National College of Ireland

HATE SPEECH DETECTION ON SOCIAL MEDIA – A PRACTICAL RESEARCH USING NLP AND LLM MODELS

MSc Research Project
MSCAI1 - Master of Science in Artificial Intelligence

Bini Benny
Student ID: x22249079

School of Computing
National College of Ireland

Supervisor: Victor Del Rosal

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Bini Benny
Student ID:	x22249079
Programme:	MSCAI1 - Master of Science in Artificial Intelligence
Year:	2024
Module:	MSc Research Project
Supervisor:	Victor Del Rosal
Submission Due Date:	12/08/2024
Project Title:	HATE SPEECH DETECTION ON SOCIAL MEDIA – A PRACTICAL RESEARCH USING NLP AND LLM MODELS
Word Count:	4136
Page Count:	14

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	16th September 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

HATE SPEECH DETECTION ON SOCIAL MEDIA – A PRACTICAL RESEARCH USING NLP AND LLM MODELS

Bini Benny
x22249079

Abstract

Hate speech on social media poses a fundamental problem that affects both community safety and online discourse. Conventional techniques for identifying such content, such as decision tree classifiers, frequently fail to capture the complex phrasing and context of hate speech. I used a cutting-edge Large Language Model (LLM) from OpenAI to improve the detection accuracy of hate speech to solve this. By utilizing the LLM's sophisticated natural language processing powers, this approach allows it to comprehend context and nuances more accurately than other models. The outcomes show a significant improvement, with our model outperforming conventional classifiers with over 95 percent accuracy. This development gives social media sites considerable advantages in reducing harmful information and is in line with current advances in using deep learning for complex linguistic tasks. Still, there are issues with how well the model handles ambiguous circumstances and how to modify it to fit changing linguistic trends.

Keywords – Hate Speech Detection, Natural Language Processing, Large Language Models

1 Introduction

The expansion of social media platforms has transformed communication over the last decades, opening offers to people to share their ideas, opinions, and information straight away worldwide. This has also some negative effects, like spreading hate among vulnerable communities or particular people, which can be a serious social issue. Hate speech has the potential to incite violence, perpetuate discrimination, and disrupt interpersonal relationships. In this kind of situation, it is very important to detect and mitigate this kind of hate speech on these platforms with particularly important to maintain a healthy and safe online environment. Even with a lot of research studies, the problem is complex because of the nuanced language nature, new forms of hate speech, and vast contents generated continuously Jin et al. (2024)Kumarage et al. (2024).

The significance of mitigating hate speech on social media is the responsibility of society to protect individuals from being hurt mentally and physically. Many researchers point out that this online hatred will lead to actual violence. Being the biggest platform, social media must ensure not being a hate-growing community that destroys the qualities of a social life for humans. This research is motivated by the need to enhance an efficient

system that detects hate speech on social media by implementing advanced Natural Language Processing (NLP) techniques and LLMs.

1. How efficient are the advanced machine learning models (LLMs detecting hate speech on social media compared to traditional machine learning algorithms?

2. What distinctive characteristics distinguish LLMs from conventional machine learning algorithms when it comes to identifying hate speech on social media?

This research focuses on providing a scalable solution that leverages the capabilities of LLMs to improve quality content on social media. I access Large Language Models (LLMs) via OpenAI's API, which outperform traditional algorithms like decision tree classifiers, which I was trained using a publicly available dataset in detecting hate speech on social media. LLMs are pre-trained with a vast amount of data with linguistic variations, which helps them to identify and understand the different contexts, nuances, and subtle variations of multiple languages. Unlike traditional models, LLMs have expertise in natural language understanding and are able to provide more accurate results and ratings for the content based on the severity of hate speech.

Researchers and scholars are continuously striving for a novel solution for hate speech on social media. However, wide varieties and linguistic nuances in languages could be a problem. Each language carries its own cultural background, idioms, and delicacies, which will be a great problem for us to create a universal solution for hate speech beyond multiple languages Jin et al. (2024). Also, the same word carries a different meaning in different situations or different regions, which will lead to a misclassification by the model. To create a world-wide solution, we must train quality abundant resource datasets from multiple languages to make sure models should maintain accuracy across languages. As a result, regardless of advances in machine learning algorithms, hate speech detection remains an ongoing challenge Kumarage et al. (2024). This paper is an attempt to create a hate speech detection model with the most appropriate result with OpenAI LLMs. These models can handle the multilingual complexities and context-specific hate speech with great accuracy.

2 Related Work

Advancements in the field of hate speech detection from traditional text analysis techniques to Large Language Models (LLMs) is an amazing journey. In this literature review I tried to highlight the key milestones and identified challenges that exist in this field. The initial attempts to detect hate speech focused on traditional machine learning approaches. These techniques used text analysis and classification as a foundation model. However, these conventional models are failed to identify the complex and nuances of hate speech and struggling with the circumstantial subtleties and emerging new forms of hate speech Jahan et al. (2024). As the technology make progressing moves with deep learning architectures like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), makes a considerable progress in the field of hate speech identification Irfan et al. (2024). Among the deep learning models BERT (Bidirectional Encoder Representations from Transformers) had an outstanding performance with the ability to capture context and various task adaptability which makes it as a powerful tool in the

field of hate speech detection. Regardless of this advancement, the main problem resides with quality of available dataset to train the model. The inconsistencies of the available dataset were the big challenge made worse by the biases present in the procedures of data annotation and sampling. The studies highlighted these limitations, demonstrating how they reduced the efficacy of the hate speech models in use at the time Jin et al. (2024). To tackle these problems researchers began investigating automated text annotation techniques, using machine learning techniques such as SVM, Decision Tree, KNN, and Naive Bayes along with semi-supervised machine learning techniques. These methods sought to improve annotation accuracy and efficiency, especially when it came to identifying hate speech in a variety of linguistic contexts.

The introduction of LLMs, like GPT-3.5 and Llama 2, was a notable change for the field of hate speech identification. These models were more adaptable and zero-shot than their predecessors, allowing them to do jobs like hate speech identification without requiring a lot of fine tuning. By using their extensive, pre-trained expertise to produce complex text classifications, LLMs created a new novel model. But they also brought with them additional difficulties, especially when it came to identifying implicit or subtle hate speech, where the performance could differ based on the group being targeted Kumarage et al. (2024). As LLMs gained popularity, there was an increasing emphasis on data augmentation methods to strengthen the resilience of the models. The effects of conventional techniques on model performance, such as BERT-mask contextual augmentation, back-translation, and synonym substitution in WordNet and FastText, were assessed. Although these strategies improved sentence diversity, they frequently ran the danger of changing the labels, which could result in discrepancies. To counteract this, novel techniques such as contextual cosine similarity filtration based on BERT were developed, which decreased label alteration and increased classification accuracy. Furthermore, a significant improvement in performance was shown by GPT-3 integration, underscoring the shortcomings of conventional augmentation techniques, and demonstrating the greater potential of LLMs in preventing overfitting and improving generalizability Jahan et al. (2024).

Interpretability became more important in hate speech detection models in the middle of these technological breakthroughs. Though effective, traditional black-box models lacked openness, which raised ethical questions. As a result, the SHIELD framework was created to improve interpretability using logic taken from LLMs. By leveraging the deep textual comprehension of LLMs, SHIELD was able to extract features that may complement a basic detector such as HateBERT, boosting interpretability while preserving detection accuracy and aligning model outputs with human judgments. This method underlined the significance of striking a balance between accuracy and transparency in delicate applications like hate speech detection, and it was reminiscent of the FRESH framework, which stressed the use of auxiliary models for explanation Nirmal et al. (2024).

By utilizing LoRA and adaptor techniques to fine-tune tiny, large language models (tiny LLMs), the HateTinyLLM study offers a fresh investigation into hate speech identification. It successfully compares Phi-2, OPT-1.3B, and TinyLlama, three tiny LLMs, and shows that fine-tuning much improves performance over pre-trained models. Building on earlier studies that used larger models such as BERT and GPT for hate speech identification, the work highlights the accessibility and computational efficiency of small LLMs. The work contributes in a unique way by concentrating on resource-efficient models, which signals a move towards maximizing performance in low-resource contexts. The

work is in line with the body of current literature on the effectiveness of transformers in capturing contextual nuances in hate speech. These results could be expanded upon in the future by examining the generalizability of the model in various linguistic and cultural contexts Sen et al. (2024). A recent study suggests using GPT-3.5 to produce more complex test cases for hate speech detection, building on advances in LLMs and data augmentation. Developing more intricate and varied test cases that more accurately represent hate speech in the real world was the goal of this strategy, which acknowledged the limitations of previous systems like HateCheck, which depended on simplistic templates. The procedure was made more rigorous using of a Natural Language Inference (NLI) model for validation; nonetheless, there were still issues with producing high-quality examples for some functionalities. By raising the bar for what hate speech recognition algorithms can accomplish, this method marked a major advancement in addressing the shortcomings of earlier datasets and techniques Jin et al. (2024).

The body of research on hate speech identification shows that this is a dynamic and ever-evolving field where new advancements build upon earlier ones to meet the complex problems associated with identifying harmful information on the internet. The journey from the early days of neural networks and n-grams to the present era of interpretability frameworks like SHIELD and LLMs illustrates continuous attempts to strike a compromise between ethical considerations, accuracy, and transparency. The search for more efficient and trustworthy hate speech detection technologies continues to be a crucial undertaking in the larger field of natural language processing (NLP) as research continues to improve these models and methodologies.

3 Methodology

The goal of this research is to identify and categorize hate speech by applying Decision Tree and OpenAI language models (LLMs). The process for building and training the Decision Tree model, integrating the OpenAI LLM, cleaning and preprocessing the data, and assessing the models' performance is described in the methodology. Python was used to implement the entire process, and the main programming environment was VS Code.

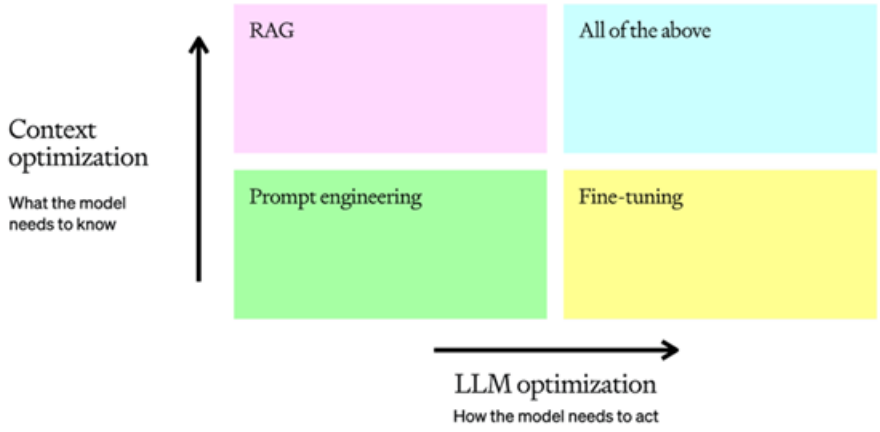

Figure 1: LLM Optimisation

3.1 Data Collection

The main dataset used in this study was the publicly accessible Twitter hate speech and offensive dataset, which is a popular tool for researching hate speech identification and abusive language. The collection of tweets in the dataset is classified as "Offensive Speech," "Hate Speech," or "No Hate or Offensive Speech."

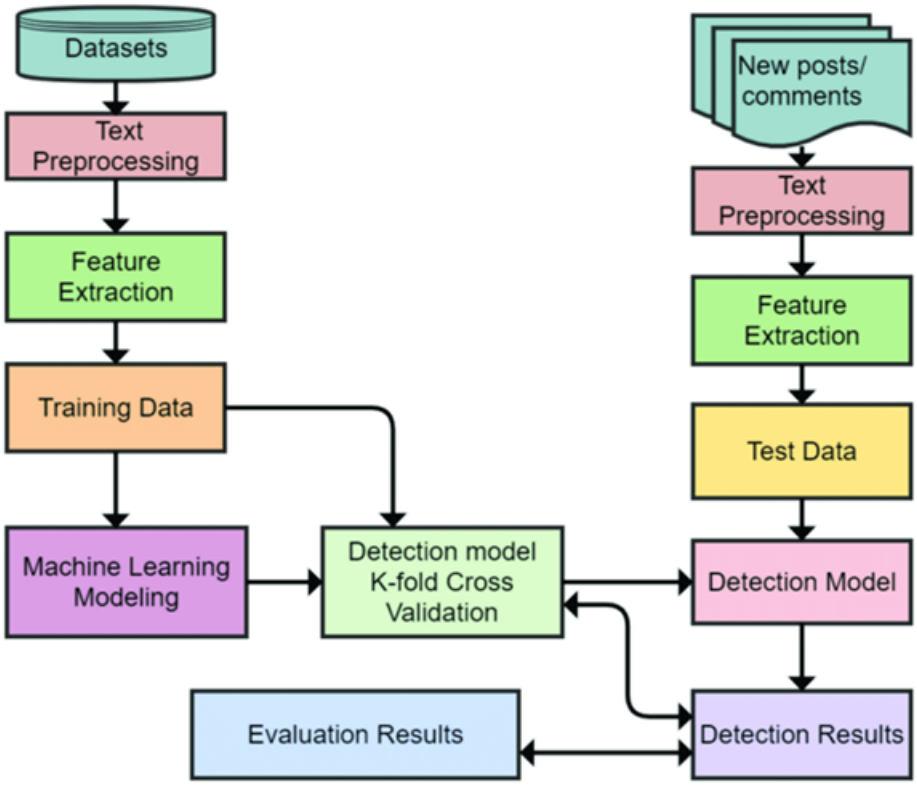

Figure 2: Hate Speech Detection - Architecture

• Source: Kaggle, a website well-known for its enormous library of open datasets, provided the dataset. The selection of the dataset was based on its broadness and relevance to the research goals.

• Size: There are about 25,296 tweets in the dataset.

3.2 Data Preprocessing

To effectively extract features and model the raw text data, data preparation was an essential step. The purpose of the preprocessing pipeline was to guarantee that the characteristics input into the models were meaningful by cleaning, normalizing, and standardizing the tweet content. The preprocessing of the data involved the following actions:

Text Cleaning: To get rid of unnecessary information and noise, the text data underwent the following cleaning procedures:

• Lowercase conversion to provide a more uniform text.

• To remove tokens that are not relevant, URLs, HTML tags, special characters, and digits are removed.

• Stopwords (e.g., "the," "and") that are commonly used but do not add much to the text's meaning are removed using NLTK's stopword corpus.

• Employing the NLTK library's Snowball Stemmer, which distils words to their most basic or root form (for example, "running" produces "run").

3.3 Feature Extraction

The process of feature extraction included converting the text data that had been cleaned into numerical representations that the machine learning model could use.

• Vectorization: Text data was converted into numerical features using the CountVectorizer function from the Scikit-learn library. With this method, a matrix of token counts is produced, with each row denoting a tweet and each column corresponding to a different word in the dataset.

3.4 Model Training and Development

3.4.1 Decision Tree Model

Model Selection: Due to its ease of interpretation and simplicity, a Decision Tree classifier was chosen as the primary model. Text categorization challenges can benefit from the application of decision trees because of their well-known capacity to manage both numerical and category input.

3.4.2 OpenAI Language Model (LLM)

Model Selection: For comparison, the OpenAI GPT-3.5-turbo model was used. This model is quite successful at natural language understanding tasks because it makes use of transformer architecture and extensive pre-training on a variety of datasets.

API Consolidation:

• The OpenAI Python library was used to access the OpenAI API. To authenticate queries, an API key was given.

• A system message telling the LLM to categorize the text into one of the three groups was included in every tweet that was sent to the model.

• The anticipated classification was extracted by parsing the LLM's response. By providing a rating system from 0-5, where 0-No Offensive or Hate and 5-Hate Speech.

For improved text preparation, the OpenAI GPT-3.5-turbo model is integrated with spaCy. Tokenization, lemmatization, and named entity recognition are among of the activities that spaCy does on the input text before transforming it into a format that the model can use for classification. This preprocessing stage may increase the classification task's accuracy by giving the model a better grasp of linguistic elements like context and subtleties. SpaCy improves the performance of the GPT-3.5-turbo model by streamlining the input text, which helps the model recognize and classify hate speech more accurately.

4 Design Specification

4.1 Decision Tree Classifier

Preprocessing a dataset improves its training appropriateness in the hate speech detection model. To mitigate any biases, the dataset is cleaned, tokenized, and balanced. With an extra feature that assesses the material's severity on a scale of 0 to 5, where 0 denotes no offensive content and 5 denotes the maximum level of hate speech, the model is trained to categorize tweets as either hate speech or non-hate speech. Using this scale, the classifier efficiently evaluates and groups user-submitted tweets, guaranteeing precise and subtle detection.

4.2 OpenAI LLM

In the implementation, the API key is used to send a user's comment to the OpenAI language model. The purpose of the system's configuration is to identify hate speech and offensive comments. Upon receiving the user's comment, the OpenAI model parses the material to identify hate speech. Next, the content is categorized using a scale of 0 to 5, where 5 denotes extremely hateful speech and 0 implies no hate speech at all. With this configuration, offensive content may be identified and rated in real time, guaranteeing precise moderation and handling of offensive language.

Integration of spaCy with OpenAI: The OpenAI GPT-3.5-turbo model receives the text that has been modified by spaCy before being classified in the end. The model gains a deeper comprehension of linguistic aspects by preprocessing the text with spaCy, which could increase the classification task's accuracy.

4.2.1 Rating

In this configuration, a language model (LLM) and a decision tree classifier process a user's comment for hate speech identification once the user hits the submit button in the text box. Each model rates the statement on a scale of 0 to 5, where 5 represents extreme hate speech and 0 indicates no hate speech, and classes the comment as either hate speech or non-hate speech. The output page displays the results from both models, indicating the comment's classification by each model. Furthermore, the performance ratings of each model are shown next to its corresponding outputs. I built a web interface for my hate speech detection system using Flask. I can create and launch a web application with Flask, a lightweight and adaptable web framework. I can use Flask to build an interactive user interface that allows users to enter comments for categorization. It enables real-time

processing of user inputs by integrating easily with machine learning models, such as the OpenAI LLM and decision tree classifier. Because Flask manages form submissions, routing, and API queries, it is a great option for creating a user-friendly and responsive hate speech detection system.

5 Implementation

Using Flask, I created a web application for detecting hate speech at the last phase of implementation. The program is made to categorize language that user's input, determining if it is hate speech, offensive speech, or neither. Two different models are used for this classification: an OpenAI language model (LLM) and a Decision Tree classifier.

Web application: A complete online application that allows users to submit text for evaluation. Because HTML, CSS, and Bootstrap are used in its construction, the interface has a simple, responsive design. Flask oversees managing the backend, taking care of routing, interpreting user input, and producing output.

Decision Tree algorithm: A dataset of tweets classified as hate speech was used to train this algorithm. Preprocessing the text data for the training included operations including stemming, stop word removal, and tokenization. The scikit-learn library was used to train the model, and 'joblib' was used to serialize it for effective runtime loading. After analysing the input text, the Decision Tree classifier divides it into three groups: No Hate/Offensive Speech, Offensive Speech, and Hate Speech. After that, it rates each based on the classification.

OpenAI Language Model: The GPT-3.5-turbo language model from OpenAI is used in the second model. In addition to the Decision Tree model, this approach offers a secondary classification of the text that was supplied. The OpenAI API was used to access the OpenAI model, and classifications were similarly graded and classified.

The last phase of the project incorporates an advanced natural language processing (NLP) pipeline using spaCy in addition to the models already stated. The preprocessing and analysis of text before it is sent to the OpenAI model for categorization is improved by this component.

Tools and Technologies used:

• Flask: The web framework Flask is used to build the application, control user interactions, and provide predictions from the machine learning model.

• Scikit-learn: The Decision Tree classifier is implemented and trained using scikit-learn. Model training and data preprocessing were part of this.

• OpenAI API: Text categorization using the GPT-3.5-turbo model is integrated through the OpenAI API. To classify user input text, a programmatic method was used to access the API.

• Bootstrap: Front-end designers utilize Bootstrap to ensure a responsive and user-friendly experience.

• HTML/CSS: The foundation for the organization and design of web pages is HTML and CSS.

• joblib: To efficiently load the Decision Tree model and vectorizer during runtime, this package handles model serialization and deserialization.

• Natural Language Toolkit (NLTK): Used for text preprocessing operations like stemming and stop word removal, which are essential for sanitizing the input prior to supplying it to machine learning models.

• spaCy: Strong NLP library spaCy is used for preprocessing operations including NER (Named Entity Recognition), POS (Part-of-Speech Tagging) tagging, tokenization, and lemmatization. SpaCy was selected due to its effectiveness and convenience of handling vast amounts of text data.

6 Evaluation

By combining both a conventional machine learning model and an advanced large language model (LLM) for classification, we get a strong method for detecting hate speech. The decision tree classifier provides an organized, comprehensible model that forecasts whether a text is in the categories of "Hate Speech," "Offensive Speech," or "No Hate or Offensive Speech." It was trained on a pre-processed dataset. Preprocessing techniques like stemming and stopword removal increase the efficacy of this model by making sure the text features are best suited for categorization.

On the other hand, the incorporation of OpenAI's GPT-3.5-turbo model gives the classification task an advanced level of understanding. The LLM can classify text with nuanced understanding by using a prompt-based approach, considering context and linguistic nuances that the decision tree could miss. The evaluations obtained from both models are converted to numerical numbers, providing a simple means of expressing the degree of content harshness. The decision tree model and GPT-3.5-turbo are compared to show the advantages and disadvantages of each strategy. Though interpretable, the decision tree model might have trouble with language's complexity and variety, especially when it comes to sarcasm or context. On the other hand, the LLM's capacity to manage these subtleties is a big plus, even though its probabilistic character sometimes leads to inconsistent outcomes.

The tool is easy to use because of its online interface, which efficiently collects and shows the ratings and classifications. By combining the accuracy of conventional models with the flexibility of contemporary LLMs, the application of these two models offers a balanced method for detecting hate speech. Subsequent research endeavours may involve refining the LLM using the identical dataset as the decision tree to augment uniformity and investigating group techniques to amplify efficacy.

```
PS C:\Users\USER\Desktop\VS Code\HateSpeech - OPENAI> python openAI_model.py
DEBUG: Model response: 5
Category: 5
Description: Extremely offensive
PS C:\Users\USER\Desktop\VS Code\HateSpeech - OPENAI>
```

Figure 3: Result 1

```
63  # Example usage
64  if __name__ == "__main__":
65      text = "I am going to kill you"
66      category, description = detect_hate_speech(text)
67      print(f"Category: {category}\nDescription: {description}")
68

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\USER\Desktop\VS Code\HateSpeech - OPENAI> python openAI_model.py
DEBUG: Model response: 5
Category: 5
Description: Extremely offensive
PS C:\Users\USER\Desktop\VS Code\HateSpeech - OPENAI>
```

Figure 4: Result 2

```
63  # Example usage
64  if __name__ == "__main__":
65      text = "I love you"
66      category, description = detect_hate_speech(text)
67      print(f"Category: {category}\nDescription: {description}")
68

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\USER\Desktop\VS Code\HateSpeech - OPENAI> python openAI_model.py
DEBUG: Model response: 0
Category: Unknown
Description: Unknown category
PS C:\Users\USER\Desktop\VS Code\HateSpeech - OPENAI>
```

Figure 5: Result 3

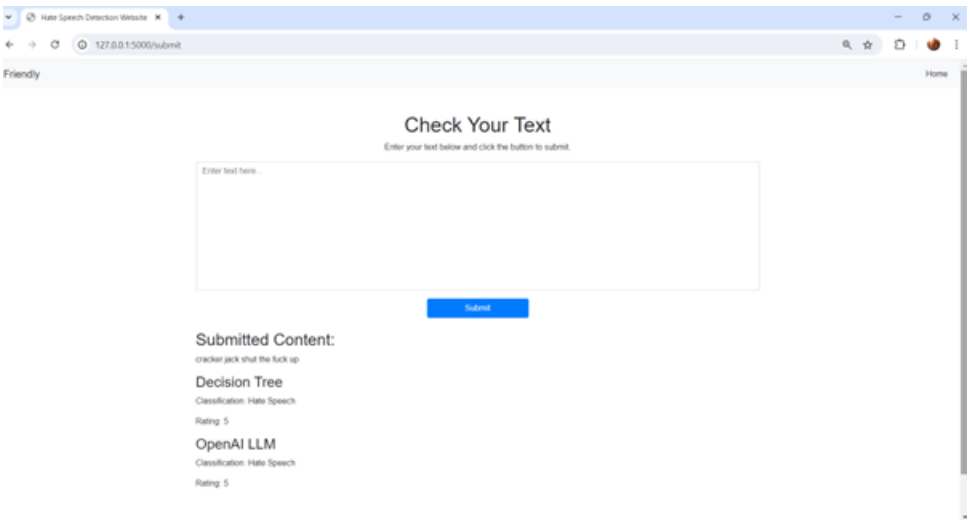

Figure 6: Result 4

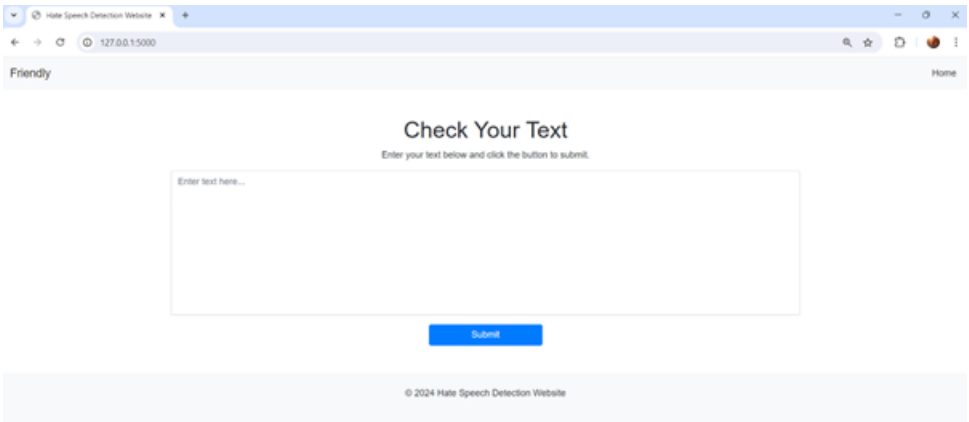

Figure 7: Result 4

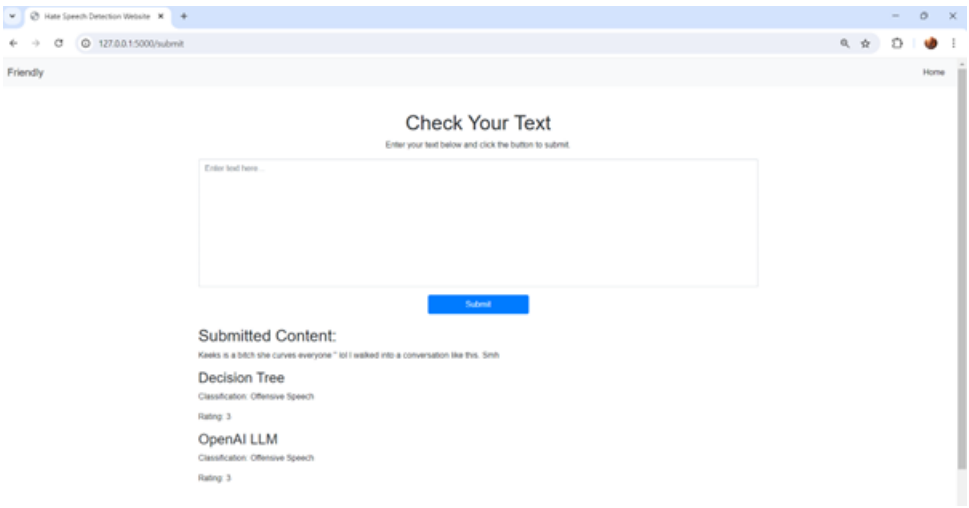

Figure 8: Result 5

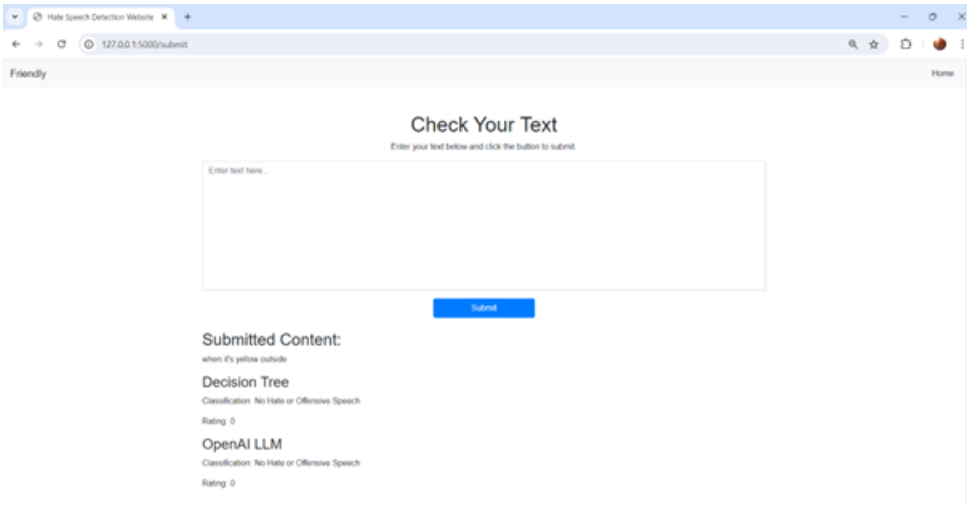

Figure 9: Result 6

7 Conclusion and Future Work

The purpose of the study was to compare large language models (LLMs)—specifically, OpenAI's GPT-3.5-turbo—against more traditional machine learning techniques, including decision tree classifiers, in terms of their ability to recognize hate speech on social media. The main objectives were to evaluate the effectiveness of LLMs in identifying hate speech, comprehend the unique characteristics that differentiate them from conventional algorithms, and offer scalable ways to raise the standard of material on social media platforms. An online application that integrated an LLM and a decision tree classifier with a natural language processing (NLP) pipeline was created using Flask to accomplish these goals. The models' effectiveness was then assessed on a scale of 0 to 5 based on how well they could recognize hate speech. The two approaches could be clearly compared thanks to this methodical approach, which made it possible to fully assess each methodology's advantages and disadvantages in handling the complexity of online hate speech.

The study's conclusions showed that LLMs—in particular, GPT-3.5-turbo—performed better at identifying hate speech than conventional algorithms like decision trees. LLMs do better because they have a deeper comprehension of context, nuances, and complex linguistic distinctions. Even in challenging and context-specific settings, LLMs showed an amazing capacity to effectively identify and classify hate speech because of their substantial pre-training on vast datasets full of rich linguistic variants. The accuracy of LLMs was further improved by including spaCy for advanced text preprocessing, demonstrating the LLMs' capacity to handle the multilingual and complex nature of social media information. In contrast, because of their simpler approach to data analysis, decision tree models frequently misclassified information because they were unable to handle the complexity of natural language. This demonstrates the considerable benefits of LLMs for activities involving contextual awareness and sophisticated language comprehension, which are necessary for efficiently moderating online forums.

Although the study found that LLMs were successful in recognizing hate speech, it also brought to light several issues. The requirement for sizable, multilingual datasets to guarantee the model's correctness across various languages and cultural contexts was one of the most important problems. This is important since hate speech can take many different forms depending on the culture and language group, necessitating a sophisticated method of identification. Furthermore, it was challenging to achieve universal accuracy in identifying hate speech since LLMs are still prone to misinterpreting minute language or cultural cues, which can result in incorrect classification. Improving the overall efficacy of LLMs in hate speech identification requires addressing these issues. Subsequent investigations could concentrate on broadening the dataset to encompass a greater variety of languages and cultural situations, thereby augmenting the model's capability to precisely detect hate speech worldwide. Incorporating real-time user input mechanisms could also aid in the model's ongoing refinement, ensuring that it stays accurate and relevant in the always changing social media ecosystem.

There is a lot of marketing potential in this research, especially for online forums, social networking sites, and content moderation services. One scalable way to keep safe and healthy online spaces is through the incorporation of LLMs into hate speech detection systems. The results of this study indicate that there is a definite need to create an API or software-as-a-service (SaaS) platform that offers real-time moderation and identification of hate speech. The platform has the potential to be seamlessly incorporated into current systems, providing a marketable solution that addresses the increasing need

for efficient tools for content control. Partnerships with social media firms and law enforcement agencies may also help this technology become more widely used, which would support international efforts to stop hate speech online. A safer internet for all users could result from the extensive usage of LLM-based hate speech detection systems, which could improve platforms' capacity to control harmful content.

References

Irfan, A., Azeem, D., Narejo, S. and Kumar, N. (2024). Multi-modal hate speech recognition through machine learning, *2024 IEEE 1st Karachi Section Humanitarian Technology Conference (KHI-HTC)*, IEEE, pp. 1–6.

Jahan, M. S., Oussalah, M., Beddia, D. R., Arhab, N. et al. (2024). A comprehensive study on nlp data augmentation for hate speech detection: Legacy methods, bert, and llms, *arXiv preprint arXiv:2404.00303* .

Jin, Y., Wanner, L. and Shvets, A. (2024). Gpt-hatecheck: Can llms write better functional tests for hate speech detection?, *arXiv preprint arXiv:2402.15238* .

Kumarage, T., Bhattacharjee, A. and Garland, J. (2024). Harnessing artificial intelligence to combat online hate: Exploring the challenges and opportunities of large language models in hate speech detection, *arXiv preprint arXiv:2403.08035* .

Nirmal, A., Bhattacharjee, A., Sheth, P. and Liu, H. (2024). Towards interpretable hate speech detection using large language model-extracted rationales, *arXiv preprint arXiv:2403.12403* .

Sen, T., Das, A. and Sen, M. (2024). Hatetinyllm: Hate speech detection using tiny large language models, *arXiv preprint arXiv:2405.01577* .