Hate Speech Detection in Roman Urdu English Tweets Through Data Pre-processing

Muhammad Asif Khan
Technology Donghua University
Jazib e nazar
Jazibenazar@cuiatd.edu.pk
Comsats University
GuohHua Liu
Technology Donghua University

Article

Keywords: code-mixed, code-switched, neural networks, speech detection, hatred speech, multilingual

Posted Date: May 7th, 2025

DOI: https://doi.org/10.21203/rs.3.rs-6345769/v1

Additional Declarations: No competing interests reported.

Hate Speech Detection in Roman Urdu English Tweets Through Data Pre-processing

Muhammad Asif Khan
School of Computer Science and Technology Donghua University, Shanghai
asif@mail.dhu.edu.cn

Jazibe e Nazar*
Department of Computer Science, COMSATS University Islamabad, Abbottabad Campus, Abbottabad, Pakistan
jazibenazar@cuiatd.edu.pk

GuoHua Liu
School of Computer Science and Technology Donghua University, Shanghai
ghliu@dhu.edu.cn

Abstract - Hate speech detection enhances internet safety by recognizing and reducing harmful or objectionable information. The increasing use of social media has made it more difficult to control and moderate hate speech. The casual and varied nature of material on Twitter presents a particular difficulty for hate speech identification because of its diversified and multilingual user base that includes code-mixed languages like Roman Urdu-English. To tackle the issue of Hate Speech in code mixed Roman Urdu-English little amount of research has been done by the NLP and machine learning community. To solve this problem, this article looks at how data pre-processing affects the ability to identify hate speech in tweets that combine Roman Urdu and English codes. We used a comprehensive 10-step data cleaning procedure followed by the Multilingual BERT (mBERT) model for Hate Speech detection. The methodology includes optimizing hyper parameters and carrying out comprehensive tests to evaluate model's accuracy. The results showed the proposed data preprocessing approach considerably increases accuracy. Compared to previous techniques, the mBERT model showed about 9.12% gain in accuracy. This demonstrates how well our pre-processing methods work and how powerful mBERT is in enhancing hate speech detection on social media networks.

Keywords- code-mixed, code-switched, neural networks, speech detection, hatred speech, multilingual

1. INTRODUCTION

Internet usage has increased dramatically over the last several years around the world. This is primarily due to the rapid rise in popularity, attraction, and addiction of social media websites [1]. Twitter is one of the most widely utilized social media sites, generating a lot of online content and massive amounts of data. Users can connect more quickly and easily on these platforms, sharing and expressing their thoughts freely by means of photos, videos, or messages/tweets. In addition to the positive aspects mentioned previously, social media also has some negative aspects. An increasing number of people are publicly using hate speech and insulting language on social media because more people are using it and becoming addicted to it [1].

According to Brown [2] , hate speech is defined as any word or expression that is meant to insult, humiliate, or encourage violence against a person or group based on characteristics such as religious belief, ethnicity, sexual orientation, disability, or gender. This definition highlights the intent behind the speech as a critical factor in classifying it as hate speech. Online anonymity enables some people to engage in abusive conduct that they are less likely to show in person, including posting hateful remarks aimed at certain groups or individuals. Hate material leads to cyberviolence and crimes, damaging

mental health and life. Many platforms prohibit hate speech, but managing communication is becoming more and harder because of the volume of material that hostile individuals create.

To prevent any potential for actual violence to occur because of hate speech, it is of foremost significance that the government and social media platforms conduct investigations, identify instances of such content, and remove it as quickly as possible [3].

The detection of hate speech is so vital that an Indian leader called for quick approval of laws to prevent hate speech that had been spreading online [4] . Many countries, like India, have put strict rules in place to stop hate speech online [5], [6]. Even with its efforts, Twitter still faces criticism for not doing enough to tackle hate speech [7]. In this work, we address the important issue of identifying hate speech on social media, specifically focusing on short text messages on Twitter a public worldwide social media platform where users may exchange quick messages, known as tweets, with up to 280 characters. People of many countries and backgrounds utilize it. Given its wide user base from various backgrounds, some tweets mix languages. Code-mixing happens when elements from one language are mixed into another's text [8].

To illustrate, examples 1 and 2 show one tweet with hate speech and one without, both in code-mixed text and their English translations.

Example 1 (hate tweet in code-mixed Roman Urdu –English). The following tweet, written in Roman Urdu with English letters, expresses negativity towards a political leader

"Yeh leader kuch nahi kar sakta, sab kuch bekaar hai, log isse hate karte hain."

Translation:

"This leader can't do anything, everything is useless, people hate him."

Example 2 (non-hate tweet in code-mixed Roman Urdu –English). The following non-hateful tweet, written in Roman Urdu with English letters, comments on a new movie release:

"Film to acchi thi, lekin thoda zyada drama tha. Still, sabko dekhni chahiye."

Translation:

"The movie was good, but it had a bit too much drama. Still, everyone should watch it."

In Pakistan and India's diverse and multilingual landscape, people frequently mix Roman Urdu and English when tweeting[9]. This code-mixed language often comes with informal expressions and varied slang and spellings [8]. Because of this mix, detecting hate speech in such texts can be quite challenging. Therefore, we need a robust software solution to automatically identify and manage hateful content across different languages, ensuring the safety and well-being of users. This study focuses on hate speech detection in tweets that mix English and Roman Urdu. While there has been extensive research on detecting hate speech in single languages such as English [10, 11] and code-mixed languages like Roman Urdu-English [8, 12, 13], the detection of hate speech in multilingual contexts has received less attention. Figure 1 illustrate the distribution of Hate speech in English and code mixed Roman Urdu tweets.

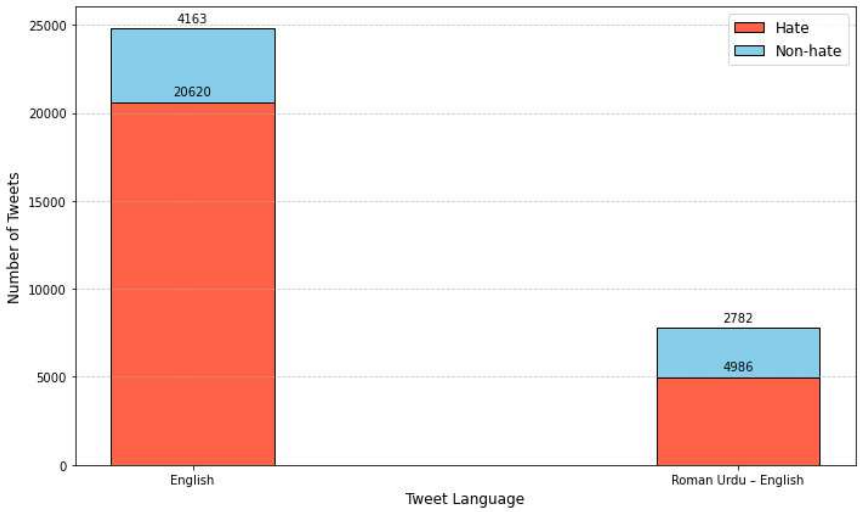

FIGURE 1. DISTRIBUTION OF HATE AND NON-HATE TWEETS BY LANGUAGE

- ***Contribution***

The main contribution of this work is to investigate how data preparation influences hate speech detection in a multiple language's context. We measure this impact by considering prediction accuracy, which determines precisely how tweets are categorized.

As far as we know, the only study that explored hate speech detection across multiple languages, including both English and code-mixed Roman Urdu-English, was conducted by Elouali et al. [7]. In this work, we present a more effective approach to precisely identify hate speech in English and Roman Urdu -English tweets. Unlike [7]., our method consists of a complete 10-step data purification process followed by the usage of multilingual BERT to create the hate speech prediction model.

2. RELATED WORK

Over the past decade, numerous studies have been carried out to detect harmful communication on social media, such as violent material, hate speech, and Cyberbullying. The goal of these initiatives has been to comprehend and minimize dangerous internet behavior [14]. Most of the existing research has concentrated on detecting toxic speech in texts written primarily in English, with limited focus on other languages. However, hate speech and offensive content are not confined to any single language, it's a global issue that affects all languages, including code-mixed and monolingual texts. In this study, we focus on short texts, particularly tweets from Twitter. To better organize the various approaches for toxic speech detection, we categorize them into three main groups: research on detecting toxic speech in English, studies targeting Roman Urdu-English code-mixed text, and research addressing multilingual content (texts written in multiple languages)

I. Detection of Toxic Speech in English Text

Numerous techniques have been developed by researchers to identify harmful content in English texts. Some studies focused solely on detecting hate speech [10,11,15] [16,17,18, 19] others on offensive language [20-22], and yet others on both [22-23]. There have been several approaches taken, such as models based on deep learning (DL) [11, 17, 20,21, 23] and classifiers based on machine learning (ML) [10, 15, 16, 20 -23].

To detect hate speech directed towards the Black community on Twitter, Kwok and Wang [10] created a naïve Bayes classifier using bag-of-words, which they were able to detect with 76% accuracy.

However, they found that inappropriate language led to the labeling of 86% of tweets as racist. This exposed a weakness in their approach by only looking at individual words; the model sometimes misjudged tweets, flagging non-hateful ones just because they had offensive words. Hate speech isn't always obvious and can be much more subtle, so identifying it requires a deeper understanding beyond just spotting offensive language.

Critical race theory was used by Waseem and Hovy [20] to create a logistic regression model that considered extra features including demographic and geographic information found in tweets. Although the accuracy of the predictions was enhanced by these demographic variables, it is frequently difficult to get or rely upon such information on Twitter. To prevent social media restrictions from being triggered, Magu et al. [16] developed a machine learning algorithm to identify racist tweets that contained hate code words. They used a support vector machine (SVM) classifier to distinguish between racist tweets that utilized code words deliberately and non-racist tweets that used those words in a normal context. Additionally, it was utilized to keep an eye on user accounts sending hateful tweets and examine the patterns of their tweeting.

Davidson et al. [22] introduced a machine learning approach to classify tweets into three groups: hate speech, offensive language, or neither. Their multi-class classification resulted in a 91% precision, 0.90% recall, and 90% F1-score using logistic regression with L2 regularization. Their study found that most racist and homophobic tweets were classified as hateful, racist tweets were often labeled as offensive, and the hardest tweets to classify were those that didn't contain clear offensive language.

Watanabe et al. [23] identified hate speech and offensive language using a combination of features including unigrams, patterns, sentiment, and semantics. Using the WEKA toolkit, they created the J48graft method, which achieved a classification accuracy of 78.4% for distinguishing among three categories and 87.4% for binary classification, separating tweets into offensive or clean categories. To prevent social media restrictions from being triggered, Magu et al. [16] developed a machine learning algorithm to identify racist tweets that contained hate code phrases. They developed a support vector machine (SVM) classifier to differentiate between racist tweets that intentionally used code phrases and non-racist tweets that used those words in a regular context. The model achieved an accuracy of 79.4%. Additionally, it was utilized to monitor and examine the tweeting patterns of user accounts that posted hateful messages.

Davidson et al. [22] proposed a machine learning approach to classify tweets into three groups: hate speech, offensive language, or neither. They achieved an F1-score of 90%, a recall of 0.90%, and a precision of 91% using logistic regression with L2 regularization for multi-class classification. Researchers observed that tweets with racist or homophobic content were predominantly classified as hateful, sexist tweets were often categorized as offensive, and tweets lacking obvious offensive language were the most difficult to categorize. Watanabe et al. [23] identified hate speech and offensive language using a combination of factors such as unigrams, patterns, sentiment, and semantics. They used the WEKA toolkit to create the J48graft method, which attained an accuracy of 87.4% for distinguishing between offensive and clean tweets, and 78.4% for categorizing tweets into three groups: hateful, offensive, or clean.

Zampieri et al. [25] presented machine learning and deep learning methods for classifying offensive content on Twitter and determining the offense's kind and target audience. They used the macro-averaged F1 score to assess and compare the effectiveness of SVM, Bi-directional LSTM (BiLSTM), and CNN models. In terms of classifying the offense, identifying the target, and finding offensive tweets, the CNN algorithm outperformed the others. De Souza and Da Costa-Abreu [22] concentrated on utilizing machine learning algorithms to identify inappropriate language in tweets. They developed

two classifiers: Naïve Bayes and Linear SVM (LSVM). The Naïve Bayes model achieved 92% accuracy with 95% recall, while the LSVM reached 90% accuracy with 92% recall. Compared to Naïve Bayes, which was easier to build and performed better, the LSVM required careful tuning due to its sensitivity to data type and standardization.
Recently, there have been advances in detecting hate speech that integrates both images and text [26,27]. While this is an important development in tackling aggressive content, we believe it differs from the specific challenge we're addressing, which centers solely on text-based hate speech detection.

II. Detection of Toxic Speech in Roman Urdu –English Code-Mixed Text

Researchers have used the identification of hate tweets [8,12,28–29] and aggressive tweets [13] to detect harmful content in code-mixed Roman Urdu-English text. These studies proposed classification methods based on machine learning (ML) and deep learning (DL) techniques [12, 30, 8, 13, 28].
Bohra et al. [12] recommended machine learning-based techniques for detecting hate speech in code-mixed Roman Urdu-English tweets. They developed classifiers using SVM and random forest (RF). SVM outperformed RF, achieving the highest accuracy of 71.7% when all features were used. Among individual features, character n-grams in SVM and word n-grams in RF yielded the best accuracy.
Kamble and Joshi [28] introduced deep learning-based classification methods for detecting hate speech in code-mixed Roman Urdu-English tweets on Twitter. They trained domain-specific word embeddings using a large sample of tweets targeting minority groups and evaluated their model using the 3849 tweets provided by Bohra et al. [12]. The use of domain-specific embeddings notably improved the identification of hate targets compared to generic embeddings. Compared to generic embeddings, domain-specific word embeddings greatly improved the identification of hate targets. Using the same baseline methods from Bohra et al. [12], SVM and RF classifiers were constructed. Additionally, the word embeddings were employed to build and compare three deep learning models: CNN-1D, LSTM, and BiLSTM. The BiLSTM model achieved the highest recall at 78.90%. On the other hand, CNN-1D excelled with the highest precision of 83.34%, the best F1-score of 80.85%, and the greatest accuracy of 82.62%. Thus, the findings demonstrate that when it came to identifying the semantics and hate speech settings, the DL models outperformed the statistical techniques.

Singh et al. [13] proposed both machine learning and deep learning classification algorithms to detect hostile speech in Facebook comments and posts that use mixed Roman Urdu and English codes. 12,000 Roman and Devanagari posts and comments from Facebook were utilized in the analysis. Kumar et al. [31] categorized the hostile speech into three distinct groups as part of their online collaborative research: Covertly Aggressive, Overtly Aggressive, and Non-Aggressive. By experimenting with various feature sets and tuning parameters, six classification models were developed: multimodal naïve Bayes, CNN, SVM, multilayer perceptron, decision tree, and LSTM. Among these, the CNN model demonstrated the highest performance, reaching an accuracy of 73.2%. When it comes to voice recognition, ML algorithms are not necessarily superior to neural networks. Because the postings and comments lacked sufficient diversity and variation in content, the models were unable to learn in an effective manner.
To detect hate speech in code-mixed Roman Urdu-English messages on Twitter, Santosh and Aravind [8] developed deep learning-based classification methods. The SVM and RF classifiers were developed using the baseline methods established by Bohra et al. [12]. Additionally, two models were created for comparison: the sub-word level LSTM model and the Hierarchical LSTM model with attention, which is based on phonemic sub-words. SVM was the most accurate at 70.7%, while Hierarchical LSTM was the most accurate at 45.1% recall and 48.7% F1-score. Sreelakshmi et al. [30] used Facebook's pre-trained fasttext embeddings as the feature matrix to classify tweets with SVM-Linear, SVM-Radial

Basis Function (RBF), and RF algorithms. They compared the results with word2vec and doc2vec features. The SVM-RBF classifier with fasttext embeddings delivered the best accuracy at 85.81%, whereas the word2vec embeddings achieved an accuracy of 75.11%. On the other hand, the doc2vec feature achieved 64.15% accuracy for the RF classification method. Compared to prior research that used document-level and word-level features for classification, this study surpassed them all by utilizing character-level characteristics in fasttext.

Research in this area offer novel and convincing methods for identifying harmful speech in text that has both Roman Urdu and English codes mixed throughout. The literature hasn't, however, gone into detail about detecting hate speech in several languages, including code-mixed Roman Urdu and English as well as other languages.

III. Detection of Toxic Speech in Multilingual Text

Kumari et al. [32] developed a technique for identifying violent content in bilingual text, focusing on English and Roman Urdu. However, their approach did not cover code-mixed text. Elouali et al. [7] sed a deep learning-based classification system to detect abusive content in multilingual text, leveraging a hate tweet dataset from Twitter. A deep learning method based on neural networks was introduced Elouali et al. [7] to identify hate speech in tweets written in seven distinct languages. They built their multilingual dataset by merging existing datasets from several languages, allowing them to address hate speech in a more diversified linguistic environment. The data was organized into two versions: the first version contained 33,727 tweets in Arabic, Italian, Portuguese, Indonesian, and English. The second version included an additional 12,446 tweets, which were in Roman Urdu-English and German, along with the tweets from the first version. To determine the most effective architecture, a CNN model with character-level embeddings was constructed, and a series of experiments were conducted by varying the parameters. The dataset's first and second versions achieved the best accuracy rates, with 88.93% and 83%, respectively. When compared to earlier research on separate datasets, a single CNN model performed better at identifying hate speech in a variety of languages.

As far as current research indicates, Elouali et al.'s study [7] is unique in its focus on multiple languages for identifying hate speech, whereas the other studies mentioned address only a single language. A solution to an issue that is closely analogous to ours was put up by Elouali et al. [7], however their technique took a very lengthy time to implement.

This work utilizes neural network architectures to detect hate speech in code-mixed tweets, which combine Roman Urdu-English and English, often in short text formats. Our proposed approach effectively identifies hate speech across these multilingual contexts. The studies included in this section are summed together in **Error! Reference source not found.**.

TABLE 1. COMPARISON TABLE

Study By	Model Used	English	Roman Urdu – English	Multilingual
Kwok and Wang [10]	Naïve Bayes	yes		
Waseem and Hovy [20]	Logistic regression	yes		
Magu et al. [16]	SVM	yes		

Davidson et al. [22]	Logistic regression	yes		
Watanabe et al. [23]	Decision Tree	yes		
Zampieri et al. [25]	SVM, BiLSTM, CNN	yes		
De Souza and Da Costa-Abreu [21]	Linear SVM, naïve Bayes	yes		
Bohra et al. [12]	SVM, random forest		yes	
Kamble and Joshi [28]	SVM, random forest, LSTM, BiLSTM		yes	
Singh et al. [13]	Naïve Bayes, decision tree, SVM, multilayer perceptron, LSTM, CNN		yes	
Santosh and Aravind [8]	SVM, random forest, LSTM		yes	
Kumari et al. [32]	LSTM		yes	yes
Elouali et al. [7]	CNN	yes	yes	yes
Our proposed work	mbert	yes	yes	yes

DATASET

For this study, we required a dataset containing both English and Roman Urdu-English (code-mixed) tweets, each labeled either hate or non-hate. Unfortunately, we could not discover a publicly accessible dataset that satisfied these requirements. To guarantee that our experiments could be reused, we chose highly recognized datasets from prior studies [12,24,22]. As shown below, tweets from three distinct sources were combined to create our final dataset:

1) The first segment of our dataset is derived from the dataset used by Davidson et al. [23], which includes English tweets labeled as "hate," "offensive," or "neither." For our study, we simplified the labels into two categories: "hate" (combining "hate" and "offensive") and "non-hate" (reclassifying "neither"). After this adjustment, we had 24,783 English tweets, with 20,620 labeled as hate and 4,163 as non-hate.
2) The second section comes from Bohra et al. [12]. The dataset also includes tweets written in a mix of Roman Urdu and English, with each tweet categorized as either "hate" or "non-hate." There are 4,579 tweets total, with 2,918 being non-hate and 1,661 being hateful.
3) The third portion of our dataset originates from Mathur et al. [24]. Their dataset consists of mixed Roman Urdu and English tweets labeled as "abusive," "non-offensive," or "hate-inducing." We classed "hate-inducing" and "abusive" as hate and "non-offensive" as non-hate to conform to our two-class system. After making this modification, we were left with a dataset of 3,189 code-mixed Roman Urdu -English tweets, of which 2,068 were classified as hate and 1,121 as non-hate.

Upon aggregating all components, our final dataset consists of two columns: "Label" (indicating either non-hate or hate) and "Tweet" (indicating text content). There are 32,551 tweets, with 8,202 classified

as non-hate and 24,349 as hate tweets. The resulting dataset and its class labels are summarized in **Error! Reference source not found.**.

TABLE 2. DATASETS

Tweet Language	**Label**	**Number of Tweets**
English	Hate	20,620
	Non-hate	4,163
Roman Urdu –English	Hate	2,918
	Non-hate	1,661
Roman Urdu –English	Hate	2,068
	Non-hate	1,121
Total	**32,551**	

A tweet, indicated by the letter t, is a sequence of letters that can hold up to 280 characters. A class label with the words "hate" and "non-hate." The tweet t and its associated class label c are part of an ordered collection, represented by the dataset record r == {r1, r2… rn}, where n = |D| indicates the total number of entries in D. The dataset D is composed of a collection of records, which are represented as D.

3. PROPOSED METHODOLOGY

In this section, we describe how we aim to distinguish both hate and non-hate tweets. In order for preparing the data for the proposed machine learning model, it starts through the data pre-processing stages applied.

The dataset's partitioning into training and test sets will be discussed next, followed by how we used resampling to address any data imbalances. The neural network designs we've suggested will be discussed last. These phases and components are summarized in Figure 2.

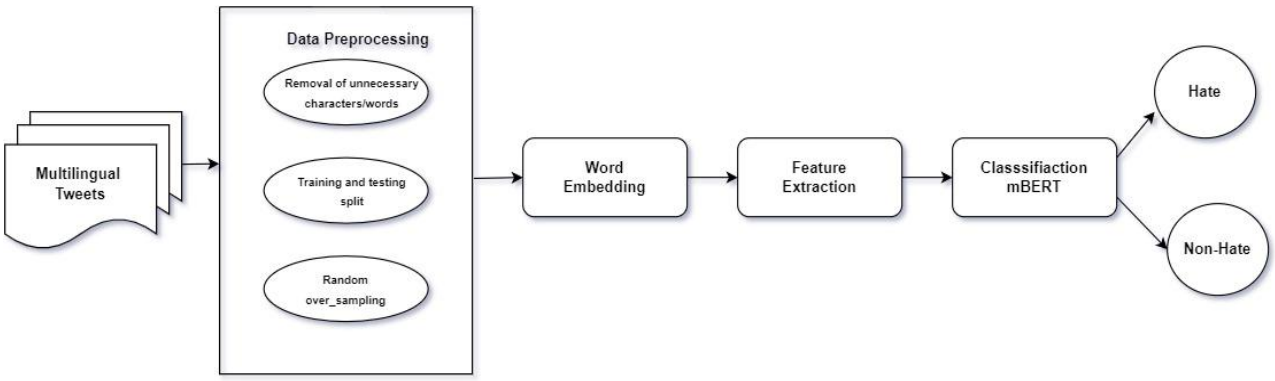

FIGURE 2 DATA PRE PROCESSING PHASES

I. Cleaning (Pre-Processing)

We're focusing on classifying short text on social media, especially tweets. Since these messages are often informal, they can include a lot of extra stuff like mentions ,emoticons , special characters (e.g., @, +), punctuation marks (e.g., ?, !), and URLs (e.g., http://t.co/hH50P5pytX). We found that these elements aren't useful for detecting hate speech, as confirmed by our experiments. So, we've implemented the following pre-processing steps for all tweets:

1) Decoding HTML: We convert any HTML entities back to their original text form. Some entities, like "&amp", don't decode properly and are removed.
2) Removing Emoticons: Emoticons encoded in UTF-8 BOM are decoded and then removed.
3) Removing Mentions: Mentions are just usernames tagged in tweets and don't add to the sentiment, so we remove them.
4) Removing URLs: All URLs and links (including those to images or videos) are removed since they don't affect sentiment.

5) Converting to Lowercase: Tweets are converted to lowercase to standardize the text for processing.
6) Expanding Negations: Apostrophes are removed, and negative contractions like "can't" are expanded to "cannot" to preserve their meaning.
7) Removing Hashtags, Punctuation, Numbers, and Special Characters: We retain the useful information from hashtags but remove the "#" symbol, along with other special characters and punctuation.
8) Removing Extra Whitespaces: Any extra spaces in the tweets are cleaned up.
9) Removing Duplicates: We eliminate duplicate tweets to ensure each one is unique and avoid overfitting. We also keep an eye for duplicates that might be mislabeled and make sure to correct or remove any errors.
10) Removing Null Values: Tweets that end up as empty strings after pre-processing are removed. For tweets that turn into empty strings after pre-processing, we simply remove them.

After the cleaning process, our dataset consists of 31,456 tweets, with 7,903 labeled as non-hate and 23,553 as hate. This imbalance in the number of labels creates a skewed dataset, where the distribution of hate and non-hate tweets is not equal.

II. Training and Test Sets Split

The dataset was partitioned into two sections using stratified sampling to guarantee that each set contains an equitable distribution of labels. 80% of the data was allocated for the model's training, while

20% was reserved for its performance assessment. There are 25,164 records in our training set, whereas the test set contains 6,292 records.

III. Handling Data Imbalance

Standard classification methods often overlook the minority class and favor the majority class due to the imbalance in the dataset. This can result in a skewed model by treating attributes of the minority class as unimportant noise and producing predictions that are biased towards the majority class [33]. For this imbalance, we employ resampling strategies such as under sampling and oversampling [34, 35]. For the training set in our instance, we concentrated on random over-sampling. To maintain equal representation for both groups and balance the dataset, records from the minority class ("non-hate") are replicated. To avoid overfitting and maintain the test set's integrity as an accurate representation of performance in the actual world, we merely oversampled the training set. Following this procedure, the training set has the same amount of data for every class that is, 37,684 records altogether, with 18,842 records for each class label.

IV. Proposed Model Architecture

Our proposed method for the detection of hate speech uses state-of-the-art classification techniques. Particularly, we use a transformer-based architecture known as the multilingual BERT (mBERT) model, which has demonstrated higher accuracy in multilingual natural language processing.

Multilingual BERT (mBERT) is a transformer-based model that excels at understanding context within text, which is crucial for tasks like classifying hate speech. Unlike older models that focus on individual characters or whole words, mBERT works at the sub word level. This means it can handle language nuances more effectively—like recognizing parts of words when faced with new or uncommon terms. For a multilingual dataset like ours, this feature is particularly beneficial, as it helps bridge the gap between different languages and dialects.

To prepare the tweets for analysis, we use mBERT's tokenizer, which breaks down each tweet into sub word units (tokens). This tokenizer is incredibly useful for dealing with misspelled words or words that might not have been seen before by the model. Instead of trying to match entire words, mBERT can understand smaller parts of the words, ensuring it captures the meaning even when spelling variations or new words are involved. Each tweet is then converted into a sequence of these subword tokens. Since Twitter has a 280-character limit, we keep the tokenized tweets within this boundary, padding shorter tweets so that all inputs have a consistent length of 280 tokens.

For Example: Let’s take the tweet "you are amazing" as an example. mBERT breaks it down into subword tokens as follows:

[‘you’, ‘are’, ‘am’, ‘##azing’]

Here’s what happens:

- The word "you" and "are" are recognized as complete tokens.
- The word "amazing" is split into two subword tokens: "am" and "##azing."

Each subword token is then converted into a corresponding index from mBERT’s vocabulary. Suppose the following is the index representation:

{‘you’: 100, ‘are’: 203, ‘am’: 345, ‘##azing’: 789}

The tweet "you are amazing" would then be represented by the sequence:

[100, 203, 345, 789]

If the tweet length is shorter than the maximum length allowed (280 characters), it is padded with zeros to maintain uniformity across the dataset:

[100, 203, 345, 789, 0, 0, ..., 0]

Each subword token is represented by a dense vector that captures the token's meaning in the context of its surrounding tokens. For example, the subword might be represented as a vector like:

[0.12, 0.43, 0.56, ..., 0.08]

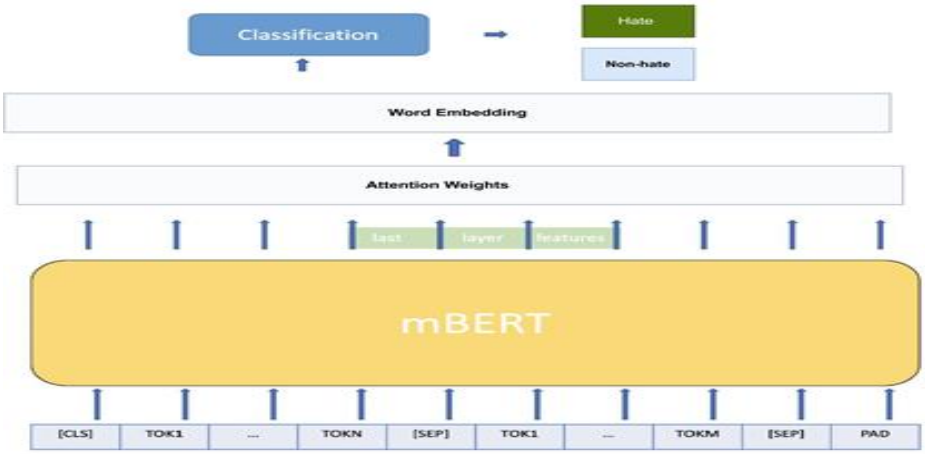

FIGURE 3. ARCHITECTURE OF FINE-TUNING MBER

Figure 3 illustrates Fine-tuning of mBERT architecture. The model's key components include:

1) Input Layer: This accepts the tweet as a sequence of up to 280 tokens.
2) BERT Embedding Layer: Each token gets transformed into a dense vector (embedding), which mBERT uses to understand its meaning within the tweet's context.
3) Transformer Layers: These layers, the heart of mBERT, use attention mechanisms to understand the relationships between words in the tweet, even if they're far apart in the sentence.
4) Dense Layers: After mBERT processes the tweet, we use two dense layers (with 1024 neurons each) to further refine the model's understanding of the tweet.
5) Dropout Layers: To prevent the model from overfitting, we add dropout layers, which randomly "drop" some neurons during training to make the model more robust.
6) Output Layer: This is where the final decision is made, classifying whether the tweet contains hate speech.

This method allows us to leverage mBERT's ability to understand multiple languages and capture the nuances of hate speech across different linguistic contexts. The fine-tuning process ensures that our model can handle the specific challenges of multilingual hate speech detection, making it a powerful tool for analyzing diverse online content. The fine-tuning of multilingual BERT (mBERT) to identify hate speech in tweets is shown in **Error! Reference source not found.**.

This architecture may be divided into three key components: the attention mechanism that helps the model focus on crucial aspects, the initial processing of tweets, and the final categorization of tweets into hate or non-hate.

The multilingual language model (mBERT) is where we begin first. Similar to a brain trained on text in several languages is mBERT. It's customized particularly for Twitter data here. This is accomplished by having certain words in the tweet randomly obscured, or masked, and having the model attempt to identify what those words may be. This technique, known as concealed language loss, aids the model in understanding the context and meaning of words in tweets, which are frequently brief and informal messages. Consider this as the model picking up the language of "Twitter."

mBERT is prepared to work with actual tweets after it has gained knowledge from this disguised language job. Consider extracted features, or embeddings, as rich representations of the words in the

tweet that the model creates after it has processed the tweet. These embeddings, which are transferred to the following section of the architecture, capture the main idea of the tweet.
The attention mechanism is the fascinating element that now arrives. This is when mBERT shows its true brilliance. It examines more than just each word in a tweet. Rather, it concentrates on the key terms and their relationships. This is where concepts like query, value, and key are useful. To put it plainly: The context from earlier tweets, such as the user's prior interactions or responses, is represented using Key and Value embeddings. The current tweet is the main topic of the Query embedding.
The algorithm gains a better understanding of which aspects of the tweet are crucial in assessing whether it is hostile by analyzing these correlations. This attention mechanism works particularly well for identifying subtle hate speech in long or complicated tweets. Additionally, there is a feature called history embedding, which maintains track of the user's previous interactions with tweets. The program receives a more complete picture by combining this history with the context of the current tweet, which may be quite useful for identifying hate speech trends. Following the attention layer's processing of the tweet, a feed-forward network processes the resultant embeddings. This is a fancy way of expressing that the model decides whether this tweet is nasty based on what it has learnt from the embeddings. After the prediction is further refined by the network, the model finally outputs "Hate" or "Non-Hate." To put it briefly, Figure 4 illustrates how multilingual BERT is trained to comprehend tweets and then used to categories tweets as dangerous or not. To do this, a combination of the attention mechanism, embeddings, and final classification is used.

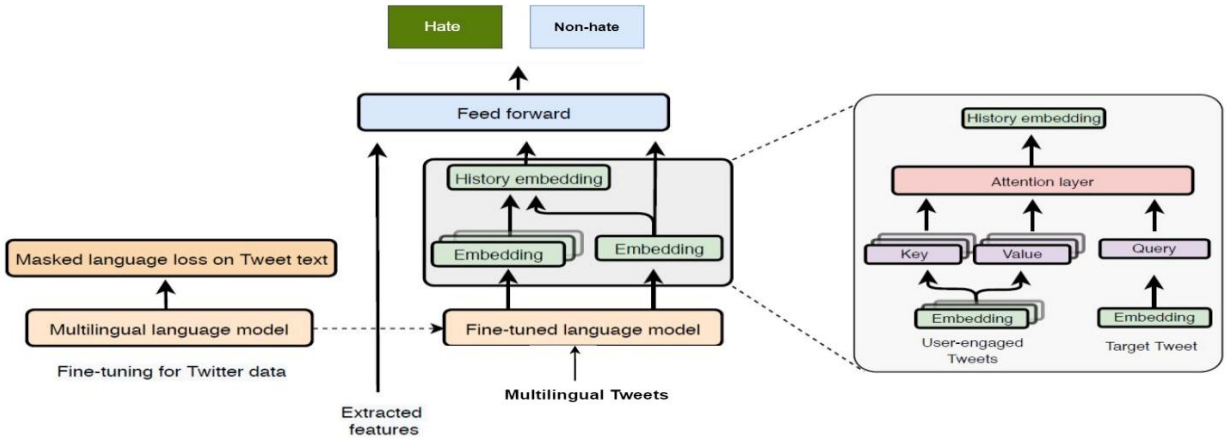

FIGURE 4. FLOW DIAGRAM OF MBERT

4. PERFORMANCE EVALUATION

In this section, the suggested mBERT-based model for hate speech identification is experimentally evaluated based on accuracy, precision and recall as shown in figure 5. The evaluation is based exclusively on accuracy. Here, accuracy is the ratio of correctly predicted instances to the total instances in the dataset and can be expressed by Eq. 1:

$$Accuracy = \frac{TP + TN}{TP + TN + TF + FN}$$

Precision is the ratio of correctly predicted positive instances to the total predicted positive instances and be defined by Eq.2:

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

Recall is the ratio of correctly predicted positive instances to the actual positive instances as given by Eq. 3:

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

With the knowledge of the genuine labels, accuracy is measured as the proportion of properly predicted labels. Our multilingual dataset, which included Roman Urdu -English and English tweets with code mixed in, allowed us to refine the mBERT model over several tests. We experimented with several hyperparameters, including the learning rate, batch size, and number of transformer layers fine-tuned during training, to optimise performance. The model was contrasted with conventional methods, such as CNN and CNN-LSTM architectures, which have been applied in related investigations in the past. However, by using mBERT, multilingual data may be handled more effectively without requiring character-level text preprocessing.

The architecture employed by Elouali et al. [7], which also aims at hate speech identification in both pure and code-mixed languages but does not make use of the pre-processing methods, was utilized as a benchmark for the performance of our mBERT model. In contrast, our model has a clear advantage over conventional character-level models as it integrates contextual knowledge and multilingual capabilities at the subword level.

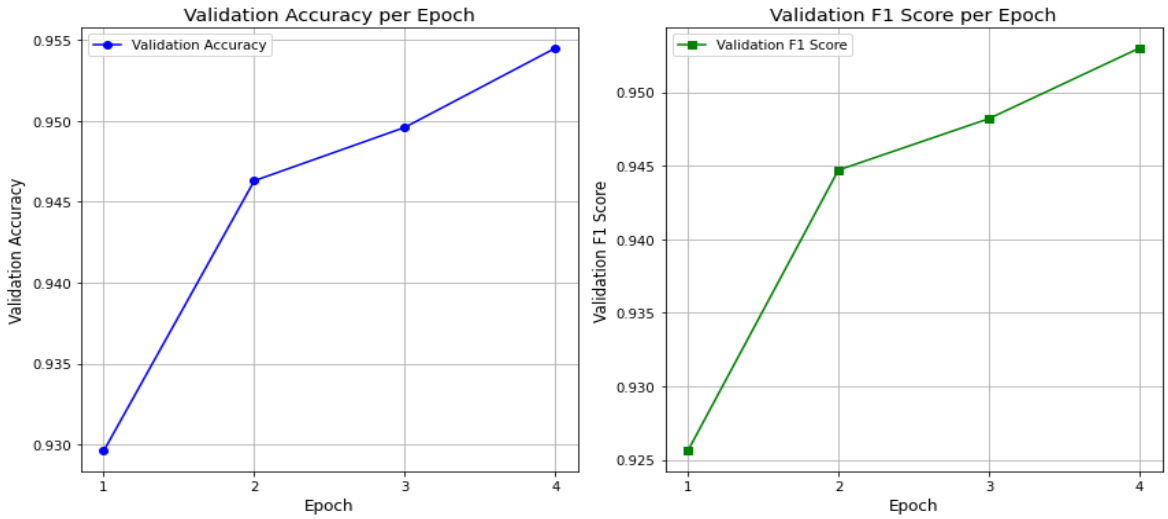

FIGURE 5. PERFORMANCE OF MODEL

Using popular libraries like NumPy, Pandas, Scikit-Learn, Matplotlib, TensorFlow, Transformers (Hugging Face), and NumPy, the model was constructed in Python. The notebook environment used for this research was based on a local machine running Windows 10, equipped with an Intel Core i7 processor, 8 GB of RAM, and a combination of 1 TB HDD plus 128 GB SSD for storage. The graphics were managed by an NVIDIA Graphics Card with 2 GB of VRAM. For scalability experiments, a virtual machine with more substantial processing power was also utilized. This virtual machine was configured with Windows 10, a six-core processor, 64 GB of RAM, and 1 TB of HDD storage.

The tests demonstrated that mBERT performed better in terms of accuracy and runtime efficiency as

compared to existing systems. It is noteworthy that the mBERT model's scalability was validated in processing huge datasets, since its runtime increased proportionately with increasing input size.

I. Word-Level mBert

In our experiments using the mBERT model for hate speech detection, we performed multiple training runs to evaluate the performance of the model with various hyperparameters and configurations. The training parameters were configured as follows: the BERT model used was bert-base-multilingual-cased, the classification task involved 2 classes, the maximum sequence length was set to 128, the batch size was 16, the number of epochs was 4, and the learning rate was 2e-5. A summary of these parameters is provided in Table 3.

TABLE 3. TRAINING PARAMETERS FOR MBERT MODEL

Parameter	**Value**
BERT Model Name	bert-base-multilingual-cased
Number of Classes	2
Max Length	128
Batch Size	16
Number of Epochs	4
Learning Rate	2e-5

During training, the model was evaluated after each epoch on the validation set, and the results are summarized in Table 4. We observed the following performance metrics: accuracy and F1 score for each epoch.

TABLE 4. PERFORMANCE METRICS FOR EACH EPOCH

Epoch	**Validation Accuracy**	**Validation F1 Score**
1	0.9296	0.9256
2	0.9463	0.9447
3	0.9496	0.9482
4	0.9545	0.9530

In each epoch, the model demonstrated progressive improvements in accuracy and F1 score as presented in Figure 6. The highest accuracy of 95.45% and an F1 score of 95.30% were achieved in the fourth epoch. These metrics demonstrate that the mBERT model, when trained with the specified parameters, performs effectively in detecting hate speech. Additionally, we tested the model on various samples and observed accurate predictions aligning with the expected labels. The results from these experiments underscore the robustness of the mBERT model for hate speech detection, validating its effectiveness in a multilingual context.

Figure 7 illustrates the accuracy of hate speech prediction (hate/non-hate) for our models. Comparing them with the model presented by Elouali et al. [7],

Table provides a summary of the model over all significance. Our experiments revealed that the mBERT model outperformed Elouali et al.'s [7] model in terms of accuracy.

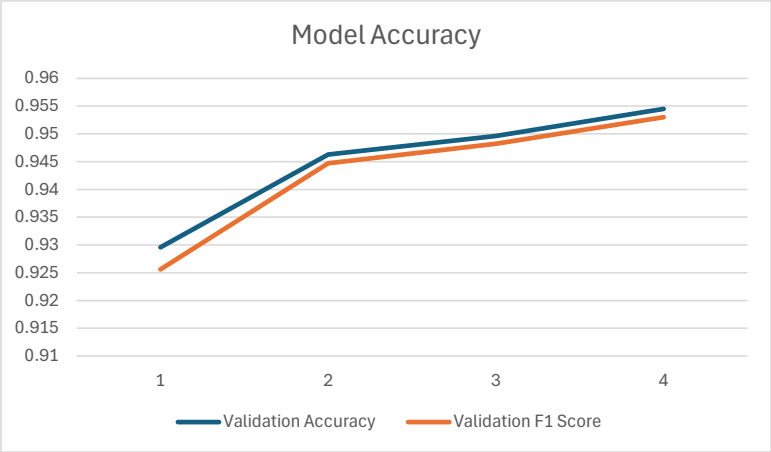

FIGURE 6. MODEL ACCURACY

The enhanced performance of our mBERT model can be attributed to two main factors: (1) the utilization of over-sampling techniques to address the inherent class imbalance in the dataset, thus reducing biased results, and (2) the effective use of multilingual BERT for feature extraction and classification, which leverages pre-trained language representations to improve predictive accuracy. Additionally, we conducted further experiments to optimize hyperparameters such as the maximum sequence length, batch size, number of epochs, and learning rate. These findings are detailed in Table 3.

Overall, our mBERT-based model demonstrated considerable improvements over Elouali et al.'s [7] approach, highlighting the effectiveness of our methodology in advancing hate speech detection.

TABLE 5. RESULTS COMPARISON

Model	**No. of Epochs**	**Accuracy**
Character-level CNN using over-sampled training dataset [7]	50	0.8661
Word level mBERT using over-sampled training dataset	4	0.9530

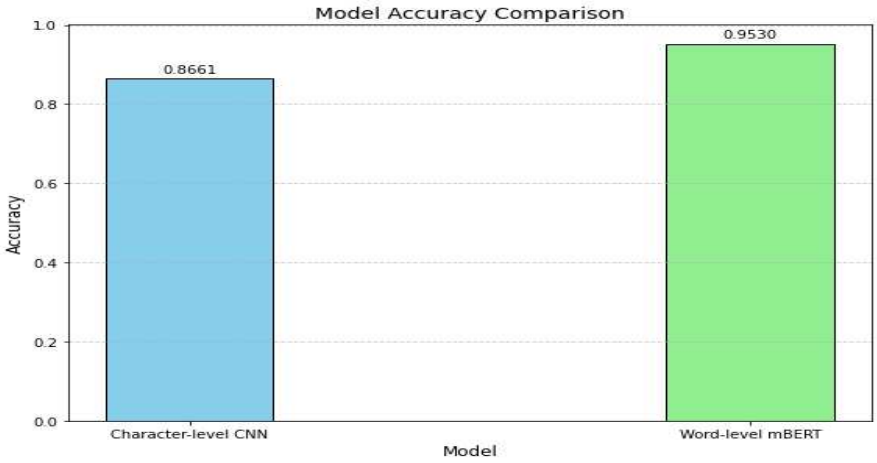

FIGURE 7. ACCURACY

The experimental results of the two classes are depicted by Figure 8. The results highlights that data preprocessing plays a crucial role in improving processing efficiency and maintaining prediction accuracy, especially when compared to closely related methods from the literature.

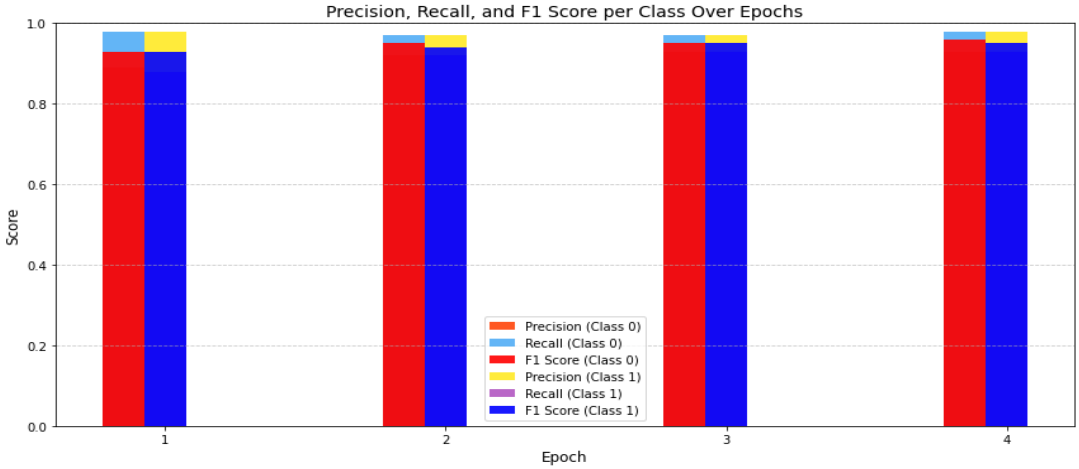

Figure 8. Performance Metrics of the two classes

5. CONCLUSION AND FUTURE WORK

This paper explores how data pre-processing affects the detection of hate speech on social media platforms, specifically targeting Twitter. The research was centered on a comprehensive data cleansing methodology that encompassed ten stages, including the removal of identifiers and URLs. An effective approach for accurately identifying hate speech in multilingual and code-mixed text, particularly in English and Roman Urdu -English, was devised. Deep neural networks (DNNs) are used for classification, while mBERT (Multilingual BERT) is used to improve character-level embedding representations.

Using an actual dataset of tweets that were released, we conducted an experimental evaluation to evaluate the accuracy of the suggested technique. Both unbalanced and oversampled versions of the training set were employed in the division of the dataset into training and test sets. Our mBERT models performed better in terms of accuracy than the approach used by Elouali et al. [7].

Specifically, our model based on mBERT performed better, with an accuracy of 94.5% on the validation set. This enhancement emphasizes how useful it is to use multilingual representations for detecting hate speech in languages with mixed codes. The suggested approach may be expanded in the future to support a wider variety of languages, both code-mixed and pure. Furthermore, investigating the psychological effects of emoticons on social media users and how they affect the identification of hate speech may yield insightful information. Subsequent research endeavors might encompass: 1) Examining the impact of distinct emoticons on human cognition; and 2) Incorporating emoticons into data models to enhance the precision of their contextual interpretations. In the future, we aim to expand the model to support additional languages and dialects, especially those frequently used in code-mixed communication on social media platforms. We also aim to Incorporate sentiment-bearing features like emoticons into hate speech detection models that could improve the contextual understanding of online discourse. Finally, we aim to focus on the role of psychological and cultural factors in hate speech, particularly examining how users' interpretations of emoticons affect hate speech classification outcomes.

REFERENCES

[1] "The Equilibrium decodes the impact of social media addiction in 2022 and its coping mechanism." Accessed: Sep. 11, 2024. [Online]. Available: https://www.business-standard.com/content/press-releases-ani/the-equilibrium-decodes-the-impact-of-social-media-addiction-in-2022-and-its-coping-mechanism-122030800604_1.html

[2] M. Hietanen and J. Eddebo, "Towards a Definition of Hate Speech-With a Focus on Online Contexts," *Journal of Communication Inquiry*, vol. 2023, no. 4, pp. 440–458, doi: 10.1177/01968599221124309.

[3] M. E. Ballard and K. M. Welch, "Virtual Warfare: Cyberbullying and Cyber-Victimization in MMOG Play", doi: 10.1177/1555412015592473.

[4] "tufts-paper-454".

[5] "Germany: Responding to 'hate speech'", Accessed: Sep. 14, 2024. [Online]. Available: www.article19.orgTw:@article19orgFb:facebook.com/article19org

[6] H. Ge, "Religion and Refugee Resettlement in New Zealand: Humanitarianism, Muslims, and the Secular State," 2024.

[7] A. Elouali, Z. Elberrichi, and N. Elouali, "Hate speech detection on multilingual twitter using convolutional neural networks," *Revue d'Intelligence Artificielle*, vol. 34, no. 1, p. 81, Feb. 2020, doi: 10.18280/RIA.340111.

[8] T. Y. S. S. Santosh and K. V. S. Aravind, "Hate speech detection in Hindi-English code-mixed social media text," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Jan. 2019, pp. 310–313. doi: 10.1145/3297001.3297048.

[9] T. Javed *et al.*, "IndicVoices: Towards building an Inclusive Multilingual Speech Dataset for Indian Languages," Mar. 2024, [Online]. Available: http://arxiv.org/abs/2403.01926

[10] I. Kwok and Y. Wang, "Locate the Hate: Detecting Tweets against Blacks," 2013, Accessed: Sep. 14, 2024. [Online]. Available: http://tempest.wellesley.edu/~ywang5/aaai/paper.html.

[11] J. H. Park and P. Fung, "One-step and Two-step Classification for Abusive Language Detection on Twitter," *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 41–45, 2017, doi: 10.18653/V1/W17-3006.

[12] A. Bohra, D. Vijay, V. Singh, S. S. Akhtar, and M. Shrivastava, “A Dataset of Hindi-English Code-Mixed Social Media Text for Hate Speech Detection,” pp. 36–41, 2018, Accessed: Sep. 14, 2024. [Online]. Available: https://github.com/deepanshu1995/HateSpeech-Hindi-
[13] V. Singh, A. Varshney, S. S. Akhtar, D. Vijay, and M. Shrivastava, “Aggression Detection on Social Media Text Using Deep Neural Networks,” pp. 43–50, 2018, Accessed: Sep. 14, 2024. [Online]. Available: https://www.gwava.com/blog/internet-data-created-
[14] F. Alkomah and X. Ma, “A Literature Review of Textual Hate Speech Detection Methods and Datasets,” Jun. 01, 2022, *MDPI*. doi: 10.3390/info13060273.
[15] Z. Waseem, “Are You a Racist or Am I Seeing Things? Annotator Influence on Hate Speech Detection on Twitter,” *NLP + CSS 2016 - EMNLP 2016 Workshop on Natural Language Processing and Computational Social Science, Proceedings of the Workshop*, pp. 138–142, 2016, doi: 10.18653/V1/W16-5618.
[16] R. Magu, K. Joshi, and J. Luo, “Detecting the Hate Code on Social Media,” 2017. [Online]. Available: www.aaai.org
[17] B. Gambäck and U. Kumar Sikdar, “Using Convolutional Neural Networks to Classify Hate-Speech,” pp. 85–90, Accessed: Sep. 14, 2024. [Online]. Available: http://github.com/zeerakw/hatespeech
[18] H. Mehta and K. Passi, “Social Media Hate Speech Detection Using Explainable Artificial Intelligence (XAI),” *Algorithms*, vol. 15, no. 8, Aug. 2022, doi: 10.3390/A15080291.
[19] N. V. Sahana, R. Prerana, S. Niharika, S. Rakshitha, and K. J. Bhanushree, “Automatic Hate Speech Detection using Ensemble Method and Natural Language Processing Techniques,” in *2023 International Conference on Network, Multimedia and Information Technology, NMITCON 2023*, Institute of Electrical and Electronics Engineers Inc., 2023. doi: 10.1109/NMITCON58196.2023.10276372.
[20] Z. Waseem and D. Hovy, “Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter,” pp. 88–93, Accessed: Sep. 14, 2024. [Online]. Available: http://github.com/zeerakw/hatespeech
[21] A. DE Souza, D. Costa Abreu, and st Gabriel Araujo De Souza, “Automatic offensive language detection from Twitter data using machine learning and feature selection of metadata”, Accessed: Sep. 14, 2024. [Online]. Available: http://shura.shu.ac.uk/26018/https://orcid.org/0000-0001-7461-7570
[22] T. Davidson, D. Warmsley, M. Macy, and I. Weber, “Automated Hate Speech Detection and the Problem of Offensive Language *”, Accessed: Sep. 14, 2024. [Online]. Available: www.facebook.
[23] H. Watanabe, M. Bouazizi, and T. Ohtsuki, “Hate Speech on Twitter: A Pragmatic Approach to Collect Hateful and Offensive Expressions and Perform Hate Speech Detection,” *IEEE Access*, vol. 6, pp. 13825–13835, Feb. 2018, doi: 10.1109/ACCESS.2018.2806394.
[24] P. Mathur, R. Sawhney, M. Ayyar, and R. Ratn Shah, “Did you offend me? Classification of Offensive Tweets in Hinglish Language,” pp. 138–148, 2018, Accessed: Sep. 14, 2024. [Online]. Available: www.github.com/pmathur5k10/
[25] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar, “Predicting the Type and Target of Offensive Posts in Social Media,” Association for Computational Linguistics. [Online]. Available: http://bit.ly/2FhLMVz
[26] K. Kumari, J. P. Singh, Y. K. Dwivedi, and N. P. Rana, “Multi-modal Aggression Identification Using Convolutional Neural Network and Binary Particle Swarm Optimization.” [Online]. Available: www.facebook.com

[27] K. Kumari and J. P. Singh, “Multi-modal cyber-aggression detection with feature optimization by firefly algorithm,” *Multimed Syst*, vol. 28, no. 6, pp. 1951–1962, Dec. 2022, doi: 10.1007/S00530-021-00785-7.
[28] S. Kamble and A. Joshi, “Hate Speech Detection from Code-mixed Hindi-English Tweets Using Deep Learning Models,” Nov. 2018, [Online]. Available: http://arxiv.org/abs/1811.05145
[29] S. Shekhar, H. Garg, R. Agrawal, S. Shivani, and B. Sharma, “Hatred and trolling detection transliteration framework using hierarchical LSTM in code-mixed social media text,” *Complex and Intelligent Systems*, vol. 9, pp. 2813–2826, Jun. 2023, doi: 10.1007/S40747-021-00487-7.
[30] K. Sreelakshmi, B. Premjith, and K. P. Soman, “Detection of Hate Speech Text in Hindi-English Code-mixed Data,” in *Procedia Computer Science*, Elsevier B.V., 2020, pp. 737–744. doi: 10.1016/j.procs.2020.04.080.
[31] R. Kumar, A. N. Reganti, A. Bhatia, T. Maheshwari, and B. Rao, “Aggression-annotated Corpus of Hindi-English Code-mixed Data”.
[32] K. Kumari *et al.*, “Bilingual Cyber-aggression Detection on Social Media using LSTM Autoencoder.” [Online]. Available: www.facebook.com
[33] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, “A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data”.
[34] P. Branco, L. Torgo, and R. P. Ribeiro, “A Survey of Predictive Modelling under Imbalanced Distributions,” 2015.
[35] B. Krawczyk, “Learning from imbalanced data: open challenges and future directions,” Nov. 01, 2016, Springer Verlag. doi: 10.1007/s13748-016-0094-0.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- linkofgithubResData.docx