



## Article

# The Impact of Data Pre-Processing on Hate Speech Detection in a Mix of English and Hindi–English (Code-Mixed) Tweets

Khalil Al-Hussaeni <sup>1,\*</sup>, Mohamed Sameer <sup>2</sup> and Ioannis Karamitsos <sup>2</sup><sup>1</sup> Computing Sciences, Rochester Institute of Technology, Dubai 341055, United Arab Emirates<sup>2</sup> Graduate Programs and Research, Rochester Institute of Technology, Dubai 341055, United Arab Emirates; ms7038@rit.edu (M.S.); ixkcad1@rit.edu (I.K.)

\* Correspondence: kxacad@rit.edu

**Abstract:** Due to the increasing reliance on social network platforms in recent years, hate speech has risen significantly among online users. Government and social media platforms face the challenging responsibility of controlling, detecting, and removing massively growing hateful content as early as possible to prevent future criminal acts, such as cyberviolence and real-life hate crimes. Twitter is used globally by people from various backgrounds and nationalities; it contains tweets posted in different languages, including code-mixed language, such as Hindi–English. Due to the informal format of tweets with variations in spelling and grammar, hate speech detection is especially challenging in code-mixed text. In this paper, we tackle the critical issue of hate speech detection on social media, with a focus on a mix of English and Hindi–English (code-mixed) text messages on Twitter. More specifically, we aim to evaluate the impact of data pre-processing on hate speech detection. Our method first performs 10-step data cleansing; then, it builds a detection method based on two architectures, namely a convolutional neural network (CNN) and a combination of CNN and long short-term Memory (LSTM) algorithms. We tune the hyperparameters of the proposed model architectures and conduct extensive experimental analysis on real-life tweets to evaluate the performance of the models in terms of accuracy, efficiency, and scalability. Moreover, we compare our method with a closely related hate speech detection method from the literature. The experimental results suggest that our method results in an improved accuracy and a significantly improved runtime. Among our best-performing models, CNN-LSTM improved accuracy by nearly 2% and decreased the runtime by almost half.

**Keywords:** speech recognition; hate speech; code-mixed; multilingual; neural networks

**Citation:** Al-Hussaeni, K.; Sameer, M.; Karamitsos, I. The Impact of Data Pre-Processing on Hate Speech Detection in a Mix of English and Hindi–English (Code-Mixed) Tweets. *Appl. Sci.* **2023**, *13*, 11104. <https://doi.org/10.3390/app131911104>

Academic Editors: Alexey Karpov, Douglas O’Shaughnessy, Ha-Jin Yu and Dongsuk Yook

Received: 11 August 2023

Revised: 22 September 2023

Accepted: 6 October 2023

Published: 9 October 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Recent years have witnessed a tremendous rise in Internet usage worldwide. One primary reason for this is the increased usage of social media websites that became popular, attractive, and addictive to users in a short time [1]. Twitter is one of the prevalent social media platforms widely used by people, generating massive online content and big data. Such platforms allow users to connect faster and easier and openly share and express their opinions and thoughts, free of cost, through photos, videos, or messages/tweets, giving them near unconditional freedom of speech.

Despite the above-mentioned positives, there is also a more negative side to social media. With increased usage and addiction to social media, there has been a significant rise in users openly using hate speech and offensive language. Nockleby [2] defined hate speech as “communications of animosity or disparagement of an individual or a group on account of a group characteristic such as race, color, national origin, sex, disability, religion, or sexual orientation”. Since all interactions are virtual, certain users take advantage of being anonymous and opt for posting hateful content against specific individuals or communities, a behavior that the same users would be much less likely to exhibit in real life.

Hateful content gives rise to cyberviolence and hate crimes, affecting individuals' social and mental health and impacting human lives. Most platforms forbid the usage of hate speech, but due to the massive amount of content generated by hateful users, controlling communication is becoming an increasingly challenging task. Hence, it is crucial for the government and social media websites to control, detect, and eliminate hateful content as early as possible to prevent real-life violence likely to be take place as a consequence [3]. Hate speech detection is so crucial that, in March 2021, an Indian politician requested for legislation to immediately regulate online hate speech since it had become uncontrollable [4]. Like India, many other countries have also imposed strict laws to combat online hate speech [5,6]. Twitter is adamant on removing hateful content but is still criticized for not being effective at it [7].

In this study, we tackle the critical issue of hate speech detection on social media, with a focus on short text messages on Twitter. Twitter is a free-of-charge global social networking platform for sharing short text (tweets), where each tweet can contain up to 280 characters. People from various backgrounds and nationalities use Twitter. Thus, some tweets may include code-mixed text. Code-mixing is defined as embedding linguistic units such as words, phrases, or morphemes of one language into the utterance of another language [8]. In other words, code-mixing involves writing the words of one language using the alphabets of another. Examples 1.1 and 1.2 demonstrate one hate and one non-hate tweet in code-mixed text, respectively, along with their English translations. We note that, due to the obscene language used in real-life hateful tweets, we resort to showing a benign version in Example 1.1.

**Example 1.1 (hate tweet in code-mixed Hindi–English).** This example shows a hate tweet in a team sport context written in Hindi–English code-mixed text. That is, the tweet is written with Hindi words using the English alphabet along with a few English words (“team”, “captain”, and “hate”):

*“Agar is team ka captain Dhoni nhi to ye team kuch nhi, chahe kisi Ko bhi le aao, Lakho log aaj apki team Ko hate krne lge h”.*

**Translation:**

*“If Dhoni is not the captain of this team, then this team is nothing, no matter whoever you bring, millions of people have started hating your team today”.*

**Example 1.2 (non-hate tweet in code-mixed Hindi–English).** This example shows a non-hate tweet in a media context written in Hindi–English code-mixed text. That is, the tweet is written with Hindi words using the English alphabet along with an English word (“media”):

*“Magar ye media walay to jaan boojh ker paise ke liye sab karrahe hain”.*

**Translation:**

*“But these media people are deliberately doing everything for money”.*

India is a large country with diverse languages within its many populations and communities [9]. Consequently, code-mixed language, namely Hindi–English, is used naturally and effectively by people when writing tweets for ease of expression and communication. Code-mixed text is usually a language with an informal format and non-grammatical structure with a significant number of variations in slang and spellings [8]. Due to the informal format caused by the mixture of different languages, hate speech detection in code-mixed text is challenging. Hence, we need to have a softwarized mechanism to automatically detect and control such hateful content regardless of the language to ensure the security as well as the social and mental well-being of users.

This study targets a mix of English and Hindi–English (code-mixed) tweets. In the area of hate speech detection, relevant research work has been conducted on pure languages such as English [10,11]. Moreover, there exists research work on code-mixed languages, such as Hindi–English [8,12,13]. However, hate speech detection in a mix of multiple languages (multilingual) has not been explored extensively in the literature.

### 1.1. Contribution

The main contribution of this study is that it evaluates the impact of data pre-processing on hate speech detection in a multilingual context. The impact of data pre-processing is evaluated in terms of prediction accuracy (percentage of correctly predicted tweet categories), prediction efficiency (runtime), and method scalability with respect to the size of the input dataset.

To the best of our knowledge, the study by Elouali et al. [7] is the only study that addresses the detection of hate speech in a mix of different languages, including pure English and code-mixed Hindi–English. In this study, we propose an efficient method to accurately detect hate speech in a mix of English and Hindi–English tweets. Our method first subjects all tweets to 10-step data cleansing (not performed by [7]); then, it uses a CNN as well as a combination of CNN and LSTM algorithms with character-level embedding to build a hate speech prediction model. We perform extensive experimental analysis on real-life tweets by evaluating the performance of our proposed method in terms of accuracy, efficiency (runtime), and scalability, and by comparing it with a closely related method proposed by [7]. Experimental evaluation suggests that our method has an improved accuracy and a significantly improved runtime.

### 1.2. Organization

This paper is organized as follows: Section 2 provides a literature review over existing studies conducted in the area of hate speech detection. Section 3 provides a detailed description of the datasets used for the experiments in our study. In Section 4, we detail our proposed model architectures for detecting hate and non-hate tweets. Experimental evaluation is provided in Section 5. This paper is concluded in Section 6.

## 2. Related Work

In the past decade, several research studies have been conducted to detect toxic (hate, offensive, or aggressive) speech on social media [14]. These studies targeted toxic speech detection in text written in English and a few other languages. Hate and abusive contents on social media are independent of the language they are written in; hence, we need to address this problem regardless of the informality or complexity of the language, whether code-mixed or pure. Our study targets short text, and particularly published tweets on Twitter. Hence, we group toxic speech detection techniques into three categories based on the targeted tweet language. The first category includes research on the detection of toxic speech in English text only, the second includes research in Hindi–English code-mixed text only, and the third includes research in multilingual text (text written in multiple languages).

### 2.1. Toxic Speech Detection in English Text

The literature includes several proposed approaches to detecting toxic content in the English text. Researchers have identified English toxic text by detecting hate tweets [10,11,15–19], offensive tweets [20–22], and a mix of both [23–25]. Various algorithms were proposed for detecting toxic content in English text, namely machine learning (ML)-based classification algorithms [10,15,16,20,22–24] and deep learning (DL)-based classification algorithms [11,17,21,25].

Kwok and Wang [10] built a bag-of-words naïve Bayes classifier model to predict English hate tweets targeting the Black community. The model resulted in an accuracy of 76%. Kwok and Wang found out that 86% of the racist tweets were categorized as racist only because they contained offensive words. Hence, by considering only the unigram features of the tweets, the model was not sufficiently accurate since the presence of offensive words in a non-hate tweet can also lead to the wrong classification of tweets as anti-Black. Hate speech needs to be carefully understood since it can even be expressed subtly without offensive words.

Waseem and Hovy [20] used critical race theory and built a logistic regression classifier that considers additional features to tweets, primarily demographic and geographic data. Even though demographic features did help enhance the accuracy of prediction, this type of information is usually not directly available or reliable on Twitter.

Magu et al. [16] proposed an ML-based classification algorithm to detect racist tweets in English that use unique hate code words referencing specific communities to avoid violating social media policies. A support vector machine (SVM) classifier was built to classify the racist tweets that used hate code words intentionally and the non-hate tweets that used the code words in a regular context. The model achieved an accuracy of 79.4%. The model was further used to identify the user handles who posted hateful tweets using code words and the frequency of tweets above a threshold value. These results helped the model understand the usage patterns of such users showing racism.

Davidson et al. [23] proposed an ML-based classification algorithm to classify hate speech, offensive language, or neither in English tweets. Logistic regression with L2 regularization was used as a multi-class classification model, which achieved a precision of 91%, recall of 0.90%, and F1-score of 90%. Predictions, accuracies, and errors were closely analyzed to understand the possibilities and difficulties in accurately classifying hate speech, offensive language, and neither. Homophobic and racist tweets were mainly classified as hateful, sexist tweets were classified as offensive, and the tweets without any hateful or offensive terms were difficult to be classified.

Watanabe et al. [24] used features such as unigrams, patterns, sentiments, and semantics to detect hate speech and offensive English language. The J48graft algorithm was built to perform classification using the WEKA toolkit. An accuracy of 87.4% was achieved for binary classification of tweets (offensive or clean) and 78.4% for ternary classification (hateful, offensive, or clean) using all the features combined.

Zampieri et al. [26] proposed ML and DL-based classification algorithms to classify offensive content on Twitter and further identify the type and target of these offences. SVM, Bi-directional LSTM (BiLSTM), and CNN models were built, and their performances were compared using a macro-averaged F1-score. The CNN model performed best in all three stages: detecting offensive tweets, categorizing the type of offense, and identifying the target.

De Souza and Da Costa-Abreu [22] proposed ML-based classification algorithms to detect offensive language in tweets and enhance the performance through the quality of features and data configuration. Linear SVM (LSVM) and naïve Bayes classification models were built to detect offensive language; LSVM achieved a 90% accuracy and a 92% recall, whereas naïve Bayes achieved a 92% accuracy and a 95% recall. LSVM was sensitive to data type and normalization and required proper parameters and a balanced input to attain good results. On the other hand, such issues were not found in the naïve Bayes classifier; hence, it performed better than LSVM and was easier to implement.

Recently, there have been some attempts to identify hate speech in contents composed of images associated with text [27,28]. While this is a great contribution to aggressive speech research, we believe the problem, and thus the proposed method, differs from that of ours, i.e., aggression in purely textual contents.

## 2.2. Toxic Speech Detection in Hindi–English Code-Mixed Text

Researchers proposed several approaches to detecting toxic content in the code-mixed language of Hindi–English text via detecting hate tweets [8,12,29–31] and aggressive tweets [13]. Various algorithms were proposed by these researchers, mainly ML-based classification algorithms [12,30] and DL-based classification algorithms [8,13,29].

Bohra et al. [12] proposed ML-based classification algorithms to introduce hate speech detection in code-mixed (Hindi–English) posts on Twitter. SVM and random forest (RF) classifiers were built. The performance of SVM was better than RF, with the highest accuracy being 71.7% when all the features were used. Considering individual features, character n-grams in SVM and word n-grams in RF achieved the highest accuracy.

Kamble and Joshi [29] proposed DL-based classification algorithms to detect hate speech in code-mixed (Hindi–English) tweets on Twitter. The 3849 tweets prepared by Bohra et al. [12] were used for model evaluation, and a large set of tweets targeting minority groups were collected to train the domain-specific word embeddings. Domain-specific word embeddings improved the representation of hate targets much better than general embeddings. SVM and RF classifiers were built using the same methodology from Bohra et al. [12] as their baseline. Additionally, three DL models, namely CNN-1D, LSTM, and BiLSTM, were built using the word embeddings for comparison. CNN-1D achieved the highest precision (83.34%), F1-score (80.85%), and accuracy (82.62%), whereas BiLSTM achieved the highest recall of 78.90%. Hence, the results show that the DL models performed much better than the statistical methods in detecting the semantics and hate speech contexts.

Singh et al. [13] proposed ML and DL-based classification algorithms to detect aggressive speech in Hindi–English code-mixed posts and comments on Facebook. The data used had 12,000 posts/comments in Roman and Devanagari from Facebook, published as part of an online shared task by Kumar et al. [32], labeled into three classes: Covertly Aggressive, Overtly Aggressive, and Non-Aggressive. Six classification models were built, namely multimodal naïve Bayes, decision tree, SVM, multilayer perceptron, LSTM, and CNN, using various combinations of features and parameter tuning. CNN performed the best, achieving an accuracy of 73.2%. Neural networks do not always serve better than ML algorithms in speech detection. The models could not learn appropriately since the posts and comments were not diverse and varied in content enough.

Santosh and Aravind [8] proposed DL-based classification algorithms to detect hate speech in code-mixed (Hindi–English) tweets on Twitter. SVM and RF classifiers were built using the same methodology from Bohra et al. [12] as their baseline. Moreover, the “sub-word level LSTM model and Hierarchical LSTM model with attention based on phonemic sub-words” were built for comparison. SVM achieved the highest accuracy of 70.7%, whereas Hierarchical LSTM achieved the highest recall (45.1%) and F1-score (48.7%). A simple architecture of DL was used due to the limited size of the data.

In the study by Sreelakshmi et al. [30], Facebook’s pre-trained embedding, fasttext, was used as a feature matrix for classifying tweets using SVM-Linear, SVM-Radial Basis Function (RBF), and RF algorithms. Performances were then evaluated and compared using word2vec and doc2vec features. The fasttext feature gave the highest accuracy of 85.81%, and the word2vec feature gave 75.11% accuracy, both using the SVM-RBF classifier, while the doc2vec feature gave 64.15% accuracy using RF. This study has outperformed all the previous studies using fasttext with character-level features for classification, which achieved better results than document-level and word-level features.

Studies in this category present different and compelling solutions for detecting toxic speech in Hindi–English code-mixed text. However, the investigation of hate speech detection in multiple languages (mix of Hindi–English code-mixed and any other language) has not been thoroughly explored in the literature.

### 2.3. Toxic Speech Detection in Multilingual Text

Kumari et al. [33] presented a method for detecting aggression in bilingual text, specifically in English and Hindi, but not in code-mixed text.

A hate tweet dataset from Twitter was used by Elouali et al. [7] to detect toxic content in multilingual text using a DL-based classification algorithm. Elouali et al. [7] proposed a DL-based classification algorithm that utilized neural networks to detect hate speech in tweets written in seven different languages. They created their multilingual dataset by combining existing monolingual datasets of different languages. Two versions of the dataset were prepared as follows: the first version contained a total of 33,727 tweets in Arabic, Italian, Portuguese, Indonesian, and English languages, and the second version contained, in addition to the tweets in the first version, 12,446 tweets written in Hindi–English and German languages. A CNN model with character-level embedding was built, and

several experiments were carried out by modifying the parameters to determine the best architecture. The best accuracies achieved for the first and second versions of the datasets, respectively, were 88.93% and 83%. A single CNN model could detect hate speech in multiple languages and perform better compared to the previous studies on individual datasets. To the best of our knowledge, the study by Elouali et al. [7] is the only study that considers several languages for detecting hate speech, whereas, in the other studies mentioned above, the number of languages considered is only one. Elouali et al. [7] proposed a solution for a problem that is closely related to ours; however, the time of execution of their method was very long. This point is further elaborated on and demonstrated in Section 5.

This work tackles a mix of English and Hindi–English (code-mixed) tweets represented, generally, in short text, to detect hate speech using neural network architectures. Our proposed method efficiently detects hate speech in a mix of multiple languages. We experimentally evaluate our method in Section 5. Table 1 summarizes the studies presented in this section.

**Table 1.** Summary of comparison of related work.

Study By	Model Used	English	Hindi–English	Multilingual
Kwok and Wang [10]	Naïve Bayes	✓		
Waseem and Hovy [20]	Logistic regression	✓		
Magu et al. [16]	SVM	✓		
Davidson et al. [23]	Logistic regression	✓		
Watanabe et al. [24]	Decision Tree	✓		
Zampieri et al. [26]	SVM, BiLSTM, CNN	✓		
De Souza and Da Costa-Abreu [22]	Linear SVM, naïve Bayes	✓		
Bohra et al. [12]	SVM, random forest		✓	
Kamble and Joshi [29]	SVM, random forest, LSTM, BiLSTM		✓	
Singh et al. [13]	Naïve Bayes, decision tree, SVM, multilayer perceptron, LSTM, CNN		✓	
Santosh and Aravind [8]	SVM, random forest, LSTM		✓	
Kumari et al. [33]	LSTM			✓
Elouali et al. [7]	CNN	✓	✓	✓
Our proposed work	CNN, LSTM	✓	✓	✓

### 3. Description of the Dataset

This study requires a dataset that consists of a mix of English tweets and Hindi–English (code-mixed) tweets. Moreover, the tweets must be classified/labeled as hate or non-hate. Unfortunately, we could not find a publicly available dataset with such specifications. To ensure the repeatability of the experiments, we opted to employ widely used datasets in the literature by combining existing datasets created and used by pertinent research work [12,21,23]. Our dataset is formed by collecting tweets from three different sources, as follows:

1. The first portion of our dataset is collected from the dataset used by Davidson et al. [23]. Davidson et al.’s dataset contains tweets written in the English language and labels tweets as “hate”, “offensive”, or “neither”. However, for the sake of integrating this dataset into our work (i.e., keeping only two class labels), we relabeled the “offensive” class as hate and “neither” as non-hate. As a result, we had a dataset containing 24,783 tweets in English, where 4163 tweets were classified as non-hate and 20,620 as hate.
2. The second portion of our dataset is collected from the dataset used by Bohra et al. [12]. This dataset contains tweets written in Hindi–English code-mixed language and tweets labeled as “hate” or “non-hate”. Overall, this dataset contains 4579 tweets, where 2918 of them are classified as non-hate and 1661 as hate.

- The third portion of our dataset is collected from the dataset used by Mathur et al. [21]. Mathur et al.’s dataset contains tweets written in Hindi–English code-mixed language tweets labeled as “hate-inducing”, “non-offensive”, or “abusive”. For the sake of having two class labels, we relabeled “hate-inducing” and “abusive” as hate and relabeled “non-offensive” as non-hate. In total, this dataset contains 3189 tweets in code-mixed Hindi–English, where 1121 of them are classified as non-hate and 2068 as hate.

The final combined dataset has two columns: “Tweet” (text content) and “Label” (hate or non-hate). This resulting dataset consists of a total of 32,551 tweets, where 8202 are classified as non-hate and 24,349 as hate. Table 2 summarizes the final dataset and its class labels.

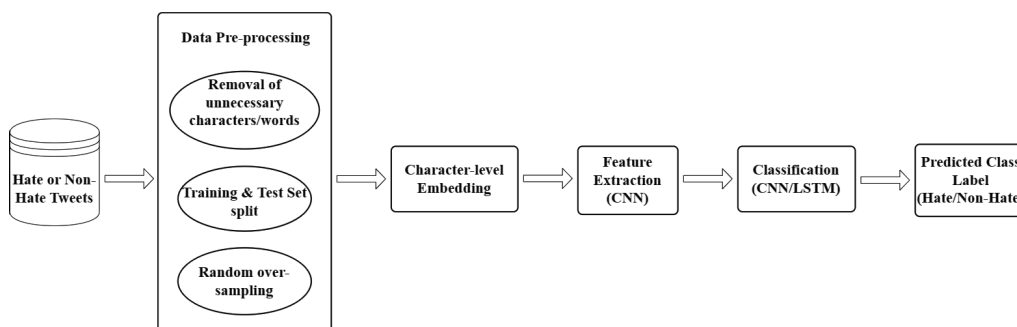
**Table 2.** Dataset class label statistics.

Language	Class	Number of Tweets
English	Hate	20,620
	Non-hate	4163
Hindi–English	Hate	4039
	Non-hate	3729
Total		32,551

Formally, a tweet,  $t$ , is a combination of characters. The maximum size of  $t$ , denoted by  $|t|$ , is 280 characters. A class label  $c \in \{“hate”, “non-hate”\}$ . A dataset record  $r = \langle t, c \rangle$  is an ordered set, where  $t$  is the tweet and  $c$  is  $t$ ’s associated class label. The dataset,  $D$ , is a set of records, i.e.,  $D = \{r_1, r_2, \dots, r_n\}$ , where  $n = |D|$  = the total number of records in the  $D$ .

#### 4. Proposed Method

This section details our proposed method for detecting hate and non-hate tweets. We first detail the data pre-processing steps carried out to prepare the data for the machine learning model. After that, we describe the training and test splits of the used dataset, followed by a description of data resampling for handling imbalances. Finally, we detail the proposed neural network architectures. Figure 1 provides an overview of all the steps and components involved in our method.



**Figure 1.** Flowchart of the proposed method.

##### 4.1. Data Cleaning (Pre-Processing)

We target classifying short text on social media platforms, and in particular, tweets. Due to the informal nature of such short text messages, they are bound to contain characters and words that may not be essential for hate speech detection, such as mentions/ usernames (e.g., @shivang), emoticons (e.g., ☺, ☹), special characters (e.g., @, +), punctuation marks (e.g., ?, !), and URLs (e.g., <http://t.co/hH50P5pytX>). Indeed, the experiments in Section 5

further corroborate this assumption. Therefore, the following pre-processing steps are carried out on all input tweets:

1. **Decoding HTML:** The encoded HTML parts are decoded by replacing the HTML entity names or numbers with their original text representation. Some encoded HTML parts, such as “&amp;”, are not correctly decoded and, thus, are removed.
2. **Removal of emoticons:** All the emoticons encoded with UTF-8 BOM (Byte Order Marks) in the original text are first decoded and then removed.
3. **Removal of mentions (usernames):** Since mentions are just other individuals’ usernames tagged in the tweets, they do not contribute to any kind of sentiment in the tweet. Hence, they are removed.
4. **Removal of URL links:** All the URLs (http/https/www) and other links (including links of pictures/videos) in the tweets are removed, since they do not contribute to any kind of sentiment in the tweet.
5. **Conversion to lower case:** All the tweets are converted to lower case in order to have only lower-case letters for character-level representation/embedding.
6. **Expansion of negations:** As a special character, all apostrophes are removed. However, in doing so, the negative contractions lose their meaning. For example, words like “can’t” end up as “can t”, which is not an English word. Therefore, to avoid this situation, all negative contractions are appropriately expanded. Thus, a contraction such as “can’t” is converted to “cannot”.
7. **Removal of hashtags, punctuation marks, numbers, and special characters:** There are cases where the words/phrases used with the hashtag symbol (#) provide useful information about the tweets and their sentiment. Hence, only the “#” symbols are removed, in addition to other special characters, punctuation marks, and numbers.
8. **Removal of extra white spaces:** Any extra white space in the tweets is removed.
9. **Removal of duplicates:** A dataset may contain duplicate tweets (same tweet repeated multiple times). All such instances of duplications are removed, keeping only unique tweets to avoid these duplicates from falling in both the training and test set during the train–test split step and hence avoid overfitting. However, some duplicate tweets are mistakenly labeled as both hate and non-hate. With careful examination, all erroneously labeled duplicates are removed.
10. **Removal of null values:** Some tweets may be composed of entirely removable parts, such as mentions, links, and emoticons. Consequently, pre-processing such tweets renders them empty strings (null). Such tweets are removed entirely.

After pre-processing our employed dataset, the final clean version consists of a total of 31,456 tweets with target class labels, where 7903 are classified as non-hate and 23,553 as hate. As a result, the dataset is imbalanced due to the unequal distribution of class labels.

#### 4.2. Training and Test Sets Split

The dataset is split into mutually exclusive training and test sets using holdout stratified sampling by reserving 80% of the total number of the dataset records for training (model construction) and 20% for testing (prediction and accuracy estimation). Eventually, the training set contains 25,164 records and the test set contains 6292 records.

#### 4.3. Handling Data Imbalance

Since we have an imbalanced dataset, standard classification algorithms tend toward the majority class and ignore the minority class when making decisions. Hence, most predictions will correspond to the majority class and treat the minority class features as noise, resulting in a highly biased model [34]. Resampling techniques, such as under-sampling and over-sampling, are widely used to address highly imbalanced datasets [35,36]. Random over-sampling is applied to our training set by replicating random records with replacements from the minority class (“non-hate”) [34]. Only the training set is over-sampled (as opposed to over-sampling the entire dataset) to avoid the problem of overfitting and poor generalization to the test set. This problem occurs if identical records are present





- f. One output layer, with two neurons and “sigmoid” activation function (where the number of neurons is the number of target class labels).

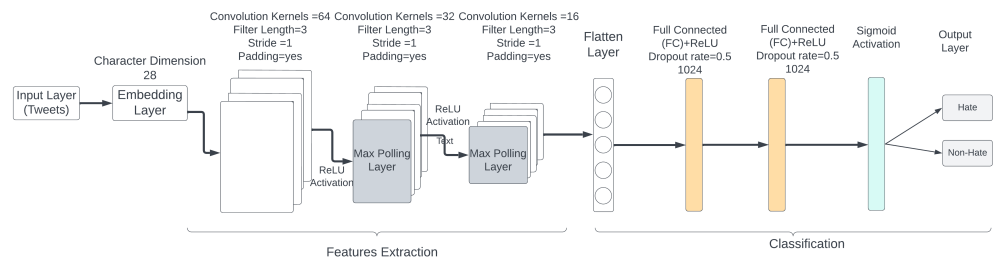


Figure 2. Character-level CNN architecture.

**Character-level CNN-LSTM.** A combination of two deep neural networks (CNN and LSTM) with character-level embedding is prepared by modifying the character-level CNN architecture by adding an LSTM layer. We leverage the benefits of both CNN and LSTM in a single architecture and use a unified model called CNN-LSTM for the detection of hate speech [38]. A CNN is commonly known to act as a good feature extractor using convolving filters, whereas LSTM is a special type of recurrent neural network (RNN) that has proven to learn and capture long-term dependencies in sequential data by considering the previous context/data. In the area of hate speech detection, the CNN can be used to extract character combinations and LSTM to learn long-term character dependencies [38].

Figure 3 illustrates our initial CNN-LSTM architecture with character-level representation. The CNN-LSTM architecture contains the following layers in sequence:

- One input layer, where the shape of the layer is 280 (maximum length of a tweet).
- One embedding layer, with an embedding size of 28 (dimension of each character).
- Three convolution layers, each with 100 filters, a window size of 3, stride 1, padding as “valid”, and “relu” activation function. Two of the convolution layers are followed by a max-pooling layer with a window size of 3.
- One LSTM layer, with a size of 60 neurons, a dropout rate of 0.5, and a recurrent dropout rate of 0.5.
- One max-pooling layer, with a window size of 3.
- One flatten layer.
- One output layer, with two neurons and “sigmoid” activation function (where the number of neurons is the number of target class labels).

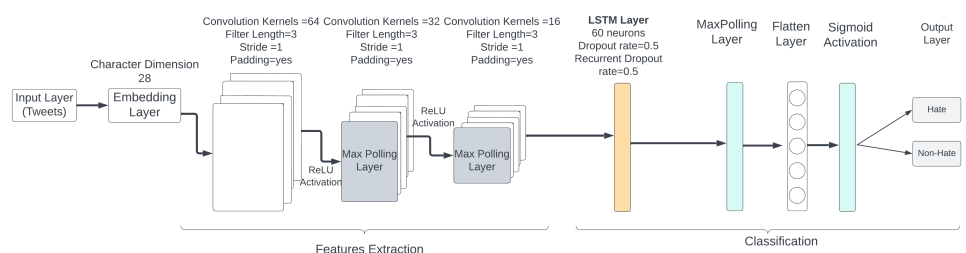


Figure 3. Character-level CNN-LSTM architecture.

Character-level embeddings are given as an input to the CNN model that has different layers, which generates/extracts character-level features. These features are provided as an input to the LSTM network to measure and learn the long-term character dependencies [39]. The output of LSTM is further fed to the max-pooling layer, and the sigmoid output generates the classification result as hate or non-hate. Hence, both CNN and LSTM are exploited and paired for feature extraction and classification.

## 5. Performance Evaluation

This section details an experimental analysis of the proposed model architectures. The experimental analysis is conducted by evaluating the performance of the model architectures in terms of accuracy, efficiency (runtime), and scalability. Accuracy is measured as the percentage of correct predictions, since the actual class values are known in advance. Runtime is calculated as the time taken to fit the model with the training set and validate the model using the test set. Scalability is the ability of the model to handle large amounts of data; that is, a model is considered as scalable if its runtime increases proportionately with the increase in input data records (tweets).

Several experiments are carried out for both character-level CNN and character-level CNN-LSTM architectures by tuning the hyperparameters, such as the number of filters of the convolution layer, dropout rate, and the number of neurons of the fully connected/LSTM layer, in order to define the best-performing architectures for the detection of hate speech in a mix of English and Hindi–English (code-mixed) text. The performance of our models is further compared to that of Elouali et al.’s [7] model, which does not employ the pre-processing steps described in Section 4.1; however, it targets hate speech detection in a mix of pure and code-mixed languages, making it a closely related study to ours (please see Section 2.3).

The proposed method was implemented using Python and its necessary associated tools (such as Natural Language Toolkit) and libraries (such as Numpy, Pandas, Scikit-Learn, Matplotlib, and Keras) on the Jupyter notebook in the Anaconda environment. The specifications of the machine used to run the experiments are Windows 10 OS, Intel Core i7, 8 GB of RAM, 1TB HDD + 128 GB SSD, and 2GB NVIDIA Graphic. Additionally, a virtual machine was used to run scalability experiments with the specifications of Windows 10 OS, 6-core, 64 GB of RAM, and 1 TB HDD. We would like to thank TDRA-ICT Fund for providing high-end computing machines to conduct our experiments, through the Digital Transformation Lab at Rochester Institute of Technology, Dubai (RIT-Dubai).

### 5.1. Character-Level CNN

In experiments 1 to 9, described in Table 3, we used the *imbalanced* version of the training dataset to train the character-level CNN architectures of different hyperparameters, and evaluated the performance of these architectures on the test dataset. The number of epochs is set to 50 for all the experiments.

**Table 3.** Experiments and results of character-level CNN (using imbalanced training dataset).

Exp No.	Model	Test Accuracy	Test Loss	No. of Epochs	Runtime	Best Accuracy	Worst Loss
1	3 conv. layers of 100 filters, 2 dense layers of 1024 neurons, dropout 0.5	0.8558	0.2846	50	18 min 4 s	0.8558 at 50th epoch	0.2846 at 50th epoch
2	3 conv. layers of 100 filters, 2 dense layers of 1024 neurons, dropout 0.2	0.8422	0.3076	50	21 min 54 s	0.8527 at 48th epoch	0.2853 at 48th epoch
3	3 conv. layers of 100 filters, 2 dense layers of 1024 neurons, dropout 0.4	0.8551	0.2970	50	19 min 23 s	0.8603 at 49th epoch	0.2795 at 49th epoch
4	3 conv. layers of 256 filters, 2 dense layers of 1024 neurons, dropout 0.2	0.8498	0.2970	50	53 min 28 s	0.8628 at 48th epoch	0.2780 at 48th epoch
5	3 conv. layers of 256 filters, 2 dense layers of 1024 neurons, dropout 0.5	0.8568	0.2833	50	64 min 29 s	0.8593 at 48th epoch	0.2804 at 48th epoch

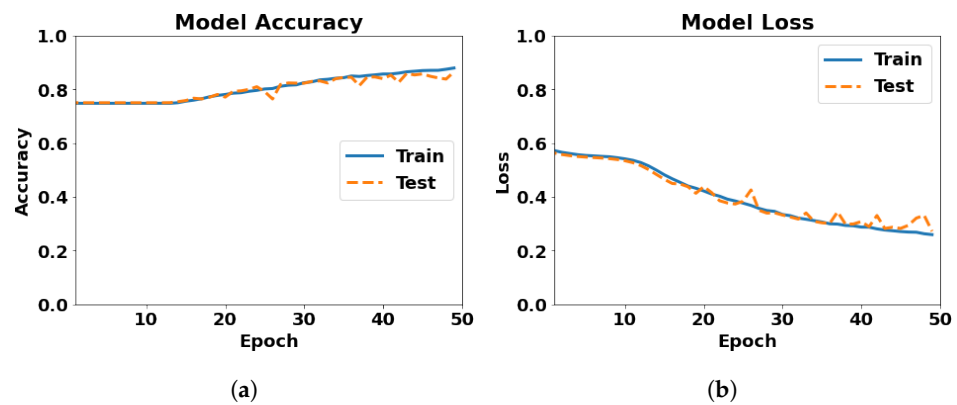
Table 3. Cont.

Exp No.	Model	Test Accuracy	Test Loss	No. of Epochs	Runtime	Best Accuracy	Worst Loss
6	3 conv. layers of 100 filters, 2 dense layers of 512 neurons, dropout 0.2	0.8636	0.2816	50	17 min 2 s	0.8636 at 50th epoch	0.2816 at 50th epoch
7	3 conv. layers of 100 filters, 2 dense layers of 512 neurons, dropout 0.5	0.8552	0.2967	50	16 min 30 s	0.8554 at 49th epoch	0.2866 at 49th epoch
8	3 conv. layers of 256 filters, 2 dense layers of 512 neurons, dropout 0.2	0.8536	0.2926	50	47 min 16 s	0.8627 at 50th epoch	0.2777 at 50th epoch
9	<b>3 conv. layers of 256 filters, 2 dense layers of 512 neurons, dropout 0.5</b>	<b>0.8652</b>	<b>0.2723</b>	<b>50</b>	<b>48 min 51 s</b>	<b>0.8652 at 50th epoch</b>	<b>0.2723 at 50th epoch</b>

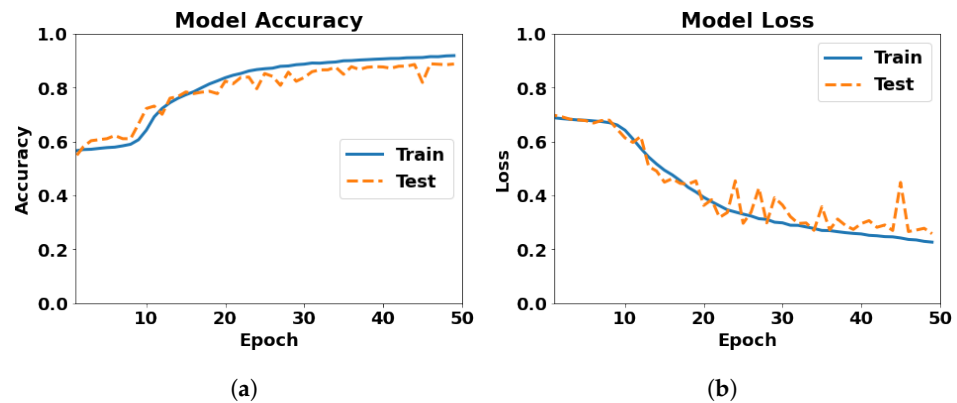
For the hyperparameters of the CNN in experiment 1, we fixed the number of filters of each convolution layer to 100, the number of neurons of each fully/densely connected layer to 1024, and the dropout rate of the dropout layer to 0.5. This is the initial version of the architecture (Figure 2). The best performance of experiment 1 on the test set was a 85.58% accuracy and a 28.46% loss in the 50th epoch with a runtime of 18 min 4 s. Loss is measured as the distance/difference between the current outcome of the model and the desired outcome, representing the summation of errors in the model.

In experiments 2 and 3, we reduced the dropout rate to 0.2 and 0.4, respectively, from the initial version of the architecture. The model in experiment 3 performed better in terms of accuracy (86.03%) compared to experiments 1 and 2. For experiment 4, we modified the hyperparameters with the number of filters as 256, the number of neurons as 1024, and the dropout rate as 0.2. The model in experiment 4 performed better in terms of accuracy (86.28%) compared to the previous three experiments, but its runtime was higher (53 min 28 s). In experiment 5, we increased the dropout rate to 0.5 from the architecture in experiment 4, but did not notice any improvement in the results. For experiment 6, we modified the hyperparameters with the number of filters as 100, the number of neurons as 512, and the dropout rate as 0.2. The model in experiment 6 performed better in terms of accuracy (86.36%) as well as runtime (17 min 2 s) compared to the previous five experiments. In experiment 7, we increased the dropout rate to 0.5 from the architecture in experiment 6, but did not see any improvement in the results. For experiment 8, we modified the hyperparameters with the number of filters as 256, the number of neurons as 512, and the dropout rate as 0.2, but also did not notice any improvement. In experiment 9, we increased the dropout rate to 0.5 from the architecture in experiment 8. The model in experiment 9 performed the best in terms of accuracy (86.52%) compared to all the previous experiments, with a runtime of 48 min 51 s. The graphs of accuracy and loss for 50 epochs of training and testing from experiment 9 are shown in Figure 4.

The model in experiment 9 has a precision of 69%, recall of 82% and F1-score of 75%. Although we trained the model using an imbalanced dataset, the model did favor the majority class (hate). However, in order to improve our results, we created another model with the same hyperparameters as in experiment 9 and trained it with the *over-sampled* version of the training dataset. The best performance of this model on the test set was a 88.84% accuracy and 26.61% loss in the 48th epoch with a runtime of 78 min 57 s, which is shown in Table 4. Hence, the accuracy of the model improved by 2.32% when training it with a balanced dataset. The model has a precision of 73%, recall of 88%, and an F1-score of 80%. The graphs of accuracy and loss for 50 epochs of training and testing of the model are shown in Figure 5.



**Figure 4.** Evaluation metrics for 50 epochs of training and testing with 3 convolutional layers of 256 filters, 2 dense layers of 512 neurons, and dropout rate of 0.5 using imbalanced training dataset: (a) Accuracy and (b) Loss.



**Figure 5.** Evaluation metric for 50 epochs of training and testing with 3 convolutional layers of 256 filters, 2 dense layers of 512 neurons, and dropout rate of 0.5 using over-sampled training dataset: (a) Accuracy and (b) Loss.

**Table 4.** Experiments and results of character-level CNN (using over-sampled training dataset).

Model	Test Accuracy	Test Loss	No. of Epochs	Runtime	Best Accuracy	Worst Loss
3 conv. layers of 256 filters, 2 dense layers of 512 neurons, dropout 0.5	0.8878	0.2582	50	78 min 57 s	0.8884 at 47th epoch	0.2661 at 47th epoch

### 5.2. Character-Level CNN-LSTM

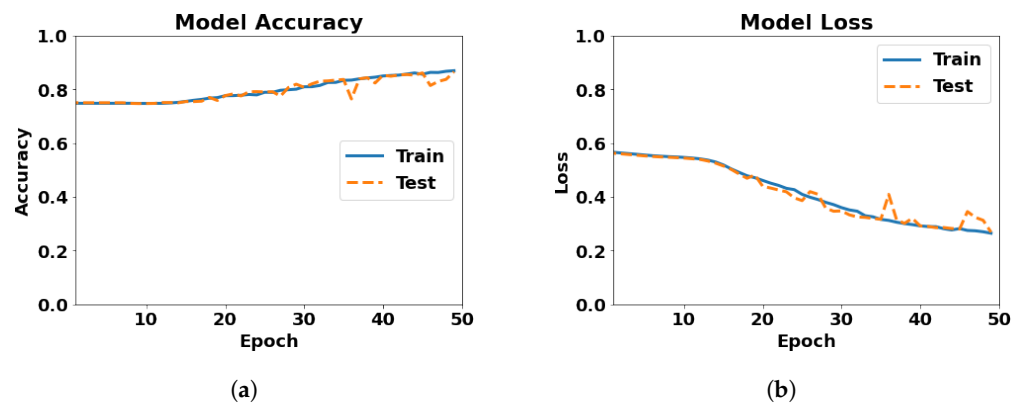
In experiments 1 to 9, described in Table 5, we trained the architectures of character-level CNN-LSTM of different hyperparameters with the cleaned and imbalanced version of the training dataset and evaluated their performance on the test dataset. The number of epochs was set to 50 for all the experiments.

For the hyperparameters of CNN and LSTM in experiment 1, we set the number of filters of each CNN convolution layer to 100, the number of neurons to 60, and the dropout rate to 0.5 for LSTM. This is the initial version of the architecture (Figure 3). The best performance of experiment 1 on the test set was a 85.84% accuracy and 28.08% loss in the 49th epoch with a runtime of 24 min 40 s.

**Table 5.** Experiments and results of character-level CNN-LSTM (using imbalanced training dataset).

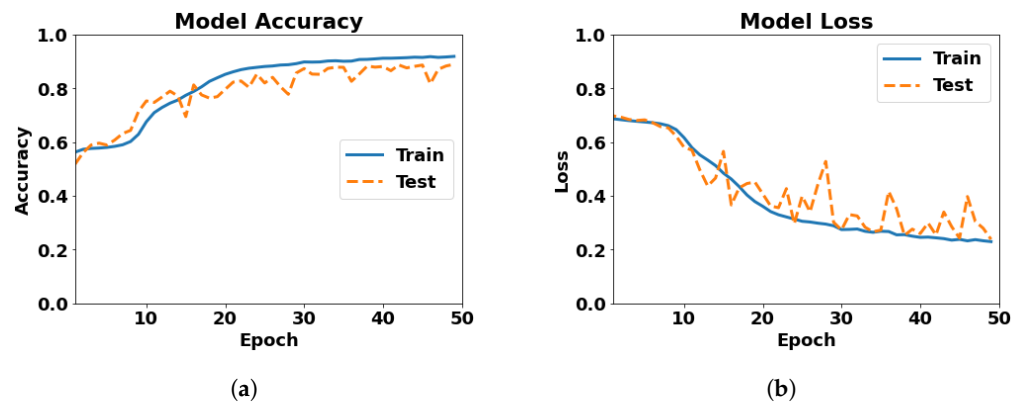
Exp No.	Model	Test Accuracy	Test Loss	No. of Epochs	Runtime	Best Accuracy	Worst LOSS
1	3 conv. layers of 100 filters, 1 LSTM layer of 60 neurons and dropout 0.5	0.8520	0.2963	50	24 min 40 s	0.8584 at 49th epoch	0.2808 at 49th epoch
2	3 conv. layers of 100 filters, 1 LSTM layer of 60 neurons and dropout 0.2	0.8369	0.3049	50	26 min 44 s	0.8608 at 49th epoch	0.2800 at 49th epoch
3	3 conv. layers of 100 filters, 1 LSTM layer of 60 neurons and dropout 0.4	0.8562	0.2808	50	20 min 51 s	0.8593 at 49th epoch	0.2885 at 49th epoch
4	3 conv. layers of 256 filters, 1 LSTM layer of 60 neurons and dropout 0.2	0.8644	0.2705	50	54 min 18 s	0.8644 at 50th epoch	0.2705 at 50th epoch
5	3 conv. layers of 256 filters, 1 LSTM layer of 60 neurons and dropout 0.5	0.8582	0.2952	50	60 min 52 s	0.8582 at 50th epoch	0.2952 at 50th epoch
6	3 conv. layers of 100 filters, 1 LSTM layer of 100 neurons and dropout 0.2	0.8493	0.3091	50	26 min 11 s	0.8500 at 47th epoch	0.3002 at 47th epoch
7	3 conv. layers of 100 filters, 1 LSTM layer of 100 neurons and dropout 0.5	0.8616	0.2781	50	25 min 17 s	0.8616 at 50th epoch	0.2781 at 50th epoch
8	3 conv. layers of 256 filters, 1 LSTM layer of 100 neurons and dropout 0.5	0.8587	0.2948	50	60 min 24 s	0.8587 at 50th epoch	0.2948 at 50th epoch
9	<b>3 conv. layers of 256 filters, 1 LSTM layer of 100 neurons and dropout 0.2</b>	<b>0.8657</b>	<b>0.2688</b>	<b>50</b>	<b>60 min 8 s</b>	<b>0.8657 at 50th epoch</b>	<b>0.2688 at 50th epoch</b>

In experiments 2 and 3, we reduced the dropout rate to 0.2 and 0.4, respectively, from the initial version of the architecture. The model in experiment 2 performed better in terms of accuracy (86.08%) compared to experiments 1 and 3. For experiment 4, we modified the hyperparameters with the number of filters as 256, the number of neurons as 60, and the dropout rate as 0.2. The model in experiment 4 performed better in terms of accuracy (86.44%) compared to the three previous experiments, but its runtime was higher (54 min 18 s). In experiment 5, we increased the dropout rate to 0.5 from the architecture in experiment 4, but did not notice any improvement in the results. Moreover, we modified the hyperparameters with the number of filters as 100, the number of neurons as 100, and the dropout rate as 0.2 for experiment 6 and 0.5 for experiment 7, but also did not see any improvement in both experiments. Furthermore, in experiment 8, we modified the hyperparameters with the number of filters as 256, the number of neurons as 100, and the dropout rate as 0.5, but still did not see any improvement in the results. In experiment 9, we decreased the dropout rate to 0.2 from the architecture in experiment 8. The model in experiment 9 performed the best in terms of accuracy (86.57%) compared to all the previous experiments, with a runtime of 60 min 8 s. The graphs of accuracy and loss for 50 epochs of training and testing from experiment 9 are shown in Figure 6.



**Figure 6.** Evaluation metrics for 50 epochs of training and testing with 3 convolution layers of 256 filters, 1 LSTM layer of 100 neurons, and dropout rate of 0.2 using imbalanced training dataset: (a) Accuracy and (b) Loss.

The model in experiment 9 has a precision of 67%, recall of 91%, and F1-score of 77%. Although we trained the model using an imbalanced dataset, the model did not favor the majority class (hate). However, in order to improve our results, we created another model with the same hyperparameters as in experiment 9 and trained it with the over-sampled version of the training dataset and evaluated its performance on the test set. The best performance of this model on the test set was an 88.97% accuracy and 23.85% loss in the 50th epoch with a runtime of 88 min 15 s, which is shown in Table 6. Hence, the accuracy of the model improved by 2.4% when training it with a balanced dataset. The model has a precision of 74%, recall of 88%, and an F1-score of 80%. Figure 7 shows the accuracy and loss for 50 epochs of training and testing of the model.



**Figure 7.** Evaluation metrics for 50 epochs of training and testing with 3 convolution layers of 256 filters, 1 LSTM layer of 100 neurons, and dropout rate of 0.2 using over-sampled training dataset: (a) Accuracy and (b) Loss.

**Table 6.** Experiments and results of character-level CNN-LSTM (using over-sampled training dataset).

Model	Test Accuracy	Test Loss	No. of Epochs	Runtime	Best Accuracy	Worst Loss
3 conv. layers of 256 filters, 1 LSTM layer of 100 neurons and dropout 0.2	0.8897	0.2385	50	88 min 15 s	0.8897 at 50th epoch	0.2385 at 50th epoch

Table 7 reports the results of running our models (Models b–e) and Elouali et al.’s [7] model (Model a) on the same dataset. Both our neural network architectures (CNN and CNN-LSTM) performed better (and similar at worse) compared to Elouali et al.’s [7] model in terms of accuracy, and significantly better in terms of runtime.

**Table 7.** Comparison of all the results.

Model No.	Model	No. of Epochs	Runtime	Accuracy	Loss
a	Character-level CNN using imbalanced training dataset (Elouali et al.'s [7] method)	50	156 min 18 s	0.8661	0.2704
b	Character-level CNN using imbalanced training dataset ( <b>our work</b> )	50	48 min 51 s	0.8652	0.2723
c	Character-level CNN using over-sampled training dataset ( <b>our work</b> )	50	78 min 57 s	0.8884	0.2661
d	Character-level CNN-LSTM using imbalanced training dataset ( <b>our work</b> )	50	60 min 8 s	0.8657	0.2688
e	Character-level CNN-LSTM using over-sampled training dataset ( <b>our work</b> )	50	88 min 15 s	0.8897	0.2385

Figure 8 reports the accuracy of prediction (hate/non-hate) of our models and compares it with Elouali et al.'s [7] model. The corresponding model for each model number in Figure 8 is listed in Table 7. CNN-LSTM performed slightly better than CNN, whether an imbalanced or over-sampled dataset was used for training the model. Moreover, Figure 8 shows that we outperformed Elouali et al.'s [7] model's accuracy by: (1) over-sampling the dataset to avoid the inherent imbalanced class labels and, in turn, biased results, and (2) adding an LSTM layer to the CNN architecture and pairing them for feature extraction and classification. We further performed more experiments by changing hyperparameters such as number of filters, number of neurons, and dropout rate, as reported in Table 5.

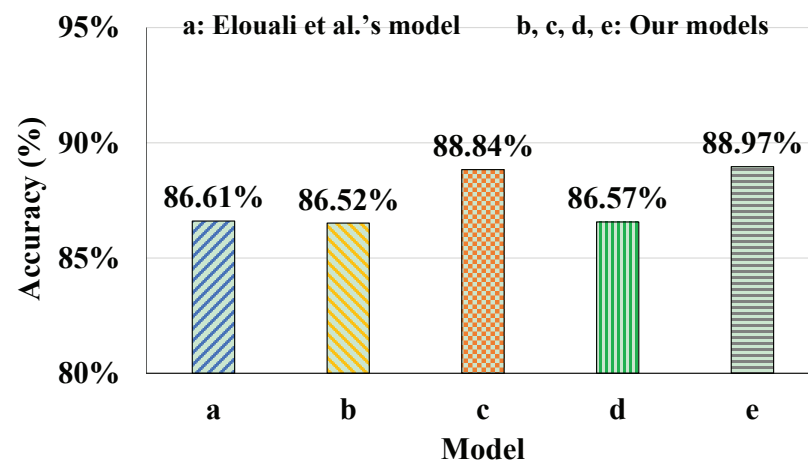
**Figure 8.** Accuracy comparison of different models.

Figure 9 shows a comparison between the time of execution of our models and that of Elouali et al.'s [7]. The corresponding model for each model number in Figure 9 is listed in Table 7. Figure 9 reports a significant improvement in runtime in our models. This is mainly due to removing emoticons encoded with UTF-8 BOM (Byte Order Marks) from the tweets. The time of execution of Elouali et al.'s [7] model trained with an imbalanced dataset was longer even when compared to our models trained with an over-sampled dataset, which is comparatively larger.

Figure 10 shows a comparison between the scalability of our models (Models c and e) and that of Elouali et al.'s [7] (Model a). In order to achieve the desired dataset size, we duplicated randomly chosen tweets. The corresponding model for each model number in Figure 10 is listed in Table 7. This experiment shows that all models are scalable because runtime increases proportionately with respect to the number of input records. However, the findings in Figure 10 further vindicate that our models successfully reduce



runtime without compromising the accuracy of prediction, thanks to the pre-processing step (Section 4.1).

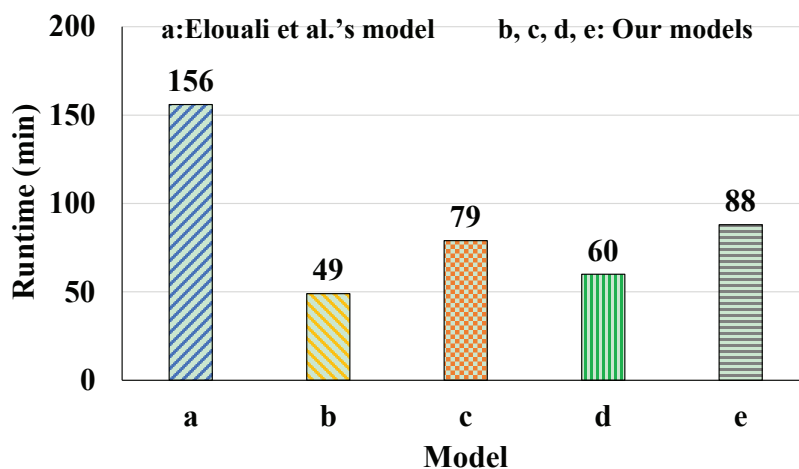


Figure 9. Runtime comparison of different models.

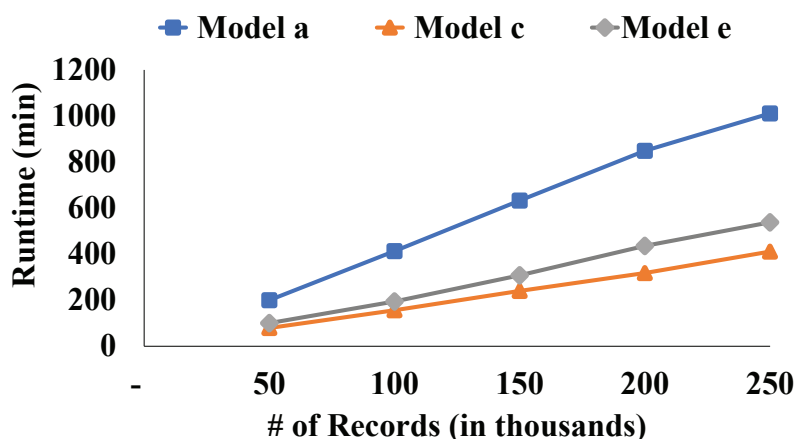


Figure 10. Scalability comparison of different models.

In summary, the experimental results on accuracy, runtime, and scalability against a closely related method from the literature indicate that data pre-processing plays a significant role in reducing processing time without compromising the accuracy of prediction.

### 6. Conclusions

This paper presented a study on the impact of data pre-processing on hate speech detection on social media platforms, in particular, Twitter. Data cleansing included 10 steps, such as the removal of URLs and hashtags. An efficient method was developed to accurately detect hate speech in text written in a mix of multiple languages including code-mixed, i.e., English and Hindi–English. The proposed method incorporates character-level embedding representations of text and deep neural networks (DNNs). Particularly, two model architectures were investigated, namely CNN and CNN-LSTM. A real-life dataset of published tweets was used to evaluate the proposed method in terms of accuracy, runtime, and scalability. The pre-processed dataset was split into training and test sets, where imbalanced and over-sampled versions of the training set were used to train the model. The experimental results suggest that data pre-processing significantly reduces runtime, without compromising the accuracy of prediction. Moreover, both the CNN and CNN-LSTM models trained with the over-sampled version of the training dataset increased the accuracy of prediction, outperforming a closely related method in the literature, i.e., Elouali et al.’s model [7].

Among our best-performing models, CNN-LSTM performed slightly better than CNN with an accuracy of 88.97%.

In the future, in addition to detecting hate speech in a mix of English and Hindi–English code-mixed tweets, the proposed method can be extended to include various other languages—pure and code-mixed. Moreover, a future research direction would be to consider the psychological effect of emoticons on social media users and study the significance of emoticons on hate speech. This can be achieved by: 1. Studying the impact of various emoticons on human perception, and 2. Integrating emoticons into building data models, which possess the ability to extract the emoticons’ contextual meaning within the provided text.

**Author Contributions:** Conceptualization, M.S.; Data curation, M.S.; Formal analysis, K.A.-H., M.S. and I.K.; Funding acquisition, K.A.-H.; Investigation, M.S.; Methodology, K.A.-H. and M.S.; Project administration, K.A.-H.; Resources, K.A.-H.; Software, M.S.; Supervision, K.A.-H.; Validation, K.A.-H. and I.K.; Visualization, M.S. and I.K.; Writing—original draft, M.S.; Writing—review and editing, K.A.-H. and I.K. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research is supported in part by the DSO-RIT Dubai Research Fund (2022-23-1003) from Rochester Institute of Technology—Dubai (RIT-Dubai).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** All the datasets used in this study are publicly available and are referenced in Section 3.

**Acknowledgments:** The authors would like to thank Rochester Institute of Technology—Dubai (RIT-Dubai) for supporting this research effort.

**Conflicts of Interest:** The authors have no competing interest to declare that are relevant to the content of this article.

## Abbreviations

The following abbreviations are used in this manuscript:

CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory
BiLSTM	Bi-Directional LSTM
RF	Random Forest
SVM	Support Vector Machine
LSVM	Linear SVM

## References

1. The Equilibrium Decodes the Impact of Social Media Addiction in 2022 and Its Coping Mechanism. *The Equilibrium*, 8 March 2022.
2. Nockleby, J.T. Encyclopedia of the American Constitution. In *Hate Speech*; Levy, L., Karst, K., Eds.; Macmillan: Detroit, MI, USA, 2000; Volume 3, pp. 1277–1279.
3. Hatzipanagos, R. Perspective | How Online Hate Turns into Real-Life Violence. *Washington Post*, 30 November 2018.
4. Kakkar, S. Supreme Court to Hear on November 22 Plea Seeking Direction to Centre to Take Steps to Deal with Hate Speech. *Live Law*, 12 November 2021.
5. Thomasson, E. German Cabinet Agrees to Fine Social Media over Hate Speech. *Reuters*, 5 April 2017.
6. McClure, T. New Zealand Moves to Toughen Hate Speech Laws in Wake of Christchurch Attacks. *The Guardian*, 25 June 2021.
7. Elouali, A.; Elberrichi, Z.; Elouali, N. Hate Speech Detection on Multilingual Twitter Using Convolutional Neural Networks. *Rev. D’Intelligence Artif.* **2020**, *34*, 81–88. [[CrossRef](#)]
8. Santosh, T.Y.S.S.; Aravind, K.V.S. Hate Speech Detection in Hindi-English Code-Mixed Social Media Text. In Proceedings of the ACM India Joint International Conference on Data Science and Management of Data, Kolkata, India, 3–5 January 2019.
9. Srivastava, A. India: The Land Of Diverse Languages And Scripts. *IITGN News*, 29 June 2020.
10. Kwok, I.; Wang, Y. Locate the Hate: Detecting Tweets against Blacks. *Proc. Aaai Conf. Artif. Intell.* **2013**, *27*, 1621–1622. [[CrossRef](#)]
11. Park, J.H.; Fung, P. One-step and Two-step Classification for Abusive Language Detection on Twitter. In Proceedings of the First Workshop on Abusive Language Online, Vancouver, BC, Canada, 4 August 2017; pp. 41–45. [[CrossRef](#)]

12. Bohra, A.; Vijay, D.; Singh, V.; Akhtar, S.S.; Shrivastava, M. 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, LO, USA, 6 June 2018; pp. 36–41. [[CrossRef](#)]
13. Singh, V.; Varshney, A.; Akhtar, S.S.; Vijay, D.; Shrivastava, M. Aggression Detection on Social Media Text Using Deep Neural Networks. In Proceedings of the 2nd Workshop on Abusive Language Online (ALW2), Brussels, Belgium, 31 October 2018; pp. 43–50. [[CrossRef](#)]
14. Alkomah, F.; Ma, X. A Literature Review of Textual Hate Speech Detection Methods and Datasets. *Information* **2022**, *13*, 273. [[CrossRef](#)]
15. Waseem, Z. Are You a Racist or Am I Seeing Things? Annotator Influence on Hate Speech Detection on Twitter. In Proceedings of the First Workshop on NLP and Computational Social Science, Austin, TX, USA, 5 November 2016; pp. 138–142. [[CrossRef](#)]
16. Magu, R.; Joshi, K.; Luo, J. Detecting the Hate Code on Social Media. *Proc. Int. Aaai Conf. Web Soc. Media* **2017**, *11*, 608–611. [[CrossRef](#)]
17. Gambäck, B.; Sikdar, U.K. Using Convolutional Neural Networks to Classify Hate-Speech. In Proceedings of the First Workshop on Abusive Language Online, Vancouver, BC, Canada, 4 August 2017; pp. 85–90. [[CrossRef](#)]
18. Mehta, H.; Passi, K. Social Media Hate Speech Detection Using Explainable Artificial Intelligence (XAI). *Algorithms* **2022**, *15*, 291. [[CrossRef](#)]
19. Agarwal, S.; Sonawane, A.; Chowdary, C.R. Accelerating automatic hate speech detection using parallelized ensemble learning models. *Expert Syst. Appl.* **2023**, *230*, 120564. [[CrossRef](#)]
20. Waseem, Z.; Hovy, D. Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. In Proceedings of the NAACL Student Research Workshop, San Diego, CA, USA, 13–15 June 2016; pp. 88–93. [[CrossRef](#)]
21. Mathur, P.; Sawhney, R.; Ayyar, M.; Shah, R. Did you offend me? Classification of Offensive Tweets in Hinglish Language. In Proceedings of the 2nd Workshop on Abusive Language Online (ALW2), Brussels, Belgium, 31 October–1 November 2018; pp. 138–148. [[CrossRef](#)]
22. Souza, G.; Da Costa-Abreu, M. Automatic offensive language detection from Twitter data using machine learning and feature selection of metadata. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–6. [[CrossRef](#)]
23. Davidson, T.; Warmusley, D.; Macy, M.; Weber, I. Automated Hate Speech Detection and the Problem of Offensive Language. *Proc. Int. Aaai Conf. Web Soc. Media* **2017**, *11*, 512–515. [[CrossRef](#)]
24. Watanabe, H.; Bouazizi, M.; Ohtsuki, T. Hate Speech on Twitter: A Pragmatic Approach to Collect Hateful and Offensive Expressions and Perform Hate Speech Detection. *IEEE Access* **2018**, *6*, 13825–13835. [[CrossRef](#)]
25. Geet d'Sa, A.; Illina, I.; Fohr, D. Classification of Hate Speech Using Deep Neural Networks. *Rev. D'Information Sci. Tech.* **2020**, *25*, 1. Available online: [https://hal.science/hal-03101938/file/SIIE\\_chap.pdf](https://hal.science/hal-03101938/file/SIIE_chap.pdf) (accessed on 5 October 2023).
26. Zampieri, M.; Malmasi, S.; Nakov, P.; Rosenthal, S.; Farra, N.; Kumar, R. Predicting the Type and Target of Offensive Posts in Social Media. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Minneapolis, MN, USA, 2–7 June 2019; pp. 1415–1420. [[CrossRef](#)]
27. Kumari, K.; Singh, J.P.; Dwivedi, Y.K.; Rana, N.P. Multi-modal aggression identification using Convolutional Neural Network and Binary Particle Swarm Optimization. *Future Gener. Comput. Syst.* **2021**, *118*, 187–197. [[CrossRef](#)]
28. Kumari, K.; Singh, J.P. Multi-Modal Cyber-Aggression Detection with Feature Optimization by Firefly Algorithm. *Multimed. Syst.* **2022**, *28*, 1951–1962. [[CrossRef](#)]
29. Kamble, S.; Joshi, A. Hate Speech Detection from Code-mixed Hindi-English Tweets Using Deep Learning Models. *arXiv* **2018**, arXiv:cs.CL/1811.05145.
30. Sreelakshmi, K.; Premjith, B.; Soman, K. Detection of Hate Speech Text in Hindi-English Code-mixed Data. *Procedia Comput. Sci.* **2020**, *171*, 737–744. [[CrossRef](#)]
31. Shekhar, S.; Garg, H.; Agrawal, R.; Shivani, S.; Sharma, B. Hatred and trolling detection transliteration framework using hierarchical LSTM in code-mixed social media text. *Complex Intell. Syst.* **2023**, *9*, 2813–2826. [[CrossRef](#)]
32. Kumar, R.; Reganti, A.N.; Bhatia, A.; Maheshwari, T. Aggression-annotated Corpus of Hindi-English Code-mixed Data. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan, 7–12 May 2018.
33. Kumari, K.; Singh, J.P.; Dwivedi, Y.K.; Rana, N.P. Bilingual Cyber-Aggression Detection on Social Media Using LSTM Autoencoder. *Soft Comput.* **2021**, *25*, 8999–9012. [[CrossRef](#)]
34. Batista, G.E.A.P.A.; Prati, R.C.; Monard, M.C. A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *SIGKDD Explor. Newsl.* **2004**, *6*, 20–29. [[CrossRef](#)]
35. Branco, P.; Torgo, L.; Ribeiro, R. A Survey of Predictive Modelling under Imbalanced Distributions. *arXiv* **2015**, arXiv:cs.LG/1505.01658.
36. Fernández, A.; García, S.; Galar, M.; Prati, R.C.; Krawczyk, B.; Herrera, F. *Learning from Imbalanced Data Sets*; Cambridge International Law Journal; Springer: Berlin/Heidelberg, Germany, 2018.
37. Zhang, X.; Zhao, J.; LeCun, Y. Character-level Convolutional Networks for Text Classification. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2015; Volume 28.

38. Zhang, Z.; Robinson, D.; Tepper, J. Hate Speech Detection Using a Convolution-LSTM Based Deep Neural Network. In Proceedings of the 2018 International World Wide Web Conference, Lyon, France, 23–27 April 2018; p. 10.
39. Alayba, A.M.; Palade, V.; England, M.; Iqbal, R. A Combined CNN and LSTM Model for Arabic Sentiment Analysis. In Proceedings of the Machine Learning and Knowledge Extraction, Hamburg, Germany, 27–30 August 2018; Holzinger, A., Kieseberg, P., Tjoa, A.M., Weippl, E., Eds.; Springer: Cham, Switzerland, 2018; pp. 179–191.

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.