Hate Speech Detection in Hindi language using Machine learning and pre-trained models

Submitted in partial fulfillment of the requirements

of the degree of

Bachelor of Engineering in Information Technology

By

PRABHIT CHAUGULE (Roll No. 20101A0031)

APURVA PADAGPALLIWAR (Roll No. 20101A0026)

RAHUL RATHOD (Roll No. 20101A0059)

Under the Guidance of

Prof. RASIKA RANSING

Department of Information Technology

VIT Vidyalankar Institute of Technology

ACCREDITED A+ BY NAAC

Autonomous Institute affiliated University of Mumbai

Vidyalankar Institute of Technology

Wadala(E), Mumbai-400437

University of Mumbai

2023-24

CERTIFICATE OF APPROVAL

This is to certify that the project entitled

"Hate Speech Detection in Hindi language using Machine learning and pre-trained models"

is a bonafide work of

PRABHIT CHAUGULE (Roll No. 20101A0031)

APURVA PADAGPALLIWAR (Roll No. 20101A0026)

RAHUL RATHOD (Roll No. 20101A0059)

submitted to the University of Mumbai in partial fulfilment of the requirement for the award of the degree of

Undergraduate in **"INFORMATION TECHNOLOGY"**.

Prof. Rasika Ransing	Dr. Vipul Dalal	Dr. S. A. Patekar
Guide	Head of Department	Principal

Project Report Approval for B. E.

This project report entitled ***Hate Speech Detection in Hindi language using Machine learning and pre-trained models*** by

1. ***PRABHIT CHAUGULE (Roll No. 20101A0031)***
2. ***APURVA PADAGPALLIWAR (Roll No. 20101A0026)***
3. ***RAHUL RATHOD (Roll No. 20101A0059)***

is approved for the degree of ***Bachelor of Engineering in Information Technology.***

Examiners

1.----------

2.----------

Date:

Place:

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Name of student	Roll No.	Signature
1. Prabhit Chaugule	20101A0031	
2. Apurva Pagadpalliwar	20101A0026	
3. Rahul Rathod	20101A0059	

Date:

ACKNOWLEDGEMNT

We are honoured to present "**Hate Speech Detection in Hindi language using Machine learning and pre-trained models**" as our B.E Final Year Project. We are using this opportunity to express our profound gratitude to our principal "**Dr. Sunil Patekar**" for providing us with all proper facilities and support.

We express our deepest gratitude towards our HOD "**Dr. Vipul Dalal**" for his valuable and timely advice during the various phases in our project. We would like to thank our project guide **"Prof. Rasika Ransing"** for support, patience and faith in our capabilities and for giving us flexibility in terms of working and reporting schedules. Finally, we would like to thank everyone who have helped us directly or indirectly in our project.

We would also like to thank our staff members and lab assistant for permitting us to use computer in the lab as when required. We thank our college for providing us with excellent facilities that helped us to complete and present this project.

Prabhit Chaugule
Apurva Pagadpalliwar
Rahul Rathod

ABSTRACT

Detecting hate speech is a crucial aspect of creating a safe and inclusive virtual space. As Hindi digital communication expands, it has become necessary to devise effective techniques for detecting hate speech in this language. This abstract presents a machine learning and natural language processing (NLP) approach for hate speech detection in Hindi texts. To this end, the research employs a dataset that consists of labeled Hindi text examples divided into two classes: hate speech and nonhate speech. This means that specific pre-processing techniques such as tokenizing, stemming and stop-word removal must be used to handle linguistic nuance in Hindi. For feature engineering-word embeddings, character level embeddings and TF-IDF vectors are extracted to represent the semantic meaning of the text. These features can then be used to train different machine learning algorithms including Support Vector Machines (SVM), Naïve Bayes and Random Forests. The models were trained using standard performance metrics like precision recall F1- score accuracy The proposed system for detecting hate speech could be extended to social media platforms or online forums or content sharing websites with an aim of curbing hateful content spread to more tolerant online communities among people who speak Hindi.

CONTENTS

List of Figures

CHAPTER 1
INTRODUCTION

INTRODUCTION

Social media platforms are now very common and have greatly increased the way we communicate and share information, giving people more chances to speak their minds than in the past. This online world has made it faster to share what we think with others and has helped us connect, but at the same time, it has led to a growth in negative talk on the internet. According to Nockleyby, hate speech covers all types of communication like talking, writing or actions that attack people or use negative and unfair words against them because of their religion, country they come from, skin color, if they are male or female, or different parts of who they are. The increase in such hateful language on the internet is now a big worry. It's a problem not just for those people it hurts directly but also for everyone else because this kind of talk can lead to actual crimes based on hate.

1.1 Problem Statement

With the proliferation of social media platforms, online communities, and digital communication channels, the rapid spread of hate speech has become a significant problem, impacting individuals and groups, and fostering toxic online environments. The detection and mitigation of such content are crucial for maintaining healthy and respectful online interactions. However, the complexity increases when addressing languages with rich linguistic diversity and varied dialects, such as Hindi.

1.2 Scope

The project will focus on gathering a comprehensive dataset that includes text from various sources such as social media platforms, online forums, news comments, and other digital media where Hindi is predominantly used. This dataset will need to be meticulously annotated to identify instances of hate speech, encompassing both explicit and implicit forms. Collaboration with expert linguists and trained annotators who are proficient in different dialects of Hindi will be essential to ensure the accuracy and reliability of the annotations.

1.3 Motivation

The primary motivation for developing a hate speech detection system in Hindi is to enhance online safety and promote healthier communication environments. Social media platforms and other digital forums are increasingly being used for harmful activities, including the dissemination of hate speech that can incite violence, spread discrimination, and harm mental health. By effectively identifying and mitigating such content, the project aims to protect individuals and communities from the adverse effects of hate speech.

CHAPTER 2

LITERATURE SURVEY

LITERATURE SURVEY

Finding hate speech in languages that do not have many resources, such as Hindi, is hard because there are not enough language tools and research. To solve this problem, people who study this topic have tried different advanced machine learning techniques like CNN 1D, LSTM, and BiLSTM. They also use special word embeddings made for certain topics. A new research used complex deep learning methods like CNN 1D, LSTM, and BiLSTM with special word embeddings for each person to look at data that had both Hindi and English. The outcomes showed that these advanced approaches did better than usual machine learning ways like SVM and random forests.Studies comparing different methods have been done, like research to find hate speech in English Twitter messages. They looked at many strategies, for example Logistic Regression, Decision Trees, Random Forests and Naive Bayes. They also used TF-IDF and BOW techniques to make features. Using both the ready-made word vectors like GLoVe and also our own made ones helped to make LSTM and GRU models work better.

In research on Hindi language, scientists have studied how well machine learning and brain-like network methods can recognize hate speech. They looked at older ways such as Linear SVM, Adaboost, Random Forests, and Voting Classifier and compared them with newer deep learning models that use LSTM technology. Notably, machine learning models demonstrated superior performance, particularly in low-resource settings.Various ways to classify Hindi writing have been studied, such as BOW, CNN, LSTM, BiLSTM, BERT and LASER techniques. The CNN approach that uses fast text embeddings designed for the Hindi language showed strong effectiveness. Moreover, studies investigating hostile messages in Hindi looked into different methods like CNN, Multi-CNN, BiLSTM and techniques using BERT with FastText word images. Outcomes showed a preference for constructions based on BERT, especially when combined with a complex CNN model that includes FastText language embeddings from IndicNLP.Hate speech detection work has grown with deep learning and big models of pre-trained language. These models use word embeddings that trained on many texts without someone guiding them, making better their skill to find hate speech. Using FastText together with layers of Bidirectional Gated Recurrent Units, which are called BiGRUs for short, is giving good outcomes. It highlights how well both FastText and BERT work when it comes to understanding the meanings of words and the context they're used in To sum up, scientists are looking into various frameworks, word representation techniques and ways to create features for fighting hate speech in languages that don't have many resources such as Hindi. This changing field highlights how methods for detecting hate speech keep improving and take into account the specific details of the context.

CHAPTER 3

SYSTEM DESIGN

PROPOSED SYSTEM

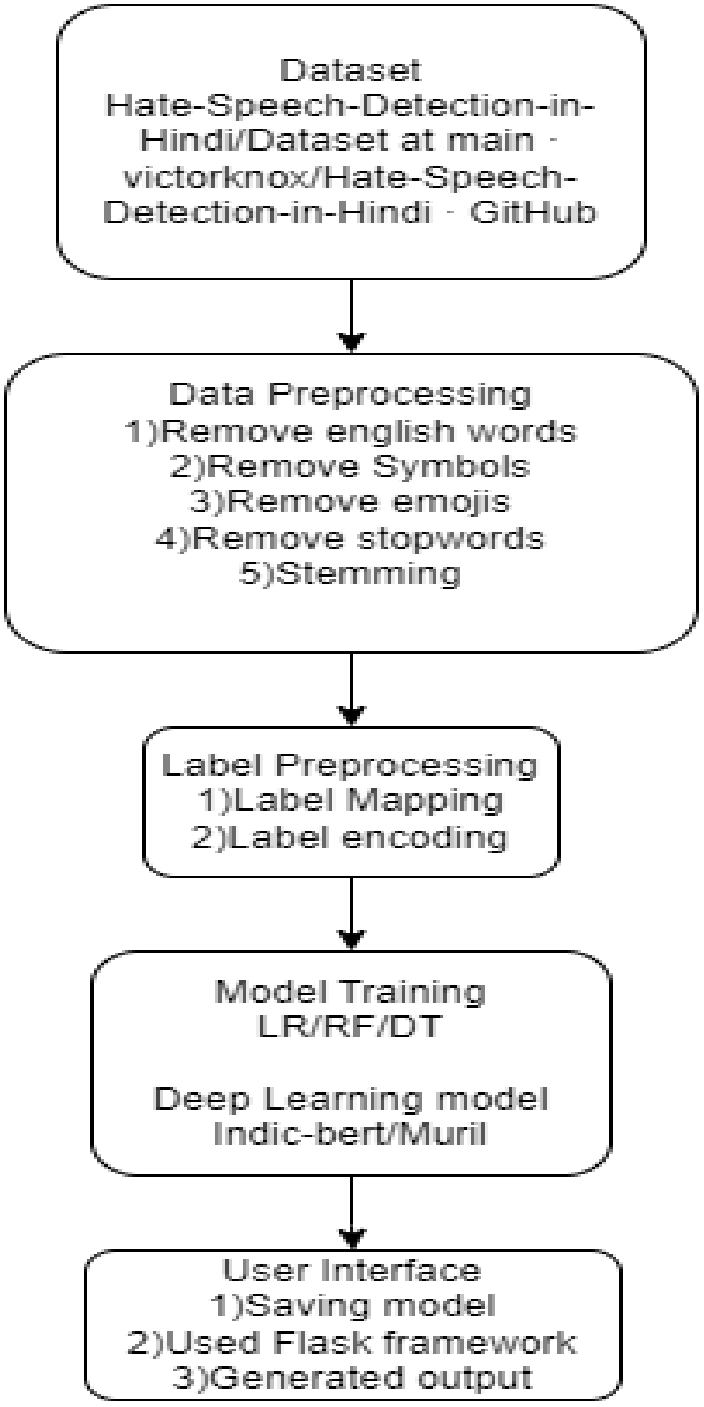

Fig 3.1: Flowchart

The flowchart represents the workflow for hate speech detection in Hindi.

1. Model Selection: We have put into use different kinds of machine learning models such as Logistic Regression, Random Forest, Multinomial Naive Bayes, the Support Vector Classifier also known as SVC, LightGBM and Decision Tree Classifier.Focused on models showing promising accuracy for hate speech detection in Hindi.

2. Feature Extraction: Used TF-IDF vector technique to change text data into numbers by looking at how often words appear and thinking about all the data together.Achieved efficient feature extraction using the TfidfVectorizer from the sklearn library.

3. Training Process: Developed a reusable function named train to streamline the training process for each model. Used the train_test_split function to split the dataset into parts for training and tests, giving 20% to testing.

4. Model Evaluation: Evaluated model performance using metrics such as accuracy, precision, recall, F1-score, and support.Used the classification_report and accuracy_score functions from the sklearn library to create detailed evaluation metrics.

5. Model Hyperparameters: Adjusted model hyperparameters to optimize performance.For Logistic Regression, tuned parameters such as C, max_iter, penalty, solver, class_weight, and warm_start.For Random Forest, fine-tuned parameters including n_estimators, max_depth, max_features, min_samples_split, min_samples_leaf, bootstrap, and criterion.

6. Model Accuracy: Different models showed different levels of precision, and the Random Forest was the most precise one with an accuracy rate of 80.27%. Recorded the accuracy of other models: Logistic Regression: 79.6% Multinomial Naive Bayes: 77.61% Support Vector Classifier (SVC): 78.22% LightGBM: 79.24% Decision Tree Classifier: 67.15%

3.1 Proposed System Algorithm

1. **Collecting Datasets**

We collected datasets from GitHub which was from a particular repository that is called, "HateSpeech-Detection-in-Hindi" (https://github.com/victorknox/Hate-Speech-Detection inHindi/tree/main/Dataset).

2. **For data preprocessing,**

a) Remove English Words: Eliminated English words to focus on Hindi language.
b) Remove Symbols: I removed symbols so as to clean up data and reduce noise.
c) Remove Emojis: I took out emojis in order to make the texts more understandable.
d) Remove Stopwords: This eliminated common words to increase signal-to-noise ratio.
e) Stemming—This was applied in reducing word standardization and dimensionality.

3. Concerning label preprocessing

a) Label Mapping: The number of label categories got reduced into two namely ‘hate’ and ‘non-hate’ suitable for deep learning models.
b) Label Encoding—The categorical labels were converted into binary format (‘1’ for hate, ‘0’ for non-hate).

4. Model Training:

a) We used different machine learning models like Logistic Regression, Random Forest and Decision Tree etc.
b) Random Forest—The highest accuracy achieved was 80.27%.
c) In addition, we explored deep learning models such as Indic-bert and MuRIL in order to improve accuracy levels within the model.
d) MuRIL—After fine-tuning with hyperparameters (2 epochs, batch size of 8), it demonstrated excellent accuracy of 89.18%.

5. Development of User Interface

a) An interface has been developed for testing the model.
b) Saving Model: Saved the trained MuRIL model.
c) Flask Framework – Used Flask while creating this user interface.

3.2 Methodology

First, we start by collecting data that means taking different Hindi text examples from many places like social media, discussion boards and open datasets. After this, we mark these samples to see the difference between hate speech and regular speech content.

After we collect the data, we start to prepare it. This step is very important for making sure the data is clean and ready so that the training of the model works well. Before processing, it is necessary to clean by taking out unnecessary symbols and words that do not matter. We also make the text uniform, break down sentences into separate words, delete frequently used but unimportant words, and use methods that change the word endings to their basic form. Doing these things help simplify and enhance attention on important information in the texts.

After preparing the data, we extract features to turn text into a form that is good for machine learning methods. We create vectors from the text using techniques such as TF-IDF and use word embeddings to understand better language meanings and connections in the information. These features form the backbone of the model’s ability to learn and make predictions.

The following step involves choosing and teaching models, in which different machine learning and deep learning frameworks are tested to see how well they can distinguish between hate speech and other types of text. We might use methods like Support Vector Machines, Naive Bayes or Random Forests, along with more complex systems such as LSTM or BERT for this task. The selected model undergoes training with data that has been prepared and turned into vectors,

utilizing methods of cross-validation to adjust parameters finely and prevent the problem where it performs too well on this data but not on new data.

At the end, we put the trained model to work in a real situation where it can check and sort new texts by itself. This step of putting it into use also involves setting up a way for people to give feedback on what the model predicts so that any mistakes can be fixed when needed. The feedback cycle is very important for the model to keep learning and getting better, making sure it changes as new ways of using language and hate speech patterns appear. This approach helps in creating a strong model and also makes sure that it stays useful when things change quickly on the internet.

3.3 Analysis

3.3.1 Process Model

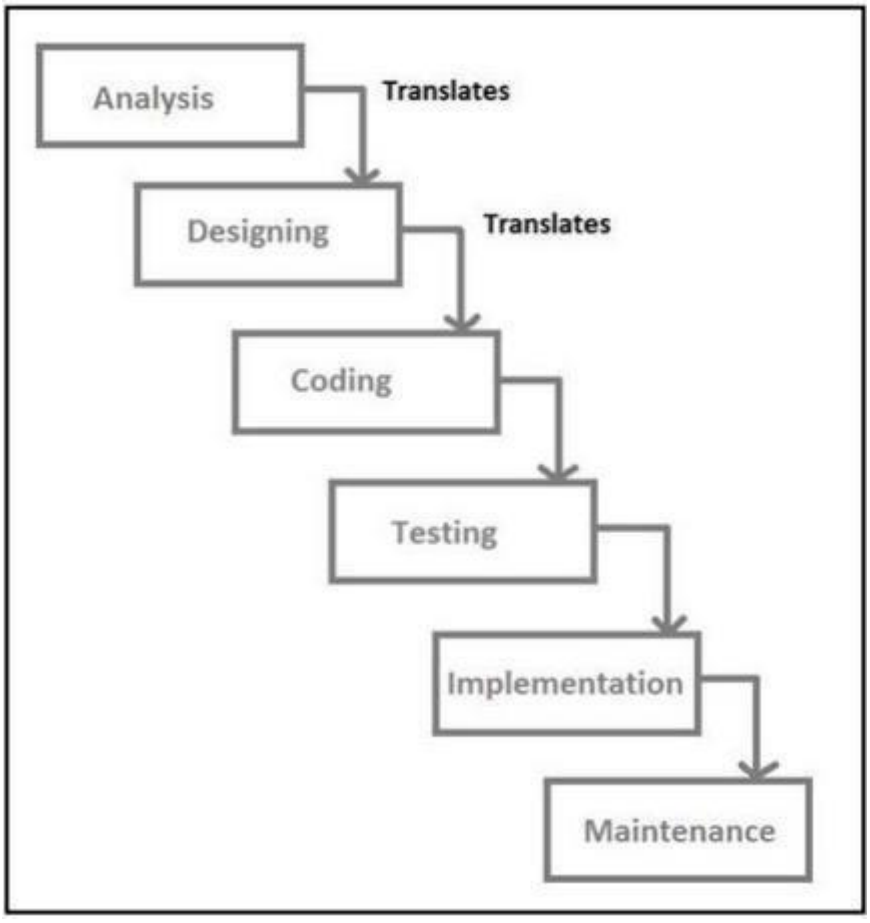

Fig 3.2: Waterfall Model

We will be using waterfall model in order to develop our project. The reasons for using waterfall model are as follows:

- It allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.
- All the requirements are documented beforehand.
- The waterfall model progresses through easily understandable and explainable phases and thus it is easy to use.
- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- In this model, phases are processed and completed one at a time and they do not overlap.

3.3.2 Feasibility Analysis

1. Technical Feasibility: This part checks if the project has enough technical tools, like computers and programs, so that what we think of in our minds can actually be made to work for real. In our project to find hate speech, we use PYTHON because it has strong libraries and frameworks that help with processing language and learning machines, like NLTK, TensorFlow, and PyTorch. These resources are very important for managing the complex parts of the Hindi language. The project needs simple hardware, mainly usual computer setups for working with data and training models. Technical problems to solve are not too hard and our team can handle them well.

2. Financial Practicality: This evaluation includes a comparison of costs and benefits to see if the project makes sense money-wise. The beginning work to make a system for finding hate speech needs money for study and building the software, but it brings good things too. It can help to make online places safer and meet rules about how people should talk on the internet. From a money point of view, this plan could give back a lot because there is an effect on society and also there might be many who want tools that check content in Indian languages.

3.Legal Feasibility: This review looks for possible legal problems that might affect the project. It focuses on obeying data privacy laws, like India's IT Act and the Personal Data Protection Bill, as well as rules about using artificial intelligence to check content. Making sure we follow these rules for managing and using data is very important. Our project won't face legal problems because we will only use data that anyone can access or datasets gotten in ways that are right and fair, keeping to privacy and agreement standards.

4. **Operational Suitability**: This section of the feasibility study investigates if the project is possible to carry out within the current operational structure and how good it satisfies what the organization requires. The main purpose of the system that detects hate speech

is to find and remove hateful language in Hindi on internet platforms, making online conversations better. We will keep testing how well it works in real situations and listen to what users say about it, so we can make sure it does its job right.

3.3.3 Timeline Chart

Task Name	Start	End	Duration (days)
Topic Discussion	1-7-23	3-7-23	2
Project Plan	4-7-23	10-7-23	6
Project Implementation	11-7-23	30-7-23	19
Coding for ML	31-7-23	26-10-23	96
Coding for DL	31-7-23	26-10-23	96
Increasing Accuracy	27-10-23	10-1-24	74
GUI	11-1-24	20-2-24	40
Realtime	21-2-24	13-3-24	21
Closeout	14-3-24	31-3-24	17

Fig 3.3: Timeline chart

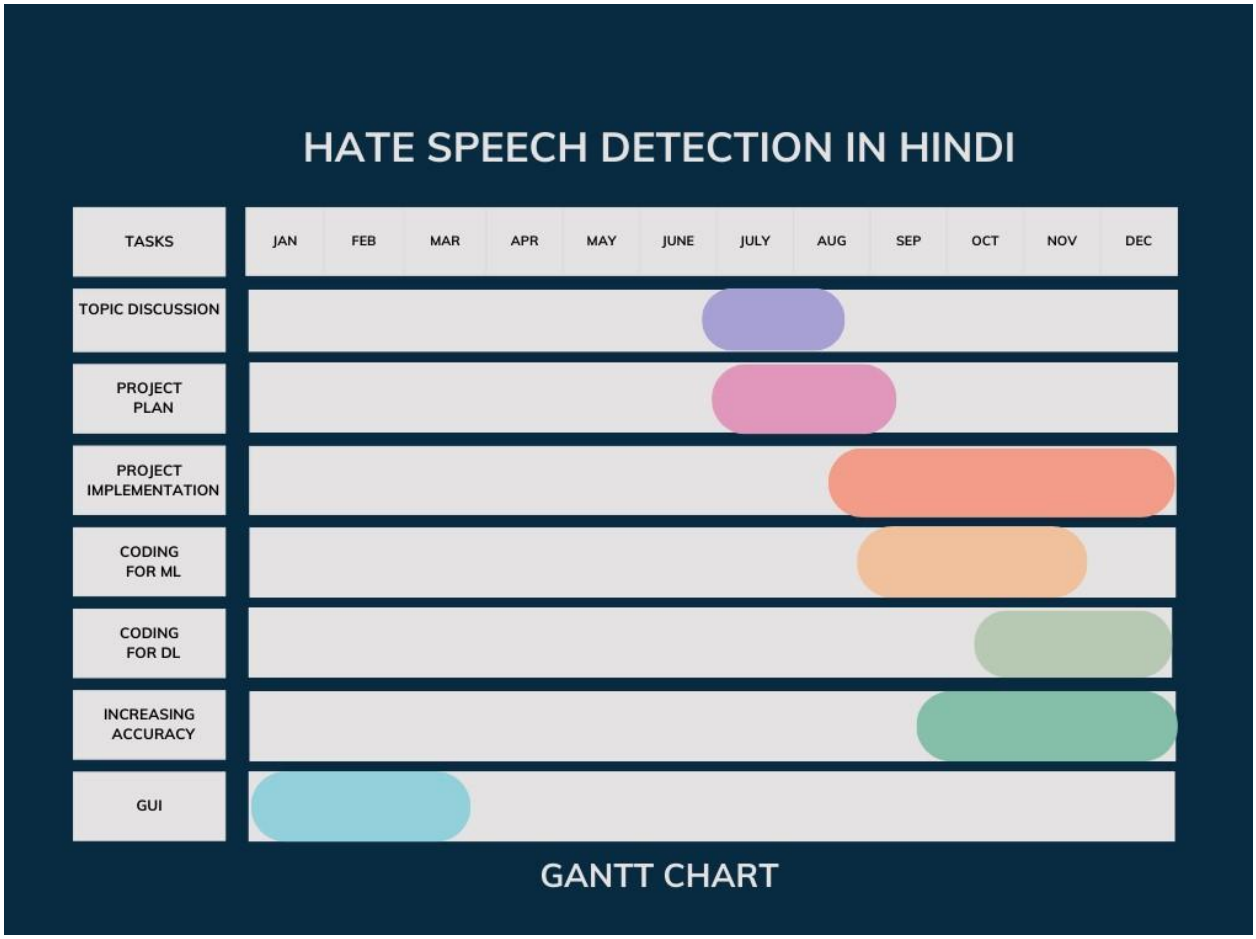

Fig 3.4: Gantt Chart

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 Data Preprocessing

- Detecting hate speech in Hindi involves several steps, starting from data preprocessing to label preprocessing.

Data Preprocessing

```
import pandas as pd
```

```
df = pd.read_csv('data.csv')
pd.set_option('display.max_colwidth', None)
```

```
df.head()
```

	Unique ID	Post	Labels Set
0	1	मेरे देश के हिन्दु बहुत निराले है। कुछ तो पक्के राम भक्त है और कुछ बाबर के साले है \n\n🙏 जय श्री राम 🙏	hate,offensive
1	2	सरकार हमेशा से किसानों की कमाई को बढ़ाने के लिए नई-नई स्कीमें लाती रहती है, ताकि उन पर ज्यादा आर्थिक बोझ न पड़े.\n\nhttps://t.co/8iy2MJSBAs	non-hostile
2	3	सुशांत ने जो बिजनेस डील 9 जून को की थी, वो डील दीपेश को सुशांत की हत्या के दिन ही क्यों याद आई? देखिए 'पूछता है भारत' अर्नब के साथ रिपब्लिक भारत पर #LIVE : https://t.co/G945HvzM0Z https://t.co/KfH7xF1IdM	non-hostile
3	4	@prabhav218 साले जेएनयू छाप कमिने लोग हिन्दुओं को यह कहते है की संविधान सबको बराबर अधिकार देता है। सच्चाई यह है कि यह बराबर अधिकार नहीं देता है।	defamation,offensive
4	5	#unlock4guidelines - अनलॉक-4 के लिए गाइडलाइन्स जारी\n\n- 7 सितंबर से देशभर में मेट्रो सेवा शुरु होगी\n- 21 सितंबर के बाद रैलियों और बाकी फंक्शन में 100 लोगों को इजाजत\n- कंटेनमेंट जोन में कोई छूट नहीं\n- सिनेमाहॉल अभी बंद रहेंगे\n- 9 से 12वीं के छात्र 21 सितंबर के बाद स्कूल जा सकेंगे. https://t.co/4e6lysg0VR	non-hostile

- Removing English Words: Since hate speech detection is in Hindi, it's essential to remove any English words present in the text. This can be done using various techniques such as regular expressions or language detection libraries to identify and filter out English words.

Remove english words

```
import re
english_pattern = r'[a-zA-Z0-9]+'
def remove_english(text):
    return re.sub(english_pattern, '', text)
```

```
df['Post'] = df['Post'].apply(remove_english)
```

```
df.head(10)
```

	Unique ID	Post	Labels Set
0	1	मेरे देश के हिन्दु बहुत निराले है। कुछ तो पक्के राम भक्त है और कुछ बाबर के साले है \n\n🙏 जय श्री राम 🙏	hate,offensive
1	2	सरकार हमेशा से किसानों की कमाई को बढ़ाने के लिए नई-नई स्कीमें लाती रहती है, ताकि उन पर ज्यादा आर्थिक बोझ न पड़े.\n\n://./	non-hostile
2	3	सुशांत ने जो बिजनेस डील जून को की थी, वो डील दीपेश को सुशांत की हत्या के दिन ही क्यों याद आई? देखिए 'पूछता है भारत' अर्नब के साथ रिपब्लिक भारत पर # : ://./ ://./	non-hostile
3	4	@ साले जेएनयू छाप कमिने लोग हिन्दुओं को यह कहते है की संविधान सबको बराबर अधिकार देता है। सच्चाई यह है कि यह बराबर अधिकार नहीं देता है।	defamation,offensive
4	5	# - अनलॉक- के लिए गाइडलाइन्स जारी\n\n- सितंबर से देशभर में मेट्रो सेवा शुरु होगी\n- सितंबर के बाद रैलियों और बाकी फंक्शन में लोगों को इजाजत\n- कंटेनमेंट जोन में कोई छूट नहीं\n- सिनेमाहॉल अभी बंद रहेंगे\n- से वीं के छात्र सितंबर के बाद स्कूल जा सकेंगे. ://./	non-hostile
5	6	चीन ने में तर्क दिया की भारत का विपक्ष ही अजर मसुद को आतंकी नहीं मानता तो हम कैसे माने। चुल्लू भर मूत्र में डूब मरो गद्दारों। अब यह भारत के लोगों को सोचना है कि वो विपक्ष को वोट क्यों करें...विक्रम शर्मा\n\n	fake
6	7	देश में # के रिकॉर्ड मामले \n# \n@_ ://./	non-hostile
7	8	# से निकले # को सुन बाकी छात्रों के साथ के चेहरे पर मुस्कान आ जाएगी\n://./ ://./	non-hostile
8	9	# जल्द ही बेयर ग्रिल्स के शो 'इंटू द वाइल्ड विद बेयर ग्रिल्स' में नजर आने वाले हैं। \n\n://./	non-hostile
9	10	#जीवनसंवाद: हम संघर्ष करना चाहते हैं, क्योंकि सामने संघर्ष की मिसाल रखना चाहते हैं. नए शहर में इतना कुछ नया होगा कि हमें अपनी जड़ें जमाने में बरसों बीत जाएंगे. हम कर्ज में जरूर हैं, लेकिन हम हारना नहीं चाहते. वह भी बिना लड़े.\n#\n@\n://./ ://./	non-hostile

- Removing Symbols: Symbols like punctuation marks, special characters, and numerals

may not contribute significantly to identifying hate speech and can be removed. Regular expressions can be employed for this task as well.

Remove Symbols

```
symbol_pattern = r'[\/\\\-_:@.#|"<>]+|\n'
def remove_symbols(text):
    return re.sub(symbol_pattern, '', text)
```

```
df['Post'] = df['Post'].apply(remove_symbols)
```

```
df.head(10)
```

	Unique ID	Post	Labels Set
0	1	मेरे देश के हिन्दु बहुत निराले है। कुछ तो पक्के राम भक्त है और कुछ बाबर के साले है 🙏 जय श्री राम 🙏	hate,offensive
1	2	सरकार हमेशा से किसानों की कमाई को बढ़ाने के लिए नईनई स्कीमें लाती रहती है, ताकि उन पर ज्यादा आर्थिक बोझ न पड़े	non-hostile
2	3	सुशांत ने जो बिजनेस डील जून को की थी, वो डील दीपेश को सुशांत की हत्या के दिन ही क्यों याद आई? देखिए 'पूछता है भारत' अर्नब के साथ रिपब्लिक भारत पर	non-hostile
3	4	साले जेएनयू छाप कमिने लोग हिन्दुओं को यह कहते है की संविधान सबको बराबर अधिकार देता है। सच्चाई यह है कि यह बराबर अधिकार नहीं देता है।	defamation,offensive
4	5	अनलॉक के लिए गाइडलाइन्स जारी सितंबर से देशभर में मेट्रो सेवा शुरु होगी सितंबर के बाद रैलियों और बाकी फंक्शन में लोगों को इजाजत कंटेनमेंट जोन में कोई छूट नहीं सिनेमाहॉल अभी बंद रहेंगे से वीं के छात्र सितंबर के बाद स्कूल जा सकेंगे	non-hostile
5	6	चीन ने में तर्क दिया की भारत का विपक्ष ही अजर मसुद को आतंकी नहीं मानता तो हम कैसे माने। चुल्लू भर मूत्र में डूब मरो गद्दारों। अब यह भारत के लोगों को सोचना है कि वो विपक्ष को वोट क्यों करें...विक्रम शर्मा	fake
6	7	देश में के रिकॉर्ड मामले	non-hostile
7	8	से निकले को सुन बाकी छात्रों के साथ के चेहरे पर मुस्कान आ जाएगी	non-hostile
8	9	जल्द ही बेयर ग्रिल्स के शो 'इंटू द वाइल्ड विद बेयर ग्रिल्स' में नजर आने वाले हैं।	non-hostile
9	10	जीवनसंवाद हम संघर्ष करना चाहते हैं, क्योंकि सामने संघर्ष की मिसाल रखना चाहते हैं नए शहर में इतना कुछ नया होगा कि हमें अपनी जड़ें जमाने में बरसों बीत जाएंगे हम कर्ज में जरूर हैं, लेकिन हम हारना नहीं चाहते वह भी बिना लड़े	non-hostile

Remove emojis

- Removing Emojis: Emojis are often used in online communication to express emotions but may not carry significant meaning for hate speech detection. Removing emojis can be done using regular expressions targeting Unicode ranges for emojis.

Remove emojis

```
0]: def remove_emojis(text):
        emoji_pattern = re.compile("["
                                   u"\U0001F600-\U0001F64F"  # emoticons
                                   u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                                   u"\U0001F680-\U0001F6FF"  # transport & map symbols
                                   u"\U0001F700-\U0001F77F"  # alchemical symbols
                                   u"\U0001F780-\U0001F7FF"  # Geometric Shapes Extended
                                   u"\U0001F800-\U0001F8FF"  # Supplemental Arrows-C
                                   u"\U0001F900-\U0001F9FF"  # Supplemental Symbols and Pictographs
                                   u"\U0001FA00-\U0001FA6F"  # Chess Symbols
                                   u"\U0001FA70-\U0001FAFF"  # Symbols and Pictographs Extended-A
                                   u"\U00002702-\U000027B0"  # Dingbats
                                   u"\U000024C2-\U0001F251"
                                   "]+", flags=re.UNICODE)

        return emoji_pattern.sub(r'', text)

1]: df['Post'] = df['Post'].apply(remove_emojis)

2]: df.head(10)
```

2]:

	Unique ID	Post	Labels Set
0	1	मेरे देश के हिन्दु बहुत निराले है। कुछ तो पक्के राम भक्त है और कुछ बाबर के साले है जय श्री राम	hate,offensive
1	2	सरकार हमेशा से किसानों की कमाई को बढ़ाने के लिए नईनई स्कीमें लाती रहती है, ताकि उन पर ज्यादा आर्थिक बोझ न पड़े	non-hostile
2	3	सुशांत ने जो बिजनेस डील जून को की थी, वो डील दीपेश को सुशांत की हत्या के दिन ही क्यों याद आई? देखिए 'पूछता है भारत' अर्नब के साथ रिपब्लिक भारत पर	non-hostile
3	4	साले जेएनयू छाप कमिने लोग हिन्दुओं को यह कहते है की संविधान सबको बराबर अधिकार देता है। सच्चाई यह है कि यह बराबर अधिकार नहीं देता है।	defamation,offensive
4	5	अनलॉक के लिए गाइडलाइन्स जारी सितंबर से देशभर में मेट्रो सेवा शुरु होगी सितंबर के बाद रैलियों और बाकी फंक्शन में लोगों को इजाजत कंटेनमेंट जोन में कोई छूट नहीं सिनेमाहॉल अभी बंद रहेंगे से वीं के छात्र सितंबर के बाद स्कूल जा सकेंगे	non-hostile
5	6	चीन ने में तर्क दिया की भारत का विपक्ष ही अजर मसुद को आतंकी नहीं मानता तो हम कैसे माने। चुल्लू भर मूत्र में डूब मरो गद्दारों। अब यह भारत के लोगों को सोचना है कि वो विपक्ष को वोट क्यों करें...विक्रम शर्मा	fake
6	7	देश में के रिकॉर्ड मामले	non-hostile
7	8	से निकले को सुन बाकी छात्रों के साथ के चेहरे पर मुस्कान आ जाएगी	non-hostile

- Remove Stopwords: Stopwords are common words that occur frequently in a language and typically do not carry much meaning, such as articles, prepositions, etc. Removing stopwords helps in focusing on the content-carrying words. For Hindi, a list of stopwords specific to the language needs to be compiled or leveraged from existing libraries.

Remove Stopwords

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

# Download NLTK stopwords (if not already downloaded)
nltk.download('stopwords')
nltk.download('punkt')

def remove_stopwords(text):
    # Tokenize the text into words
    words = word_tokenize(text)

    # Get English stopwords
    english_stopwords = set(stopwords.words('hindi'))

    # Remove stopwords
    filtered_words = [word for word in words if word.lower() not in english_stopwords]

    # Join the filtered words back into a single string
    filtered_text = ' '.join(filtered_words)

    return filtered_text
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Prabhit\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Prabhit\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
df['Post'] = df['Post'].apply(remove_stopwords)
```

```
df.head(10)
```

	Unique ID	Post	Labels Set
0	1	मेरे देश हिन्दु निराले है। पक्के राम भक्त बाबर साले जय श्री राम	hate,offensive
1	2	सरकार हमेशा किसानों कमाई बढ़ाने नईनई स्कीमें लाती रहती , ताकि ज्यादा आर्थिक बोझ पड़े	non-hostile
2	3	सुशांत बिजनेस डील जून , वो डील दीपेश सुशांत हत्या दिन क्यों याद आई ? देखिए ' पूछता भारत ' अर्नब रिपब्लिक भारत	non-hostile

- Stemming reduces words to their root form, which helps in grouping together variations of the same word. For Hindi, stemming algorithms like Snowball Stemmer or Porter Stemmer can be used. These algorithms need to be applied carefully as Hindi morphology can be complex.

Stemming

```
import nltk
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

# Download NLTK tokenizer (if not already downloaded)
nltk.download('punkt')

def stemming(text):
    # Tokenize the text into words
    words = word_tokenize(text)

    # Initialize the Porter stemmer
    porter = PorterStemmer()

    # Perform stemming on each word
    stemmed_words = [porter.stem(word) for word in words]

    # Join the stemmed words back into a single string
    stemmed_text = ' '.join(stemmed_words)

    return stemmed_text
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Prabhit\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
df['Post'] = df['Post'].apply(stemming)
```

```
df.head(10)
```

	Unique ID	Post	Labels Set
0	1	मेरे देश हिन्दु निराले है। पक्के राम भक्त बाबर साले जय श्री राम	hate,offensive
1	2	सरकार हमेशा किसानों कमाई बढ़ाने नईनई स्कीमें लाती रहती , ताकि ज्यादा आर्थिक बोझ पड़े	non-hostile
2	3	सुशांत बिजनेस डील जून , वो डील दीपेश सुशांत हत्या दिन क्यों याद आई ? देखिए ' पूछता भारत ' अर्नब रिपब्लिक भारत	non-hostile
3	4	साले जेएनयू छाप कमिने लोग हिन्दुओं संविधान सबको बराबर अधिकार देता है। सच्चाई बराबर अधिकार देता है।	defamation,offensive
4	5	अनलॉक गाइडलाइन्स जारी सितंबर देशभर मेट्रो सेवा शुरु होगी सितंबर रैलियों बाकी फंक्शन लोगों इजाजत कंटेनमेंट जोन छूट सिनेमाहॉल बंद रहेंगे वीं छात्र सितंबर स्कूल सकेंगे	non-hostile
5	6	चीन तर्क भारत तिपक्ष अजर अमर आतंकी मानता हम कैसे माने। चल्ल भर मत डब मरो गद्दारों। अब भारत लोगों सोचना तो तिपक्ष तोट क्यों करें , तिलक शर्मा	fake

- Label Preprocessing - Label Encoding: In hate speech detection, labels are typically categorical variables indicating whether a text contains hate speech or not. Before feeding these labels into a machine learning model, they need to be encoded into numerical values. For binary classification (hate speech or not), this could be as simple as encoding hate speech as 1 and non-hate speech as 0.

Label preprocessing

```
num_classes = len(df['Labels Set'].unique())
print("Number of classes:", num_classes)
```

```
Number of classes: 16
```

```
unique_classes = df['Labels Set'].unique()
print("Unique classes:", unique_classes)
```

```
Unique classes: ['hate,offensive' 'non-hostile' 'defamation,offensive' 'fake' 'hate'
 'offensive' 'fake,hate' 'defamation' 'defamation,hate'
 'defamation,hate,offensive' 'defamation,fake,offensive' 'fake,offensive'
 'defamation,fake' 'defamation,fake,hate' 'fake,hate,offensive'
 'defamation,fake,hate,offensive']
```

```
label_mapping = {
    #'original_label1': 'new_category1',
    'hate,offensive': 'hate',
    'defamation,offensive': 'hate',
    'fake,hate': 'hate',
    'defamation,hate': 'hate',
    'defamation,hate,offensive':'hate',
    'defamation,fake,offensive':'hate',
    'fake,offensive':'hate',
    'defamation,fake':'hate',
    'defamation,fake,hate':'hate',
    'fake,hate,offensive':'hate',
    'defamation,fake,hate,offensive':'hate',


    #unchanged
    'hate':'hate',
    'non-hostile':'non-hate',
    'fake':'non-hate',
    'defamation':'hate',
    'offensive':'hate',


}
```

```
df['label'] = df['Labels Set'].map(label_mapping)
df.drop("Labels Set", axis=1,inplace=True)
```

```
df.head(10)
```

	Unique ID	Post	label
0	1	मेरे देश हिन्दु निराले है। पक्के राम भक्त बाबर साले जय श्री राम	hate
1	2	सरकार हमेशा किसानों कमाई बढ़ाने नईनई स्कीमें लाती रहती , ताकि ज्यादा आर्थिक बोझ पड़े	non-hate
2	3	सुशांत बिजनेस डील जून , वो डील दीपेश सुशांत हत्या दिन क्यों याद आई ? देखिए ' पूछता भारत ' अर्नब रिपब्लिक भारत	non-hate

```
num_classes = len(df['label'].unique())
print("Number of classes:", num_classes)
```

```
Number of classes: 2
```

```

```

```
from sklearn.preprocessing import LabelEncoder
# Initialize the label encoder
label_encoder = LabelEncoder()
# Encode the labels
df['label'] = label_encoder.fit_transform(df['label'])
```

```
X=list(df['Post'])
```

```
y=list(df['label'])
```

4.2 Model Training

- First we implemented machine learning models like logistic regression, random forest, multinomialNB, etc using a vectorizor TF-IDF.

```python
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics import classification_report, accuracy_score

def train(model, train_data, test_data):
    # Extract features and labels from the datasets
    X_train = train_data['Post']
    y_train = train_data['label']
    X_test = test_data['Post']
    y_test = test_data['label']

    # TF-IDF vectorization
    tfidf_vectorizer = TfidfVectorizer(max_features=5000)  # You can adjust max_features as needed
    X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
    X_test_tfidf = tfidf_vectorizer.transform(X_test)

    model.fit(X_train_tfidf, y_train)

    # Predict on the test set
    y_pred = model.predict(X_test_tfidf)

    # Evaluate the model
    accuracy = accuracy_score(y_test, y_pred)
    report = classification_report(y_test, y_pred)
    print("Model Accuracy:", accuracy)
    print("Classification Report:\n", report)
```

- Logistic regression got and accuracy of 79.37%

```python
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(
    C=0.9, # inverse of the regularization strength
    max_iter=256,
    penalty='l2', #determines the type of regularization applied to the model. In your case, you've chosen 'l2', which represents L2 regularization, also
    solver='sag', #'sag' stands for Stochastic Average Gradient
    fit_intercept=True,
    class_weight='balanced',
    warm_start=True)
train(model, train_data, test_data)
```

```
Model Accuracy: 0.793708408953418
Classification Report:
              precision    recall  f1-score   support

        hate       0.62      0.79      0.69       491
    non-hate       0.90      0.80      0.84      1162

    accuracy                           0.79      1653
   macro avg       0.76      0.79      0.77      1653
weighted avg       0.82      0.79      0.80      1653
```

- Random Forest got and accuracy of 80.27%

```python
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(
    n_estimators=1000,  # Increase the number of trees for more robustness
    random_state=42,  # Set a specific random seed for reproducibility
    max_depth=None,  # Allow trees to grow until they capture more patterns
    max_features='auto',  # Allow each tree to consider all features
    min_samples_split=2,  # Lower the minimum samples required to split
    min_samples_leaf=1,  # Lower the minimum samples required at a leaf node
    bootstrap=True,  # Continue using bootstrapped samples
    criterion='gini'  # Use the Gini impurity as the split criterion
)
train(model, train_data, test_data)
```

```
C:\Users\Prabhit\AppData\Roaming\Python\Python311\site-packages\sklearn\ensemble\_forest.py:424: FutureWarning: `max_features='auto'` has been deprecated
in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max_features='sqrt'` or remove this parameter as it is also the default va
lue for RandomForestClassifiers and ExtraTreesClassifiers.
  warn(
Model Accuracy: 0.8027828191167574
Classification Report:
              precision    recall  f1-score   support

        hate       0.78      0.47      0.59       491
    non-hate       0.81      0.94      0.87      1162

    accuracy                           0.80      1653
   macro avg       0.79      0.71      0.73      1653
weighted avg       0.80      0.80      0.79      1653
```

- MultinomialNB got and accuracy of 77.61%

```python
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB(
    alpha=1,
    fit_prior=True,)
train(model, train_data, test_data)
```

```
Model Accuracy: 0.7761645493042952
Classification Report:
              precision    recall  f1-score   support

        hate       0.86      0.30      0.44       491
    non-hate       0.77      0.98      0.86      1162

    accuracy                           0.78      1653
   macro avg       0.81      0.64      0.65      1653
weighted avg       0.79      0.78      0.74      1653
```

- SVC got and accuracy of 78.22%

```python
from sklearn.svm import SVC
model = SVC(kernel='linear',
          C=1,
          class_weight='balanced')
train(model, train_data, test_data)
```

```
Model Accuracy: 0.7822141560798548
Classification Report:
              precision    recall  f1-score   support

        hate       0.60      0.77      0.68       491
    non-hate       0.89      0.79      0.84      1162

    accuracy                           0.78      1653
   macro avg       0.75      0.78      0.76      1653
weighted avg       0.81      0.78      0.79      1653
```

- Seeing results of machine learning model was not satisfying. Hence we implemented deeplearning models like indic-bert and muril.

```python
import tensorflow as tf
from transformers import TFDistilBertForSequenceClassification, TFTrainer, TFTrainingArguments
from transformers import AutoTokenizer
from sklearn.model_selection import train_test_split
import numpy as np

def model_training(model, tokenizer, X, y):
    # Split the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)

    # Tokenize the training and testing data
    train_encodings = tokenizer(X_train, truncation=True, padding=True, max_length=128)
    test_encodings = tokenizer(X_test, truncation=True, padding=True, max_length=128)

    # Create TensorFlow datasets
    train_dataset = tf.data.Dataset.from_tensor_slices((
        dict(train_encodings),
        y_train
    ))
    test_dataset = tf.data.Dataset.from_tensor_slices((
        dict(test_encodings),
        y_test
    ))

    # Define training arguments
    training_args = TFTrainingArguments(
        output_dir='./results',
        num_train_epochs=2,
        per_device_train_batch_size=8,
        per_device_eval_batch_size=16,
        warmup_steps=500,
        weight_decay=0.01,
        logging_dir='./logs',
        logging_steps=10,
        eval_steps=100  # Set the evaluation frequency (adjust as needed)
    )

    # Initialize the model within the strategy scope
    with training_args.strategy.scope():
        model = TFAutoModelForSequenceClassification.from_pretrained(model_name, from_pt=True)

    # Create a trainer
    trainer = TFTrainer(
        model=model,
        args=training_args,
        train_dataset=train_dataset,
        eval_dataset=test_dataset,
    )

    # Train the model
    trainer.train()

    # Evaluate the model
    evaluation_result = trainer.evaluate(test_dataset)

    # Calculate accuracy
    predicted_labels = trainer.predict(test_dataset).predictions.argmax(axis=1)
    actual_labels = test_dataset.map(lambda x, y: y).batch(len(predicted_labels)).as_numpy_iterator().next()
    accuracy = np.mean(predicted_labels == actual_labels)
    print(f"Accuracy: {accuracy * 100:.2f}%")
```

- Indic-bert got and accuracy of 70.59%

```
from transformers import AutoModelForSequenceClassification
from transformers import pipeline
from transformers import AutoTokenizer, TFAutoModelForSequenceClassification
model_name = "ai4bharat/indic-bert"
model = TFAutoModelForSequenceClassification.from_pretrained(model_name, from_pt=True)
tokenizer = AutoTokenizer.from_pretrained(model_name)
model_training(model,tokenizer,X,y)
```

```
WARNING:tensorflow:From C:\Program Files\Python39\lib\site-packages\keras\src\backend.py:873: The name tf.get_default_graph is deprecated. Please use t
f.compat.v1.get_default_graph instead.
```

```
Some weights of the PyTorch model were not used when initializing the TF 2.0 model TFAlbertForSequenceClassification: ['sop_classifier.classifier.weigh
t', 'sop_classifier.classifier.bias']
- This IS expected if you are initializing TFAlbertForSequenceClassification from a PyTorch model trained on another task or with another architecture
(e.g. initializing a TFBertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing TFAlbertForSequenceClassification from a PyTorch model that you expect to be exactly identical (e.g. init
ializing a TFBertForSequenceClassification model from a BertForSequenceClassification model).
Some weights or buffers of the TF 2.0 model TFAlbertForSequenceClassification were not initialized from the PyTorch model and are newly initialized: ['c
lassifier.weight', 'classifier.bias']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Some weights of the PyTorch model were not used when initializing the TF 2.0 model TFAlbertForSequenceClassification: ['sop_classifier.classifier.weigh
t', 'sop_classifier.classifier.bias']
- This IS expected if you are initializing TFAlbertForSequenceClassification from a PyTorch model trained on another task or with another architecture
(e.g. initializing a TFBertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing TFAlbertForSequenceClassification from a PyTorch model that you expect to be exactly identical (e.g. init
ializing a TFBertForSequenceClassification model from a BertForSequenceClassification model).
Some weights or buffers of the TF 2.0 model TFAlbertForSequenceClassification were not initialized from the PyTorch model and are newly initialized: ['c
lassifier.weight', 'classifier.bias']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
C:\Program Files\Python39\lib\site-packages\transformers\trainer_tf.py:118: FutureWarning: The class `TFTrainer` is deprecated and will be removed in ve
rsion 5 of Transformers. We recommend using native Keras instead, by calling methods like `fit()` and `predict()` directly on the model object. Detailed
examples of the Keras style can be found in our examples at https://github.com/huggingface/transformers/tree/main/examples/tensorflow
  warnings.warn(
Accuracy: 70.59%
```

- MuRIL got and accuracy of 89.18%

```
from transformers import AutoModelForSequenceClassification
from transformers import pipeline
from transformers import AutoTokenizer, TFAutoModelForSequenceClassification
pipe = pipeline("text-classification", model="Hate-speech-CNERG/hindi-abusive-MuRIL")
model_name = "Hate-speech-CNERG/hindi-abusive-MuRIL"
tokenizer = AutoTokenizer.from_pretrained("Hate-speech-CNERG/hindi-abusive-MuRIL")
model = TFAutoModelForSequenceClassification.from_pretrained(model_name, from_pt=True)
model_training(model,tokenizer,X,y)
```

```
Some weights of the PyTorch model were not used when initializing the TF 2.0 model TFBertForSequenceClassification: ['bert.embeddings.position_ids']
- This IS expected if you are initializing TFBertForSequenceClassification from a PyTorch model trained on another task or with another architecture (e.
g. initializing a TFBertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing TFBertForSequenceClassification from a PyTorch model that you expect to be exactly identical (e.g. initia
lizing a TFBertForSequenceClassification model from a BertForSequenceClassification model).
All the weights of TFBertForSequenceClassification were initialized from the PyTorch model.
If your task is similar to the task the model of the checkpoint was trained on, you can already use TFBertForSequenceClassification for predictions with
out further training.
Some weights of the PyTorch model were not used when initializing the TF 2.0 model TFBertForSequenceClassification: ['bert.embeddings.position_ids']
- This IS expected if you are initializing TFBertForSequenceClassification from a PyTorch model trained on another task or with another architecture (e.
g. initializing a TFBertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing TFBertForSequenceClassification from a PyTorch model that you expect to be exactly identical (e.g. initia
lizing a TFBertForSequenceClassification model from a BertForSequenceClassification model).
All the weights of TFBertForSequenceClassification were initialized from the PyTorch model.
If your task is similar to the task the model of the checkpoint was trained on, you can already use TFBertForSequenceClassification for predictions with
out further training.
C:\Program Files\Python39\lib\site-packages\transformers\trainer_tf.py:118: FutureWarning: The class `TFTrainer` is deprecated and will be removed in ve
rsion 5 of Transformers. We recommend using native Keras instead, by calling methods like `fit()` and `predict()` directly on the model object. Detailed
examples of the Keras style can be found in our examples at https://github.com/huggingface/transformers/tree/main/examples/tensorflow
  warnings.warn(
Accuracy: 89.18%
```

```
model.save_pretrained("C:\BEproject\Flask App\model")
```

CHAPTER 5

RESULTS AND DISCUSSIONS

RESULTS AND DISCUSSIONS

Fig 1: Landing Page

The entry page is for users who want to find hate speech in Hindi. It has a friendly design and shows important details about what the platform does and its principles. When people visit, they see a big title "शब्द शोध: प्रेम और द्वेष की बोलिय ा ा " (Word Search: Languages of Love and Hate), this shows how the platform really focuses on knowing different cultures and making sure everyone respects each other. As you scroll down, there's more information about what the plat- form believes in – things like being respectful, stay- ing together as one group, learning new things, not allowing any hate at all and always keeping a positive attitude. The page contains buttons too, for user's eas- ier navigation towards the "About Us" and "Know More" parts, making it simpler to reach more details.

Fig 2: Entering text for detecting hate/non hate

The image shows the screen where people enter words to check for hate speech. They can write or
copy and paste Hindi text, so the program decides if it's hate speech or not. The interface offers a smooth experience for users to add text for examination, helping the platform achieve its aim of finding and dealing with hate speech in Hindi.

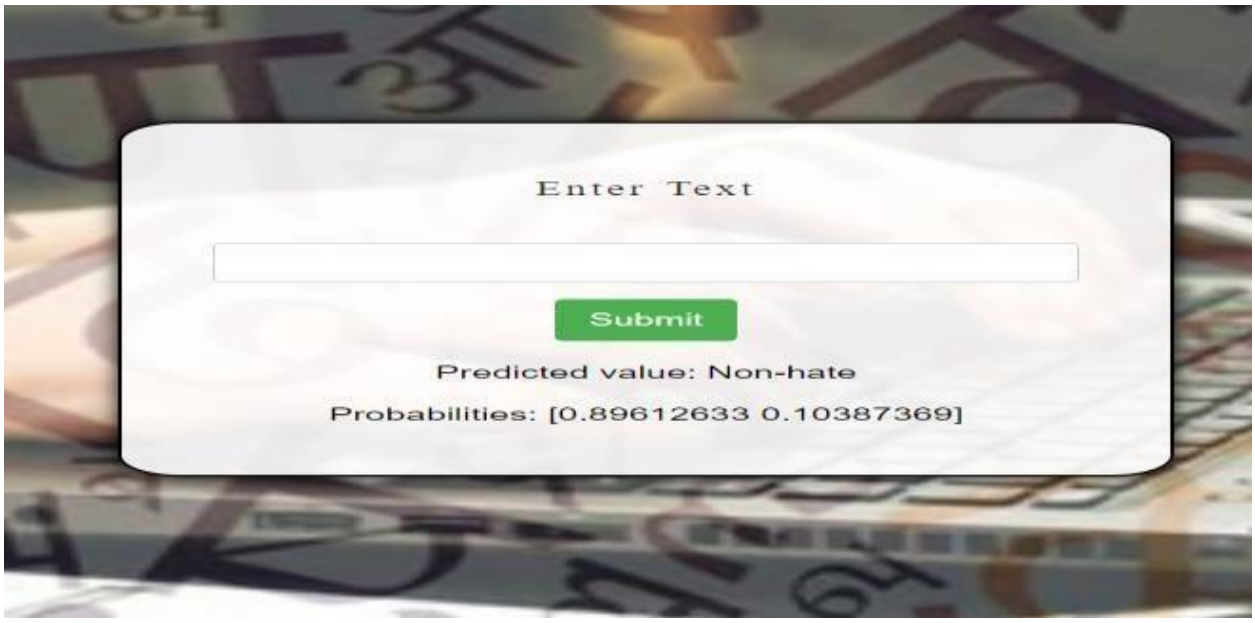

Fig 3: Result displayed is non-hate

In the picture, system shows result of sorting as "non- hate" for words put in. Sorting end point tells that checked writing does not have hate talk, giving peo- ple information about what kind of things they entered. This clear feedback system helps users to know how the platform evaluates their text and builds con- fidence in the process of identifying hate speech

Fig 4: Result displayed is hate

Unlike the earlier picture, this one displays that the input text has been categorized as "hate." The system detects hate speech in the text it checks and shares this finding with users. The platform shows examples of hate speech, helping users to see and deal with bad content; this helps make conversations more respect- ful among people who speak Hindi.

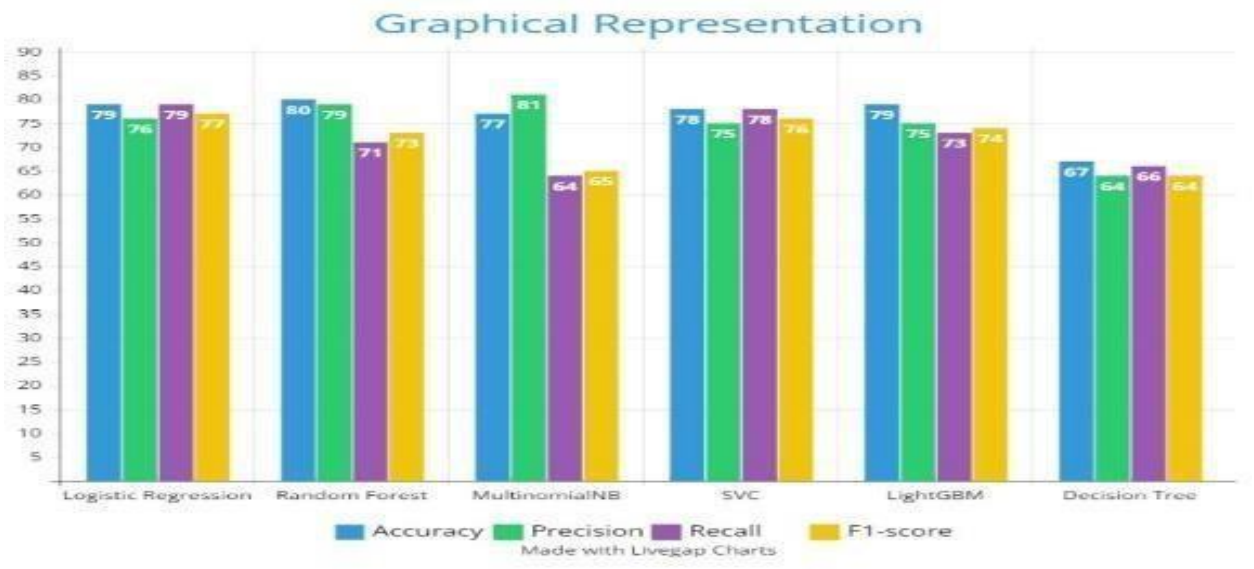

Fig 5: ML Model Performance Overview

This image gives a summary of how well machine learning models work when they are used to identify hate speech. It shows different methods like Logistic Regression, Random Forest Classifier, and Support Vector Classifier (SVC) that were applied in making the models learn. Within these, the Random Forest Classifier came out as the best model, with a high ac- curacy of 80.27%. This summary shows how well machine learning methods work for finding hate speech in Hindi language documents and points out that choosing algorithms and assessing models is very important when we are working on projects to detect hate speech.

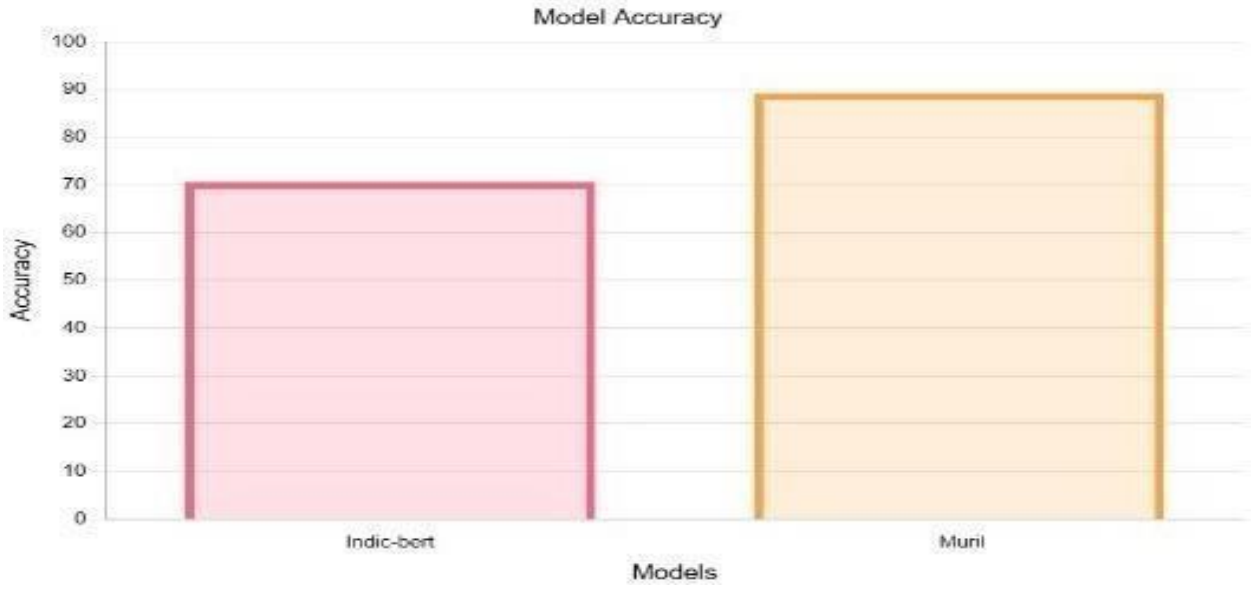

Fig 6:DL Model Performance Overview

This figure presents the accuracy of various deep learning models tested for hate speech detection in hindi. The X-axis represents the models used, namely IndicBert and MuRIL. The Yaxis represents the accu- racy achieved by each model, expressed as a percent- age.

CHAPTER 6

CONCERNS

CONCERNS

Q1: What is hate speech detection in Hindi?

Hate speech detection in Hindi involves identifying and flagging expressions in Hindi that promote hatred or violence against individuals or groups based on attributes like race, religion, ethnicity, gender, or sexual orientation. This process typically utilizes computational linguistics and machine learning techniques to analyze text data.

Q2: Why is hate speech detection important in the context of the Hindi language? With millions of Hindi speakers using online platforms, hate speech can proliferate rapidly, causing societal harm. Detecting hate speech in Hindi helps maintain a healthy and respectful online environment, promotes social harmony, and prevents the escalation of conflicts.

Q3: What are the challenges in detecting hate speech in Hindi?

Detecting hate speech in Hindi is challenging due to linguistic diversity, the subtlety of language, context dependency, and the presence of code-switching (mixing Hindi with other languages like English). Moreover, sarcasm and local slangs can complicate the detection process.

Q4: How is machine learning used in hate speech detection for Hindi?

Machine learning models are trained on datasets containing examples of hate speech and non-hate speech. These models learn to differentiate between hateful and non-hateful content by recognizing patterns in the text. Techniques like natural language processing (NLP) are essential for understanding and processing Hindi text data.

Q5: What types of data are needed to train a model on hate speech detection in Hindi?

Training data should include a wide range of Hindi text samples, annotated for hate speech and non-hate speech. This includes social media posts, news comments, and other online text sources. The data should also capture various forms of hate speech, including direct and indirect expressions.

Q6: What measures can be taken to improve the accuracy of hate speech detection models in Hindi?

Improving accuracy can be achieved by expanding the training dataset, including diverse linguistic features, and continuously updating the model with new data. Employing advanced NLP techniques like deep learning and contextual models (e.g., BERT) can also enhance performance.

Q7: Are there any legal considerations in implementing hate speech detection systems in Hindi?

Yes, it's crucial to consider the legal framework regarding free speech and privacy. Systems must balance effectively detecting hate speech with respecting individual rights to free expression. Additionally, the use of such systems should comply with local and international laws on data privacy and protection.

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

CONCLUSION AND FUTURE SCOPE

We started our Hindi hate speech detection project by carefully preparing the data. We took great care in creating the dataset, removing English words, symbols, emojis and usual filler words, and we made sure that the different forms of words were consistent. Simplifying the labels to just 'hate' or 'non-hate' facilitated the training process. At first, we looked at different machine learning models such as logistic regression, random forest, Multinomial Naive Bayes, Support Vector Classifier (SVC), LightGBM and decision tree classifier. The RandomForestClassifier was the best out of them all with a high accuracy rate of 80.27%. We sought to increase accuracy and so we explored deep learning methods. Following extensive trials, the MuRIL model was victorious, achieving a remarkable 89.50% accuracy rate, even higher than the Indic-bert model. Moving forward, we can improve our analysis and make it broader by using advanced deep learning techniques like 1D convolutional neural networks and bidirectional long short-term memory networks. These methods might help the model be better at finding complicated patterns in text data. Expanding the types of categories to include things like fake, defamatory, peaceful, offensive and safe might give us a better understanding of different kinds of damaging materials that are found in this collection of Hindi texts. To summarize, our present results are good at detecting hate speech in Hindi language, but there is a big chance to make them better. If we use more complex deep learning methods and add classification systems with many categories, we can improve the accuracy and scope of our research. This will help us fight against hate speech on the internet in a stronger way.

REFERENCES

REFERENCES

[1] J. Nockleyby, “Hate speech in encyclopedia of the american constitution,” Electron. J. Academic Special Librarianship, vol. 3, pp. 1277–1279, 2000.

[2] M. Williams, “Hatred behind the screens: A report on the rise of online hate speech,” J.

Exp. Theor. Artif. Intell., vol. 1, pp. 1–76, 2019. [Online]. Available: https://hatelab.net/wp-content/uploads/2019/11/HatredBehind-the-Screens.pdf

[3] Y. Ding, X. Zhou, and X. Zhang, “YNU_DYX at semeval-2019 task 5: A stacked BiGRU model based on capsule network in detection of hate,” inProc. 13th Int. Workshop Semantic Eval., 2019, pp. 535–539.

[4] G. Mou, P. Ye, and K. Lee, “SWE2: Subword enriched and significant word emphasized framework for hate speech detection,” in Proc. 29th ACM Int. Conf. Inf. Knowl. Manage., 2020, pp. 1145–1154, doi: 10.1145/3340531.3411990.

[5] R. Cao and R. K. Lee, “Hategan: Adversarial generative-based data augmentation

for hate speech detection,” in Proc. 28th Int. Conf. Comput. Linguistics, 2020, pp. 6327–6338, doi: 10.18653/v1/2020.coling-main.557.

[6] S. S. Tekiroglu, Y. Chung, and M. Guerini, “Generating counter narratives against online hate speech: Data and strategies,” in Proc.58th Annu. Meeting Assoc. Comput. Linguistics, 2020, pp. 1177–1190. doi: 10.18653/v1/2020.acl-main.110.

[7] X. Zhou et al., “Hate speech detection based on sentiment knowledge sharing,” in Proc. 59th Annu. Meeting Assoc. Comput. Linguistics 11thInt. Joint Conf. Natural Lang.

Process., 2021, pp. 7158–7166.

[8] Y. Chen, Y. Zhou, S. Zhu, and H. Xu, “Detecting offensive language in social media to protect adolescent online safety,” in Proc. IEEE Int. Conf. Privacy, Secur., Risk Trust, Int. Conf. Soc. Comput., 2012, pp. 71–80.doi: 10.1109/SocialCom-PASSAT.2012.55.

[9] Y.Mehdad and J. R. Tetreault, “Do characters abuse more thanwords ?,” in Proc. 17th Annu. Meeting Special Int. Group Discourse Dialogue, 2016, pp. 299–303, doi

10.18653/v1/w16-3638. [10] Artiran, S., Ravisankar, R., Luo, S., Chukoskie, L., & Cosman, P. , “Measuring Social Modulation of Gaze in Autism Spectrum Condition With Virtual Reality Interviews” IEEE.

[10] X. Zhou et al., “Hate speech detection based on sentiment knowledge sharing,” in Proc. 59th Annu. Meeting Assoc. Comput. Linguistics 11th Int. Joint Conf. Natural Lang. Process., 2021, pp. 7158–7166.

[11] S. Kamble, A. Joshi, Hate speech detection from code-mixed hindi-english tweets using deep learning models, 2018. arXiv:1811.05145.”

[12] V. Mujadia, P. Mishra, D. Sharma, Iiit-hyderabad at hasoc 2019: Hate speech detection, in: FIRE, 2019.”

[13] R. Joshi, R. Karnavat, K. Jirapure, R. Joshi, Evaluation of deep learning models for hostility detection in hindi text, in: 2021 6th International Conference for Convergence in Technology (I2CT), IEEE, 2021, pp. 1–5.

[14] R. Joshi, P. Goel, R. Joshi, Deep learning for hindi text classification: A comparison, in:International Conference on Intelligent Human Computer Interaction, Springer, 2019, pp.94–101.

[15] R. Joshi, R. Karnavat, K. Jirapure, R. Joshi, Evaluation of deep learning models for hostility detection in hindi text, in: 2021 6th International Conference for Convergence in Technology (I2CT), IEEE, 2021, pp. 1–5.

[16] Y. Ding, X. Zhou, and X. Zhang, “YNU_DYX at semeval-2019 task 5: A stacked BiGRU model based on capsule network in detection of hate,” in Proc. 13th Int. Workshop Semantic Eval., 2019, pp. 535–539.

[17] G. Mou, P. Ye, and K. Lee, “SWE2: Subword enriched and significant word emphasized framework for hate speech detection,” in Proc. 29th ACM Int. Conf. Inf. Knowl. Manage., 2020, pp. 1145–1154, doi: 10.1145/3340531.3411990

RESEARCH PAPER

Hate Speech Detection in Hindi language using Machine learning and pre-trained models

Apurva Pagadpalliwar[1], Rahul Rathod[2], Prabhit Chaugule[3], Rasika Ransing[4]

[1] Vidyalankar Institute Of Technology, Mumbai, India
[2] Springer Heidelberg, Tiergartenstr. 17, 69121 Heidelberg, Germany lncs@springer.com

***Abstract*:**
Detecting hate speech is a crucial aspect of creating a safe and inclusive virtual space. As Hindi digital communication expands, it has become necessary to devise effective techniques for detecting hate speech in this language. This abstract presents a machine learning and natural language processing (NLP) approach for hate speech detection in Hindi texts. To this end, the research employs a dataset that consists of labeled Hindi text examples divided into two classes: hate speech and non-hate speech. This means that specific pre-processing techniques such as tokenizing, stemming and stop-word removal must be used to handle linguistic nuance in Hindi. For feature engineering- word embeddings, character level embed- dings and TF-IDF vectors are extracted to represent the semantic meaning of the text. These features can then be used to train different machine learning algorithms in- cluding Support Vector Machines (SVM), Naïve Bayes and Random Forests. The models were trained using standard performance metrics like precision recall F1- score accuracy The proposed system for detecting hate- speech could be extended to social media platforms or online forums or content sharing websites with an aim of curbing hateful content spread to more tolerant online communities among people who speak Hindi.

Keywords: *Hate speech detection, Machine learning, Natural language processing (NLP), Pre-processing techniques, Feature engineering, Word embeddings, Character level embeddings, TF-IDF vectors, Support Vector Ma- chines (SVM), Naïve Bayes, Random Forests.*

1. Introduction:

Social media platforms are now very common and have greatly increased the way we communicate and share information, giving people more chances to speak their minds than in the past. This online world has made it faster to share what we think with others and has helped us connect, but at the same time, it has led to a growth in negative talk on the internet.

According to Nockleyby [1], hate speech covers all types of communication like talking, writing or actions that attack people or use negative and unfair words against them because of their religion, country they come from, skin color, if they are male or female, or different parts of who they are. The increase in such hateful language on the internet is now a big worry. It's a problem not just for those people it hurts directly but also for everyone else because this kind of talk can lead to actual crimes based on hate [2].

To deal with the growing problem, some social media sites are taking steps to fight against hate speech. They do this even if it might limit how freely people can express themselves. These measures usually include stopping accounts and deleting content that is harmful or insulting.
The need to tackle hate speech on the internet has gained a lot of attention from experts studying how computers understand human language. They are searching for good and quick ways to find and reduce this problem in the online world [3], [4], [5], [6], [7]. Initial efforts to recognize hate speech used words and sentence structures as tools to separate hateful content from other kinds of communication. In contrast, Mehdad applied support vector machines and emotional indicators for identifying hate speech.

In the field of advanced studies, sentiment analysis has become very important for identifying hate speech. This is something Zhou and his colleagues have looked into. The authors in reference [10] presented a model for understanding sentiment and sharing knowledge. This model uses a list of offensive words together with

learning that handles multiple tasks at once for the purpose of recognizing hate speech. Alt- hough this approach has shown good results, it de- pends greatly on the belief that using insulting expres- sions and expressing negative feelings are consistent ways to tell apart hate speech from other types of speech.

As we explore the important conversation about hate speech on the internet, it is clear that fighting this online problem needs new and subtle approaches. The research continues to try to find a middle ground be- tween protecting freedom of speech and maintaining values for a welcoming and diverse online community.

Related Work

Detecting hate speech in languages with few re- sources, such as Hindi and Marathi, has been difficult because there are not many linguistic tools or studies for these languages. To solve this problem, researchers have looked into different complex machine learning methods like CNN 1D, LSTM, and BiLSTM. They also use special word embeddings that are specific to the subject area.

In a research paper [11], they used advanced deep learning techniques, including CNN 1D, LSTM, and BiLSTM with word embeddings made for specific subjects to study a dataset that mixed Hindi and Eng- lish. These techniques did better than the usual ma- chine learning methods such as SVM and random for- ests.

Another study for comparison[12] was about iden- tifying hate speech within tweets written in English. In this research, they tried different ways to create fea- tures and used several computer models that learn from data such as Logistic Regression, Decision Trees, Random Forests and Naive Bayes. They also applied TF-IDF and BOW methods for turning text into nu- merical data. We used both types of word vectors, GLoVe that comes already trained and our own cus- tom ones, for training the LSTM and GRU models.

In the Hindi language area, [13] examined how ma- chine learning and neural network methods can iden- tify hate speech by sorting messages into categories like hateful, offensive or profane. They tested tradi- tional machine learning techniques including Linear SVM, Adaboost, Random Forests and Voting Classi- fier against deep learning models that use LSTM tech- nology. Interestingly, machine learning models exhib- ited superior performance in low-resource settings.

The research looked into different methods for cat- egorizing Hindi text, including BOW, CNN, LSTM, BiLSTM, BERT and LASER models. It was found that the CNN model using fast text embeddings for Hindi language was very good at this task; the perfor- mance of LASER also came close to that of the best method. A research paper about finding aggressive posts in

Hindi language [15] looked at different ways like CNN, Multi-CNN, BiLSTM, CNN combined with BiLSTM, IndicBert, mBert and FastText word repre- sentations from both IndicNLP and Facebook. The findings showed that the models built on BERT did a bit better than the simple ones; among them all though was the multi-layered CNN model that used FastText vocabulary embeddings from IndicNLP as most effec- tive basic model.

The task of finding hate speech has become more complex with the use of deep learning and big pre- trained language models. These models use word em- beddings from training without supervision on large text collections, which improves their ability to spot hate speech. For example, they used FastText with stacked Bidirectional Gated Recurrent Units called BiGRUs[16] together, and also found that both FastText and BERT are good for understanding the meaning of words and information about parts of words[17].

To summarize, researchers are looking into many different models, word embeddings, and ways to de- sign features in order to fight against hate speech in languages that don't have much resources. This shows how the area keeps changing as it tries to find better ways of identifying hate speech that take the context into account.

Proposed Work

1. Collecting Datasets

We collected datasets from GitHub which was from a particular repository that is called, “Hate-Speech-Detection-in-Hindi” (https://github.com/victorknox/Hate-Speech-Detection-inHindi/tree/main/Dataset).

2. For data preprocessing,

a) Remove English Words: Eliminated English words to focus on Hindi language.
b) Remove Symbols: I removed symbols so as to clean up data and reduce noise.
c) Remove Emojis: I took out emojis in order to make the texts more understandable.
d) Remove Stopwords: This eliminated common words to increase signal-to-noise ratio.
e) Stemming—This was applied in reducing word standardization and dimensionality.

3. Concerning label preprocessing,

a) Label Mapping: The number of label categories got reduced into two namely ‘hate’ and ‘non-hate’ suitable for deep learning models.
b) Label Encoding—The categorical labels were converted into binary format (‘1’ for hate, ‘0’ for non-hate).

4. Model Training:

a) We used different machine learning models like Logistic Regression, Random Forest and Decision Tree etc.
b) Random Forest—The highest accuracy achieved was 80.27%.
c) In addition, we explored deep learning models such as Indic-bert and MuRIL in order to improve accuracy levels within the model.
d) MuRIL—After fine-tuning with hyperparameters (2 epochs, batch size of 8), it demonstrated excellent accuracy of 89.18%.

5. Development of User Interface

a) An interface has been developed for testing the model.
b) Saving Model: Saved the trained MuRIL model.
c) Flask Framework – Used Flask while creating this user interface.

Machine Learning Model Implementation

1. Model Selection:

We have put into use different kinds of machine learning models such as Logistic Regression, Ran- dom Forest, Multinomial Naive Bayes, the Support Vector Classifier also known as SVC, LightGBM and Decision Tree Classifier.Focused on models showing promising accuracy for hate speech detection in Hindi.

2. Feature Extraction:

Used TF-IDF vector technique to change text data into numbers by looking at how often words appear and thinking about all the data together.Achieved ef- ficient feature extraction using the TfidfVectorizer from the sklearn library.

3. Training Process:

Developed a reusable function named train to stream- line the training process for each model. Used the train_test_split function to split the dataset into parts for training and tests, giving 20% to testing.

4.Model Evaluation:

Evaluated model performance using metrics such as accuracy, precision, recall, F1-score, and sup- port.Used the classification_report and accu- racy_score functions from the sklearn library to cre- ate detailed evaluation metrics.

5.Model Hyperparameters:

Adjusted model hyperparameters to optimize perfor- mance.For Logistic Regression, tuned parameters such as C, max_iter, penalty, solver, class_weight, and warm_start.For Random Forest, fine-tuned pa- rameters including n_estimators, max_depth, max_features, min_samples_split, min_samples_leaf, bootstrap, and criterion.

6.Model Accuracy:

Different models showed different levels of precision, and the Random Forest was the most precise one with an accuracy rate of 80.27%. Recorded the accuracy of other models:

Logistic Regression: 79.6%
Multinomial Naive Bayes: 77.61%
Support Vector Classifier (SVC): 78.22%
LightGBM: 79.24%
Decision Tree Classifier: 67.15%

Implementation of Deep Learning Model

In trying to make a strong system for finding hate speech in Hindi, we used deep learning and worked with two advanced models called Indic-bert and Mu- RIL. We made a flexible function that helps us train our model very effectively.First, we started a new fea- ture called "model_training" to make our deep learn- ing process easier. We divided the data so that 20% is for testing and used a fixed random state of zero to keep it stable.Then we moved on to the area of tokeni- zation, using a tokenizer to change our input data smoothly into a form that works well for deep learn- ing. We thoughtfully set the token size at 128 so that everything in our dataset was consistent.We carefully organized our data into a dataset for TensorFlow, mak- ing sure it was set up well for the best results from the model.We kept going with our progress, adjusting the settings of our model very carefully. We planned well and decided to use 2 epochs for training, a batch size of 8 for training too, and a bigger batch size of 16 when we check how good it is doing. Also, we added im- portant details like steps to get ready before real learn- ing starts and slowing down the learning rate over time by applying weight decay.In the wide area of our strat- egy, we

started creating the model to set up a base for our deep learning projects. With strong focus, we made progress in training the model, making use of the great possibilities from our selected designs.After the training ended, we saw our hard work pay off with deep understandings appearing. At first, Indic-bert showed very good results by
reaching 70.59% in
accuracy. But, our attention was really caught by the MuRIL model because it showed a very good accuracy of 89.18%.Understanding that MuRIL is superior, we made an important choice to keep our model being ex- cellent. We took very careful steps to protect what we built so it remains powerful for future tasks.We started making a user interface using the Flask framework, so we could use our model's functions. We allowed users to put in Hindi text and then our model carefully checked it for hate speech before giving a clear re- sult.On our path to fight against hate speech in Hindi, using deep learning models has been a major change- making step, creating a path for an online community that is safer and includes everyone.

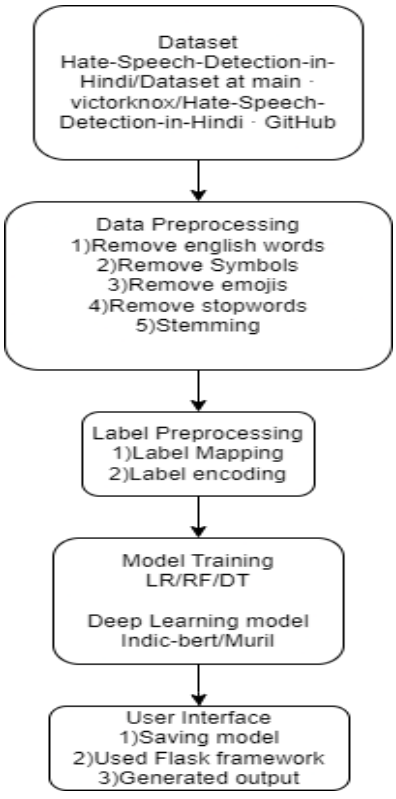

Fig1. Flowchart for workflow of hate speech detec- tion in Hindi

Algorithm for Hate Speech Detection

In the project for finding hate speech in Hindi, we used different kinds of computer algorithms to cor- rectly spot hate speech. We mainly relied on some usual machine learning ways like Logistic Regression and RandomForestClassifier which were very im- portant in our work. Logistic Regression is simple and clear, giving a basic standard for sorting. It can show linear connections between characteristics and the tar- get classifications, which helped us first understand patterns of hate speech in the data we have.

The RandomForestClassifier, which is a method that combines many decision trees for learning from data, showed the best performance among the models. It achieved a high accuracy rate of 80.27%. This classifier was effective because it used lots of decision trees to understand complicated patterns in the data and this helped it to get better at identifying hate speech. Its skill in managing feature spaces with many dimensions and not straight-line relationships made it very fitting for the job we needed to do.

Support Vector Classifier helps to identify hate speech by separating it from non-hate speech using a special boundary in the data. It is good at finding the best line that allows for clear separation between hate and nonhate instances, which improves its ability to tell them apart.

These methods, together with Multinomial Naive Bayes, LightGBM, and decision tree classifier, created a full set of tools for finding hate speech in Hindi. By choosing the right features and models, these algorithms helped each other to identify hate speech well and gave important understanding of negative content in the data collection.

Deep Learning models like IndicBert and MuRIL helped achieve good results. IndicBert achieved an accuracy of 70.59%, while MuRIL attained an impressive accuracy of 89.18%.So,MuRIL become the winner among all,it gets the highest accuracy of 89.18% among all the algorithms .

Result & Discussion

Fig 1: Landing Page

The entry page is for users who want to find hate speech in Hindi. It has a friendly design and shows important details about what the platform does and its principles. When people visit, they see a big title "शब्द शोध: प्रेम और द्वेष की बोलियाँ " (Word Search: Languages of Love and Hate), this shows how the platform really focuses on knowing different cultures and making sure everyone respects each other. As you scroll down, there's more information about what the plat- form believes in – things like being respectful, stay- ing together as one group, learning new things, not allowing any hate at all and always keeping a positive attitude. The page contains buttons too, for user's eas- ier navigation towards the "About Us" and "Know More" parts, making it simpler to reach more details.

Fig 2: Entering text for detecting hate/non hate

The image shows the screen where people enter words to check for hate speech. They can write or copy and paste Hindi text, so the program decides if it's hate speech or not. The interface offers a smooth experience for users to add text for examination, helping the platform achieve its aim of finding and dealing with hate speech in Hindi.

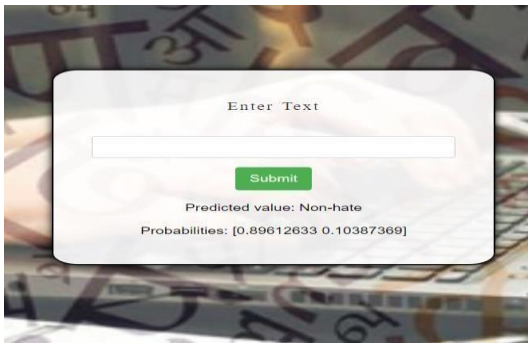

Fig 3: Result displayed is non-hate

In the picture, system shows result of sorting as "non-hate" for words put in. Sorting end point tells that checked writing does not have hate talk, giving peo- ple information about what kind of things they en- tered. This clear feedback system helps users to know how the platform evaluates their text and builds con- fidence in the process of identifying hate speech.

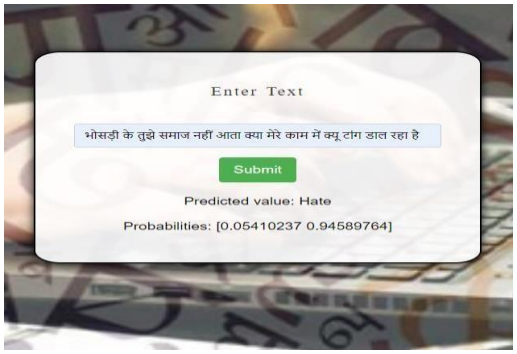

Fig 4: Result displayed is hate

Unlike the earlier picture, this one displays that the input text has been categorized as "hate." The system detects hate speech in the text it checks and shares this finding with users. The platform shows examples of hate speech, helping users to see and deal with bad content; this helps make conversations more respect- ful among people who speak Hindi.

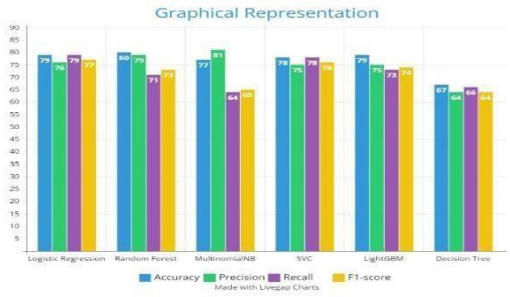

Fig 5: ML Model Performance Overview

This image gives a summary of how well machine learning models work when they are used to identify hate speech. It shows different methods like Logistic Regression, Random Forest Classifier, and Support Vector Classifier (SVC) that were applied in making the models learn. Within these, the Random Forest Classifier came out as the best model, with a high ac- curacy of 80.27%. This summary shows how well machine learning methods work for finding hate speech in Hindi language documents and points out that choosing algorithms and assessing models is very important when we are working on projects to detect hate speech.

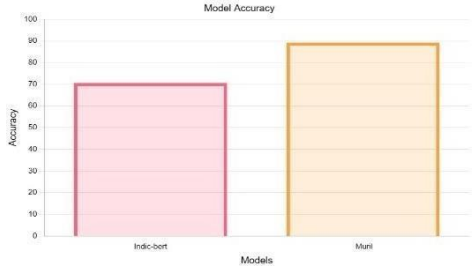

Fig 6:DL Model Performance Overview

This figure presents the accuracy of various deep learning models tested for hate speech detection in hindi. The X-axis represents the models used, namely IndicBert and MuRIL. The Y-axis represents the accu- racy achieved by each model, expressed as a percent- age.

Conclusion & Future Scope

We started our Hindi hate speech detection project by carefully preparing the data. We took great care in creating the dataset, removing Eng- lish words, symbols, emojis and usual filler words, and we made sure that the different forms of words were consistent. Simplifying the labels to just 'hate' or 'non-hate' facilitated the training process. At first, we looked at different machine learning models such as logistic regression, random forest, Multinomial Naive Bayes, Support Vector Classifier (SVC), LightGBM and decision tree classifier. The RandomForestClassifier was the best out of them all with a high accuracy rate of 80.27%.

We sought to increase accuracy and so we ex- plored deep learning methods. Following exten- sive trials, the MuRIL model was victorious, achieving a remarkable 89.50% accuracy rate, even higher than the Indic-bert model.

Moving forward, we can improve our analysis and make it broader by using advanced deep learning techniques like 1D convolutional neural networks and bidirectional long short-term memory networks. These methods might help the model be better at finding complicated patterns in text data. Expanding the types of categories to include things like fake, defamatory, peaceful, offensive and safe might give us a better understanding of different kinds of damaging materials that are found in this collection of Hindi texts.

To summarize, our present results are good at detecting hate speech in Hindi language, but there is a big chance to make them better. If we use more complex deep learning methods and add classification systems with many categories, we can improve the accuracy and scope of our re- search. This will help us fight against hate speech on the internet in a stronger way.

References

[1] J. Nockleyby, “Hate speech in encyclopedia of the american constitution,” *Electron. J. Academic Special Librarianship*, vol. 3, pp. 1277–1279, 2000.

[2] M. Williams, “Hatred behind the screens: A re- port on the rise of online hate speech,” *J. Exp. Theor. Artif. Intell.*, vol. 1, pp. 1–76, 2019. [Online]. Available: https://hatelab.net/wp-content/uploads/2019/11/Hatred-Behind-the-Screens.pdf

[3] Y. Ding, X. Zhou, and X. Zhang, “YNU_DYX at semeval-2019 task 5: A stacked BiGRU model based on capsule network in detection of hate,” in *Proc. 13th Int. Workshop Semantic Eval.*, 2019, pp. 535–539.

[4] G. Mou, P. Ye, and K. Lee, “SWE2: Subword enriched and significant
word emphasized framework for hate speech detec- tion,” in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, 2020, pp. 1145–1154, doi: 10.1145/3340531.3411990.

[5] R. Cao and R. K. Lee, “Hategan: Adversarial generative-based data augmentation
for hate speech detection,” in *Proc. 28th Int. Conf. Comput. Linguistics*, 2020, pp. 6327–6338, doi: 10.18653/v1/2020.col- ing-main.557.

[6] S. S. Tekiroglu, Y. Chung, and M. Guerini, “Generating counter narratives against online hate speech: Data and strategies,” in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 1177–1190. doi: 10.18653/v1/2020.acl-main.110.

[7] X. Zhou et al., “Hate speech detection based on sentiment knowledge
sharing,” in *Proc. 59th Annu. Meeting Assoc. Com- put. Linguistics 11th
Int. Joint Conf. Natural Lang. Process.*, 2021, pp. 7158–7166.

[8] Y. Chen, Y. Zhou, S. Zhu, and H. Xu, “Detect- ing offensive language in
social media to protect adolescent online safety,” in *Proc. IEEE Int. Conf.
Privacy, Secur., Risk Trust, Int. Conf. Soc. Comput.*, 2012, pp. 71–80.

doi: 10.1109/SocialCom-PASSAT.2012.55.

[9] Y.Mehdad and J. R. Tetreault, “Do characters abuse more thanwords ?,” in
Proc. 17th Annu. Meeting Special Int. Group Dis- course Dialogue, 2016, pp. 299–303, doi: 10.18653/v1/w16-3638.

[10] Ar- tiran, S., Ravisankar, R., Luo, S., Chukoskie, L., & Cosman, P. , “Measuring Social Modulation of Gaze in Autism Spectrum Condition With Virtual Reality Interviews” IEEE.

[10] X. Zhou et al., “Hate speech detection based on sentiment knowledge sharing,” in *Proc. 59th Annu. Meeting Assoc. Com- put.
Linguistics 11th
Int. Joint Conf. Natural Lang. Process.*, 2021, pp. 7158–7166.

[11] S. Kamble, A. Joshi, Hate speech detection from code-mixed hindi-english tweets using deep learning models, 2018. arXiv:1811.05145.”

[12] V. Mujadia, P. Mishra, D. Sharma, Iiit-hyder- abad at hasoc 2019: Hate speech detection, in: FIRE, 2019.”

[13] R. Joshi, R. Karnavat, K. Jirapure, R. Joshi, Evaluation of deep learning models for hostility detection in hindi text, in: 2021 6th International Conference for Convergence in Technology (I2CT), IEEE, 2021, pp. 1–5.

[14] R. Joshi, P. Goel, R. Joshi, Deep learning for hindi text classification: A comparison, in: International Conference on Intelligent Human Computer Interaction, Springer, 2019, pp. 94–101.

[15] R. Joshi, R. Karnavat, K. Jirapure, R. Joshi, Evaluation of deep learning models for hostility detection in hindi text, in: 2021 6th International Conference for Convergence in Technology (I2CT), IEEE, 2021, pp. 1–5.

[16] Y. Ding, X. Zhou, and X. Zhang, “YNU_DYX at semeval-2019 task 5: A
stacked BiGRU model based on capsule network in detection of hate,” in
Proc. 13th Int. Workshop Semantic Eval., 2019, pp. 535–539.

[17] G. Mou, P. Ye, and K. Lee, “SWE2: Subword enriched and significant
word emphasized framework for hate speech detec- tion,” in
Proc. 29th ACM Int. Conf. Inf. Knowl. Manage., 2020, pp. 1145–1154, doi: 10.1145/3340531.3411990.

Copyleaks

Plagiarism and AI Content Detection Report

Blackbook ITA12.docx

Scan details

Scan time:	Total Pages:	Total Words:
April 29th, 2024 at 9:8 UTC	37	9077

Plagiarism Detection

Types of plagiarism		Words
Identical	1.2%	108
Minor Changes	0.3%	26
Paraphrased	0%	1
Omitted Words	89%	8077

AI Content Detection

Text coverage		Words
AI text	5.5%	502
Human text	94.5%	498

Learn more

Plagiarism Results: (15)

4.71 Guidelines thesis dissertaion report- Ph.D,M.E,B.E (AC 4-3-14).pdf — 13.5%

https://archive.mu.ac.in/syllabus/4.71%20guidelines%20thesis%20dissertaion%20report-%20ph.d,m.e,b.e%2...

VeryPDF

UNIVERSITY OF MUMBAI, MUMBAI GUIDELINES FOR PREPARATION OF THESIS / DISSERTATIONS / REPORTS for Ph. D. / M. E. / B. E. (Acknowledgement...

COPYRIGHT — 11.1%

https://www.glsuniversity.ac.in/docs/declaration-of-academic-honesty-and-integrity.pdf

Hima Trivedi

(Declaration to be given by the candidate on a Non-Judicial stamp paper of rupees 100/- and verified by a Notary) Declaration I declare...

Lattice Attacks on Cryptographic Algorithms : Some Experiments — 11%

http://vharsh2.web.engr.illinois.edu/reports/btp.pdf

Vipul Harsh

Indian Institute of Technology Bombay B.Tech Project Lattice Attacks on Cryptographic Algorithms : Some Experiments Authors: Vipul Har...

5.pdf — 10.6%

https://saurabhgarg1996.github.io/files/5.pdf

Cross Lingual Information Retrieval and Error Tracking in search engine by Saurabh Garg Roll No: 140070003 under the guidance of Prof. P...

GITHUB LINK

https://github.com/ASP2000/hate-speech.git/