information

*Article*

# Online Multilingual Hate Speech Detection: Experimenting with Hindi and English Social Media

**Neeraj Vashistha** * and **Arkaitz Zubiaga**

School of Electronic Engineering and Computer Science, Queen Mary University of London, London E1 4NS, UK; a.zubiaga@qmul.ac.uk
* Correspondence: ec19471@qmul.ac.uk

**Abstract:** The last two decades have seen an exponential increase in the use of the Internet and social media, which has changed basic human interaction. This has led to many positive outcomes. At the same time, it has brought risks and harms. The volume of harmful content online, such as hate speech, is not manageable by humans. The interest in the academic community to investigate automated means for hate speech detection has increased. In this study, we analyse six publicly available datasets by combining them into a single homogeneous dataset. Having classified them into three classes, abusive, hateful or neither, we create a baseline model and improve model performance scores using various optimisation techniques. After attaining a competitive performance score, we create a tool that identifies and scores a page with an effective metric in near-real-time and uses the same feedback to re-train our model. We prove the competitive performance of our multilingual model in two languages, English and Hindi. This leads to comparable or superior performance to most monolingual models.

**Keywords:** social media; hate speech; text classification



**Citation:** Vashistha, N.; Zubiaga, A. Online Multilingual Hate Speech Detection: Experimenting with Hindi and English Social Media. *Information* **2021**, *12*, 5. https://dx.doi.org/10.3390/info12010005

Received: 23 November 2020
Accepted: 18 December 2020
Published: 22 December 2020



## 1. Introduction

Hate speech is a characterisation of communication that is 'hateful', controversial, generates intolerance and in some way is divisive and demeaning. There is no legal definition of hate speech, but on several accounts accepted meaning of the term deals with communication in speech, behaviour or writing, remarks which are pejorative or discriminatory concerning a person or a group of persons, either directly or indirectly. Such remarks are based on their religion, ethnicity, nationality, descent, race, colour, gender or other identity factors [1].

Many countries have adopted new laws and frameworks have been constituted. However, due to the pervasive nature of online communications, only 3% of malicious communication offenders are charged [2]. This is because of a lack of clarity and certainty in this area. Several of these frameworks prohibit the incitement to discrimination, hostility and violence rather than prohibiting hate speech. There is a need for quick identification of such remarks and an automatic system which can identify and take measures to prevent the instigation and incitement.

There are several examples of hate speech (Figure 1) which either implicitly or explicitly target an individual or a group of individuals and inflict mental pain that may eventually cause of social revolts or protests.

Reminding feminists world-wide that they are only mad because they're ugly...

Good morning WHITE people! F* you N*

**Figure 1.** Examples of hate text sequences.

In this work, we study the existing methods designed to tackle hate speech. We curated the data from the most prominent social media platform, Twitter. We also combine the existing datasets into one, as these are pre-annotated and previous researchers have used these to target several cases of hate speeches. Individually, these are small datasets, but, after we combine the existing datasets into one dataset, we obtain a large corpus of hate speech sequences. This dataset is a combination of the hateful content of five different categories: sexual orientation, religion, nationality, gender and ethnicity. We create models which classify the content of a text as hateful, abusive or neither. The total size of the dataset is around seventy-six thousand samples, and, due to the high variance and low bias, we avoid sub-categorisation of hate classes.

We also study some state-of-the-art models which claim to give superior accuracy. Since these models are built on specific languages (English or Hindi), we utilise the work and propose our model which is multilingual. Finally, we use our optimised models to create a simple tool which identifies and scores a page if hateful content is found and uses the same as the feedback to re-train the model. While the vast majority of previous works have investigated the development of hate speech detection models for specific languages [3], here we propose a multilingual model which we experiment in two languages, English and Hindi, leading to competitive performance and superior to most monolingual models.

The main contributions of this work are as follows:

- We create a system which is trained on a sufficiently large corpus of hate speech text.
- A multilingual system that can work on different languages is proposed, and another language be added and trained quickly using our transfer learning models.
- We propose system which can learn a new form of hateful remark or Zipfian nature of language by re-training in an online environment.
- The resulting system has models which are simple, lightweight and optimised to be used in a near-real-time online environment.

In this way, the system holds potential as a product for safer social media utilisation as well as reduces the need for human annotators and moderators to tag disturbing online messages.

The rest of the paper is organised as follows. An overview of hate speech datasets and existing models is provided in the Related Work Section. In the Model Section, we critically analyse the existing models and discuss their limitations. We also discuss our model and several optimisations we performed to achieve a desirable performance score. Furthermore, we talk about the feedback mechanism for the optimised model and its usage in an online environment.

## 2. Related Work

Here, we discuss related work on hate speech datasets, hate speech detection models and different approaches. This serves as a motivation for our work and helps us bridge our study to the existing research available.

### *2.1. Related Datasets*

As mentioned above, six publicly available datasets [4–9] are manually curated and of modest size. In the following, we discuss the characteristics and the generation process of each of these datasets. Using these annotated datasets, we create our own resource of hate speech sequences.

#### 2.1.1. Data Gathering Process

The data in the online diaspora can be curated from various technology companies and social media platform providers such as Facebook, Google, Internet Service Provider Association, Oath, Twitter, Snap Group Limited, etc. Twitter's Public API, with its ease of use, high availability and accessibility, it is one of the most sought and targeted online social media platforms. We can set up a mechanism, and relevant data for hate speech can

be sourced. The dataset we use in this study comes from Twitter and is labelled by the annotators into pre-specified categories and subcategories of hate speech.

The examples sourced from Twitter are based on keyword matching with terms which have a strong connection with hate. These are annotated and tagged as abusive or hateful. This task has to be done manually and is very time-consuming. The same process has been adopted for code-mixed and pure Hindi language. Since the sample space of P. Mathur et al. [9] and HASOC2019 [4] is small, we try to update these datasets with our own curated samples.

To describe the multilingual aspect, the dataset consists of three different types of target language: English, Hindi and code-mixed Hindi. Table 1 describes various attributes of the dataset.

**Table 1.** Multilingual dataset attributes.

| Language | Attributes | | |
|---|---|---|---|
| | *Total Records* | *Vocab Size* | *Max Seq. Len* |
| English-EN | 61,319 | 23,216 | 89 |
| Hindi-HI | 9330 | 6082 | 104 |
| Hindi Code-Mix | 3161 | 6094 | 48 |

Although there are some limitations such as small sample text size of 160 characters or under-representation of all forms of hate speech sequences, this does not limit the use of data. Other issues related to this kind of dataset is the filtering of unwanted details such as email address, URL, date, phone number, username, etc. due to which textual and contextual information is lost. We also aim to ensure that the collation of different datasets does not impact the attributes. As these datasets are tagged by different authors as per their own perception and with the daily evolution of slang and jargon, evaluation becomes a tedious task.

#### 2.1.2. Existing Datasets

**HASOC2019**: The dataset presented in [4] shares task on hate speech and offensive content identification in Indo-European languages and contains around 7005 posts in English and 9330 posts in the Hindi language, with 36% and 52% abusive content in the respective languages. The task is focused on three sub-tasks:

- Whether the tweet text is either hate or offensive, or neither
- Whether tweet text is hate, offensive or profane
- Whether the tweet text is targeted towards an individual/group or is untargeted

For this study we only consider the data presented for the first sub-task.

**TDavidson et al.**: The dataset presented in [5] includes tweets labelled by human annotators into three categories: hate speech, offensive or neither. There are 24,802 English tweets with only 6% abusive content.

**ElSherif et al.**: The dataset presented in [6] is procured from Twitter and it is categorised into seven different categories of hate. The data are sought from [5,10] with their own annotation from No Hate Speech Movement (www.nohatespeechmovement.org) and Hatebase (hatebase.org). The total number of samples is 10,760 with abusive content accounting for around 10%. This dataset is in the English language.

**Ousidhoum et al.**: Here, the author categorised Twitter data into 46 different sentiments of hateful, offensive, fearful, abusive, disrespectful, normal and different combinations of each. The dataset presented [7] is in English, and it also captures the annotators' sentiments, directness, intended targets and groups. These attributes are important, but, to create a homogeneous dataset, they have been left out from this study. The total size of this dataset is around 5647 instances which are reorganised for our purpose into the three categories hateful, abusive or neither, with abusive content of around 26%.

**SemEval 2019 Task 5**: The dataset presented [8] comprises 12,906 text instances, of which 42% are abusive in nature. Again, the data are sourced from Twitter. This is a shared task with three sub-tasks in it. The first task is a binary task to classify if the text instance is hate or not, which is what we consider for our study. The other tasks are within hate, whether the text is directed towards a group or individual and whether the text is aggressive or not. This dataset is hate speech sequences against immigrants and women in English language.

**PMathur et al.**: The dataset presented in [9] contains tweets of offensive nature along with a profanity word list in code-mixed Hindi. Code-mixed Hindi is a phonetic translation of Hindi words written in the English language. These tweets are annotated into three different categories of normal, abusive and hate. The resulting size of the profanity word list is around 226 words in code-mixed Hindi, with the meaning of each in English, profanity score and Devanagari script Hindi word. The size of the dataset is around 3189 tweets with 55% of hateful nature.

For our study, these datasets (https://github.com/neerajvashistha/online-hate-speech-recog/tree/master/data/) were curated according to the guidelines provided by the author [9]. Originally, the Hindi language dataset [4] contains only two classes, non-hate and hate, so we added more data in accordance with the above-mentioned datasets. The summary of all the datasets used in this research is provided in Table 2.

**Table 2.** Datasets hate composition.

| Dataset Name | *Total Records* | *Hate %* | *Abuse %* |
|---|---|---|---|
| HASOC2019—EN | 7005 | 36.38 | — |
| HASOC2019 - HI | 9330 | 52.92 | - |
| TDavidson et al. | 24,783 | 77.43 | 5.77 |
| ElSherif et al. | 10,760 | 90.94 | - |
| Ousidhoum et al. | 5647 | 65.66 | 22.63 |
| SemEval 2019 Task 5 | 12,906 | 42.15 | - |
| PMathur et al. | 3189 | 55.34 | 9.5 |
| Total | 76,403 | 61.84 | 4.55 |

The data from all sources were collated and sorted into a homogeneous form, the details of which are illustrated in the Model Section.

### *2.2. Related Models*

Davidson et al. [5] followed a simple approach and created a simplistic model. They created simple feature vectors for each sample instance. The feature vector consisted of Part of Speech (POS), term frequency–inverse document frequency (TFIDF) vectors, etc. The final model was a logistic regression and linear support vector machine (SVM) which claimed to achieve an overall precision of 0.99, recall of 0.90 and F1 score of 0.90.

As we expanded the dataset for our experiments, the 94% accuracy for English language dataset claimed by the author was not applicable in this case, and therefore we had to re-run it again. This is why in our case the accuracy of the model was close to 68%. Similar to this work, Ref. [11] discussed the use of SVM, Random Forest and other simpler models to achieve comparable performance score in classifying hate speech sequence. We could not use Hindi or code-mixed Hindi because the embeddings used by the author, Babylon multilingual word embeddings [12] and MUSE [13], were not compatible with our Hindi or code-mixed Hindi dataset.

Ref. [7] created two types of models: Bag-of-word (BOW), a feature-based logistic regression model, and a deep learning-based model [14] containing Bi-directional Long Short Term Memory (BiLSTM) layer with one hidden layer. They claimed that, due to the small size of the dataset, deep learning-based models performed poorly. They suggested using Sluice network [15], as it is suitable for loosely related tasks such as the annotated aspects of the corpora. State-of-art performance scores are obtained using Sluice networks.

One of the initial research on hate speech detection from Hindi–English tweets was done by [16]. The research was based on 4575 code-mixed Hindi tweets. They used features such as character n-gram, emoticon count, word n-gram, punctuation count and applied dimension reduction techniques. An accuracy of 71.7% on SVM and 66.7% on Random Forest was obtained.

The most relevant research done for code-mixed Hindi dataset was done by [9]. They claimed that term frequency, inverse document frequency (TF-IDF) and BoW features with SVM model gave peak performance when compared with other configurations of baseline supervised classifiers. The other features such as Linguistic Inquiry and word count features (LIWC) [17], profanity vector [18] with the glove and Twitter embeddings gave best results against the baseline model, as argued by the authors. In addition, a transfer learning-based approach called Multi-Channel Transfer Learning-based Model (MTLM) achieved on its best configuration a precision of 0.851, recall of 0.905 and F1 score of 0.893. Similar scores have been obtained in our study as well.

Ref. [19] applied deep learning-based approaches. In the experiment, the author created a sub-word level LSTM model and Hierarchical LSTM model with attention-based phonemic sub-words. The architecture of these models is important to research in this field as attention-based models perform highly in comparison to basic deep learning-based models. Hierarchical LSTM with attention achieved an accuracy of 66.6% while sub-word level LSTM model scored 69.8%.

In the work of [20], three deep learning models using domain-specific word embeddings were created. These word embeddings are available and comprise of 255,309 code-mixed Hindi text, which the author collected from Twitter. The word embeddings are trained using gensim's word2vec model and used in the three deep learning models. For 1D-CNN, an accuracy of 82.62% was achieved, which was the highest, against BiLSTM with 81.48% and LSTM with 80.21%. Similar to this work, Ref. [21] dedicated his research to discovering the best embeddings to predict the occurrence of hate speech in the English language.

In the above-mentioned research, we can see that all the models were created to process only a single language at a time. In our project, we focused on creating a single model that will inculcate multiple languages at the same time, leading to a more generalisable approach. The only exception of a multilingual model found in the literature is that by [22], who focused on English and Chinese. However, the authors used third party translation APIs, such as Google translation API, to convert the raw text into English or Chinese.

In the following sections, we propose a Logistic regression model and two Deep Learning-based models for both English and Hindi (code-mixed Hindi) languages. The model architecture for all the languages is the same; however, the feature building process is different for different languages.

## 3. Model

In this section, we discuss the model-building procedure. We are building a system that learns from the feedback, is multilingual and is trained on large hate sequence text. We also provide optimisation carried out and performance verification.

### *3.1. Data Preprocessing*

Since the data in our use case are sourced from various datasets [4–9], it becomes essential to organise them into a homogeneous form. To achieve this, we consider the text and the class type of the existing datasets. In many datasets, as shown in Table 2,

the data are classified as hate or not, and abuse class is not present. The original text is taken without any formatting. The class types (if necessary) are converted from original to hate, abusive or normal. To identify the source of the text, we labelled each record instance with specific dataset identifiers.

Samples for Hindi and code-mixed Hindi are sourced from [4,9]. This dataset is relatively small in comparison to the English language dataset. Thus, we updated these data by adding new tweet samples to Hindi and code-mixed Hindi dataset by following the below procedure.

1. We added and updated the profane word list provided by [9] with more words in code-mixed Hindi language.
2. We used this profane word list and added corresponding Hindi Devnagari scripted text against each code-mixed Hindi profane word.
3. To assign classes to newly curated text from Twitter Public API, we followed the guideline below.
   (a) Tweets involving sexist or racial slur to target a minority may contain abusive word and are annotated as hate.
   (b) Tweets which represent undignified stereotypes are marked as hate.
   (c) Tweets which contain problematic hashtags are marked as hate.
   (d) Tweets which contain only abusive words are tagged as abusive.
   (e) Other tweets are marked as normal.

   Some typical instances of our seventy-six thousand data are shown in Table 3.

**Table 3.** Example texts.

| Dataset Type | Example | Class Type |
|---|---|---|
| English-EN | I am literally too mad right now a ARAB won #MissAmerica | Hateful |
| English-EN | Black on the bus | Hateful |
| English-EN | I can not just sit up and HATE on another bitch. I got too much shit going on! | Abusive |
| Hindi-HI | ये लिटन की गाँड में लिप्टन की चाय डालता हूँ अभी विथ गर्म केतली | Hateful |
| Hindi Code-Mixed | Main jutt Punjabi hoon aur paka N league. Madarchod Imran ki Punjab say nafrat clear hai. | Hateful |

One of the major issues related to Twitter text and to which other studies have indicated as the potential cause for degrading in performance is the small sequence of text followed by ever-evolving use of unknown words such as slang and hashtags. In Table 1, Max Seq. Len. denotes the maximum sequence length of the various language datasets; the sequence length for all the languages is very small (less than 100), but the vocabulary size (vocab. size) is way too high. Ref. [23] described the limitations of unusually large vocabulary leading to poor performance.

In our research, when we performed exploratory data analysis (EDA), we concluded that, due to the above-mentioned issues, the performance of our classifier was also being affected. Thus, during prepossessing of text, we decided to use ekphrasis [24]. This preprocessor performs the following:

- Normalise the url, email, percent, money, phone, user, time, date and number in Twitter text.
- Annotate the hashtag, allcaps, elongated, repeated, emphasis, censored and words in the text.
- Segment words.
- Convert Hashtags to unpacked word list (if possible).

- Convert English contractions such as "can't" and "we'll" into "cannot" and "we will", respectively.
- Tokenise the sentences into words.
- Convert emoticons and slang into actual expressions/phrases.

The output from this preprocessing technique contains more information than previously studied research. Although ekphrasis works well for the English language, it does not have a multilingual capability built into it. Therefore, we leveraged [25] Indic NLP and [26] NLTK library support for Hindi Language. During the EDA, we observed some issues related to unpacking of slang, emoticons and English contractions in ekphrasis and tokenisation issues in Indic NLP [25]. These are now being rectified by raising an issue on Github and by making an open-source contribution to the original work.

### *3.2. Model Building*

In previous research [5,9], we found that BoW- or TFIDF-based feature vectors along with other features tend to work best with logistic regression-based models, and they generally give a fine baseline model performance. A similar approach was carried out in this study.

We created a logistic regression model as the baseline model. We applied an L2 penalty giving equal class weights to three classes. To comprehend the sheer volume of data and to make the model converge, the model's hyperparameter maximum iteration was set to 5000 iterations for English and 3000 for Hindi and code-mixed Hindi. Other hyperparameters were obtained in a grid search, with five fold cross-validation, and the performance scores are described in Table 4.

**Table 4.** Performance Comparison of F1 scores and accuracy between classes for Logistic regression model.

| Dataset Type | *F1-Neither* | *F1-Hate* | *F1-Abuse* | *Acc* |
|---|---|---|---|---|
| EN | 0.68 | 0.47 | 0.80 | 0.687 |
| HI | 0.95 | 0.96 | - | 0.956 |
| HI-Code Mix | 0.84 | 0.62 | 0.91 | 0.855 |

After building a baseline model, we explored the possibility of building a Hierarchical Deep Neural Network by combining several Convolutional Neural Network (CNN) filters into Bi-directional Long Short Term Memory network (BiLSTM). During the EDA, we discovered valuable sequential information in Twitter text for each class. When we applied a BiLSTM layer on top of a CNN layer, we could capture the sequential information as well as the low lying textual representation. Thus, we improved the simple contextual BiLSTM classification with use of CNN layers. This model has the following architecture:

1. Word Embedding—to capture and convert text into sequences
2. CNN—to capture low lying information using different filter size
3. BiLSTM—to capture contextual information of the sequence
4. Fully-connected output

This model was inspired by the works of [27–29]. In these studies, with the use of CNN, the character level property of the text is explored.

Figure 2 describes the architecture and different layers. The word embedding layer converts sparse representation of word sequences into dense vector representation. The three CNN layers consist of three parallel convolutional; two-dimensional filters, of sizes 3, 4 and 5 with batch normalisation; and max pooling layer. This CNN layer is time distributed over the LSTM layer, which captures the contextual information and finally the model is fed forward to a dense layer, where sigmoid activation layer performs the classification. This model is built to analyse the use of deep neural networks and find out whether there is any improvement in the previous benchmarks with respect to models performance.

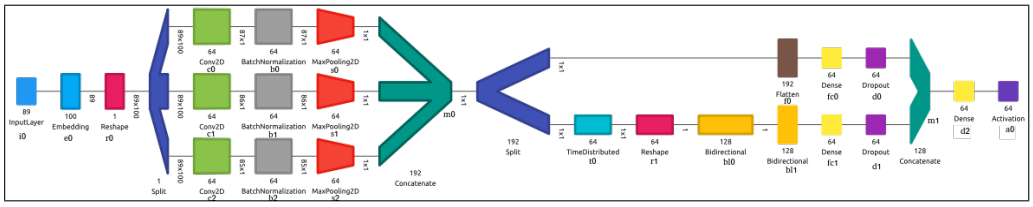

**Figure 2.** CNN LSTM network.

The above network is trained on all three datasets. After optimisations, the results obtained are relatively similar to the logistic regression model.

Table 5 describes the performance of three language datasets. This model was trained on GPU. Thus, the model convergence time was much quicker in comparison to the logistic regression model.

We used random search to find the parameters and hyperparameters. Table 6 describes the parameters which resulted in the best performance.

**Table 5.** Performance comparison of F1 scores and accuracy between classes for CNN-LSTM model.

| Dataset Type | *F1-Neither* | *F1-Hate* | *F1-Abuse* | *Acc* |
|---|---|---|---|---|
| EN | 0.74 | 0.63 | 0.72 | 0.78 |
| HI | 0.86 | 0.86 | - | 0.85 |
| HI-Code Mix | 0.83 | 0.62 | 0.70 | 0.86 |

**Table 6.** Parameter optimisation for CNN LSTM model.

| Dataset Type | *Epochs* | *Batch* Size | *Optimiser* | *Learning* Rate | *Dropout* | *Hidden* Size |
|---|---|---|---|---|---|---|
| EN | 22 | 30 | sgd | 0.001 | 0.2 | 64 |
| HI | 20 | 30 | sgd | 0.001 | 0.1 | 32 |
| HI-Code Mix | 22 | 30 | adam | 0.01 | 0.2 | 64 |

In the previous model, the word embeddings are learned from data that have low dimension and are denser than TFIDF features. However, the model seemed to be less performant than the logistic regression model built previously. Therefore, in the next model, we incorporate Bidirectional Encoder Representations from Transformers (BERT) [30] to leverage contextual word embeddings in place of CNN layers we previously added to the CNN LSTM model. This model has the following architecture:

1. Pre-trained BERT Embedding—to capture contextual word embeddings
2. Bidirectional LSTM—to capture contextual information of the sequence
3. Fully-connected output

Ref. [31] claimed to achieve a best in class F1 and GLUE score of above 90% for SQUAD 1.1 dataset classification. Using the transfer learning method, we leveraged the pre-trained embeddings of this transformer model into our CNN LSTM model. There are multiple pre-trained embeddings of various sizes (layers and hidden nodes) and languages. In our study, for English and code-mixed Hindi dataset, we used BERT-Base, Uncased: 12-layer, 768-hidden, 12-heads and 110 M parameters. For Hindi dataset, we employed BERT-Base, Multilingual Cased: 12-layer, 768-hidden, 12-heads and 110 M parameters.

Figure 3 describes the architecture and different layers. Instead of word embedding and CNN layers, for English, we used 24-layer contextual aware BERT language embeddings. There are only 12 hidden layers in the pre-trained model for BERT to support multilingual aspects which we implemented for our Hindi dataset. The BERT model results in 768 hidden layers, which are then time distributed over the BiLSTM layers and finally concatenated on a dense layer. For classification, the sigmoid activation layer is applied to the dense layer [32] used transfer learning approach and built a BERT-based model, but our model is architecturally different from the one proposed by them. In our study, we

split the BERT model's dimensions and later concatenated the results of BERT-LSTM for classification.

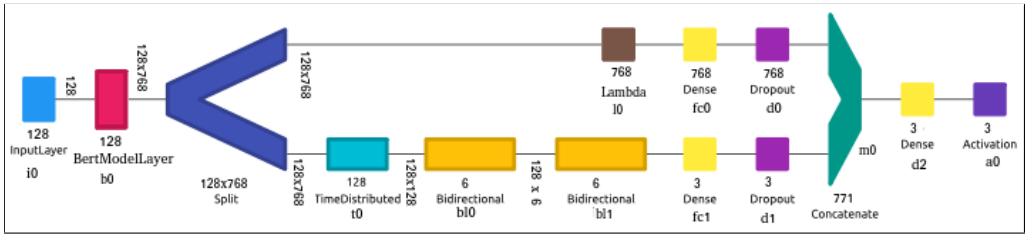


**Figure 3.** BERT-based network.

Table 7 describes the performance score for the BERT based model. The model was trained on Google's Tensor Processing Unit (TPU) and requires different mechanisms to process the data and feed them into the network. The model's performance is better than the logistic regression model and CNN LSTM model.

**Table 7.** Performance comparison of F1 scores and accuracy between classes for BERT model.

| Dataset Type | *F1-Neither* | *F1-Hate* | *F1-Abuse* | *Acc* |
|---|---|---|---|---|
| EN | 0.75 | 0.55 | 0.90 | 0.80 |
| HI | 0.94 | 0.94 | - | 0.95 |
| HI-Code Mix | 0.83 | 0.65 | 0.92 | 0.86 |

In Table 8, we describe the best parameters obtained using the technique of random search and evaluation.

**Table 8.** Parameter optimisation for BERT model.

| Dataset Type | *Epochs* | *Batch* Size | *Optimiser* | *Learning* Rate | *Dropout* | *Hidden* Size |
|---|---|---|---|---|---|---|
| EN | 13 | 121 | sgd | 0.01 | 0.2 | 3 |
| HI | 15 | 9 | sgd | 0.01 | 0.2 | 3 |
| HI-Code Mix | 8 | 11 | sgd | 0.01 | 0.2 | 3 |

The models developed in this study are oriented to be used in an online environment. Therefore, it becomes important that we pursue a state-of-the-art model which can be re-trained in a given frame of time within a resource-constrained environment. It is critical to analyse the prediction time as well as the time taken by the model to train.

Table 9 summaries some of the important performance features of the models built thus far.

- The logistic regression model takes an excessive amount of time for preprocessing and training. The BERT and CNN LSTM models take much less time in comparison. This is due to the fact that the logistic regression model requires TFIDF features on the entire vocabulary size.
- The model size for logistic regression models are 1000 times less than the BERT model and 100 times less than CNN LSTM model. Thus, the inference time for the logistic model is the least as compared to other models.
- The logistic regression model is well suited for smaller datasets as it is trained on CPU. For larger datasets, GPU- or TPU-based CNN LSTM- or BERT-based models could be used to reduce the training and inference time.
- One of the reasons for high inference time for CNN LSTM models is the inability to find the right set of hyperparameters, and thus we observed a higher performance for BERT models as compared with other models.

**Table 9.** Model evaluation and performance scores.

| Model | *Preprocess* Type | *Training* Time | *Infer.* Time | *System* Req. | *Model* Size | *Test* Acc. |
|---|---|---|---|---|---|---|
| LR-EN | 126 s | 9287 s | 0.7 s | CPU | 498.8 KB | 68% |
| LR-HI | 14,927 s | 412 s | 0.7 s | CPU | 161.2 KB | 95% |
| LR-HI Code-Mix | 435 s | 210 s | 0.7 s | CPU | 402.2 KB | 85% |
| CNN-LSTM-EN | 41 s | 340 s | 13 s | GPU | 19.9 MB | 78% |
| CNN-LSTM-HI | 1453 s | 68 s | 11 s | GPU | 5.7 MB | 85% |
| CNN-LSTM HI-Code-Mix | 0.6 s | 27 s | 13 s | GPU | 6.8 MB | 83% |
| BERT-EN | 28 s | 726 s | 0.9 s | TPU | 438.6 MB | 80% |
| BERT-HI | 0.8 s | 1037 s | 0.9 s | TPU | 712.2 MB | 95% |
| BERT-HI Code-Mix | 0.1 s | 218 s | 0.9 s | TPU | 438.6 MB | 86% |

## 4. Results and Discussion

We show that the logistic regression model supplemented with TFIDF and POS features gave relatively good results in comparison to other models. However, the time taken for the model to converge is very long. The results presented are in line with those of previous research [5,9,33].

The deep learning model gave similar performance scores without much feature engineering and the model converged quickly too. Such results can be attributed to the use of embedding layers in the neural network. We also tried to use the glove and Twitter embedding layers, but, due to the high number of unknown words, desirable results were not obtained.

The logistic regression model has fewer system requirements, is very quick to infer a single sample instance and has great performance scores. Due to its inability to scale on a large dataset and the time taken to build the model is very high, it can only be used as a benchmark for other models and cannot be used in an online environment, where models are continuously re-trained on the feedback loop.

The CNN LSTM model is an average model which has a mediocre performance but its performance can be perfected. The random search optimisation used in this study is a test-driven approach. Other optimisation techniques such as Bayesian optimisation can find better hyperparameters. However, the time taken to build the model will increase considerably. The CNN LSTM model requires moderate GPU processing and infers a single sample instance in near-real-time. Thus, we utilise this model in our online web application.

The BERT model is a high-performance state-of-the-art model. It has the highest accuracy among all the models. If the system requirement (TPU) criteria are fulfilled, it can be used in an online environment. This model has much less training time and is very quick to infer a single sample instance. The only constraint is the use of the TPU. Although the model size is 1000 times the size of a logistics regression model, memory and disk space consumption is hardly a worry these days due to their easy availability. Unlike GPU, the cost of a single TPU instance is very high. However, the scope of TPU usage as the mainstream processing unit for machine learning in the future is high. Thus, the research done in this study is futuristic and this novel state-of-the-art model is very much applicable to be used in the real world.

To give a fair comparison of our models to existing ones, we apply our models to the datasets in isolation. We find that our models outperform on most of the datasets. Table 10 shows these results. The dataset was segmented in the exact proportions of test and train as it was done in the original research. Further, out of the many models described here, we chose the best results (after optimising it to work on smaller datasets) of CNN LSTM model as it works more quickly in loading the data as well as giving inference. The results

are consistent with the performance and the performance is consistent across the datasets as well.

**Table 10.** Comparison of different dataset models with our model.

| Datasets | Existing Best Scores | | | Our Model Best Scores | | |
|---|---|---|---|---|---|---|
| | *Precision* | *Recall* | *F1* | *Precision* | *Recall* | *F1* |
| HASOC 2019 EN | n.a. | n.a. | 0.79 | 0.91 | 0.9 | **0.9** |
| HASOC 2019 HI | n.a. | n.a. | **0.81** | 0.87 | 0.82 | **0.81** |
| Davidson et al., 2017 | 0.91 | 0.9 | 0.9 | 0.93 | 0.9 | **0.92** |
| ElSherif et al. | n.a. | n.a. | n.a. | 0.85 | 0.86 | **0.83** |
| Ousidhoum et al., 2019 | n.a. | n.a. | **0.94** | 0.91 | 0.91 | 0.9 |
| SemEval 2019 Task 5 | 0.69 | 0.68 | 0.65 | 0.83 | 0.8 | **0.8** |
| Mathur et al., 2018. | 0.85 | 0.9 | **0.89** | 0.88 | 0.9 | 0.87 |

We also undertook and extended our study to compare our models with the state-of-the-art models RoBERTa [34] for English and code-mixed Hindi and XLM-R (cross-lingual RoBERTa model) [35] for Hindi. We found that our model's performance is comparable to the existing pre-trained models. Our BERT-based model and CNN-LSTM model performance match fairly against RoBERTa. Figure 4 describes the accuracy scores of different models in different languages.

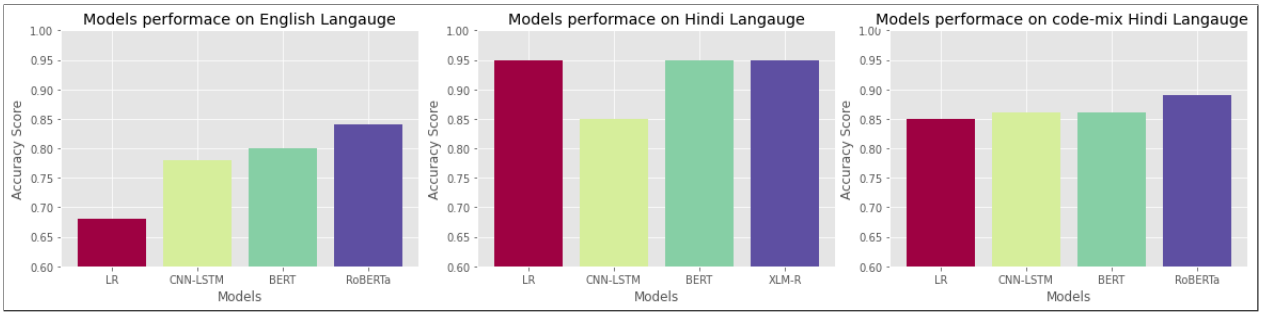


**Figure 4.** Our model comparison with other state-of-the-art models.

*Ablation Study and Modes of Errors*

**Performance layer-by-layer, without fine-tuning.** To understand which layers are critical for classification performance, we analysed results on the English dataset for CNN-LSTM- and BERT-based models. We checked the importance of CNNs and LSTM in CNN-LSTM network and LSTM in the BERT-based model.

In Figure 2, we have three parallel convolution kernels that are convoluted with the reshaped input layer to produce a tensor of outputs each are of shape $1 \times 6$, which are then concatenated at m0 resulting into a $1 \times 192$ tensor. As explained above, the main reason to employ three parallel CNNs is to learn textual information in each utterance, which is then used for classification. The output of CNN is time distributed in t0 over Bidirectional LSTMs (bl0,bl1). The outputs of both CNNs and LSTMs are passed through dense layer fc0 and fc1, respectively, and are finally concatenated at m1 before passing through another dense layer fc2 and activation a0. We focus the usage of first CNN C1 (c0,b0,s0), second CNN C2 (c1,b1,s1) and third CNN C3 (c2,b2,s2) along with the role of BiLSTM bl0 and bl1, on the performance of the entire network. The first five rows of Table 11 give the details of the performance of our network on a layer-by-layer basis. We understand that both CNN and BiLSTM are important in the network to attain good performance score. If we remove

the CNN layers entirely, then we get an average model but there is a considerable increase in performance when CNN layers are added.

**Table 11.** Performance of CNN-LSTM and BERT model layer-by-layer.

| Layers | Model | *F1-Neither* | *F1-Hate* | *F1-Abuse* | *Acc* |
|---|---|---|---|---|---|
| Without C1 and C2 and C3 | CNN-LSTM | 0.60 | 0.59 | 0.60 | 0.60 |
| C1 only | CNN-LSTM | 0.64 | 0.63 | 0.62 | 0.62 |
| C1 and C2 | CNN-LSTM | 0.70 | 0.73 | 0.73 | 0.72 |
| Without bl0 and bl1 | CNN-LSTM | 0.54 | 0.56 | 0.53 | 0.53 |
| bl0 only | CNN-LSTM | 0.70 | 0.69 | 0.70 | 0.71 |
| Without bl0 and bl1 | BERT | 0.71 | 0.67 | 0.83 | 0.79 |
| bl0 only | BERT | 0.75 | 0.62 | 0.80 | 0.80 |
| Without FT | CNN-LSTM | 0.74 | 0.61 | 0.70 | 0.76 |
| With FT | CNN-LSTM | 0.74 | 0.62 | 0.73 | 0.78 |
| Without FT | BERT | 0.75 | 0.62 | 0.88 | 0.79 |
| With FT | BERT | 0.75 | 0.55 | 0.91 | 0.80 |

In Figure 3, we use BERT + BiLSTM to capture both utterance level and sentence level understanding of hate speech sequence. To understand how the performance is affected, we removed bl1 and bl0, simultaneously. Rows 6–7 present the performance of the model when both BiLSTM layers are removed and when only one is kept. The overall performance of the model is unaffected with or without the presence BiLSTM as there are 109,584,881 parameters of which only 102,641 belong to BiLSTM part of the network, but, with the split and BiLSTM layers, a slight improvement is added nonetheless.

**Performance layer-by-layer, with fine-tuning.** We now analyse the results from our CNN-LSTM and BERT models after having fine-tuned their parameters on the English dataset. The improvement is marginal in both the cases (Table 11, Rows 8–11): fine-tuning increases the accuracy by 1–2%. The best hyperparameters to give this result are shown in Tables 6 and 8.

**Modes of Error** Further, we discuss some categories of errors that were observed in the deep learning and logistic regression models:

- The noisy and repetitive nature of the data present on social media creates a skewness in the class distribution. Although it is taken care of by the hyperparameter tuning, it still caused overfitting in both logistic regression models and CNN LSTM models.
- The code-mixed words tend to be biased as they appear at specific locations. [36] showed that bilingual people favour code-mixing of specific words at specific locations.
- Almost all the class labels are hand-annotated. Since there are no defined criteria of how one should classify, it can lead to the ambiguity between classes. This is the reason for lower F1 scores between abusive and hate classes in both logistic regression and CNN LSTM models for all the languages.

## 5. Online Feedback Mechanism

In this section, we discuss the use of created models in an online environment. Once the model is created, it is essential to understand how the model will behave. Here, we also discuss the complete execution of the machine learning pipeline, by adding an external feedback loop to the model. This allows the model to learn the evolution of text and textual context over time. We are aware that there are several disadvantages of doing this, as the model may become biased towards one class if used over a stretch of time, but we can overcome this by adding human-in-the-loop. In this way, the weight of tagging text by moderator and annotators could be reduced significantly.

To achieve this, we create a RESTful API which connects the machine learning model to an online webchat system. The webchat we create here is a live application demonstrating a chat room. The most important aspect is the API which scans the page and scores each comment by the user. It summarises the total score of all classes, viz. normal, hateful or offensive. In Figure 5, we see the metric which understands the page content concerning its hateful or abusive nature.

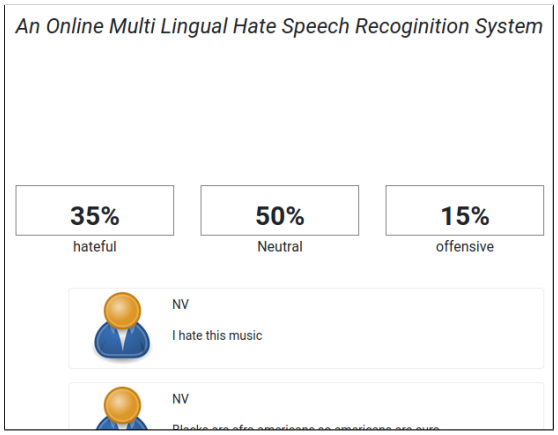


**Figure 5.** Real time monitoring of content and scoring page based on percentage if hateful content is present.

The above use case is a passive method showcasing the capability of our model in an online application. An active use case would be if we can prevent the use of hateful/abusive comments. To do so, we create a notifier on the submit button of the comments. If the comment is hateful or abusive, the user gets notified and we can actively prevent users from commenting derogatory remarks. Thus, users can be made self-aware and can be advised to refrain from sending something incinerating to the social world. Figure 6 depicts the mentioned use case.

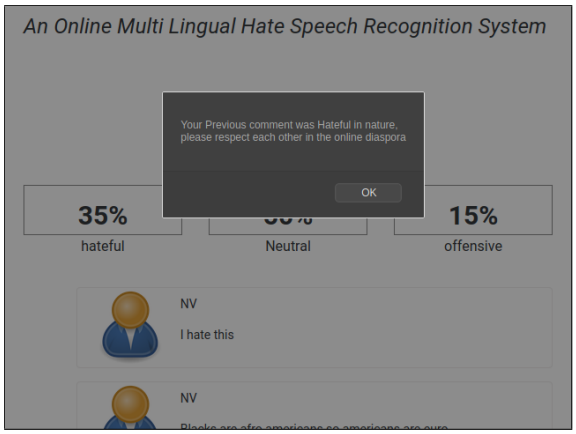


**Figure 6.** Proactively notifying users that their comment is hateful in nature.

The above application has the ability to automatically switch between different machine learning models depending on the language of the user. If the user uses Hindi, the model automatically switches to the Hindi model.

We capture these online comments into a database and use this database to update our existing datasets. The database can be reviewed by an annotator for comments which have lower confidence of belonging to a single class. Thus, a small portion of the comments has to be reviewed. Such comments can be added to the training dataset for future addition.

## 6. Conclusions

The objective of this study was to bring to light a model which was trained on a large dataset of multiple languages. By experimenting on an aggregated dataset combining six datasets in English, Hindi and Code-mixed Hindi, we demonstrate that our models achieve comparable or superior performance to a wide range of baseline monolingual models. The model leads to competitive performance on combined data and works in an online environment in near-real-time.

Further work can be done in the space of improving the model architecture and performance—we can apply and test other feature selection methods and extend the model to other code-mixed languages. Other strategies can be incorporated such as using CNN with BERT or other BERT-based model, e.g. RoBERTa or distilBERT, which support multilingual aspect of language. In this process of fine-tuning the transfer-learning methods, the model can be used to understand some types of biases to help in annotation. We can also look at different embeddings. both co-occurrence and contextual. which can be used in CNN-LSTM model to improve its performance. Besides fine-tuning deep learning models and hyperparameter selection/optimisations, other advanced hierarchical models such as Spinal, GRU and Bayesian optimisation are areas of future research. These models have shown significant results in text classification and are quite robust.

**Author Contributions:** Conceptualisation, N.V. and A.Z.; methodology, N.V.; software, N.V.; validation, N.V. and A.Z.; formal analysis, N.V. and A.Z.; investigation, N.V.; resources, N.V.; data curation, N.V.; writing—original draft preparation, N.V.; writing—review and editing, N.V. and A.Z.; visualisation, N.V.; supervision, A.Z.; project administration, N.V. and A.Z.; All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

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

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not available.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Cortese, A.J.P. *Opposing Hate Speech*; Greenwood Publishing Group: Westport, CT, USA, 2006.
2. *Abusive and Offensive Online Communications: A Scoping Report*; The Law Commission: London, UK, 2018.
3. Vidgen, B.; Derczynski, L. Directions in Abusive Language Training Data: Garbage In, Garbage Out. *arXiv* **2020**, arXiv:2004.01670.
4. Mandl, T.; Modha, S.; Majumder, P.; Patel, D.; Dave, M.; Mandlia, C.; Patel, A. Overview of the HASOC Track at FIRE 2019: Hate Speech and Offensive Content Identification in Indo-European Languages. In Proceedings of the 11th Forum for Information Retrieval Evaluation, Kolkata, India, 12–15 December 2019; Association for Computing Machinery: New York, NY, USA, 2019; FIRE '19; pp. 14–17. [CrossRef]
5. Davidson, T.; Warmsley, D.; Macy, M.W.; Weber, I. Automated Hate Speech Detection and the Problem of Offensive Language. In Proceedings of the Eleventh International Conference on Web and Social Media, ICWSM 2017, Montréal, QC, Canada, 15–18 May 2017; AAAI Press: Palo Alto, CA, USA, 2017; pp. 512–515.
6. ElSherief, M.; Nilizadeh, S.; Nguyen, D.; Vigna, G.; Belding, E. Peer to Peer Hate: Hate Speech Instigators and Their Targets. *arXiv* **2018**, arXiv:cs.SI/1804.04649.
7. Ousidhoum, N.; Lin, Z.; Zhang, H.; Song, Y.; Yeung, D.Y. Multilingual and Multi-Aspect Hate Speech Analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*; Association for Computational Linguistics: Hong Kong, China, 2019; pp. 4675–4684. [CrossRef]
8. Basile, V.; Bosco, C.; Fersini, E.; Nozza, D.; Patti, V.; Rangel Pardo, F.M.; Rosso, P.; Sanguinetti, M. SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter. In Proceedings of the 13th International Workshop on Semantic Evaluation; Association for Computational Linguistics: Minneapolis, MN, USA, 2019; pp. 54–63. [CrossRef]
9. 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)*; Association for Computational Linguistics: Brussels, Belgium, 2018; pp. 138–148. [CrossRef]

10. 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*; Association for Computational Linguistics: San Diego, CA, USA, 2016; pp. 88–93.
11. 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]
12. Smith, S.L.; Turban, D.H.P.; Hamblin, S.; Hammerla, N.Y. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. *arXiv* **2017**, arXiv:cs.CL/1702.03859.
13. Lample, G.; Conneau, A.; Denoyer, L.; Ranzato, M. Unsupervised Machine Translation Using Monolingual Corpora Only. *arXiv* **2017**, arXiv:cs.CL/1711.00043.
14. Minaee, S.; Kalchbrenner, N.; Cambria, E.; Nikzad, N.; Chenaghlu, M.; Gao, J. Deep Learning Based Text Classification: A Comprehensive Review. *arXiv* **2020**, arXiv:cs.CL/2004.03705.
15. Ruder, S.; Bingel, J.; Augenstein, I.; Søgaard, A. Sluice networks: Learning what to share between loosely related tasks. *arXiv* **2017**, arXiv:abs/1705.08142.
16. 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*; Association for Computational Linguistics: New Orleans, LA, USA, 2018; pp. 36–41. [CrossRef]
17. Sawhney, R.; Manchanda, P.; Singh, R.; Aggarwal, S. A Computational Approach to Feature Extraction for Identification of Suicidal Ideation in Tweets. I*n Proceedings of ACL 2018, Student Research Workshop*; Association for Computational Linguistics: Melbourne, Australia, 2018; pp. 91–98. [CrossRef]
18. Jay, T.; Janschewitz, K. The pragmatics of swearing. *J. Politeness Res. Lang. Behav. Cult.* **2008**, *4*, 267–288. [CrossRef]
19. Santosh, T.; Aravind, K. 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; pp. 310–313. [CrossRef]
20. Kamble, S.; Joshi, A. Hate Speech Detection from Code-mixed Hindi-English Tweets Using Deep Learning Models. *arXiv* **2018**, arXiv:cs.CL/1811.05145.
21. Kshirsagar, R.; Cukuvac, T.; McKeown, K.; McGregor, S. Predictive Embeddings for Hate Speech Detection on Twitter. *arXiv* **2018**, arXiv:cs.CL/1809.10644.
22. Sohn, H.; Lee, H. MC-BERT4HATE: Hate Speech Detection using Multi-channel BERT for Different Languages and Translations. In Proceedings of the 2019 International Conference on Data Mining Workshops (ICDMW), Beijing, China, 8–11 November 2019; pp. 551–559.
23. Chen, W.; Su, Y.; Shen, Y.; Chen, Z.; Yan, X.; Wang, W.Y. How Large a Vocabulary Does Text Classification Need? A Variational Approach to Vocabulary Selection. 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)*; Association for Computational Linguistics: Minneapolis, MN, USA, 2019; pp. 3487–3497. [CrossRef]
24. Baziotis, C.; Pelekis, N.; Doulkeridis, C. DataStories at SemEval-2017 Task 4: Deep LSTM with Attention for Message-level and Topic-based Sentiment Analysis. In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), Vancouver, QC, Canada, 3–4 August 2017; Association for Computational Linguistics: Vancouver, QC, Canada, 2017; pp. 747–754.
25. Kunchukuttan, A. The IndicNLP Library. Available online: https://github.com/anoopkunchukuttan/indic_nlp_library/blob/master/docs/indicnlp.pdf (accessed on 20 December 2020).
26. Loper, E.; Bird, S. NLTK: The Natural Language Toolkit. In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics, Philadelphia, PA, USA, 7–12 July 2020; Association for Computational Linguistics: Philadelphia, PA, USA, 2002.
27. Zhang, X.; LeCun, Y. Text Understanding from Scratch. *arXiv* **2015**, arXiv:cs.LG/1502.01710.
28. Jozefowicz, R.; Vinyals, O.; Schuster, M.; Shazeer, N.; Wu, Y. Exploring the Limits of Language Modeling. *arXiv* **2016**, arXiv:cs.CL/1602.02410.
29. Kim, Y.; Jernite, Y.; Sontag, D.; Rush, A.M. Character-Aware Neural Language Models. *arXiv* **2015**, arXiv:cs.CL/1508.06615
30. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 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)*; Association for Computational Linguistics: Minneapolis, MN, USA, 2019; pp. 4171–4186. [CrossRef]
31. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv* **2018**, arXiv:1810.04805.
32. Mozafari, M.; Farahbakhsh, R.; Crespi, N. A BERT-Based Transfer Learning Approach for Hate Speech Detection in Online Social Media. In *Complex Networks and Their Applications VIII*; Cherifi, H., Gaito, S., Mendes, J.F., Moro, E., Rocha, L.M., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 928–940.
33. Badjatiya, P.; Gupta, S.; Gupta, M.; Varma, V. Deep Learning for Hate Speech Detection in Tweets. In Proceedings of the 26th International Conference on World Wide Web Companion—WWW '17 Companion, Perth, Australia, 3–7 April 2017. [CrossRef]
34. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv* **2019**, arXiv:cs.CL/1907.11692.

35. Conneau, A.; Khandelwal, K.; Goyal, N.; Chaudhary, V.; Wenzek, G.; Guzmán, F.; Grave, E.; Ott, M.; Zettlemoyer, L.; Stoyanov, V. Unsupervised Cross-lingual Representation Learning at Scale. *arXiv* **2020**, arXiv:cs.CL/1911.02116.
36. Singh, R. Grammatical Constraints on Code-Mixing: Evidence from Hindi-English. *Can. J. Linguist. Rev. Can. Linguist.* **1985**, *30*, 33–45. [CrossRef]