute a sequence of hidden states. We took the final hidden state, which summarizes the entire input sequence, as the fixed-length representation. We denoted the fixed-length representation as p.

p = Encoder([27, 30,888, 618, 4480, 18, 299, 51, 529, 31,416, 1]).

Finally, we fed the fixed-length representation z into the mBART decoder network to generate the output sequence in the target language. The decoder network applies a series of self-attention, encoder-decoder attention, and feedforward layers to the input representation to compute a sequence of hidden states, which are then used to generate the output sequence token by token. Since the input sentence is in French, we have set the target language to be English, which is the code language for the mBART model. We denoted the output sequence as y.

y = Decoder(z, [2])

Here, [2] is the numerical index for the special start-of-sequence token in English. The decoder network generates the output sequence by predicting the next token in the sequence given the previous tokens and the input representation z. The output sequence is terminated by the special end-of-sequence token, which has a numerical index of 0 in English.

3.1.3 BOW with N-gram

In this step output from mBART is given into BOW with N-gram that splits words from the sentence. Features are extracted by BOW that tokenizes the sentence into words and then these tokens are taken as an array of the frequency called it as bow vector; finally, these vectors are mapped at a fixed dimension in the space given. In simple words, a positive, negative or neutral sentence is converted to numbers by the following methodology of mapping the words.

Let us take all the sets of finite sequences of words as A and pre-trained dictionary labels as D. Mapping of finite set A to the dimensional feature space is done by

$$\varnothing : A \to R^M \tag{1}$$

The sentiment label set is taken as $\gamma = \{1, \ldots .., K\}$, assuming $K = 2$ shows whether it is positive or negative. The labeled trained data set trains γ and the input dataset is given as $x = (w1, \ldots .wN)$ where N is the input sequence length. In bags of n-gram $\varnothing(x)$ maps x to $M = |\Gamma|$ where M is dimensional latent space where Γ is the vocabulary of n-grams and Γ is evaluated using $|\Gamma| = O(|D|^n)$.

The embedding of the γ_j is evaluated by,

$$P_{\gamma_j} = h(F \times z_{\gamma_j} \tag{2}$$

where F: projection matrix

$$h(.) = \tanh(.)^3$$

$$\gamma_j = (w_{j,} w_{j+1}, \ldots .., w_{j+n-1})$$

F: projection matrix

z_{γ_j} is an operator string of trained set

Finally, the vector representation of a document is represented as

$$\varnothing(x) \equiv d_x = 1/N \sum_{j=1}^{N} P_{\gamma_j} \tag{3}$$

where, $d_x \in \mathbb{R}^M$ and $x = (w1, \ldots., wN)$

The reason for formulating this evaluation is to fix the length n of the sentence with the number of phrases.

A simple N-gram calculation is done by

$$Ngram_K = X - (N - 1) \tag{4}$$

where,

$Ngram$: Weightage

K: Given sentence

X: Number of words

Equations 3 and 4 help in the easy and correct calculation of n-grams. The output of the n-gram is an n-gram dictionary with an actual valid n-gram id, n-gram phrases, document frequency, length of n-gram, global term frequency, and other information [18]. We use these details to find the output in the next step of our process.

3.1.4 Similarity checker and log-likelihood test

The similarity checker detects, removes, and corrects the spelling mistakes in the respective input from the n-gram dictionary. We use a computer-programmed similarity checker, Grammarly and plag checker, that compares the present vocabulary with the already stored one. Furthermore, to evaluate whether the output of the checker is a good one or not, Log-Likelihood Test is performed. If the log-likelihood value is higher than the dataset better suits the model. It ranges from a Negative infinity value to a Positive infinity value. Values are obtained by Eq. 5,

$$l(\theta) = lnL(\theta) \tag{5}$$

where

$L(\theta)$:the output of the similarity checker

$l(\theta)$: the output of the log-likelihood test

From Eq. (5), the validated vocabularies are given out as output, but it is not precise in its emotion analysis as the classification of words may fail to detect the nature of the sentence correctly. Thus "Profanity Check Technique" is implemented which detects the correct emotion of the sentence, whether it is sad or happy, or angry.

3.2 Profanity check technique

We proposed a method called Profanity Check Technique (PCT) that uses a neural network to identify negative or profane language in a given string of text. The neural network is built using Rectified Linear Unit (ReLU), a popular activation function used in deep learning models. The PCT method utilizes a library of words and phrases that are known to be associated with negativity or profanity. The input string is compared with this library to determine whether it contains any of these words or phrases. If the input string is found to have negative or profane language, it is flagged as such. The neural network used in PCT consists of multiple layers and is trained using sigmoidal function and logistic regression. The sigmoidal function is used to activate the vectors provided to the neural network. However, this function can sometimes suffer from vanishing gradient issues, which can lead to slow convergence during training. To mitigate this problem, We have used ReLU in the calculation, which does not use any transforms and thus avoids vanishing gradient issues.

ReLu function $f(x)$ is calculated by

$$f(x) = max\ (0, x) \tag{6}$$

From Eq. (6) the output of the function is found. As if it returns 0 it is negative, or else it is positive. Though, complex functions cannot be learned using this ReLu function which may lead to regression problems. A generalized linear classifier model known as logistic regression is implemented to overcome this issue. That undergoes supervised learning with a labeled dataset and analyzes pre-trained binary data where the output is continuous with the parameters and summing of input and brings out the relation between independent and dependent variables. Logistic regression uses the sigmoid activation function to take back the label's probability to code any value between 0 and 1. The S curve shows that the output is negative, where it cannot go behind 1. This is calculated using the Eq. 7 below.

We have used the Loglikelihood test along with the Profanity Check Technique to improve the performance of the hate speech detection model. While the PCT is useful in identifying profane words or phrases, it may not be enough to accurately identify hate speech since not all hate speech contains profanity. The log-likelihood ratio test is a statistical method that can help determine the likelihood of a given text being hate speech based on the frequency of certain words or phrases in hate speech versus non-hate speech. By combining the PCT with the log-likelihood ratio test, the model can potentially achieve a higher accuracy in detecting hate speech. We have proposed the novel PCT algorithm 2 for the OHS problem.

$$Loglikelihood\ test = 2\log_e\left\{L_s(\theta)/L_g(\theta)\right\} \tag{7}$$

where,

s :pre-processed sample

g :pre-trained parameters

From equation [7] the output of the log-likelihood test is evaluated by giving the sample attained with the pre-trained dataset available.

```
Input: $twt$: A text string
Output: decision: A binary label indicating whether the text contains profanity or not
Begin
1. Input: A string of text
2. Perform pre-processing on the input text:
        a. Convert the text to lowercase
        b. Remove all punctuation marks
        c. Tokenize the text into individual words
        d. Remove stop words
        e. Perform stemming and lemmatization
3. Use a pre-defined list of profanity words and compare each word in the pre-processed text to the list of
   profanity words.
4. If any profanity words are found, assign a score of 1 for each profanity word.
5. Calculate the loglikelihood score for the input text using a pre-trained language model:
    a. Tokenize the pre-processed text into a sequence of tokens
    b. Use the language model to predict the probability distribution of the next token given the previous tokens.
    c. Calculate the log-likelihood score of the input text based on the predicted probability distribution
6. Combine the PCT score and the loglikelihood score to classify the input text:
    a. If the PCT score is greater than a pre-defined threshold AND the loglikelihood score is negative,
       classify the input text as hate speech
    b. If the PCT score is greater than a pre-defined threshold but the loglikelihood score is positive,
       classify the input text as profanity
    c. If the PCT score is below the pre-defined threshold, classify the input text as non-offensive
7. Output the classification of the input text.
```

```
Example: for the above Algorithm
Input: "I hate this person, she is very mean"
Pre-processed text: "hate person mean"
Profanity words: ["hate", "mean"]
PCT score: 2
Loglikelihood score: -7.5
Threshold: 1.5
Since the PCT score is greater than the threshold and the loglikelihood score is negative, the input text is
classified as hate speech.
```

Algorithm 2 **PCT for OHS**

The final output from the above algorithm is then passes to the logistic regression algorithm. It is a machine learning algorithm that is commonly used for binary classification tasks. It takes a set of input features and learned a set of weights for those features that can predict the probability of a given example belonging to a particular class (in this case, the class of hate speech vs. non-hate speech). Once the input text has been preprocessed and features have been extracted using Bag-of-Words with Ngram, logistic regression is trained on these features to learn the weights that best predict the probability of a given example being classified as hate speech. During training, the logistic regression algorithm iteratively adjusted the weights of the features until it achieves the highest possible accuracy on the training data. Once the model has been trained, it is used to predict the probability of new examples belonging to each class. In the case of hate speech detection, the logistic regression model would output a probability score for each input text indicating the likelihood that it is hate speech. The output of the PCT with Loglikelihood test is used as input features for the logistic regression algorithm, which can then learn to predict the probability of a given example being classified as hate speech.

So, it is clear that the proposed technique utilizes high resource language for hate speech detection and a much more pre-trained data set is placed in the dataspace with the help of a multi-layer perceptron embedded with BERT. All the forms of text, images, and emojis are taken into account to detect the OHS, and finally the emotion is efficiently predicted with the help of the above profanity check technique.

4 Experimental result

BERT with a multi-layer perceptron-based word embedded system has been recently used by most researchers who undergo research on hate-speech detection. The technique used in this study was a Multilingual hate speech detection technique with word embedding and detected hate speech on the Internet. The proposed technique is used to detect the pre-processed dataset's vocabulary. Good words have been evaluated, and hate speeches have been detected by sensing the exact emotion of the sentence formulation and reporting an individual or a group.

4.1 Implementation setup

Python programming language (version 3.6) [38] and its machine learning (ML) module Scikit-Learn [39] was utilized for all experiments and model construction. All calculations were performed on a system with an Intel(R) Core i7 8750H CPU 2.20 GHz processor, 8 GB RAM, and a 4 GB NVIDIA GeForce 1050 TI Graphics Processing Unit running Windows 10 Home edition. Google Collaboratory Notebook [40] was utilized for certain intensive computations and model training. Further in this section, we have discussed the four datasets collected from the open-sourced repositories, the experimentation results we obtained after training the proposed technique on these Datasets, and a brief discussion and comparison report on the results obtained. This section is further divided into three subsections to incorporate all this information.

4.2 Data set description

The simulation result has been discussed in this section in detail. The hate speech detection uses the Multilingual Hate Speech Technique, which uses BERT with a multi-layer neural network that is further encoded with PCT to hold the emotion of the data set. The methodology carries tweets on Twitter media as the dataset for performing the methodology.

1 T. Davidson [41]: The dataset consists of 24,783 tweets divided into three categories: Hate speech (5.8%), Offensive Language (77.4%) and Neither (16.8%). This open-sourced dataset was collected from GitHub. The tweets labeled as Offensive or Neither were included in the Non-Hate Category. Furthermore, the data set column labeled as having Hate Speech was included in Hate Category.
2 Hinglish Hate Speech Dataset: The first set of data was drawn from the paper [42]. Two linguists, one fluent in Hindi and the other fluent in English, annotated the tweets that were gathered through the Twitter API. Data from the research classification of Offensive Tweets in the Hinglish Language [43] was used for the second batch of data. As a result of our current task, the dataset has been relabeled into two classes: Not-Abusive and Hate-inducing categories have been labelled as hate, while the non-offensive category has been relabeled as not hate. The final piece of data comes from the HASOC task, which was a collaborative effort.
3 Hot (Hate Offensive Text) [43]: It was collected by using the Twitter Streaming API[3], in which the tweets had more than three Hinglish words. These tweets were then manually annotated. This collection of tweets covered the period spanning November 2017 and February 2018. In order to limit the scope of our data mining to a particular geographic region, we only considered tweets that originated from the Indian subcontinent. It has been shown that in a Hindi Offensive Text (HOT) data set with many numbers, nearly 24,779 tweets are taken for pre-processing. The tweet has been divided into Hate speech, offensive language, and other classes. Then the given tweets are pre-processed by the BERT with a multi-layer perceptron in the Multilingual Hate-speech Detection Technique.

The tweets are pre-processed using BERT, as the upper-case letters are converted into lower-case letters, removal of tags, URLs, and stopping notations makes the machine easy to understand and validate the given tweet. The first part of the processed tweet holds the tweets with the capitalization of letters and the second part comprises pre-processed tweets without capitalization and has no tags like #, @, and so on. This simulation result has been obtained by applying the proposed technique. In the above simulation, the tweets have been pre-processed using BERT. Followingly, the Hinglish comment has been given to the mBART to convert them into English. Then forwarded to the similarity checker where the incomplete or misspelled words are vocabulary and spell-checked using an online checker forwarded to the log-likelihood test where the tweet's negative, positive or neutral nature is verified. Additionally, the sentiment of the tweet has been analyzed by the Profanity check technique. Here, with ReLu, sigmoid function, and logistic regression, the last validated words are compared with the nearby words in the sentence. The emotion has been found out whether it's negative, positive, or neutral. Finally using the above simulation, the input tweet has been classified into three sub-categories. When we come to Hate speech, 4744 words are optimized with 38,419 reference words, and from those, 4259 words are validated.

4.3 Performance evaluation metrics

The performance of the proposed technique has been evaluated in this subsection with parameters [44] such as Precision, Recall, Accuracy, and F1-score. Then this performance evaluation graph comprises the result with other techniques such as Bidirectional Encoder Representation from Transformer- Convolutional Neutral Network (BERT-CNN), Convolutional Neutral Network (CNN), Bidirectional Long-Short Term Memory (Bi-LSTM), Ternary Trans-CNN, Deep Learning (DL) Ensemble Stacked, Embeddings from Language Model (ELMO). Equations 8–11 show the mathematical formulas of these evaluation metrics.

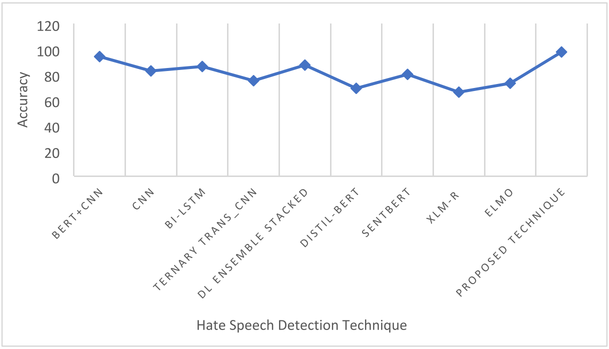


Fig. 6 Accuracy result of the HateDetetctor technique with previously Implemented techniques

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

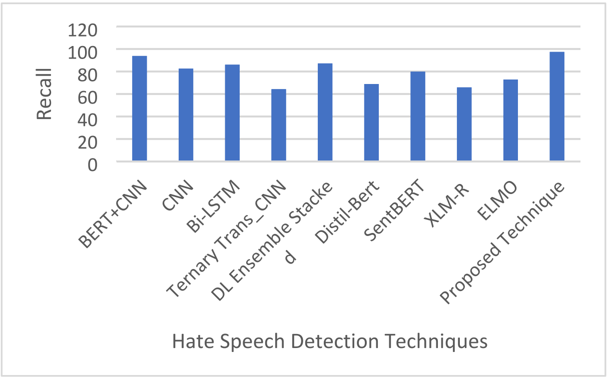
$$F1SCORE = \frac{2 * Precision * Recall}{Precision + Recall} \tag{11}$$

TP: True Positive, FP: False Positive, TN: True Negative, and FN: False Negative.

4.3.1 Evaluation of accuracy

In Fig. 6, the performance of the proposed technique based on accuracy is shown. The degree of achieving the standard level of the calculation is accuracy by using Eq. 8. From the result, it has been clear that the accuracy attained to 97.6% when it has been analyzed with the existing technique like BERT-CNN has 94% accuracy, CNN has 82.7% accuracy,

Fig. 7 Recall of the HateDetetctor technique with previously Implemented techniques



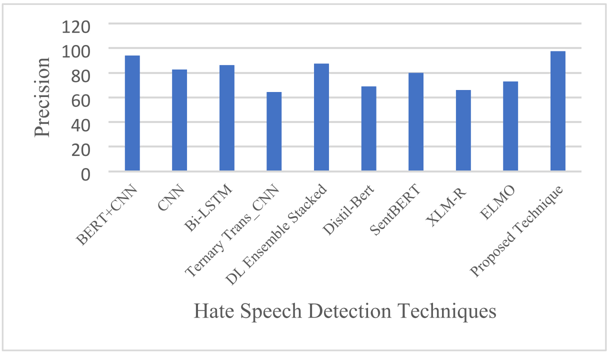


Fig. 8 Precision result of the HateDetetctor technique with previously Implemented techniques

Bi-LSTM has 86.2% accuracy, Ternary Trans-CNN has 75% accuracy, DL Ensemble Stacked has 87.3% accuracy, Disttil-BERT has 69%, SentiBERT has 80%, XLR-R has 66% and ELMO has 73% accuracy. On over-viewing it, the proposed technique reaches high in its accuracy and ELMO Technique was low in its accuracy.

4.3.2 Evaluation of recall

In Fig. 7 the recall evaluation of the proposed technique is shown. The capability to identify the positive tweets is recall. From the result, the recall value by using Eq. 10 of the proposed technique attained is 97.6% when it has been analyzed with the existing technique, BERT-CNN has 94%, CNN has 82.7%, Bi-LSTM has 86.2%, Ternary Trans-CNN has 64.4%, DL Ensemble Stacked has 87.3%, Disttil-BERT has 69%, SentiBERT has 80%, XLR-R has 66%and ELMO has 73%. On over-viewing it, the proposed technique performs high in recall and the Ternary Trans-CNN Technique is low in its recall.

4.3.3 Evaluation of precision

In Fig. 8, the Precision evaluation of the proposed technique is shown. The quality measure of positive samples is called precision. From the result, the precision value by using Eq. 9 of the proposed technique has been attained to 97.5% when it has been analyzed with the existing technique, BERT-CNN has 94%, CNN has 82.7%, Bi-LSTM has 86.2%, Ternary Trans-CNN has 64%, DL Ensemble Stacked has 87.4%, Disttil-BERT has 69%, SentiBERT

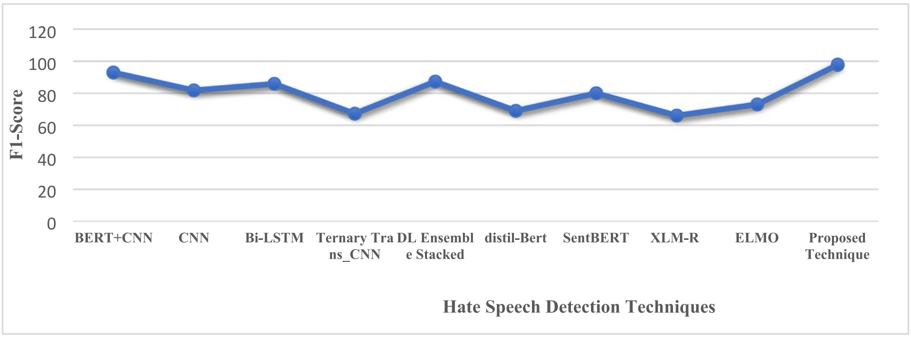


Fig. 9 F1-Score result of the HateDetetctor technique with previously Implemented techniques

Table 2 Comparison Analysis of the Accuracy of HateDetetctor technique with previously Implemented techniques

Hate Speech Models	Davidson dataset	Hinglish Hate dataset	Hot dataset
BERT + CNN [45]	94	94.6	95.6
CNN [46]	84.7	85.7	86.7
Bi-LSTM [47]	86.2	87.2	88.2
Ternary Trans_CNN [48]	84.4	86.4	87.4
DL Ensemble Stacked [49]	87.4	85.4	86.4
Distil-Bert [50]	69	69.1	69.5
SentBERT [51]	80	80.3	80.2
XLM-R [52]	66	66.2	66
ELMO [53]	73	75	78
Proposed Technique: HateDetector	**97.5**	**97.3**	**97.7**

has 80%, XLR-R has 66% and ELMO has 73%. The results show that the proposed technique gives high value of precision and the Ternary Trans-CNN Technique was low in its precision.

4.3.4 Evaluation of F1 score

Figure 9 shows the F1- Score result of the proposed technique, which is calculated using Eq. 11. The proposed technique has achieved 97.8% when it has been analyzed with the existing technique. The other technique like BERT-CNN has 94%, CNN has 82%, Bi-LSTM has 86%, Ternary Trans-CNN has 71%, DL Ensemble Stacked has 88%, Disttil-BERT has 69%, SentiBERT has 80%, XLR-R has 66% and ELMO has 73% on overviewing it, proposed achieved high in F1-Score performance and Ternary Trans-CNN Technique is low in its F1-Score.

4.4 Comparative analysis of classifiers performance

This subsection compares three different types of data sets with proposed and existing Techniques BERT-CNN, CNN, Bi-LSTM, Ternary Trans-CNN, DL Ensemble Stacked, and ELMO.

From Table 2, it shows the comparative analysis of proposed and existing techniques based on three different types of data sets: the Davidson Dataset, the Hinglish Hate Dataset, and the HOT dataset. The three types of datasets maintain the same level of accuracy in the proposed technique when compared with the other existing technique. Each technique taken for comparison has undergone certain deviations from its accuracy value for all three data sets. Here too, the accuracy does not change. The proposed technique has achieved high in its accuracy without any such deviations and the Ternary Trans-CNN and ELMO techniques have undergone more deviations when compared with others.

In Fig. 10 the comparison analysis of proposed and existing technique was viewed based on the above said three data set. The three types of datasets maintain the same level of

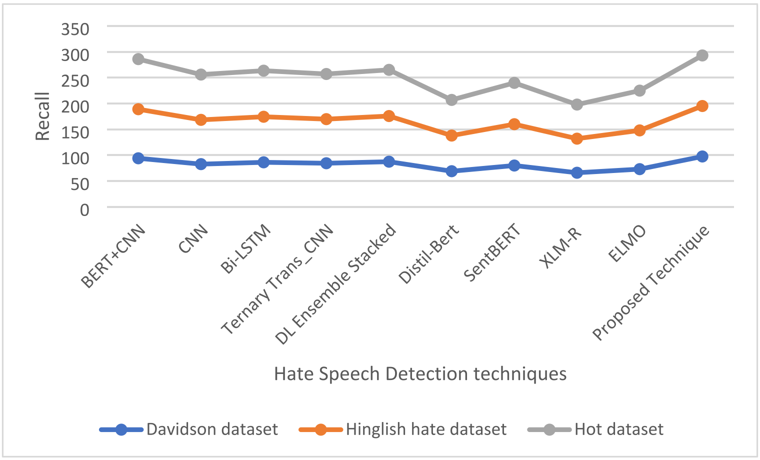


Fig. 10 Comparison Analysis of Recall of HateDetetctor technique with previous Implemented techniques

recall in the proposed when compared with the other existing technique. The proposed has achieved high recall without any such deviations and the CNN, and Bi-LSTM techniques have undergone more deviations when compared with others.

In Fig. 11 the proposed and existing technique is compared with three datasets. The proposed technique has achieved high precision too and all other techniques have highly deviated for all three data sets.

From Fig. 12 it has been clear that the proposed technique does not vary for any dataset given. The output result shows that all the three data set that works under this proposed brings out the output with a high-performance range. All technique has high deviations and Bi-LSTM, Ternary Trans-CNN, and DL Ensemble Stacked technique has equal deviations.

Thus, comparing all the parameters of the proposed technique for all three datasets, it attained the utmost high performance. The proposed technique shows an accuracy level of 97.9%, precision 97.9%, recall 98%and F1-score 97.8%. On comparing with the existing technique, namely Bidirectional Encoder Representation from

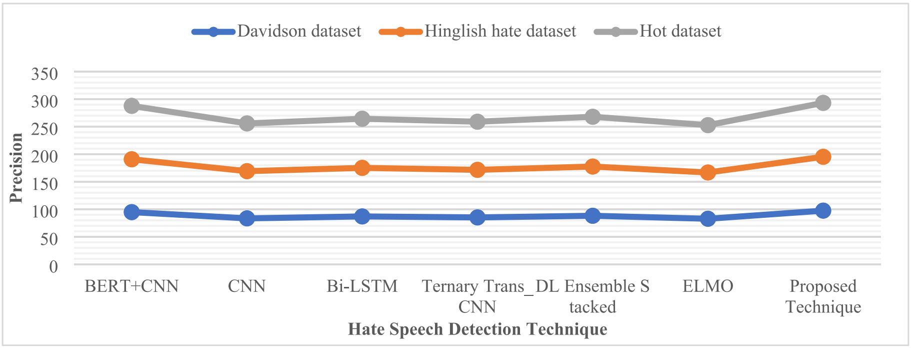


Fig. 11 Comparison Analysis of Precision

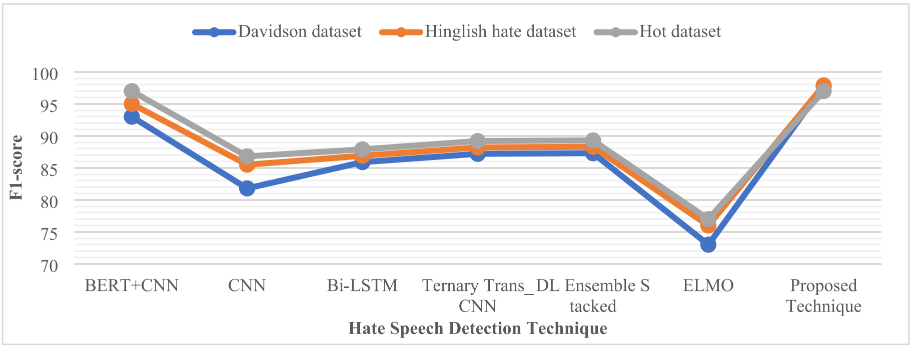


Fig. 12 Comparison Analysis of F1-Score of HateDetetctor technique with previous Implemented techniques

Transformer- Convolutional Neutral Network (BERT-CNN), Convolutional Neutral Network (CNN), Bidirectional Long-Short Term Memory (Bi-LSTM), Ternary Trans-CNN, Deep Learning (DL) Ensemble Stacked, Embeddings from Language Model (ELMO) the proposed technique gives higher and efficient performance.

5 Conclusion and future work

We have worked on the Detection and Analysis of Multilingual OHS in Twitter Using Artificial Intelligence with our proposed techniques. Multilingual Hate Speech Detection and Profanity Check is new as the step-by-step performance of several methods or techniques is carried over within these two proposed techniques. The Hinglish Tweets are pre-processed using BERT and coded using mBART, weighted by numbers using BOW with N-gram, and spell-checked then finally, the emotion is analyzed using the Profanity check technique which comprises of ReLu, sigmoid function, and logistic regression and finally the hate speech is detected. Thus, it is concluded that the performance of the existing technique BERT-CNN, CNN, Bi-LSTM, Ternary Trans-CNN, DL Ensemble Stacked, Disttil-BERT, SentiBERT, XLR-R and ELMO has been taken as a reference in performance evaluation and comparison was low, compared with the performance of the proposed which is ultimately high. This perfectly removes the code-mixing, with fast processing capability, is easy to train, and is highly reliable with a top performance of 97.9% of accuracy, precision, and F1-Score and 98% recall. For future work, the unlabeled data should be examined for the unsupervised machine learning model as labeling data is a very time-consuming task. Therefore, further study of the deep learning model is essential and advantageous to address hate speech problems.

Author Contribution Ms. Anjum

Dr. Rahul Katarya (Corresponding Author)

1. Conceived and designed the analysis
2. Collected the data
3. Contributed data or analysis tools
4. Performed analysis
5. Wrote the paper
6. All other necessary contributions related to this paper

Data availability Data sharing not applicable – no new data generated: Data sharing is not applicable to this article as no new data were created or analyzed in this study.

Declarations

Conflict of interest There is "NO" Conflict of interest for this manuscript and with any author.

References

1. Dinakar K, Reichart R, Lieberman H (2021) Modeling the Detection of Textual Cyberbullying. Proceedings of the International AAAI Conference on Web and Social Media 5(3):11–17. https://doi.org/10.1609/icwsm.v5i3.14209
2. Zhang Z, Luo L (2019) Hate speech detection: A solved problem? The challenging case of long tail on Twitter. Semant Web 10(2019):925–945
3. Pereira-Kohatsu JC, Quijano-Sánchez L, Liberatore F, Camacho-Collados M (2019) Detecting and monitoring hate speech in twitter. Sensors (Switzerland) 19(21):1–37. https://doi.org/10.3390/s19214654
4. Guiora A, Park EA (2017) Hate Speech on Social Media. Philos (United States) 45(3):957–971. https://doi.org/10.1007/s11406-017-9858-4
5. Corazza M, Menini S, Cabrio E, Tonelli S, Villata S (2020) A Multilingual Evaluation for Online Hate Speech Detection. ACM Trans Internet Technol 20(2). https://doi.org/10.1145/3377323
6. Ousidhoum N, Lin Z, Zhang H, Song Y, Yeung DY (2020) Multilingual and multi-aspect hate speech analysis. EMNLP-IJCNLP 2019 - 2019. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing Conference, pp 4675–4684. https://doi.org/10.18653/v1/d19-1474
7. Sreelakshmi K, Premjith B, Soman KP (2020) Detection of Hate Speech Text in Hindi-English Code-mixed Data. Procedia Comput Sci 171(2019):737–744. https://doi.org/10.1016/j.procs.2020.04.080
8. Mandl T (2020) Overview of the HASOC Track at FIRE 2020 : Hate Speech and Offensive Language Identification in Tamil, Malayalam, Hindi, English and German, pp 29–32. https://doi.org/10.1145/3368567.3368584
9. Mozafari M, Farahbakhsh R, Crespi N (2022) Cross-Lingual Few-Shot Hate Speech and Offensive Language Detection Using Meta Learning. IEEE Access 10:14880–14896. https://doi.org/10.1109/ACCESS.2022.3147588
10. Ridenhour M, Bagavathi A, Raisi E, Krishnan S (2020) Detecting Online Hate Speech: Approaches Using Weak Supervision and Network Embedding Models. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 12268 LNCS:202–212. https://doi.org/10.1007/978-3-030-61255-9_20
11. Wang L, Niu J, Yu S (2020) SentiDiff: Combining Textual Information and Sentiment Diffusion Patterns for Twitter Sentiment Analysis. IEEE Trans Knowl Data Eng 32(10):2026–2039. https://doi.org/10.1109/TKDE.2019.2913641
12. Alonso P, Saini R, Kovács G (2020) Hate Speech Detection Using Transformer Ensembles on the HASOC Dataset. In: Speech and Computer, pp 13–21
13. Wang Y, Huang G, Li J, Li H, Zhou Y, Jiang H (2021) Refined Global Word Embeddings Based on Sentiment Concept for Sentiment Analysis. IEEE Access 9:37075–37085. https://doi.org/10.1109/ACCESS.2021.3062654
14. Cao R, Lee RK-W, Hoang T (2020) DeepHate: Hate Speech Detection via Multi-Faceted Text Representations, pp 11–20. https://doi.org/10.1145/3394231.3397890
15. Mridha MF, Wadud MAH, Hamid MA, Monowar MM, Abdullah-Al-Wadud M, Alamri A (2021) L-Boost: Identifying Offensive Texts From Social Media Post in Bengali. IEEE Access 9:164681–164699. https://doi.org/10.1109/ACCESS.2021.3134154
16. Park JH, Fung P (2017) One-step and Two-step Classification for Abusive Language Detection on {T}witter. In: Proceedings of the First Workshop on Abusive Language Online, Aug., pp. 41–45. https://doi.org/10.18653/v1/W17-3006
17. Paschalides D, Stephanidis D, Andreou A, Orphanou K, Pallis G, Dikaiakos MD, Markatos E (2020) MANDOLA: A Big-Data Processing and Visualization Platform for Monitoring and Detecting Online Hate Speech. ACM Trans Internet Technol. https://doi.org/10.1145/3371276

18. Modha S, Majumder P, Mandl T, Mandalia C (2020) Detecting and visualizing hate speech in social media: A cyber Watchdog for surveillance. Expert Syst Appl 161:113725. https://doi.org/10.1016/j.eswa.2020.113725
19. Roy PK, Tripathy AK, Das TK, Gao X-Z (2020) A Framework for Hate Speech Detection Using Deep Convolutional Neural Network. IEEE Access 8:204951–204962. https://doi.org/10.1109/ACCESS.2020.3037073
20. Maxime (2019) What is a Transformer?No Title. Medium. https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04
21. Horev R (2021) BERT Explained: State of the art language model for NLP Title. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270
22. Mozafari M, Farahbakhsh R, Crespi N (2020) A BERT-Based Transfer Learning Approach for Hate Speech Detection in Online Social Media. Stud Comput Intell 881 SCI:928–940. https://doi.org/10.1007/978-3-030-36687-2_77
23. Mutanga RT, Naicker N, Olugbara OO (2020) Hate speech detection in twitter using transformer methods. Int J Adv Comput Sci Appl 11(9):614–620. https://doi.org/10.14569/IJACSA.2020.0110972
24. Qian J, Niu Z, Shi C (2018) Sentiment analysis model on weather related tweets with deep neural network. ACM Int Conf Proc Ser 31–35. https://doi.org/10.1145/3195106.3195111
25. Plaza-Del-Arco FM, Molina-González MD, Ureña-López LA, Martín-Valdivia MT (2020) Detecting Misogyny and Xenophobia in Spanish Tweets Using Language Technologies. ACM Trans Internet Technol 20(2). https://doi.org/10.1145/3369869
26. Wang G, Wang B, Wang T, Nika A, Zheng H, Zhao BY (2014) Whispers in the dark: Analysis of an anonymous social network. In: Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC, pp 137–149. https://doi.org/10.1145/2663716.2663728
27. Qureshi KA, Sabih M (2021) Un-Compromised Credibility: Social Media Based Multi-Class Hate Speech Classification for Text. IEEE Access 9:109465–109477. https://doi.org/10.1109/ACCESS.2021.3101977
28. Kapil P, Ekbal A (2020) A deep neural network based multi-task learning approach to hate speech detection. Knowl-Based Syst 210. https://doi.org/10.1016/j.knosys.2020.106458
29. Corazza M, Menini S, Cabrio E, Tonelli S, Villata S (2020) A Multilingual Evaluation for Online Hate Speech Detection. ACM Trans Internet Technol 20(2):1–22. https://doi.org/10.1145/3377323
30. Greevy E, Smeaton AF (2004) Classifying racist texts using a support vector machine. In: Proceedings of Sheffield SIGIR - Twenty-Seventh Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 468–469. https://doi.org/10.1145/1008992.1009074
31. Agarwal S, Sureka A (2017) But i did not mean it! - Intent classification of racist posts on tumblr. In: Proceedings - 2016 European Intelligence and Security Informatics Conference, EISIC 2016, pp 124–127. https://doi.org/10.1109/EISIC.2016.032
32. Davidson T, Bhattacharya D, Weber I (2019) Racial Bias in Hate Speech and Abusive Language Detection Datasets. In Proceedings of the Third Workshop on Abusive Language Online, Florence, Italy. Association for Computational Linguistics, pp 25–35
33. Miró-Llinares F, Moneva A, Esteve M (2018) Hate is in the air! But where? Introducing an algorithm to detect hate speech in digital microenvironments. Crime Sci 7(1):1–12. https://doi.org/10.1186/s40163-018-0089-1
34. Paetzold GH, Malmasi S, Zampieri M (2019) UTFPR at SemEval-2019 Task 5: Hate Speech Identification with Recurrent Neural Networks. International Workshop on Semantic Evaluation
35. Gambäck B, Sikdar UK (2017) Using Convolutional Neural Networks to Classify Hate-Speech. In Proceedings of the First Workshop on Abusive Language Online, Vancouver, BC, Canada. Association for Computational Linguistics, pp 85–90
36. Liu Y et al (2020) Multilingual Denoising Pre-training for Neural Machine Translation. Trans Assoc Comput Linguist 8:726–742. https://doi.org/10.1162/tacl_a_00343
37. Liu Y, Gu J, Goyal N, Li X, Edunov S, Ghazvininejad M, Lewis M, Zettlemoyer L (2020) Multilingual Denoising Pre-training for Neural Machine Translation. Trans Assoc Comput Linguistics 8:726–742
38. Bloomberg M (n.d.) Google, “Python programming language (version 3.6).” https://www.python.org/downloads/
39. Pedregosa F et al (2011) Scikit-learn: Machine Learning in Python. J Mach Learn Res 12:2825–2830
40. Bisong E (2019) Google Colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform. Apress, Berkeley, CA, pp. 59–64. https://doi.org/10.1007/978-1-4842-4470-8_7
41. Davidson T, Warmsley D, Macy M, Weber I (2017) Automated Hate Speech Detection and the Problem of Offensive Language. Proceedings of the International AAAI Conference on Web and Social Media. https://doi.org/10.1609/icwsm.v11i1.14955
42. Bohra A, Vijay D, Singh V, Akhtar SS, Shrivastava M (2018) A Dataset of Hindi-English Code-Mixed Social Media Text for Hate Speech Detection. In Proceedings of the Second Workshop on Computational

Modeling of People's Opinions, Personality, and Emotions in Social Media, New Orleans, Louisiana, USA. Association for Computational Linguistics, pp 36–41

43. Mathur P, Sawhney R, Ayyar M, Shah R (2018) Did you offend me? Classification of Offensive Tweets in {H}inglish Language. In: Proceedings of the 2nd Workshop on Abusive Language Online ({ALW}2), pp 138–148. https://doi.org/10.18653/v1/W18-5118
44. Omar A, Mahmoud TM, Abd-El-Hafeez T (2020) Comparative Performance of Machine Learning and Deep Learning Algorithms for Arabic Hate Speech Detection in OSNs. Adv Intell Syst Comput 1153 AISC:247–257. https://doi.org/10.1007/978-3-030-44289-7_24
45. Zhou Y, Yang Y, Liu H, Liu X, Savage N (2020) Deep Learning Based Fusion Approach for Hate Speech Detection. IEEE Access 8:128923–128929. https://doi.org/10.1109/ACCESS.2020.3009244
46. Chaudhari A, Parseja A, Patyal A (2020) CNN based Hate-o-Meter: A Hate Speech Detecting Tool. In: 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), pp 940–944. https://doi.org/10.1109/ICSSIT48917.2020.9214247
47. Jain R, Goel D, Sahu P, Kumar A, Singh JP (2021) Profiling Hate Speech Spreaders on Twitter.
48. Mathur P, Shah RR, Sawhney R, Mahata D (2018) Detecting Offensive Tweets in Hindi-English Code-Switched Language. Proc Ann Meet Assoc Comput Linguist 18–26. https://doi.org/10.18653/v1/w18-3504
49. Mazari AC, Boudoukhani N, Djeffal A (2023) BERT-based ensemble learning for multi-aspect hate speech detection. Cluster Comput 0123456789. https://doi.org/10.1007/s10586-022-03956-x
50. Ali R, Farooq U, Arshad U, Shahzad W, Beg MO (2022) Hate speech detection on Twitter using transfer learning. Comput Speech Lang 74:101365. https://doi.org/10.1016/j.csl.2022.101365
51. Madhu H, Satapara S, Modha S, Mandl T, Majumder P (2023) Detecting offensive speech in conversational code-mixed dialogue on social media: A contextual dataset and benchmark experiments. Expert Syst Appl 215:119342. https://doi.org/10.1016/j.eswa.2022.119342
52. Ryzhova A, Devyatkin D, Volkov S, Budzko V (2022) Training Multilingual and Adversarial Attack-Robust Models for Hate Detection on Social Media. Procedia Comput Sci 213:196–202. https://doi.org/10.1016/j.procs.2022.11.056
53. STUFIIT at SemEval-2019 Task 5: Multilingual Hate Speech Detection on Twitter with MUSE and ELMo Embeddings. In Proceedings of the 13th International Workshop on Semantic Evaluation, pp. 464–468, Minneapolis, Minnesota, USA. Association for Computational Linguistics. https://doi.org/10.18653/v1/s19-2082