Hate Speech in Code Mix

Vanshika(066) Shalini Mittal(051) Megha Raghav(024)
Priya Gupta(035) Shivangi Pandey(052) Tripti(063)
Anisha(00202112020)

July 2021

Work Done by Team Members:

Anisha:

For dataset 3(ENGLISH):

- SVM(fastText,doc2vec)+HyperparameterTuning(Random Search and Grid Search))
- RF(fastText,doc2vec)+HyperparameterTuning(Random Search and Grid Search))
- LSTM(FastText,doc2vec)
- Gaussian Naive Bayes(fastText)
- KNN(fastText)
- Report

Megha Raghav:

For Dataset 1(HINGLISH)

- SVM(word2Vec,doc2Vec,TF-IDF,BOW)+HyperparameterTuning(Random Search and Grid Search))
- RF(word2Vec,doc2Vec,TF-IDF,BOW) +HyperparameterTuning(Random Search and Grid Search))
- LSTM(word2Vec,doc2Vec)+hyperparameterTuning(Random Search)

- Report

Shivangi Pandey:
For Dataset 2(HINDI)

- SVM(word2Vec,doc2Vec,TF-IDF,BOW)+HyperparameterTuning(Random Search and Grid Search))
- RF(word2Vec,doc2Vec,TF-IDF,BOW) +HyperparameterTuning(Random Search and Grid Search))
- LSTM(word2Vec,doc2Vec)+hyperparameterTuning(Random Search)
- Report

Priya Gupta:
For Dataset 1(HINGLISH)

- SVM(FastText)+HyperparameterTuning(Random Search and Grid Search))
- RF(Fasttext) +HyperparameterTuning(Random Search and Grid Search))
- Naive Bayes (FastText)
- Report

Shalini Mittal:
For Dataset 3(ENGLISH)

- SVM(word2vec)+Hyperparameter Tuning(Random Search and Grid Search)
- RF(word2vec)+Hyperparameter Tuning(Random Search and Grid Search)
- Naive Bayes (FastText)
- Report

Tripti:
For Dataset 1(HINGLISH)

- Data Preprocessing
- Data visualization

- Feature Extraction (Doc2vec,Bow,TF-IDF)
- Report

Vanhika:
For Dataset 2(HINDI)

- Data visualization
- SVM(FastText)+HyperparameterTuning(Random Search and Grid Search))
- RF(Fasttext) +HyperparameterTuning(Random Search and Grid Search))
- Naive Bayes (FastText)
- Coversion of LATEX Report

Google Colab folder link: 'https://drive.google.com/drive/folders/18JaWLXq3MtAm0bOvAggUQ13_9KwhVKUk?usp=sharing'

1 Abstract:

The use of hate speech and offensive language is overpowering these days and therefore it is essential to detect it as it hurts the sentiments of people. The hate content is code mixed with hindi and english language. This paper projects a machine learning model for the detection of hate speech. By using Facebook's pre trained word embedding library, fastText to represent 45758 data samples as hate and not hate. The performance of this is compared with word2vec and doc2vec features and observed that it gives better results with the Support Vector Machine(SVM) Radial Basis Function(RBF) classifier.

2 Introduction:

We all are witnessing how social media is affecting us by becoming a major part of our lives. With the easily accessible internet and surge in population, the number of social media users are growing exponentially[11].Websites and apps like twitter provide its users an opportunity to express their thoughts through their platform. But these thoughts also include the offensive and

abusive content which is known as Hateful speech[11]. Social media is pervasive with this offensive content that can be categorised as hate speech and offensive language.

Hate speech is an act of expressing prejudice and aggression in any form of communication to a community or an individual[1]. Hurting,disgracing, humiliating, or threatening on the basis of someone's belief, religion, sexual orientation, or gender is also considered as Hateful speech[7]. Whereas offensive Language is considered as an aim of hurting someone and making them feel bad and disrespectful[1].

This hate speech encourages a lot of violence and Companies like Twitter,Facebook etc. face a lot of condemnation because of this offensive and abusive content on their sites[1]. Hence it's really necessary for lawmakers and social media platforms to cast down these hateful activities which are being used by millions of users belongs to different age groups[4],[3].

However, In a diverse country like India where people speak more than one language, they tend to express their thoughts with code-mixed languages in which Hinglish(a combination of Hindi and English) is slightly regular for expressing opinions on social media[9].
Code mixed language is a way of mixing two languages i.e. using phrases or morphemes or words of one language with pronunciation of another language[1].

Original	**Translated**
Text-1: Hate: Aise logo se sakht nafrat karta hu Jo caste ko naam ke sath jod ke chaude hote h but real me vo piddu hote h	**Text-1: Hate:** I hate such people who show arrogance in the name of caste but in reality they are cowards.
Text-2: Non Hate: I am very sorry to say saaf dil shilpa ke fans hiten ke dil mein kya hai woh bhi samjhte hain hadh hoti hai nafrat ki bhi	**Text-2: Non Hate:**I am very very sorry to say what is in the heart of shilpa's fns hiten , they also understand that there is hate even

Popularity of opinion-rich online platforms like these social media sites(Facebook, Twitter) and microblogging sites has encouraged users to express and convey their thoughts all across the world in real time. .This often results in users posting offensive and abusive content online using hateful speech.

The intent behind using these Code-Switched languages for writing hateful messages is to get undetected by those English trained classifiers which are responsible for detecting this kind of offensive content automatically[4].

And this linguistic complexity and growing usage of code mixed languages have necessitated an efficient classifier which is able to detect hateful and offensive language in code-mixed language automatically[1].Hinglish extends its grammatical setup from native Hindi accompanied by a hundreds of slurs, slang and phonetic variations due to regional influence. Randomized spelling variations and multiple possible interpretations of Hinglish words in different contextual situations make it extremely difficult to deal with automatic classification of this language. Another challenge worth consideration in dealing with Hinglish is the demographic divide between the users of Hinglish relative to total active users globally. This poses a serious limitation as the tweet data in Hinglish language is a small fraction of the large pool of tweets generated, necessitating the use of selective methods to process such tweets in an automated fashion[10].

In this paper, we introduced the machine learning model which detects whether tweets are of Hate or non hate category by using the binary classification of code mixed Hinglish tweets and Hindi and English tweets. We have used pre-trained models such as Fasttext, word2vec and doc2vec as features for the classification.

The paper proceeds as follows.Section 2 Related work provides a brief description about the existing works done in the area. of hate speech detection. Section 3 provides a description about the dataset used for the experiments. Section 4 accords description of the proposed methodology. Section 5 provides the details of the experiments design and results and the paper is concluded in the final result section.

3 Related Work:

There is an immense amount of research conducted in the area of hate speech detection but the most often hindrance to it is the availability of datasets especially of the native and local languages.

Base Papers: Sreelakshmi k. et.al (2020) work on detection of hate speech text inspired us to continue the research done by them to get more accurate results. They have used a pre-trained word embedding library of

Facebook called fastText to represent data collected from different sources as hate and non-hate. This method of fasttext representation is compared with word2vec and doc2vec features. They have also shown that character level features provide better results.

Important Papers: Aditya Bohra et.al (2018) looked into the challenge of detecting hate speech in different language texts and presented a dataset of tweets from Twitter. The language of the tweets is annotated at the word level, as well as the class to which they belong i.e. Hate Speech or Non Hate Speech. They also presented a supervised classification method that uses character, word, and lexicon-based characteristics to detect hate speech in text.

On Twitter data, Shivang Chopra et.al (2020) utilised pre-trained Word2Vec embeddings and a seq2seq model. They also fine-tuned the datasets using Word2Vec Embeddings. Their research indicates that combining targeted hatred embeddings with social network-based characteristics yields positive outcomes. They also provided their bias removal technique and investigated the prevalence and performance impact of debiasing.

Neeraj Vashishta et.al (2020) looked at six datasets and combined them into a one dataset and then categorised them into three categories like abusive, hateful, or neither. They built a base model and used optimization techniques to enhance it. Then they created a tool that detects and rates a page in near-real time using an effective metric, and then utilises that information as feedback to retrain the model.

Raghav Kapoor et.al (2019) utilized transfer learning to create an LSTM-based hate speech categorization model. This significantly makes the Hinglish offensive text categorization methods widely accessible. For training embeddings for the processed tweets, they used Glove algorithm and Twitter word2vec code. These embeddings assist in the learning of distributed tweet representations.

Aditya Gaydhani et.al (2018) studied n-grams as attributes, feeding their term frequency-inverse document frequency (TFIDF) values to different machine learning models. They compared models with values of n-grams and TFIDF normalising techniques. Upon assessment, they scored 95.6 percent accurate. They compared Logistic Regression (LR), Naive Bayes (NB), and Support Vector Machines (SVM) for various feature parameter mixture.

Santosh et.al (2019) tested hate speech identification using two architectures: a normal LSTM model and a hierarchical LSTM model with attention on phonemic sub-words, using a code-mixed dataset. Character n-grams,

word n-grams, negation words, and punctuation marks were taken out and used as attributes in the supervised machine learning model. They utilised Random Forest and SVM with radial basis function. They have used the chi-square attribute selection technique to lessen the size of the feature vector.

Vivek Kumar et al.(2019): Combining bi-directional sequence models with basic text augmentation techniques like synonym replacement, random insertion, random swap, and random deletion, they built a classifier. The best recall rate for hate speech was 77 percent for a Bidirectional LSTM with 32 units and a recurrent drop out rate of 0.2. For hate speech identification, the GRU type of RNN sequence model performed better for precision. GRU vs BiLSTM based models achieved 92 percent accuracy and an 88 percent recall rate for abusive tweets.

Puneet Mathur et.al (2018) article introduces the Multi-Input Multi-Channel Transfer Learning based model (MIMCT). This will use transfer learning and multiple feature inputs to find tweets from the given Hinglish Offensive Tweet (HOT) dataset. They employed several primary word embeddings as inputs, and also secondary extracted features, so that training is done using multi-channel Conventional Neural Network-LSTM architecture on English tweets. This MIMCT model is better than supervised classification methods like transfer learning-based Conventional Neural Network models, and Long Short Term Memory models.

Table 1: Comparative Analysis of Papers used same dataset as the Base Paper

S.No.	Author	Brief Description	Results
1.	Aditya Bohra et.al(2018)	They displayed a collection of Hindi-English code-mixed text, which includes tweet ids and annotations. They also exhibited the supervised algorithm that was utilized to detect hate speech present in mixed language data.	When all characteristics are included in the feature vector and SVM is used as the classification algorithm, the best accuracy of 71.7 percent is attained.

2.	Satyajit et al.(2018)	They contrasted and assessed 3 deep learning-based techniques(CNN, LSTM, BiLSTM) with a statistical approach.They have trained word embeddings on a large corpus of relevant code-mixed data.	They found that CNN-1D has the best performance, with an F-score of 80.85% and an accuracy of 82.62%. There was an improvement of about 12% in F-score.

3.	Shivang Chopra et.al (2020)	They have used deep graph embeddings. Pre-trained Word2vec and Seq2seq model is used on twitter data. They improved hate speech identification in coding mixed languages by adding social media-based characteristics and capturing the usage of offensive terms into these models.	For baseline classifiers, the greatest results came from a combination of CNN, Bidirectional LSTM, and Attention. The model performed better than the other models by the Convolution, LSTM, and Attention-based layers, respectively(Accuracy=85+, F1-score=77+). The concatenation of graph embeddings with the encoded text resulted in a 7% increase in accuracy.The Bias Elimination Algorithm employed was able to erase prejudices (such as "Islam" and LGBT groups) and accurately identified phrases that had previously been misclassified by the mode

4.	Santosh et.al (2019)	They tested hate speech identification using two architectures: a normal LSTM model and a hierarchical LSTM model especially with phonemic sub-words, using a code-mixed dataset.	Support vector machines using character n-grams, word n-grams as attributes offer high accuracy, but Hierarchical LSTM models, in association with phonemic sub-words, produced a significant margin of recall(45.1) and F1-score(48.7).

Table 2: Prior Research work related to the topic(Papers having different datasets regarding Base paper)

S.no	Author	Dataset	Brief Description
1.	Puneet Mathur et al.(2018)	Hinglish-Offensive-Text-Classification https://github.com/pmathur5k10/Hinglish-Offensive/-Text-Classificat/ion	From the suggested Hinglish Offensive Tweet (HOT) dataset, this study offers the MIMCT model for identifying abusive Hinglish tweets.
2.	Neeraj Vashishtha et.al(2020)	Hate Speech and Offensive Language dataset https://github.com/t-davidson/hate-speech-and-/offensive-language/tree/master/data	They combined six datasets into one similar dataset before categorising them into three categories i.e. abusive, hateful, or neither. They built a base model and used optimization techniques to enhance it.

3.	Vivek Kumar et al.(2019)	Hate Speech and Offensive Language dataset https://github.com/t-davidson/hate-speech-and-\offensive-language/tree/master/data	They created a classifier by combining bi-directional sequence models with simple text augmentation techniques including synonym substitution, random insertion, random swap, and random deletion.
4.	Aditya gaydhani et al.(2018)	Detecting Hate Speech and Offensive Language on Twitter https://data.world/crowdflower/hate-speech-\identification	They built a classifier model through n-gram and TFIDF as features and assessed it using publicly accessible Twitter datasets. The researchers then compared the findings obtained using LR, NB and SVM as classifier models, finding that LR outperforms them all

4 Dataset Description:

This section explains the dataset we have used in the experimentation[1] Authors of the base paper had constructed a corpus of Hindi-English code-mixed tweets and scrapped the tweets using Twitter Python API.

They carried out the annotations of corpus by 'Language at Word Level' and then 'Classifying Hate Speech or Normal Speech'. After that, they successfully scraped weets with a kappa score, 0.982. Based on this we chose three datasets.

Table 3.1: Description of dataset 1:

The first dataset consists of hinglish data.

Type Of Speech	Labelled as	Number of Tweets
Hate Speech	Yes	1664
Non-Hate Speech	No	2914

We had taken two more databases individually of English as well as of Hindi, which is taken from HASOC website.[13]

Hinglish Dataset The dataset contains 4578 rows, having 2 columns , the first column contains tweets and the second column has class.The tweets are labelled as "YES" or "NO". "YES" for Hate Speech , "NO" for Non-Hate Speech. In the dataset, Hate Speech is present in 1664 tweets and the remaining, 2914 tweets are Non-Hate.

Dataset 1 : https://github.com/pmathur5k10/Hinglish-Offensive-Text-Classification

Table 3.2: Description of dataset 2:

This dataset consists of hindi language tweets and is taken from HASOC 2019.

Type Of Speech	Labelled as	Number of Tweets
Hate & Offensive Tweets	Yes	713
Not Hate & Offensive Tweets	No	606

Hindi Dataset contains 1319 rows and two columns, one of tweet and other of class i.e. Yes and No. Yes is the class defined for Hate and Offensive Tweets and No class is defined for Non-Hate and Non-Offensive Tweets. There are 713 No class tweets and 606 Yes class tweets.

Table 3.3: Description of dataset 3:

This dataset consists of english language tweets and is taken from HASOC 2019.

Type Of Speech	Labelled as	Number of Tweets
Hate & Offensive Tweets	Yes	865
Not Hate & Offensive Tweets	No	288

English Dataset contains 1153 rows and two columns,one of tweet and other of class i.e. Yes and No. Yes is the class defined for Hate and Offensive Tweets and No class is defined for Non-Hate and Non-Offensive Tweets. There are 865 No class tweets and 288 Yes class tweets.

5 Dataset Visualization:

- **DATASET 1:**

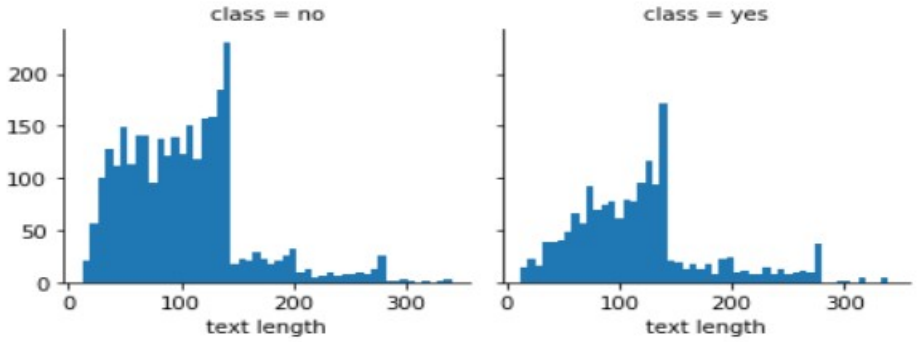

Figure 1: Basic visualization of data using histograms(Yes=Hate,No=Non-Hate)

- **DATASET 2:**

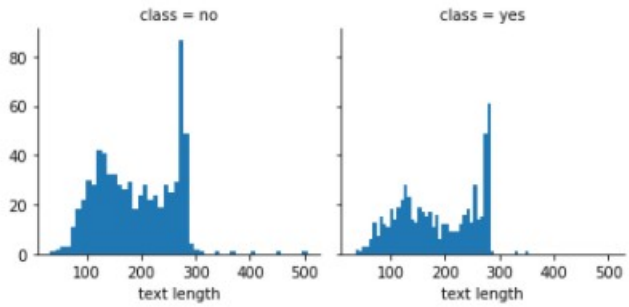

Figure 2: Basic visualization of data using histograms(Yes=Hate,No=Non-Hate)

- **DATASET 3:**

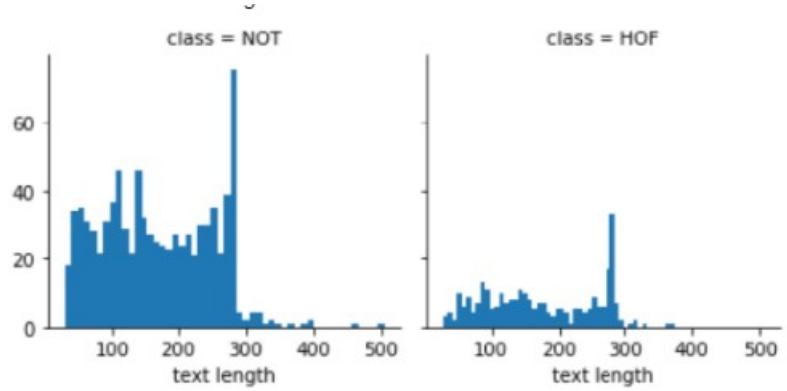

Figure 3: Basic visualization of data using histograms(HOP=Hate,NOT=Non-Hate)

6 Proposed Methodology:

This section is about Proposed Methodology we used in detecting Hate Speech and Offensive Tweets.

Training:

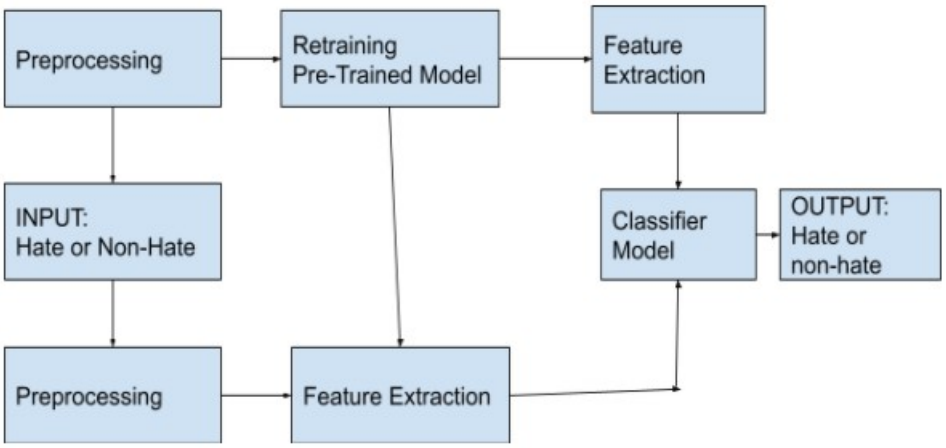

Figure 4: WorkFlow Diagram of Proposed Methodology

Testing: Proposed Methodology is done in a way:

- **Preprocessing:** The dataset we used is not so proper for training. There were many Null and Duplicate values present in the dataset. Firstly we removed extra columns that we do not need according to our problem statement. Then we removed special characters, numbers, punctuations and short words using different functions.

  Then we pre-processed data by performing TOKENIZATION and STEMMING.

  - Tokenization separated and divided the text into words, characters or subwords, that are called "tokens" using lambda function. Tokens are further used in stemming and lemmatization.
    Tokenization is an important step in pre-processing as analysis and understanding of text can be done by words itself. We had done tokenization using split() function.
  - Stemming reduced the word to stem and used tokens in it.

    Preprocessing made the dataset proper for performing further models and algorithms.

- **Retraining the model:** Then we trained the pre-trained model using Pre-processed and cleaned data. We used three pre-trained models that are fastText, word2vec and doc2vec.

Word2vec:

We have used Genism implementation of Word2vec. Word2vec algorithm learned the word associations from tokens, text that we had preprocessed using Tokenization.

After training the word2vec model on the data, it detects for the similar word or synonymous as it suggested synonyms for word hate when we implemented it. Then we converted the tokens into vectors using a function.We set the hyper parameters for this experiment at 200 for the vector length and 5 for the window length. Each sentence's vector representation was created by concatenating the vector representations of each word.

fastText:

fastText is a method of word embedding which is a supplement of the word2vec model. Rather than learning vectors for words straight, n-gram of character representation is used in the fasttext. . Though the fasttext model takes much time to train, the performance of fasttext is better than word2vec. Working with rare words is good in fasttext. Usage of n-gram for

the word representation, it gives better results for classification purpose on language, sentiment investigation.

Doc2vec:

Doc2Vec model, gives the vectorised depiction of the words (forming any sentence or paragraph). An unsupervised learning method is used just like in word2vec.Representation of the document into the numerical values is the basic idea behind doc2vec. The task of the vectors being generated is to find the similarity between documents, sentences or paragraphs. The basic idea of Doc2Vec is that document representation must be capable enough to foresee the words in the document.Doc2vec, we used CBOW to train the hyper-parameters with a vector length of 300, a window size of 5, and a minimum count of 1.

TF-IDF:

Based on the number of times the keyword appears, it provides importance to that keyword by weighing it. Moreover, it checks how important that keyword is throughout the entire web, which is called a corpus. Each word or term that appears in the text has its own TF as well as IDF score. TF*IDF is basically used by search engines to understand the content that is undervalued . It has several applications, such as in automated text analysis, and is very useful for scoring words in machine learning algorithms for Natural Language Processing (NLP).

For a term t in document d, the weight Wt,d of term t in document d is given by

Wt,d = TFt,d log (N/DFt)

BOW:

Bag of Words (BOW) is an approach to bring out the features from text documents. For training machine learning algorithms, these features can be used.. It is a collection of words to represent a sentence with word count and disregarding their order in which they occur.

Supervising the word counts and disregarding the grammatical details and the word order. It is called a "bag" of words as the reason any information about the order or structure of words in the document is not considered.BOW is an approach widely used with:

- Natural language processing (NLP)
- Information retrieval from documents
- Document classifications

7 Experiment Design:

We trained our dataset using models :Support Vector Machine, Random Forest,KNN and LSTM. We have used all the methods of feature extraction that are mentioned above i.e word2vec,doc2vec,fasttext,bow,tfidf on each model one by one. and then perform Hyperparameter Tunning using Both Random Search And Grid Search.

Random Search: Arbitrary fusion of the hyperparameters are used to find the optimal answer. Somehow, random search is alike the grid search but gives better results as compared to the grid search. One of the limitation of random search is its high variance during the calculation process.

Grid Search: For a specific model, grid search technique finds the correct set of hyperparameters. From the training data, it is not possible to find the right set of parameters. During the training phase, model parameters are learned by optimising the loss function using gradient descent. By building a model for every combination, the model which gives the best results wins.

- **SVM:** Support vector machines (SVMs) are well built and flexible supervised machine learning algorithms that are used for both the classification and regression process. The major aim of SVM is to divide the datasets into respective classes to search for a maximum marginal hyperplane (MMH). SVM algorithm is applied with the kernel that modifies the input data space into requisite form. Kernel trick is used by the SVM algorithm in which a low dimensional input is taken by the kernel space and modifies into a higher dimensional space. In other words, the kernel turns the non-separable queries into separable problems by adding more dimensions to it. It makes SVM more powerful, flexible and accurate. SVM uses the following mentioned types of kernels:
  - Linear
  - Polynomial
  - Radial basis function

  SVM optimizer uses the C parameter to know how much misclassifying is needed to avoid in each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly.

- **RF:** In order to reach a single output, Random forest machine algorithm is used which works by integrating the results of multiple decision trees. Through cross validation, random forest provides high accuracy and handles the missing values and hence maintaining the accuracy of high amount of data. Limitation of decision tree are removed by random forest. Overfitting of datasets is reduced by it and hence the precision increases.

  Some parameters are:

  N_estimators: This is the number of trees you want to build before taking the maximum voting or averages of predictions. More the number of trees leads to better performance but slows the code. **Max_features:** maximum number of features considered when finding the best split. This enhances the performance of the model as each node of each tree is now considering a higher number of options.

- **LSTM:** An LSTM has a related control flow model as recurrent neural network (RNN). It operates by data passing on information as it moves forward. The differences are the operations within the LSTM's cells. Keep or forget information methodology is used by the LSTM. Complex problem domains like machine translation, speech recognition make use of this methodology. Long-term dependencies problem was handled by it(RNN cannot predict the word stored in the long term memory but can give more accurate predictions from the recent information). As the length of gap varies, RNN does not give good performance. LSTM can by default retain the information for a long period of time. It is used for processing, predicting and classifying on the basis of time series data.

- **KNN:** K-Nearest Neighbour Machine Learning algorithms based on Supervised Learning technique. By assuming the similarity between the data and the test cases available, it puts the data in that category which is the most similar. Thus algorithm can be used for classification as well as regression but mostly it is used for classification. By not making any assumption on the underlying data, it is a non-parametric algorithm.

8 RESULTS AND OBSERVATIONS::

For Dataset 1(Hinglish):

1.1 SVM WORD2VEC: The accuracy of 0.64 attained. After doing Hyperparameter tuning by using Random Search method, the best parameters we have attained are kernel:'rbf' and C:1 and accuracy of 0.64 was achieved and Same With the Grid Search Method of Hyperparameter Tuning.

1.2 SVM DOC2VEC: The accuracy of 0.63 attained. After doing Hyperparameter tuning by using Random Search method, the best parameters we have attained are kernel:'rbf' and C:1 and accuracy of 0.64 was achieved and Same With the Grid Search Method of Hyperparameter Tuning.

1.3 SVM BOW: The accuracy of 0.64 attained. After doing Hyperparameter tuning by using Random Search method, the best parameters we have attained are kernel:'rbf' and C:10 and accuracy of 0.66 was achieved and Same With the Grid Search Method of Hyperparameter Tuning.

1.4 SVM TF-IDF: The accuracy of 0.64 attained. After doing hyperparameter tuning by using Random Search method, the best parameters we have attained are kernel:'rbf' and C:1 and accuracy of 0.65 was achieved and Same With the Grid Search Method of Tuning.

1.5 SVM FASTTEXT: The accuracy of 0.49 attained. After doing hyperparameter tuning by using Random Search method, the best parameters we have attained are kernel:'rbf' and C:1 and accuracy of 0.74 was achieved and Same With the Grid Search Method of Tuning.

2.1 RF WORD2VEC: The accuracy of 0.65 attained before hyperparameter tuning. After doing hyperparameter tuning by using Random Search method, the best parameters, we have attained for the following values:-

n_estimators: 767
min_samples_split: 10
min_samples_leaf: 4
max_features: 'auto'
max_depth: 36
bootstrap: True

The accuracy of 0.64 was achieved and by using the Grid Search Method , the best parameters are

bootstrap: True
max_depth: 110
max_features: 3

min_samples_leaf: 4
min_samples_split: 12
n_estimators: 100
The accuracy of 0.65.

2.2 RF Doc2Vec: The accuracy of 0.64 is attained before hyperparameter tuning. After doing hyperparameter tuning by using Random Search method, the best parameters we have attained are

n_estimators: 964
min_samples_split: 2
min_samples_leaf: 4
max_features: sqrt
max_depth: 27
bootstrap: True

The accuracy of 0.64 was achieved and by using the Grid Search Method , the best parameters are

bootstrap: True
Max_depth: 90
max_features: 3
min_samples_leaf: 4
min_samples_split: 10
n_estimators: 100

2.3RF BOW: The accuracy of 0.64 is attained by using Random Search method, the best parameters we have attained are

n_estimators: 964
min_samples_split: 2
min_samples_leaf: 4
max_features: sqrt
max_depth: 27
bootstrap: True

and accuracy of 0.64 was achieved and by using the Grid Search Method , the best parameters are

bootstrap: True
max_depth: 110
max_features: 3
min_samples_leaf: 4
min_samples_split: 12
n_estimators: 100
accuracy of 0.64.

2.4 RF TF-IDF: TThe accuracy of 0.66 is attained by using Random Search method, the best parameters we have attained are

n_estimators: 934
min_samples_split: 5
min_samples_leaf: 2
max_features: sqrt
max_depth: 41
bootstrap: True

and accuracy of 0.64 was achieved and by using the Grid Search Method , the best parameters are

max_depth: 100
max_features: 3
min_samples_leaf: 3
min_samples_split: 12
n_estimators: 100

and accuracy of 0.63.

2.5 RF Fasttext: The accuracy of 0.745 is attained by using Random Search method, the best parameters we have attained are

n_estimators: 767
min_samples_split: 10
min_samples_leaf: 4
max_features: 'auto'
max_depth: 36
bootstrap: True

and accuracy of 0.64 was achieved and by using the Grid Search Method ,the best parameters are

bootstrap: True
max_depth: 110
max_features: 3
min_samples_leaf: 4
min_samples_split: 12
n_estimators: 100

accuracy of 0.746.

3.1 LSTM Word2Vec: We used Embedding layer of 200 neurons with LSTM layer of 100 neurons and dropout of 0.1 and dense layer with activation function softmax. And then compile the model using categorical cross entropy loss and Adam optimizer and fit our model using number of Epochs is equal to 5 and batch size equal to 32 and achieved accuracy of 0.37 before

hyperparameter tuning .after using Random Search Method of hyperparameter tuning we have achieved Best score of 0.37 using the parameters

var_optimizer: sgd

Var_activation: tanh

batch_size: 16

3.2 LSTM Doc2Vec: We used Embedding layer of 200 neurons with LSTM layer of 176 neurons and dropout of 0.2 and dense layer with activation function softmax. And then compile the model using binary cross entropy loss and Adam optimizer and fit our model using number of Epochs is equal to 10 and batch size equal to 32 and achieved accuracy of 0.37 before hyperparameter tuning before hyperparameter tuning .after using Random Search Method of hyperparameter tuning we have achieved

Best score of 0.37 using

var_optimizer: sgd

var_activation: tanh

batch_size: 16

Table 4: Accuracy of Hinglish Dataset:

	HT	**SVM**	**RF**
word2vec	RS	0.63	0.65
	GS	0.64	0.65
Doc2Vec	RS	0.64	0.64
	GS	0.64	0.64
BOW	RS	0.66	0.68
	GS	0.66	0.63
TF-IDF	RS	0.65	0.66
	GS	0.65	0.63
fasttext	RS	0.74	0.74
	GS	0.74	0.75

Table 5: Accuracy of LSTM Using RS:

Word2Vec	**Doc2Vec**
0.37	0.37

Classification report of best model for HINGLISH dataset: The best model for the first dataset is with accuracy 0.75 and the model is Random Forest using Grid Search and Fasttext is used for obtaining the vectors. This figure is not better than the base paper as SVM RBF is giving the highest accuracy of 0.8581.

```
              0       0.75      1.00      0.85       859
              1       1.00      0.00      0.01       294

    accuracy                            0.75      1153
   macro avg        0.87      0.50      0.43      1153
weighted avg        0.81      0.75      0.64      1153

RF(Fasttext) after HT+GS, Accuracy Score: 0.7458803122289679
```

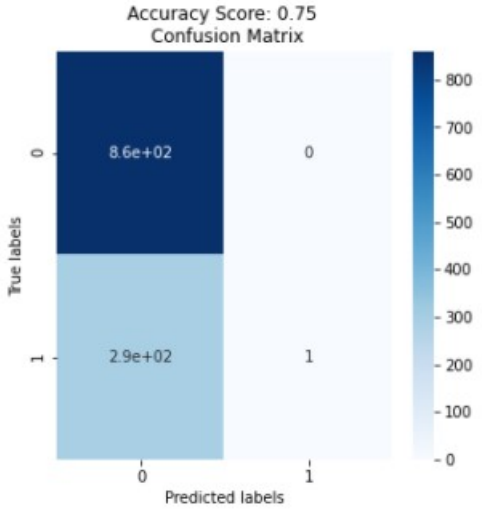

Figure 5: confusion Matrix of best model using RS[RF(Fasttext)]

Table 6:Comparison Table With Base Paper

Model	Base Paper (Accuracy)	Our Paper (ACccuracy)
SVM(word2vec)	0.7511	0.64
Rf (Word2vec)	0.7267	0.65
SVM (Doc2vec)	0.613	0.64
RF (Doc2vec)	0.6415	0.64
SVM(Fasttext)	0.8581	0.74
RF(Fasttext)	0.7834	0.75

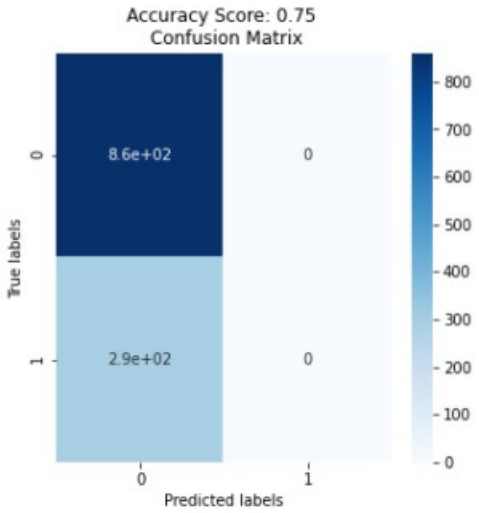

Figure 6: confusion Matrix of best model using GS[RF(Fasttext)]

From the comparison table above it has been observed that base paper have performed well with accuray 0.8581 using SVM(fastText) if compared with our model for dataset with accuracy 0.75 using RF(FastText).

For Dataset 2(Hindi):

1.1 SVM WORD2VEC: After doing Hyperparameter tuning by using Random Search method, the best parameters we have attained are kernel:'rbf' and C:1 and accuracy of 0.73 was achieved and Same Accuracy with the Grid Search Method of Hyperparameter Tuning.

1.2 SVM DOC2VEC: The accuracy of 0. 76 attained. After doing Hyperparameter tuning by using Random Search method, the best parameters we have attained are kernel:'rbf' and C:1 and accuracy of 0.76 was achieved and Same With the Grid Search Method of Hyperparameter Tuning.

1.3 SVM BOW: The accuracy of 0.71 attained. After doing Hyperparameter tuning by using Random Search method, the best parameters we have attained are kernel:'rbf' and C:10 and accuracy of 0.71 was achieved and Same With the Grid Search Method of Hyperparameter Tuning.

1.4 SVM TF-IDF: The accuracy of 0.73 attained. After doing hyperparameter tuning by using Random Search method, the best parameters we have attained are kernel:'rbf' and C:1 and accuracy of 0.73 was achieved and Same With the Grid Search Method of Tuning.

1.5 SVM FASTTEXT: The accuracy of 0.75 attained. After doing hy-

perparameter tuning by using Random Search method, the best parameters we have attained are kernel:'rbf' and C:1 and accuracy of 0.76 was achieved and Same With the Grid Search Method of Tuning.

2.1 RF WORD2VEC: The accuracy of 0.72 attained before hyperparameter tuning. After doing hyperparameter tuning by using Random Search method, the best parameters we have attained accuracy of 0.73 was achieved and by using the Grid Search Method , the Accuracy is 0.73.

2.2 RF Doc2Vec: The accuracy of 0.64 is attained before hyperparameter tuning. After doing hyperparameter tuning by using random Search method accuracy of 0.76 was achieved and by using the Grid Search Method , the accuracy of 0.76 was achieved.

2.3 RF BOW: The accuracy of 0.71 is attained by using Random Search method, the best parameters we have attained are and accuracy of 0.64 was achieved and by using the Grid Search Method , the accuracy of 0.60 was achieved.

2.4 RF TF-IDF: The accuracy of 0.72 is attained by using Random Search method, accuracy of 0.64 was achieved and by using the Grid Search Method accuracy of 0.73 was achieved.

2.5 RF Fasttext: The accuracy of 0.75 is attained by using Random Search method, accuracy of 0.76 was achieved and by using the Grid Search Method accuracy of 0.74 was achieved.

3.1 LSTM Word2Vec: We used Embedding layer of 200 neurons with LSTM layer of 100 neurons and dropout of 0.1 and dense layer with activation function softmax. And then compile the model using categorical cross entropy loss and Adam optimizer and fit our model using number of Epochs is equal to 5 and batch size equal to 32 and achieved accuracy of 0.37 before hyperparameter tuning .after using Random Search Method of hyperparameter tuning we have achieved Best score of 0.45 using Best Score of 0.46 using

{'var_optimizer': 'sgd','var_activation':'tanh','batch_size':16}

3.2 LSTM Doc2Vec: We used Embedding layer of 200 neurons with LSTM layer of 176 neurons and dropout of 0.2 and dense layer with activation function softmax. And then compile the model using binary cross entropy loss and Adam optimizer and fit our model using number of Epochs is equal to 10 and batch size equal to 32 and achieved accuracy of 0.46 before hyperparameter tuning before hyperparameter tuning .after using Random Search Method of hyperparameter tuning we have achieved.Best Score of 0.46 using

{'var_optimizer': 'sgd','var_activation':'tanh','batch_size':16}

Table 7: Accuracy of Hindi Dataset:

	HT	SVM	RF
word2vec	RS	0.73	0.733
	GS	0.733	0.738
Doc2Vec	RS	0.76	0.76
	GS	0.761	0.76
BOW	RS	0.716	0.74
	GS	0.716	0.605
TF-IDF	RS	0.72	0.727
	GS	0.73	0.73
fasttext	RS	0.76	0.76
	GS	0.76	0.74

Table 8: Accuracy of LSTM:

HT	Word2Vec	Doc2Vec
RS	0.45	0.34
GS	0.45	0.34

Classification report of best model for HINDI dataset: The best result for hindi dataset received is on F1 score of 0.77. The classification model SVM is giving best F1 score and also after hyperparameter tuning, we are getting same results for SVM from both Grid Search and Random Search (i.e 0.77 f1 score) and word2cev is used for obtaining these vectors.

This F1 score is better than the work done previously on this dataset whose F1 score was 0.53.

	precision	recall	f1-score	support
0	0.76	0.79	0.77	100
1	0.78	0.75	0.77	100
accuracy			0.77	200
macro avg	0.77	0.77	0.77	200
weighted avg	0.77	0.77	0.77	200

SVM(word2vec) after HT+GD, Accuracy Score: 0.77

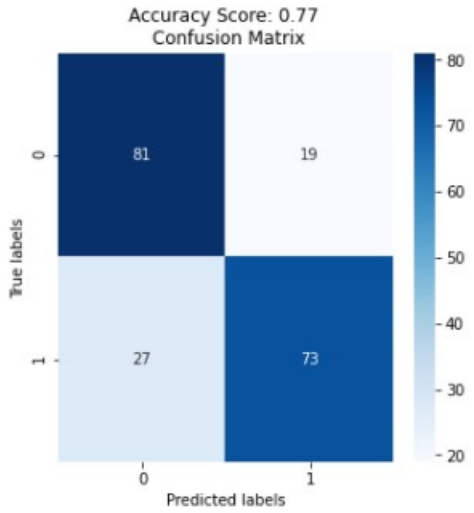

Figure 7: confusion Matrix of best model using RS[RF(Fasttext)]

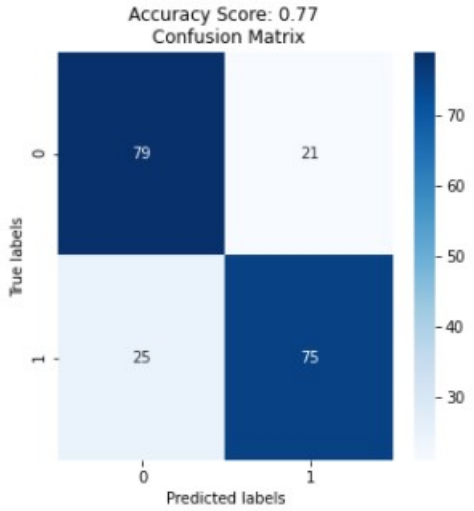

Figure 8: confusion Matrix of best model using GS[RF(Fasttext)]

For Dataset 3(English):

1.Fasttext:

SVM: By using FastText on SVM we attained accuracy is 0.4329004329004329 before Hyperparameter tuning.

SVM(Grid Search): After using Hyperparameter tuning , that is, GRID SEARCH on SVM we attained accuracy of 0.7445887445887446.

SVM(Randomized Search): After using Hyperparameter tuning , that is, Randomized Search we attained accuracy of 0.7445887445887446.

RF: By using fastText on Random Forest we attained an accuracy of 0.7489177489177489 before tuning.

RF(Randomized Search): After tuning our model using Randomized Search ,we attained an accuracy of 0.7445887445887446.

RF(Grid Search) : After tuning our model using Randomized Search ,we attained an accuracy of 0.7445887445887446.

LSTM: By using fastText features on LSTM model we got an accuracy of 0.2693 Without doing any Hyperparameter tuning.

Gaussian NB: By using fastText on Naive Bayes before tuning our model we attained an accuracy of 0.5454545454545454.

KNN: By using fastText on KNN before tuning we attained an accuracy of 0.636363636363636.

2.Doc2vec:

SVM: By using DOC2VEC on SVM we attained accuracy is 0.7965367965367965 before Hyperparameter tuning.

SVM(Grid Search): After using Hyperparameter tuning , that is, GRID SEARCH on SVM we attained accuracy of 0.8051948051948052.

SVM(Randomized Search): After using Hyperparameter tuning , that is, Randomized Search we attained accuracy of 0.8051948051948052.

RF: By using DOC2VEC on Random Forest we attained an accuracy of 0.7835497835497836 before tuning.

RF(Randomized Search): After tuning our model using Randomized Search ,we attained an accuracy of 0.7922077922077922.

RF(Grid Search): After tuning our model using Randomized Search ,we attained an accuracy of 0.7748917748917749.

LSTM: By using DOC2VEC features on the LSTM model we got an accuracy of 0.2626 Without doing any Hyperparameter tuning.

3.Word2vec:

SVM: By using WORD2VEC on SVM we attained accuracy is 0.7532467532467533 before Hyperparameter tuning.

SVM(Grid Search): After using Hyperparameter tuning , that is, GRID SEARCH on SVM we attained accuracy of 0.7792207792207793

SVM(Randomized Search): After using Hyperparameter tuning , that is, Randomized Search we attained accuracy of 0.7792207792207793

RF: By using WORD2VEC on Random Forest we attained an accuracy of 0.7922077922077922 before tuning.

RF(Randomized Search): After tuning our model using Randomized Search ,we attained an accuracy of 0.7835497835497836.

RF(Grid Search): After tuning our model using Randomized Search ,we attained an accuracy of 0.7705627705627706.

LSTM: By using WORD2VEC features on the LSTM model we got an accuracy of 0.2592 Without doing any Hyperparameter tuning.

Table 9: Accuracy of English Dataset:

	HT	**SVM**	**RF**
word2vec	RS	0.78	0.78
	GS	0.78	0.77
Doc2Vec	RS	0.80	0.79
	GS	0.80	0.77
fasttext	RS	0.74	0.74
	GS	0.74	0.74

Table 10: Accuracy of LSTM:

Word2Vec	**Doc2Vec**	**fasttext**
0.26	0.26	0.2693

Classification report of best model for ENGLISH dataset: The best model for the english dataset is with accuracy 0.80 and the model is Supported Vector Machine using Randomized Search and Grid Search and Doc2Vec is used for obtaining the vectors and better than the results mentioned in base paper as their highest F1 Score mentioned is 0.51 using LSTM and of our model is 0.77.

```
precision    recall  f1-score   support

           0       0.83      0.94      0.88       172
           1       0.69      0.42      0.53        59

    accuracy                           0.81       231
   macro avg       0.76      0.68      0.70       231
weighted avg       0.79      0.81      0.79       231

SVM_RBF(doc2vec) after HT+RS, Accuracy Score: 0.8051948051948052
```

Table 11:Comparison Table With Base Paper

	Base paper(Accuracy)	**Our Paper(Dataset 1)(Accuracy)**
Svm(word2vec)	0.7511	0.779
Rf (Word2vec)	0.7267	0.792
Svm(Doc2vec)	0.613	0.8051
Rf (Doc2vec)	0.6415	0.7922
Svm(Fasttext)	0.8581	0.7445
Rf(Fasttext)	0.7834	0.7489

Word2vec and doc2vec have outperformed the results from base paper whereas the fasttext model of the base paper has higher accuracy than our paper.

9 Conclusion:

As the number of people using social media grows, so does the volume of hate content on those platforms, leading to a rise in cyber violence and crimes. As a result, detecting hate speech on social media platforms like Twitter and Facebook before it reaches the general public is critical. However, because individuals try to utilise code-mixed language, this identification becomes difficult. In our paper we have used three different datasets which includes english, hindi and hinglish language. For comparison we chose three different algorithms applied to each one of them i.e. word2vec, doc2vec and fasttext. Through performance evaluation parameters all the three dataset behaves differently and gives us different results on each algorithm used. The results also suggest that for code-mixed text classification, character level features

give more information than word and document level features, and that in English text, document level information performs better. Additionally, the suggested technique will be expanded in the future to include a finer definition of hate tweets, such as insulting, abusive, and profane.

References

[1] Sreelakshmi, K., B. Premjith, and K. P. Soman. "Detection of hate speech text in Hindi-English code-mixed data." Procedia Computer Science 171 (2020): 737-744.

[2] Mandl, Thomas, et al. "Overview of the hasoc track at fire 2019: Hate speech and offensive content identification in indo-european languages." Proceedings of the 11th forum for information retrieval evaluation. 2019.

[3] Bohra, Aditya, et al. "A dataset of hindi-english code-mixed social media text for hate speech detection." Proceedings of the second workshop on computational modeling of people's opinions, personality, and emotions in social media. 2018.

[4] Chopra, Shivang, et al. "Hindi-english hate speech detection: Author profiling, debiasing, and practical perspectives." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 34. No. 01. 2020.

[5] Vashistha, Neeraj, and Arkaitz Zubiaga. "Online multilingual hate speech detection: experimenting with Hindi and English social media." Information 12.1 (2021): 5

[6] Kapoor, Raghav, et al. "Mind your language: Abuse and offense detection for code-switched languages." Proceedings of the AAAI conference on artificial intelligence. Vol. 33. No. 01. 2019.

[7] Gaydhani, Aditya, et al. "Detecting hate speech and offensive language on twitter using machine learning: An n-gram and tfidf based approach." arXiv preprint arXiv:1809.08651 (2018).

[8] Santosh, T. Y. S. S., and K. V. S. Aravind. "Hate speech detection in hindi-english code-mixed social media text." Proceedings of the ACM India Joint International Conference on Data Science and Management of Data. 2019.

[9] Gupta, Vivek Kumar. "" Hinglish" Language–Modeling a Messy Code-Mixed Language." arXiv preprint arXiv:1912.13109 (2019).

[10] Mathur, Puneet, et al. "Detecting offensive tweets in hindi-english code-switched language." Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media. 2018.

[11] https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/

[12] https://www.sciencedirect.com/science/article/abs/pii/S1359178918300028

[13] HASOC(hasocfire.github.io)