MINI PROJECT REPORT

ON

Hate Speech Detection Across Code-Mixed Indo-Aryan Languages

School of Digital Sciences Kerala University of Digital Sciences, Innovation and Technology

Hate Speech Detection across Code-Mixed Indo-Aryan Languages

Submitted by

Sourudra Nag (Register Number: 233024)
Sourikta Nag (Register Number: 233023)
Poulomi Banerjee (Register Number: 233310)

In partial fulfillment of the requirements for the award of Master of Science in

School of Digital Sciences Kerala University of Digital Sciences, Innovation and Technology
Technocity Campus, Thiruvananthapuram, Kerala - 695317

BONAFIDE CERTIFICATE

This is to certify that the project report entitled "*Hate Speech Detection across Code-Mixed Indo-Aryan Languages*" submitted by SOURUDRA NAG, SOURIKTA NAG, and POULOMI BANERJEE in partial fulfilment of the requirements for the award of Master of Science in Computer Science and Data Analytics is a Bonafide record of the work carried out at "Kerala University of Digital Sciences, Innovation and Technology".

Supervisor	**Course Coordinator**
Dr. ANOOP V.S	**Dr. ANOOP V.S**
School of Digital Sciences	**School of Digital Sciences**
DUK	**DUK**

Head Of Institution
Prof. Ciza Thomas
Vice Chancellor DUK

DECLARATION

I, Sourudra Nag, a student of MSc Computer Science, I affirm that this report primarily reflects my team's efforts, with explicit acknowledgment of any external contributions within the text, and was completed during the designated timeframe **August 2024 – January 2025**.

Place: Trivandrum ……………………………….

Date: 03/01/2025 Signature of Student

ACKNOWLEDGEMENT

I would like to convey my sincere appreciation to **Dr. Anoop V.S**, who serves as both a Professor and Course Coordinator at the School of Digital Sciences, Digital University Kerala, located in Trivandrum. His exceptional guidance and mentorship have been truly invaluable, which were instrumental in the successful completion of this project. I would like to extend my heartfelt thanks to him for creating a supportive environment and facilitating access to educational resources that contributed to this achievement. I extend my appreciation to **Prof. Ciza Thomas,** Vice Chancellor, Digital University Kerala, for providing access to their institutional facilities and resources. Additionally, I am profoundly thankful for the unwavering encouragement, motivation, and assistance offered by my friends and family throughout the course of this project.

ABSTRACT

Hate-speech detection in the domain of code-mixed Indo-Aryan languages, like Hinglish and Banglish, is a challenge due to the linguistic complexity, informal nature, and unavailability of structured data. This project solves the above-mentioned challenges using specialized datasets and advanced machine learning (ML) and deep learning (DL) techniques to examine comments within defined subcategories. We apply preprocessing techniques like tokenization, padding, and attention masking to prepare the data effectively for embedding in a BERT based multilingual model that offers enhanced contextualized representations. Experimental results show that BERT embeddings significantly outperform traditional feature extraction methods with higher accuracy in hate speech classification. Integrating machine learning classifiers with deep learning models offers a novel method for comprehending and identifying hate speech. This approach opens up opportunities to create comprehensive, resilient, and effective solutions within the field of natural language processing.

CONTENTS

TABLE OF FIGURES

CHAPTER-1
INTRODUCTION

Hate speech is a type of speech where the speaker targets a person or group with hate. They sometimes use humiliating words or derogatory terms based on their religion, gender, race, disability, or sexual orientation. It can also initiate violence against these groups of people. Nowadays, the use of social media has risen, and also the amount of hate speech is also risen. Because of this un, undetected hate speech can harm a person and even society. This research can help reduce discrimination and violence. It will also help policymakers and technology developers make further improvements in this domain.

This unique challenge emerges in multilingual regions like South Asia, in which languages are deeply interwoven with communication and culture, with the extensive use of code-mixed languages. Hindi is used by more than 609 million people, and Bengali by 272 million, ranking among the most widely used languages worldwide. Code-mixed forms that have grown in popularity particularly among young users include forms such as Hinglish (mix of Hindi and English) and Banglish (mix of Bengali and English) on web-based environments. While enriching the communication, the same brings complexity to tasks like natural language processing. Standard monolingual NLP models fail to capture the subtle nature of codemixed text due to the informal and constantly changing nature of code-mixed text. The task of hate speech detection thus becomes even harder.

Despite this advancement in hate speech detection, most tools built today are specifically for monolingual datasets, failing to solve the complexity issues in code-mixed text. This gap is especially concerning for Indo-Aryan languages, where limited linguistic resources, annotated datasets, and robust models exacerbate the challenges. The detection of hate speech in code-mixed Indo-Aryan languages like Hinglish and Banglish remains an underexplored problem.

This study aims at bridging this gap through curating representative datasets, coupled with machine learning and deep learning techniques, towards hate speech detection in code-mixed Hinglish and Banglish. Building on rigorous preprocessing methods, it seeks to make the online space safer while at the same time contributing towards the multilingual NLP research. This work is highly relevant to society, as it provides tools for countering the harmful effects of hate speech and advances the understanding of low-resource and code-mixed languages in NLP.

CHAPTER-2
RELATED WORK

Hate speech detection is currently a vibrant field of study of research within NLP community, motivated by the surge in online platforms and accompanying harmful content. Over the years, several approaches have been proposed, from the Conventional machine learning techniques to Current deep learning architectures. However, much of existing research focuses on monolingual text, with few explorations of hate speech detection in the field of code-mixed languages, especially in the context of Indo-Aryan.

Early work mostly used techniques such as Support Vector Machines, Naïve Bayes Algorithm, and Logistic Regression Algorithm, which are traditional methods of machine learning. As an example, Davidson et al. (2017) used logistic regression model coupled with bag-of-words approach representation in order to classify hate speech in English, thus showing competitive performance. While the results were interpretable, they required extensive feature engineering and were not quite capable of capturing the semantic complexity of hate speech.

Deep learning introduced its own models that advanced the hate speech detection process very significantly. For example, CNNs and RNNs have proved to do well in models like them. Zhang et al., (2018) did prove the extraction of context-based features through CNN, for classification of text. Even Badjatiya et al., (2017) showed how introducing LSTM along with boosting gradient could better the hatred speech classification task on any English dataset. These models are particularly effective for more nuanced tasks because they automate feature extraction. More recently, transformer-based models like Bidirectional Encoder Representations from Transformers (Devlin et al., 2019) have set Cutting-edge benchmarks through contextual embeddings and self-attention mechanisms.

However, these advancements notwithstanding, the area remains quite uncharted regarding the detection of hate speech in case of code-mixed languages. Texts which are code mixed, like Hinglish (Hindi-English) and Tanglish (Tamil-English), has unique challenges due to complex syntax, language switching, also the absence of annotated datasets. Mandal et al. (2020) suggests one of the first Hinglish hate speech datasets and applied traditional machine learning techniques, showing the need for tailored solutions to handle code-mixed text. Chakravarthi et al. (2021) extended this line of work using multilingual transformer models like mBERT for effective capture of linguistic nuances in code-mixed data.

Techniques to preprocess are important for handling code-mixed languages because of their irregular nature. Language tagging, transliteration, and removal of stop words has shown to improve the model's performance as reported by Mandal et al., 2020). Transfer learning techniques with the use of multi-language supporting models like XLM-R (Conneau et al., 2020) and MuRIL (Khanuja et al., 2021) also have proven promising outcomes with shared embeddings through multiple languages. The models overcome the major limitation of code-mixed NLP research, which is data scarcity.

Some studies outline the issues of hate speech detection in code-mixed languages such as Hindi-English due to the unavailability of traditional methods when the traditional methods fail against the complications of language switch and syntactic irregularities. Here, the authors suggest a machine learning-based model based on character level and word level feature extraction that combines features and results in surpassing the outcomes of traditionally used embeddings fastText and GloVe against hate speech classification in the code-mixed datasets. Additionally, it highlights value of preprocessing techniques, such as normalizing text and the removal of irrelevant elements for improvement in model performance.

These studies have greatly benefited the field, but such issues as data scarcity, linguistic ambiguity, and a dynamic nature of hate speech still prevail. This study expands on previous work with a well-curated, high quality dataset for code-mixed Indian languages hate speech detection in which experiments are conducted through machine learning and deep learning models, including approaches which are based on transformers. Our work bridges the gap of hate speech detection in low-resource and code-mixed language, which advances theory as well as provides a practical contribution to the area of NLP.

CHAPTER-3
METHODOLOGY

This Methodology section outlines the approach to detecting hate speech in Banglish and Hinglish languages (code-mixed). This is a multi-stage process, from data collection to preprocessing of the dataset before model training. Here, we explain the techniques for feature extraction, model selection, and model training to classify hate speech effectively. With all these complexities of code-mixed text, special importance is placed on handling the linguistic complexity and choosing an appropriate model to capture the nuances of the mixed languages' syntax and semantics. This section provides a comprehensive account of steps to build a hate-speech detection system by concentrating on state-of-the-art methods for code-mixed language processing.

3.1 Dataset Description

3.1.1 The Banglish Dataset

The Banglish dataset is suitable for hate speech detection. Banglish means Bengali written in English script. This dataset contains 5000 comments. These comments are divided into two classes. They are ‘hate speech’ and ‘non-hate speech’. Hate speech has 2836 comments, and non-hate speech has 2164 comments. The hate and non-hate speech ratio is 1.31: 1. The hate speech class is further divided into six subcategories. These are: ‘Racial,’ ‘Religious,’ ‘Appearance,’ ‘Sexual,’ ‘Slang,’ and ‘Others.’ The sexual subcategory has 928 comments, Slang has 575, racial has 135, religious has 198, appearance has 410, and others has 590 comments. These subcategories help us understand hate speech even more deeply. Here, the average comment length is 46.81 characters, the shortest is four, and the longest is 491. The median of the text length is 38 characters; it has a standard deviation of 35.98 characters, which shows variability in comment length. The banglish data shows deeper insights through the percentile matrix. The 25 percentile is 25 characters, the 50 percentile is 38, and the 75 percentile is 55. This type of detail annotations, category, and subcategory can help us build a system that detects hate speech using machine learning.

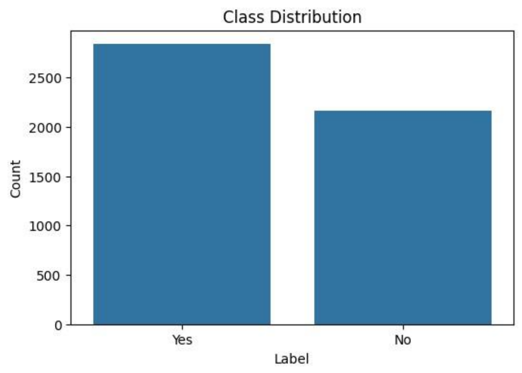


Figure 1: Class Distribution of Banglish Dataset

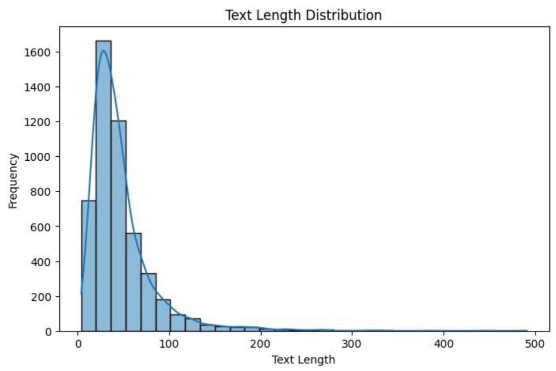


Figure 2: Text Length Distribution of Banglish Dataset

3.1.2 The Hinglish Dataset

The Hinglish dataset contains 4579 comments. These comments are web-scrapped from tweets. These comments are written in English, which is Hindi, and written in English. This dataset has two categories: ‘non-hate speech’ and ‘hate speech’. The hate speech had 1661 comments, and the non-hate speech had 2918 comments. The ratio of non-hate and hate speech is 0.57:1. Here, the average length is 107.34 characters. The minimum length is 12 characters, and the maximum is 341 characters. The median length is 104 characters, with a standard deviation of 55.77 characters. Percentiles show the 25th percentile at 66 characters, the median at 104, and the 75th percentile at 136. These insights are crucial for developing natural language processing (NLP) models that detect hate speech in code-mixed Aryan languages.

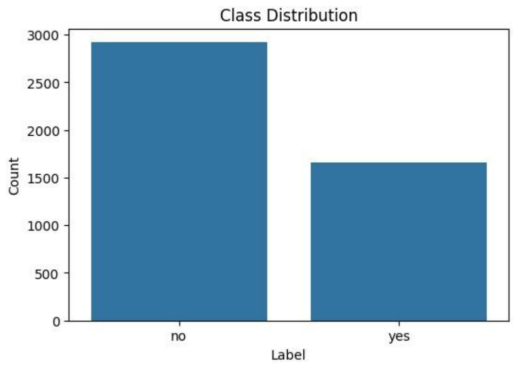


Figure 3: Class Distribution of Hinglish Dataset

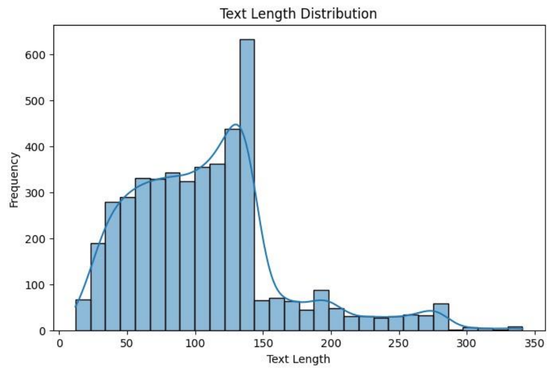


Figure 4: Text Length Distribution of Hinglish Dataset

3.2 Preprocessing

During preprocessing, the labels were changed to binary for classification, ensuring consistency by trimming spaces, standardizing text to lowercase, and mapping hate speech to 1 and non-hate to 0. Also, all non-alphanumeric characters were removed, and a clean dataset was created for further analysis.

There are further preprocessing requirements to ensure the data is presented in the proper format to the BERT to generate the embeddings. Some of the most crucial preprocessing includes tokenization: it splits text into subword tokens and also includes unique tokens such as [CLS] and [SEP]. Then, it converts the tokens into input_ids. After that, every token is mapped to its unique numerical ID in BERT vocabulary. At last, attention_mask was created to help understand the position of valid tokens. This ensures uniform input length through truncation or padding.

3.2.1 Tokenization

Tokenization is the first step. It breaks down raw text into smaller tokens or units. The BertTokenizer handles this process from the Transformers library. Also, the tokenizer splits words into subwords when necessary (for example, the word "programming" is split into "program" and "##ming"). That way, all rare or unknown words can still be represented quite well because they can be split into smaller units of meaningful parts.

3.2.2 Conversion to Input IDs

When the text is tokenized, each token gets mapped to a unique numerical ID from BERT's vocabulary. This conversion transforms text into a numeric version that is acceptable for the model to process.

3.2.3 Adding Special Tokens

BERT uses special tokens to format and comprehend input sequences properly. The sequence starts with the [CLS] token, while the [SEP] token is added at the end to represent the end of a sentence or segment. These special tokens will allow BERT to know the context and the boundaries of the input text.

3.2.4 Padding and Truncation

BERT operates on inputs of fixed length, so tokenized sequences need to be adjusted. Padding: Smaller sequences are padded with zero to match the length of the sequence (128 tokens). Truncation: Larger sequences are truncated or chopped off to ensure their length does not exceed the predefined length (128 tokens).

3.2.5 Attention Masks

An attention mask is generated to distinguish meaningful tokens from padding. Tokens are marked as one if they correspond to actual text and a zero when it is used for padding. This model then pays attention only to the significant portions of the sequence, while the padded elements are not considered. After preprocessing, every text comment is represented by two key arrays; Input IDs: The numerical form of the text, including all the padding or truncation applied to it, Attention Mask: A binary mask highlighting the meaningful tokens and padding elements. These arrays are then fed into the BERT model, which can perform classification with contextual understanding.

3.3 Embedding

We leverage BERT-base-multilingual-cased, a pre-trained Transformer model designed for multilingual applications to represent Hinglish and Banglish text effectively. This model accommodates 110,000 tokens in its multilingual vocabulary and provides robust embeddings by processing input text bidirectionally through self-attention mechanisms. The architecture has 12 encoder layers, which include 12 self-attention heads and a hidden size of 768. Every token's embedding is expressed as a 768-dimensional vector, thereby allowing the model to capture complex dependencies between words in both the left and right neighbouring tokens. The feed-forward network of each encoder layer has a hidden size of 3072, which facilitates more complex transformations of token embeddings. This model is cased, distinguishing between lowercase and uppercase letters, which is essential to preserve the case-sensitive meaning of English and other languages.

BERT creates contextualized embeddings for tokens, and it adapts based on its surrounding context. This feature is particularly valuable for handling polysemous words (words with multiple meanings) and the nuanced linguistic features inherent in code-mixed text. To adapt BERT for tasks like hate speech detection, the embedding corresponding to the [CLS] token, which summarizes the entire input sequence, is extracted. This contextualized representation is passed to subsequent layers for classification or other tasks. BERT's architecture is highly versatile, and it can successfully handle various NLP challenges, such as sentiment analysis, question answering, and, in this case, detecting hate speech in Hinglish and Banglish. By fine-tuning BERT on our datasets, we ensure the model learns domain-specific features and nuances of code-mixed languages, enabling accurate and context-aware hate speech detection.

3.4 Machine Learning Models

The machine learning algorithms applied in this project to classify comments into Hate along with Non-Hate categories using the 768-dimensional CLS token embedding from BERT include Logistic Regression, Support Vector Machine Algorithm (SVM), Gradient Boosting Algorithm, Decision Tree, K-Nearest Neighbors (KNN) Algorithm, Random Forest, and Bernoulli Naive Bayes (BernoulliNB). Logistic Regression is a type of linear classifier that calculates the probability of the sample affiliated to the 'Hate' class by assigning weights to the input features, 768dimensional CLS embeddings. The model then applies a sigmoid function to obtain the output. SVM creates a hyper-plane in the high-dimensional vector space of BERT embedding to discover the decision boundary that maps the value of margin to its maximum between the Hate and Non-Hate class samples. Decision Tree, on the other hand, splits the data based on the feature value and then it learns the decision rules, making it easier to interpret the classification process. Random Forest algorithm is an ensemble technique, it integrates multiple decision trees to increase classification capabilities. Each decision tree is trained on a subset of data which is derived from randomization, and then last prediction is obtained by adding the individual tree predictions. Gradient Boosting sequentially builds weak learners (typically decision trees), where each new tree corrects itself by the estimating the errors of the previous one, producing a stronger predictive model. K-Nearest Neighbors (KNN) is a algorithm where it uses non-parametric method that classifies new samples which is based on the majority class of their nearest neighbors in the specific feature space, with the distance between data points computed using metrics such as Euclidean distance. Finally, Bernoulli Naive Bayes is a probabilistic classifier that applies Bayes' theorem to calculate the posterior probability of a comment belonging to the Hate class, assuming binary features (presence or absence of certain features), and classifies based on the highest probability. Each of these models uses the rich, context aware representations provided by BERT embeddings to effectively capture the semantic nuances necessary for accurate classification.

3.5 Dense Layer Classifier Model

The dense layer in this architecture takes the 768-dimensional [CLS] embedding from the BERT model as input and performs binary classification. It comprises 768 input nodes corresponding to the [CLS] embedding dimensions and is connected directly to a single output node with no hidden layers. Hence, it is a single-layer perceptron. Every input node has an associated weight W and bias b, and the dense layer computes the output z as:

$$z = W . x + b$$

Now, the output goes into a sigmoid function, which is:

$$\sigma(z) = 1 / (1 + e^{-z})$$

It produces a probability $\hat{y}$ between 0 and 1, where probabilities closer to 1 mean "Hate," and closer to 0 mean "Non-Hate." This model is optimized with ‘binary cross-entropy’ loss:

$$Loss = -1/N \Sigma [y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})]$$

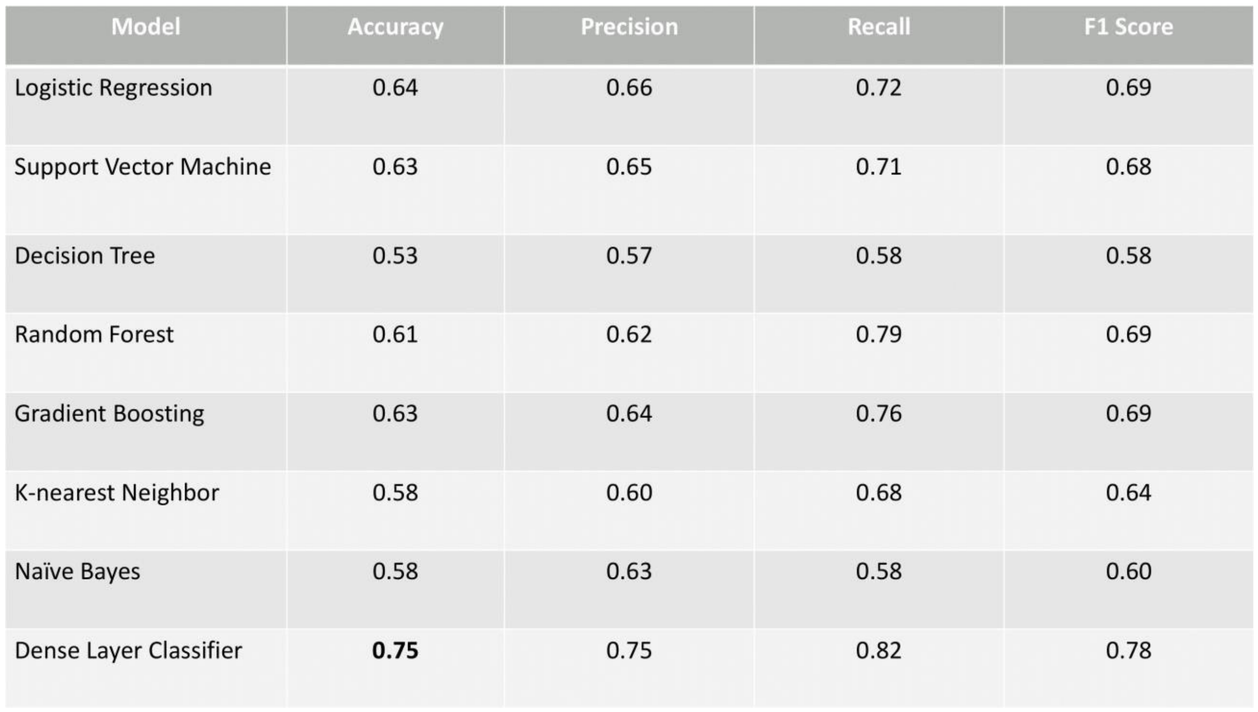
Here, $\hat{y}$ is the predicted probability, and y is the actual label. Adam optimizer (an adaptive gradient-based algorithm) optimizes weights W and bias b for efficient and stable training. After training, the final output probabilities are thresholded (e.g., 0.5) for classification of the comment or text as ‘Hate’ or ‘Non-Hate.’

CHAPTER-4
RESULTS

In this section the results of our study is shown, highlighting the performance of various feature extraction techniques and classification models. Our experiments show that BERT-base-multilingual-cased does a significantly better job compared to traditional feature extraction approaches, like TF-IDF, for capturing semantic as well as contextual subtleties of code-mixed text. The strength of handling multiple languages in BERT along with sub word tokenization helps make it stronger in processing more common as well as out-of-vocabulary words robustly. These strengths in BERT make it really effective to represent the inherent complex linguistic structures in datasets like Banglish and Hinglish.

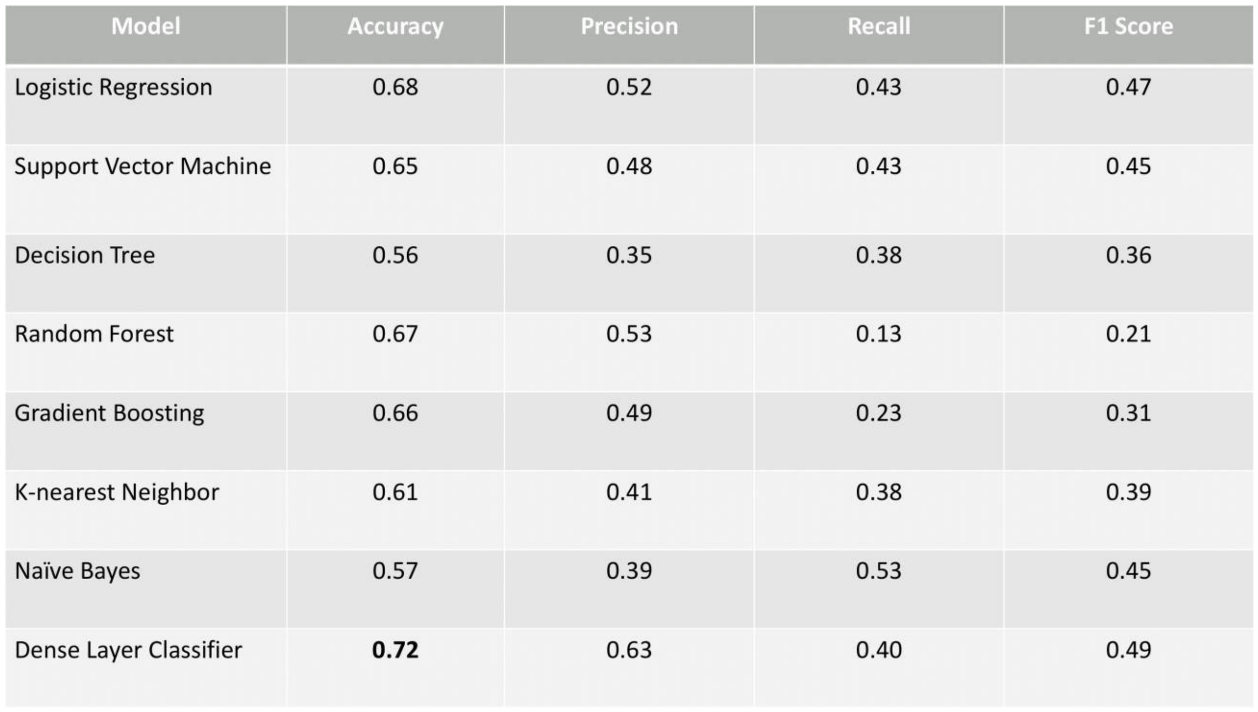
Regarding the execution of classification, dense neural network architecture using BERT embeddings gave better performance than the classic machine learning algorithms like Logistic Regression algorithm, and Random Forest algorithm, Support Vector Machine algorithm. The dense layer uses the rich context-aware embedding produced by BERT, which means it results in better accuracy and generalization in the domain of hate speech classification tasks. Study shows that the balance of datasets is important. In this case, an imbalanced distribution of hate and non-hate speech data has a negative effect on the performance metrics, particularly the F1-score, which is crucial in calculating the model's competence to balance precision, also the recall. A dataset which is balanced ensures that both classes are fairly represented and reduces the possibility of the model becoming biased towards the majority class.

In summary, our results highlight the effectiveness of modern deep learning approaches, especially BERT, to portray the challenges in hate speech detection in code-mixed languages. These findings emphasizes that this field requires a careful selection of data preparation models to ensure better results.



Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.64	0.66	0.72	0.69
Support Vector Machine	0.63	0.65	0.71	0.68
Decision Tree	0.53	0.57	0.58	0.58
Random Forest	0.61	0.62	0.79	0.69
Gradient Boosting	0.63	0.64	0.76	0.69
K-nearest Neighbor	0.58	0.60	0.68	0.64
Naïve Bayes	0.58	0.63	0.58	0.60
Dense Layer Classifier	**0.75**	0.75	0.82	0.78

Figure 5: Comparison of Banglish Results



Model	Accuracy	Precision	Recall	F1 Score
Logistic Regression	0.68	0.52	0.43	0.47
Support Vector Machine	0.65	0.48	0.43	0.45
Decision Tree	0.56	0.35	0.38	0.36
Random Forest	0.67	0.53	0.13	0.21
Gradient Boosting	0.66	0.49	0.23	0.31
K-nearest Neighbor	0.61	0.41	0.38	0.39
Naïve Bayes	0.57	0.39	0.53	0.45
Dense Layer Classifier	**0.72**	0.63	0.40	0.49

Figure 6: Comparison of Hinglish Results

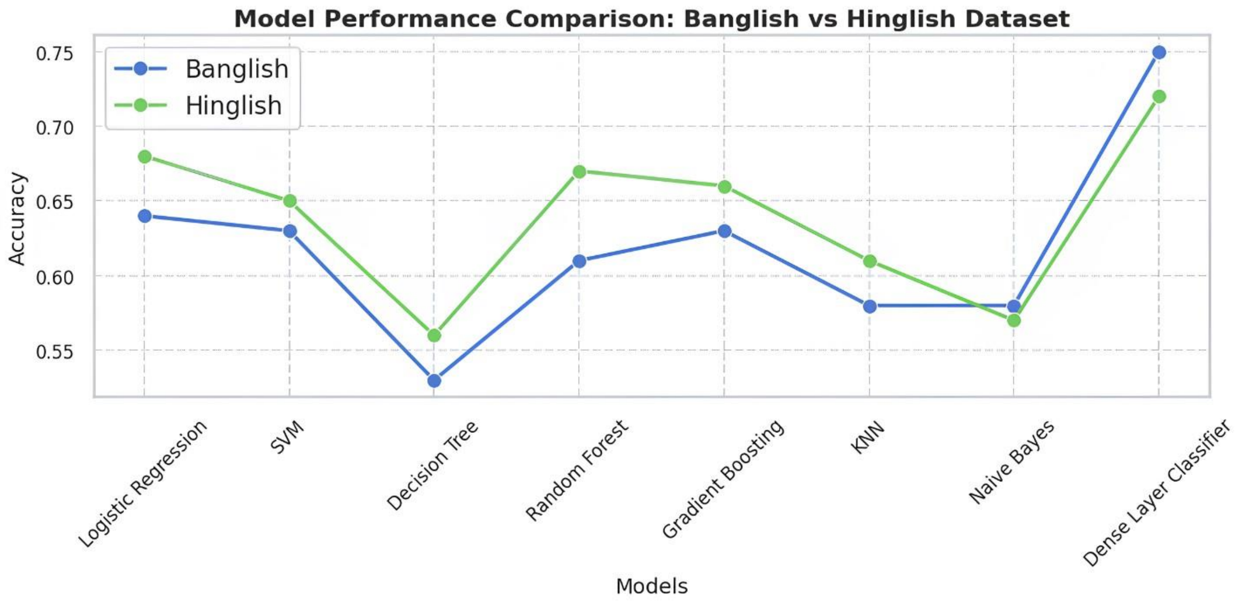
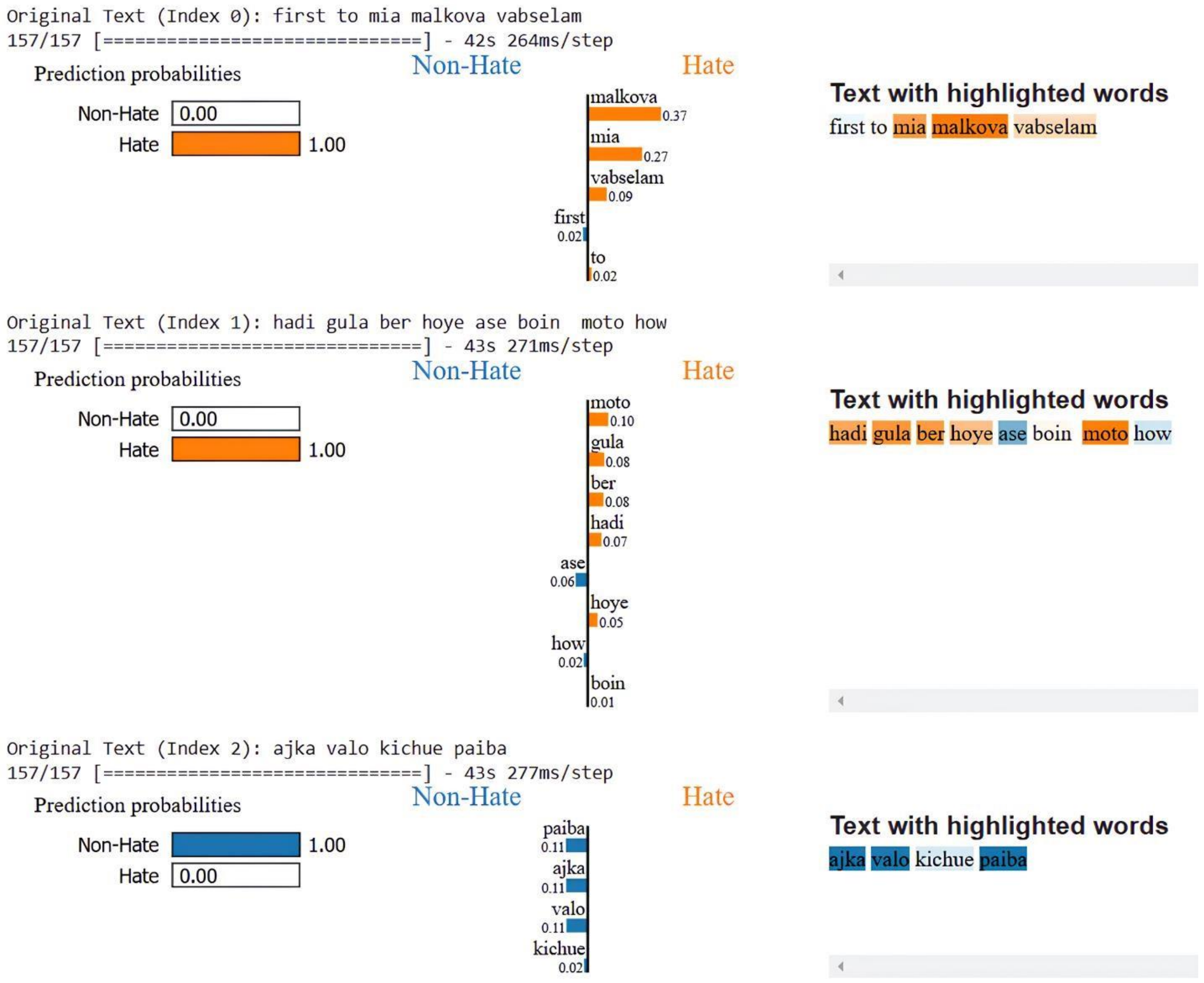


Figure 7: Model Performance Comparison: Banglish vs Hinglish

4.1 Model Explainability Using LIME

Lime is a technique that can help us understand the prediction of a machine learning or a deep learning system. Lime or Local Interpretable Model-Agnostic Explanations identifies the factors influencing the machine learning model to predict a specific output. Lime also helps the developer by generating visuals for the result so that the developer can easily understand the outcome. It highlights a particular word or phrase being fed as an input, and that word or phrase is responsible for the prediction. This also helps us identify bias in the model. Here, lime showed that the model learned to detect hate speech while maintaining interpretability. Interpretability is very important for hate speech detection.



```
Original Text (Index 0): first to mia malkova vabselam
157/157 [==============================] - 42s 264ms/step
```

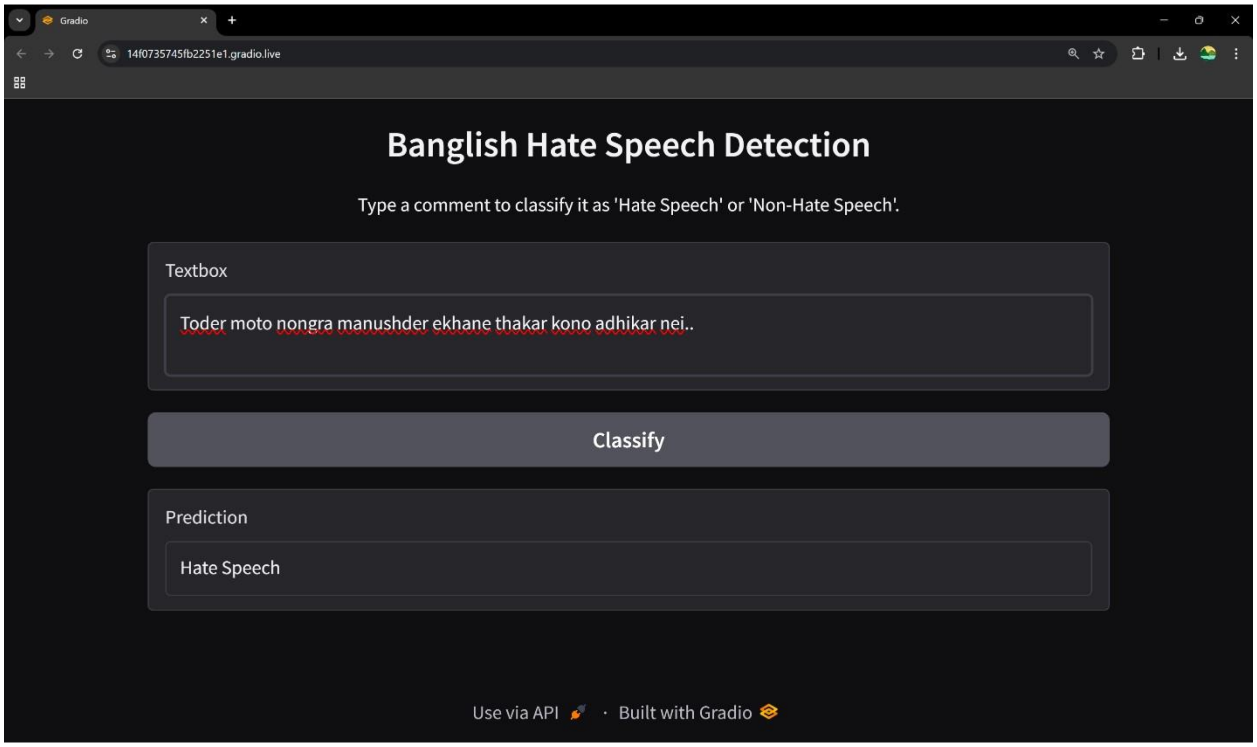
```
Original Text (Index 1): hadi gula ber hoye ase boin  moto how
157/157 [==============================] - 43s 271ms/step
```

```
Original Text (Index 2): ajka valo kichue paiba
157/157 [==============================] - 43s 277ms/step
```

Figure 8: Model interpretability with LIME

CHAPTER-5
IMPLEMENTATION

Gradio is a sound open-source Python library. This library can be used to develop user-friendly interfaces for machine learning systems and deep learning systems. This is also user-friendly and customizable. This helps developers to deploy the model quickly in the cloud. This is also good for reaching a larger audience of researchers, regular users, and NLP enthusiasts through interactive radio web applications. Radio makes it easy for non-technical users to use the ML models easily. Here, we have chosen the top-performing model for Banglish, Hinglish and used that model to develop the Gradio app. This application enables the user to detect hate speech or non-hate speech.


Gradio
14f0735745fb2251e1.gradio.live
Classify
Use via API · Built with Gradio

Banglish Hate Speech Detection

Type a comment to classify it as 'Hate Speech' or 'Non-Hate Speech'.

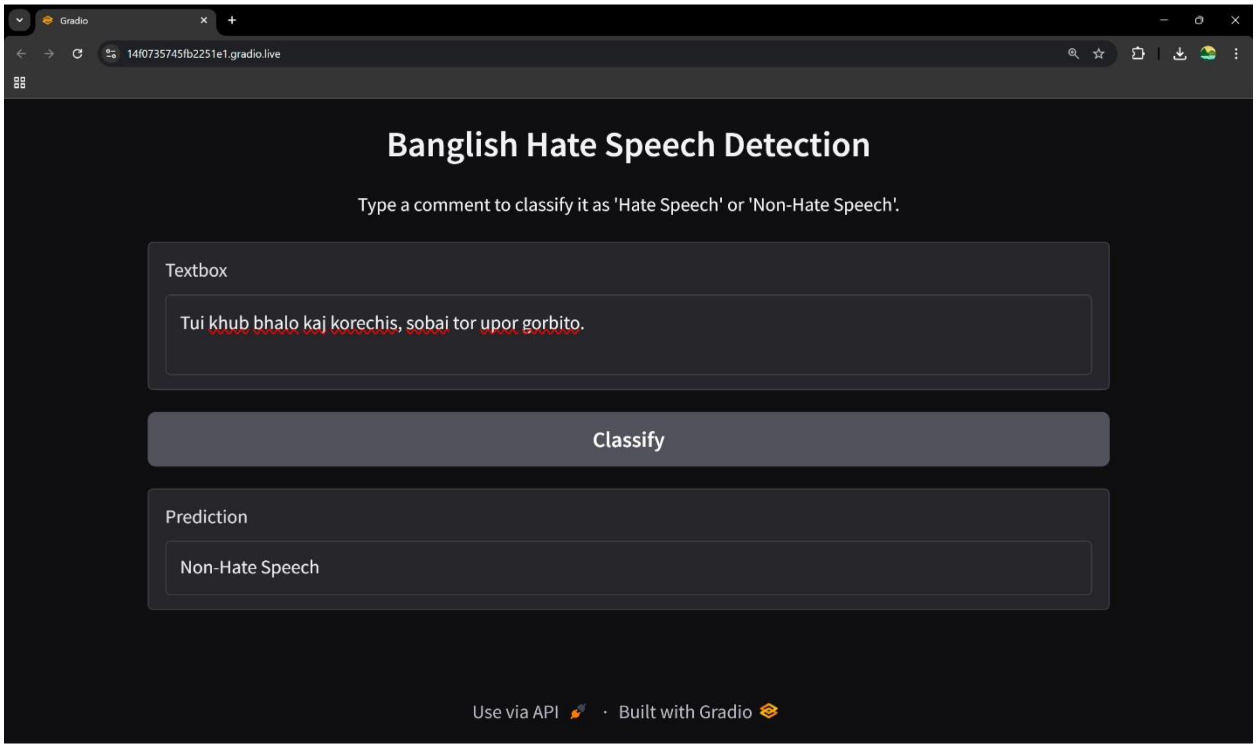
Textbox

Toder moto nongra manushder ekhane thakar kono adhikar nei..

Prediction

Hate Speech

Figure 9: Detection of Hate Speech in Banglish


Gradio
14f0735745fb2251e1.gradio.live

Banglish Hate Speech Detection

Type a comment to classify it as 'Hate Speech' or 'Non-Hate Speech'.

Textbox

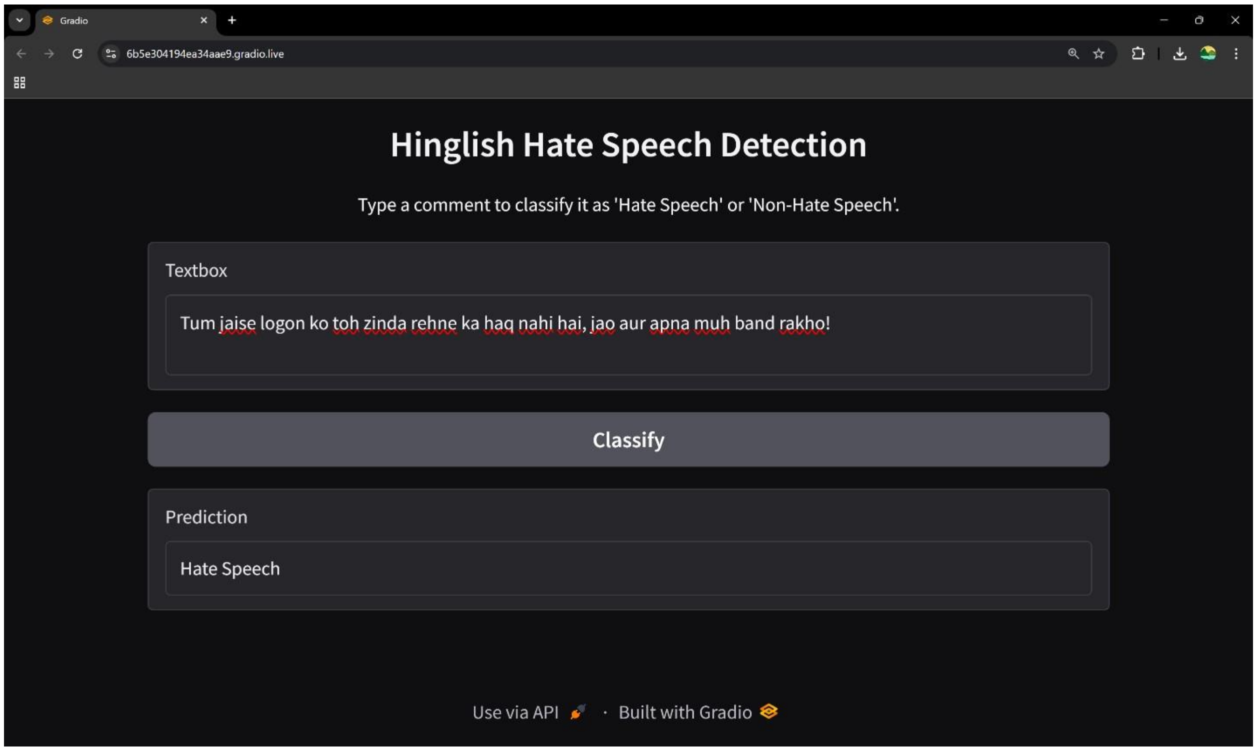
Tui khub bhalo kaj korechis, sobai tor upor gorbito.

Prediction

Non-Hate Speech

Use via API · Built with Gradio

Figure 10: Detection of Non-Hate Speech in Banglish


Gradio
6b5e304194ea34aae9.gradio.live

Hinglish Hate Speech Detection

Type a comment to classify it as 'Hate Speech' or 'Non-Hate Speech'.

Textbox

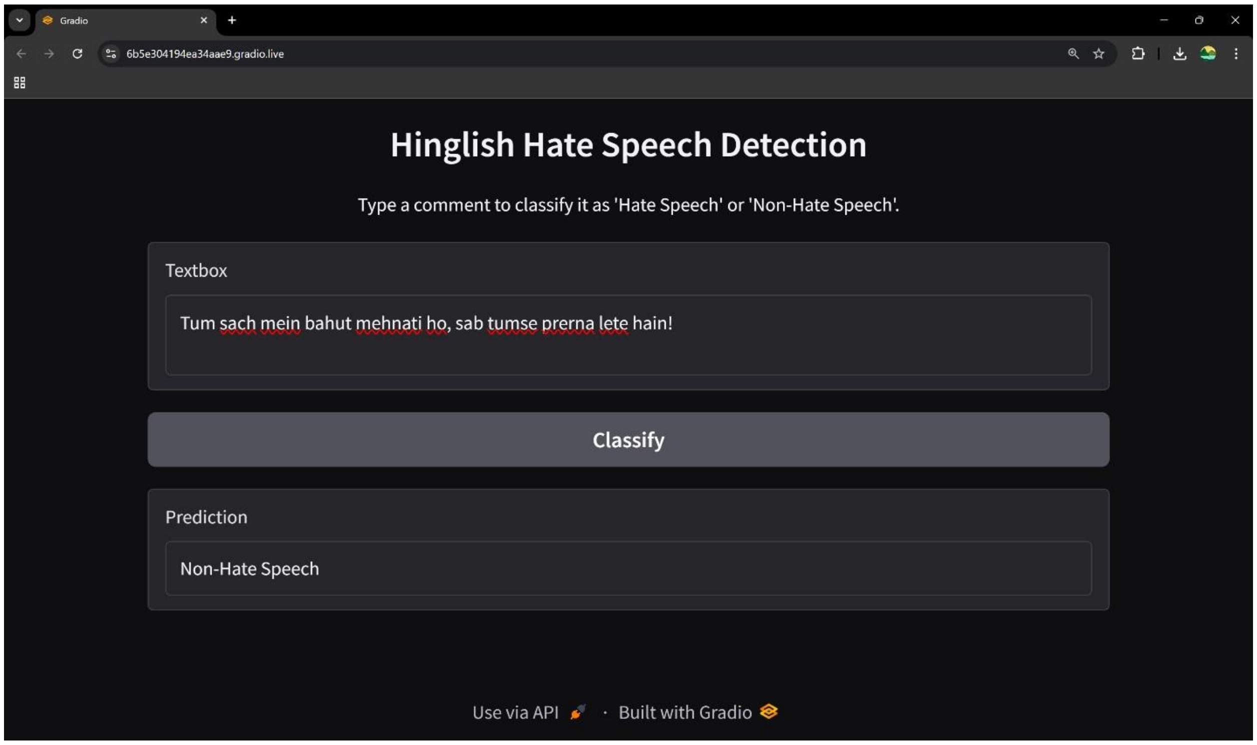
Tum jaise logon ko toh zinda rehne ka haq nahi hai, jao aur apna muh band rakho!

Prediction

Hate Speech

Use via API · Built with Gradio

Figure 11: Detection of Hate Speech in Hinglish


Gradio

6b5e304194ea34aae9.gradio.live

Hinglish Hate Speech Detection

Type a comment to classify it as 'Hate Speech' or 'Non-Hate Speech'.

Tum sach mein bahut mehnati ho, sab tumse prerna lete hain!

Non-Hate Speech

Figure 12: Detection of Non-Hate Speech in Hinglish

CHAPTER-6
CONCLUSION

This Mini Project dives into the difficulties and the unique challenges of working with code mixed in Aryan languages, especially Banglish and Hinglish. By employing a BERT-based multilingual model, we effectively address issues such as irregular syntax and linguistic ambiguity using specially curated datasets. Our findings indicate that deep learning models, especially those utilizing BERT embeddings, surpass traditional machine learning classifiers in performance. We emphasize the crucial role of balanced datasets in achieving optimal model efficacy, ensuring a fair representation of both hate and non-hate speech. Looking ahead, we aim to explore hyperparameter tuning, implement advanced architectures, and investigate cross-lingual applications. This research contributes to understanding multilingual hate speech detection and has significant implications for academia and society, enhancing online safety and advancing the processing of low-resource languages.

REFERENCES

1. Davidson, T., Warmsley, D., Macy, M. W., & Weber, I. (2017). Automated hate speech detection and the problem of offensive language. *Proceedings of the Eleventh International Conference on Web and Social Media (ICWSM)*.
2. Zhang, Z., Robinson, D., & Tepper, J. (2018). Detecting hate speech on Twitter using a convolutional neural network. *Proceedings of the International Conference on Neural Information Processing*.
3. Badjatiya, P., Gupta, S., Gupta, M., & Varma, V. (2017). Deep learning for hate speech detection in tweets. *Proceedings of the 26th International Conference on World Wide Web Companion*.
4. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*.
5. Mandal, A., Das, D., & Chakraborty, S. (2020). Hate speech detection in code-mixed Hindi-English social media text. *Proceedings of the 14th International Conference on Natural Language Processing (ICON)*.
6. Chakravarthi, B. R., Priyadharshini, R., & McCrae, J. P. (2021). A sentiment analysis dataset for codemixed Tamil-English text. *Proceedings of the 2021 Workshop on Computational Approaches to Linguistic Code-Switching*.
7. Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., ... & Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
8. Khanuja, S., Dandapat, S., Srinivasan, A., & Bali, K. (2021). MuRIL: Multilingual Representations for Indian Languages. *arXiv preprint arXiv:2103.10730*.
9. Bohra, A., Vijay, D., Singh, V., Akhtar, S. S., and Shrivastava, M. (2018). A dataset of hindi-english code-mixed social media text for hate speech detection. In Proceedings of the second workshop on computational modeling of people's opinions, personality, and emotions in social media, pages 36–41.
10. K. Thapliyal, M. Thapliyal, and D. Thapliyal, "Social media and health communication: A review of advantages, challenges, and best practices," Emerg. Technol. Health Literacy Med. Pract., vol. 1, pp. 364–384, Jul. 2024.
11. E. Hashmi, M. M. Yamin, and S. Y. Yayilgan, "Securing tomorrow: A comprehensive survey on the synergy of artificial intelligence and information security," AI Ethics, vol. 1, pp. 1–19, Jul. 2024.
12. S. J. Johnson, M. R. Murty, and I. Navakanth, "A detailed review on word embedding techniques with emphasis on word2vec," Multimedia Tools Appl., vol. 83, no. 13, pp. 37979–38007, Oct. 2023.
13. D. S. Asudani, N. K. Nagwani, and P. Singh, "Impact of word embedding models on text analytics in deep learning environment: A review," Artif. Intell. Rev., vol. 56, no. 9, pp. 10345–10425, Sep. 2023.
14. F. Mehmood, H. Ghafoor, M. N. Asim, M. U. Ghani, W. Mahmood, and A. Dengel, "Passion-net: A robust precise and explainable predictor for hate speech detection in Roman Urdu text," Neural Comput. Appl., vol. 36, no. 6, pp. 3077–3100, Feb. 2024
15. J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532–1543.
16. K. Kumari, J. P. Singh, Ai_ml_nit_patna@ trac-2: deep learning approach for multi-lingual aggression identification, in: Proceedings of the second workshop on trolling, aggression and cyberbullying, 2020, pp. 113–119.
17. K. Kumari, J. P. Singh, Ai ml nit patna at hasoc 2019: Deep learning approach for identifi- cation of abusive content., FIRE (working notes) 2517 (2019) 328–335

18. S. Abro, S. Shaikh, Z. H. Khand, A. Zafar, S. Khan, G. Mujtaba, Automatic hate speech detection using machine learning: A comparative study, International Journal of Advanced Computer Science and Applications 11 (2020).
19. Chakravarthi, B. R., Arcan, M., and McCrae, J. P. (2018). Improving wordnets for under-resourced languages using machine translation. In Proceedings of the 9th Global WordNet Conference (GWC 2018), page 78.
20. Nobata, C., Tetreault, J., Thomas, A., Mehdad, Y., and Chang, Y. (2016). Abusive language detection in online user content. In Proceedings of the 25th International Conference on World Wide Web, WWW '16, pages 145– 153, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.
21. Mathur, P., Shah, R., Sawhney, R., and Mahata, D. (2018). Detecting offensive tweets in hindi-english code-switched language. In Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media, pages 18–26.
22. MacAvaney, S., Yao, H.-R., Yang, E., Russell, K., Goharian, N., and Frieder, O. (2019). Hate speech detection: Challenges and solutions. PLOS ONE, 14(8):1–16, 08.
23. S. Nagar, F. A. Barbhuiya, and K. Dey, ''Towards more robust hate speech detection: Using social context and user data,'' Social Netw. Anal. Mining, vol. 13, no. 1, p. 47, Mar. 2023.
24. A. Founta, C. Djouvas, D. Chatzakou, I. Leontiadis, J. Blackburn, G. Stringhini, A. Vakali, M. Sirivianos, and N. Kourtellis, ''Large scale crowdsourcing and characterization of Twitter abusive behavior,'' in Proc. Int. AAAI Conf. Web Social media, vol. 12, 2018, pp. 1–26.
25. A. A. Nagra, K. Alissa, T. M. Ghazal, S. Kukunuru, M. M. Asif, and M. Fawad, ''Deep sentiments analysis for Roman Urdu dataset using faster recurrent convolutional neural network model,'' Appl. Artif. Intell., vol. 36, no. 1, Dec. 2022, Art. no. 2123094.
26. S. Chen, J. Wang, and K. He, ''Chinese cyberbullying detection using XLNet and deep bi-LSTM hybrid model,'' Information, vol. 15, no. 2, p. 93, Feb. 2024.
27. R. S. Satpute and A. Agrawal, ''A critical study of pragmatic ambiguity detection in natural language requirements,'' Int. J. Intell. Syst. Appl. Eng., vol. 11, no. 3s, pp. 249–259, 2023.
28. K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, ''Electra: Pretraining text encoders as discriminators rather than generators,'' 2020, arXiv:2003.10555.
29. M. M. Khan, K. Shahzad, and M. K. Malik, ''Hate speech detection in Roman Urdu,'' ACM Trans. Asian Low-Resource Lang. Inf. Process., vol. 20, no. 1, pp. 1–19, Jan. 2021.
30. N. Seemann, Y. S. Lee, J. Höllig, and M. Geierhos, ''Generalizability of abusive language detection models on homogeneous German datasets,'' Datenbank-Spektrum, vol. 23, no. 1, pp. 15–25, Mar. 2023.
31. J. A. García-Díaz, M. Cánovas-García, R. Colomo-Palacios, and R. Valencia-García, ''Detecting misogyny in Spanish tweets. An approach based on linguistics features and word embeddings,'' Future Gener. Comput. Syst., vol. 114, pp. 506–518, Jan. 2021.
32. F. E. Ayo, O. Folorunso, F. T. Ibharalu, I. A. Osinuga, and A. Abayomi-Alli, ''A probabilistic clustering model for hate speech classification in Twitter,'' Expert Syst. Appl., vol. 173, Jul. 2021, Art. no. 114762. 33. J. C. Pereira-Kohatsu, L. Quijano-Sánchez, F. Liberatore, and M. Camacho-Collados, ''Detecting and monitoring hate speech in Twitter,'' Sensors, vol. 19, no. 21, p. 4654, Oct. 2019.
34. I. Bigoulaeva, V. Hangya, I. Gurevych, and A. Fraser, ''Label modification and bootstrapping for zeroshot cross-lingual hate speech detection,'' Lang. Resour. Eval., vol. 57, no. 4, pp. 1515–1546, Dec. 2023.

35. J. Fillies, M. P. Hoffmann, and A. Paschke, "Multilingual hate speech detection: Comparison of transfer learning methods to classify German, Italian, and Spanish posts," in Proc. IEEE Int. Conf. Big Data, Dec. 2023, pp. 5503–5511
36. Kumar, G. and Singh, J.P., 2022. Hate Speech and Offensive Content Identification in English and Indo-Aryan Languages using Machine Learning Models. In *FIRE (Working Notes)* (pp. 542-551).