Princess Sumaya University for Technology
Department of Data Science
Semester 2023/2024

Aspect-Based Hate Speech Detection

Prepared By:
Mohammad Al Malalha
Hamzeh Bseiso
Zaid Abdullah

Supervised By:
Dr. Omar Alqawasmeh

Project Submitted in partial fulfillment for the degree of Bachelor of Science in Data Science

First Semester-2023-2024

Declaration of Originality

This document has been written entirely by the undersigned team members of the project. The source of every quoted text is clearly cited and there is no ambiguity in where the quoted text begins and ends. The source of any illustration, image or table that is not the work of the team members is also clearly cited. We are aware that using non-original text or material or paraphrasing or modifying it without proper citation is a violation of the university's regulations and is subject to legal actions.

Names and Signatures of team members:

Acknowledgments

Express your appreciation to whoever helped you during your work or even before!

Summary

Give a complete but concise description of your work. The summary is a brief overview of your motivation, statement of purpose, general methodological approach, major results, discussion and conclusion. The abstract should not exceed one page.

List of Abbreviations

List the abbreviations you have used in your project if there are any and what they stand for.

NGO: Non-Governmental Organization.

CNN: Cable News Network.

NB: Naive Bayes.

SVM: Support Vector Machine.

LSTM: Long Short-Term Memory.

FCM: Feature Concatenation Model.

NLP: Natural Language Processing.

RAM: Random Access Memory.

SSD: Solid State Drive.

TF-IDF: Term Frequency-Inverse Document Frequency.

UI: User Interface.

Table of Contents

The table of contents should be automatically generated by going to: *Insert >> Index And Tables>> Table of Contents.* Choose *Classic* as the format of the table and set the number of levels to be 3.

In order for the table of contents to be generated correctly:

- Each chapter title should be formatted using the style “Chapter”.
- Each section title should be formatted using the style “Section”.
- Each subsection title should be formatted using the style “Subsection”.

The table of contents should replace all of the text that is in this page.

Table of Figures

The table of Figures should be automatically generated by going to: *Insert >> Index And Tables>> Table of Figures.* Choose *Classic* as the format of the table and set the tab leader to dots.

In order for the table of figures to be generated correctly, the label of each figure should be formatted using the style “Figure”.

The table of figures should replace all of the text that is in this page.

Table of Contents

Table of Figures

Chapter 1
Introduction

1.1 Overview

In a world increasingly dominated by digital communication, the ability to communicate with others is an important aspect of our lives. However, there are challenges that people might face while communicating with others online such as hateful and discriminatory language. Therefore, our project introduces an application that extracts hate-speech from text and takes action.

The main aim of our project is detecting, categorizing and dealing with hate speech using machine learning and natural language processing techniques. We hope this will result in a safer environment for those who are seeking safer platforms. Additionally, raising awareness and improving safety by allowing concerned entities (e.g., news platforms, social media platforms, and/or NGOs) to take legal actions towards those who introduce hate speech against people.

Data from multiple news outlets such as CNN, EuroNews and Council of Europe show that cases of online hate speech have substantially increased in the past few years. Hate speech in online conversations can cause real-world harm, affecting the user's mental and emotional health. This will give users the ability to identify and respond to hate speech, so the app helps create a safer online environment. They can also choose to withdraw from conversations involving hate speech, promoting a sense of agency and personal safety.

1.2 Problem Statement

Our project aims to address the growing issue of online hate speech and create a safer environment for users. Online hate speech, including racism, sexism, offensiveness, and other forms of discrimination, can cause emotional distress. By detecting hate speech the project seeks to reduce the harmful impact of such content, promote respectful communication, and empower users to make informed decisions about their online interactions.

We propose to deal with the hate speech problem in terms of aspects. In order to achieve our goal, we propose to categorize our output based on different aspects (i.e., Aspect based hate detection). To formalize, we propose three main tasks:

- Task1. Nature Language Processing
    - Transform text into vectors and embeddings to get the meaning and context.
- Task 2. Aspect Term Categories Extraction
    - Racism
    - Sexism
    - Offensive
    - None

The project will result in the development of a web application equipped with hate speech detection capabilities. The application will take raw text transcripts, sentences, files, etc. and classify them into various categories of hate speech (e.g., racist, sexist, offensive or none). The system will provide users with alerts and visual cues to identify hate speech occurrences, enabling them to respond appropriately or disengage from harmful conversations.

The primary audience for our application are organizations, in more details, we target the following audience set:

- Governments: where they can use our application to derive meaningful insights for specific topics (e.g., a political party, a specific event).

- News channels: they could use our application to detect hate speech on live-air to prevent any racist or offensive comments to their audience.

- NGOs: Our system provides NGOs with a powerful tool to enhance their efforts in creating a safer online space. By swiftly identifying and responding to hate speech.

- University-related studies/activities: By leveraging our system, university students can actively participate in promoting responsible digital communication within their academic communities. The tool empowers students to contribute to a secure online environment, where instances of hate speech are promptly detected and addressed.

Another major audience for this application includes individuals who engage in social interactions in public or online (zoom , facetime, chat rooms,etc).

Moreover, we also propose a speech-to-text feature designed for users of diverse backgrounds, ages, and experiences who are concerned about encountering hate speech during their online conversations. This is important for users who have difficulties or disabilities in using smart devices. It is particularly relevant for vulnerable communities (hard of hearing people, slow/low-awareness people,etc..), minors, and individuals seeking a safer online environment.

Users will utilize the web application during their online conversations. The app will process spoken words in real-time, identifying hate speech and categorizing it based on its type and will receive instant alerts and visual notifications when hate speech occurs. This information empowers them to make informed decisions about continuing or discontinuing interactions, ensuring personal safety and well-being.

1.3 Related Work

Table 1.1

Paper	Method	Dataset	Problem Statement	Results	Compare it with my own solution
[1]	Char N-Grams	hatespeech/NAACL_SRW_2016.csv at master · zeeraktalat/hatespeech · GitHub	Racism Sexism	F1 - 73.89% Precision - 72.87% Recall - 77.75%	To be decided once starting implementing GP2
[2]	NB & SVM	Toxic Comment Classification Challenge | Kaggle	toxic severe_toxic obscene threat-insult identity_hate	0.9772%	To be decided once starting implementing GP2
[3]	Logistic Regression & Linear SVM	hate-speech-and-offensive-language/data/labeled_data.csv at master ·	Hate Speech Offensive	precision - 0.91 recall - 0.90 F1 - 0.90. (a lot of	To be decided once starting implemen

		t-davidson/hate-speech-and-offensive-language · GitHub		misclassification based on their report)	ting GP2
[4]	LSTM & FCM	MMHS150K.zip - Google Drive	Hate Not Hate	LSTM - F-Score - 0.703 - accuracy - 68.3 FCM - F-Score - 0.697 - accuracy - 67.8	To be decided once starting implementing GP2

Authors in [1] proposed predictive features for Hate Speech Detection on Twitter by Zeerak Waseem and Dirk Hovy. They presented a list of criteria based on critical race theory to identify racist and sexist slurs. These can be used to gather more data and address the problem of a small, but highly prolific number of hateful users. Their best results were by using a character n-gram based approach providing a recall of 77.7%.

Authors in [2] created a kaggle competition for toxic comment classification, our main goal is to compare our results on the same data with the top 3 groups in the leaderboard. The number 1 group got a score of 0.98856 NB - SVM.

Authors in [3] Automated hate speech Detection. They used a crowd-sourced hate speech lexicon to collect tweets containing hate speech keywords. They labeled a sample of these tweets into three categories: those containing hate speech, only offensive language, and those with neither. They trained a multi-class classifier to distinguish between these different categories. In their conclusion they found that racist and homophobic tweets are more likely to be classified as hate speech but that sexist tweets are generally classified as offensive. precision - 0.91, recall - 0.90, F1 - 0.90.

Authors in [4] targeted the problem of hate speech detection in multimodal publications formed by a text and an image. We gather and annotate a large scale dataset from Twitter, MMHS150K, and propose different models that jointly analyze textual and visual information for hate speech detection, comparing them with unimodal detection. Their results were as follows using LSTM - F-Score - 0.703 - accuracy - 68.3, and these were their results using FCM (clustering method) - F-Score - 0.697 - accuracy - 67.8.

1.4 Contribution:

- Compared to the state of the art, we newly introduced the concept of Aspect Based Hate Speech Detection.
- Based on the papers we read and the insights we obtained, the majority of high-accuracy models used SVM and LSTM. Therefore, We assume using SVM and LSTM will enhance and yield the best results.
    - This makes sense, because when it comes to NLP models and using text or images for classification problems, the data is almost 100% of the time non-linear, and SVM is the best model for this problem due to its use of n-dimensions and optimal hyperplanes.
- Our project realistically aims to help any group of individuals who want to be aware of any hate-speech being spoken in their event to be able to take action (eg, company, government, ngo, university, online meetings, etc…).
- Novelty in the structure of the pipeline is new because it incorporates unique feature extraction techniques, specific aspect identification, and a purposeful combination of SVM and LSTM models for improved aspect-based hate speech detection.

1.5 Document Outline

In Chapter 2, we present the Project Plan, serving as our project blueprint, outlining deliverables, tasks, team contributions, and potential challenges.

In Chapter 3, Requirements Specification, delves into crucial components and stakeholder needs, defining the system's platform and specifying requirements.

In Chapter 4, System Design, explores the architectural and logical aspects of our hate speech detection system, utilizing visual aids for a comprehensive understanding of the design process.

Chapter 2
Project Plan

This chapter will be our blueprint for this project. We will go through the details of our plan, including what our project deliverables will be, what tasks we need to perform, what each member of the team has done for this project, what potential problems we might face that might have an impact on our project.

2.1 Project Deliverables

In this section, we will investigate the datasets that will be used for training and testing our model. We will also discuss what our project deliverables will be, which are:

- The model that is used to detect and categorize hate speech.
- The web application, which acts as a layer of communication between the user and our model.

2.1.1 Datasets

Our methodology to use the datasets is:

- For Category extraction, we will take advantage of the following data sources based on the category:
- For Racism and Sexism categories we will use the following datasets:
    - Hate Speech and Offensive Language Dataset (kaggle.com): Twitter data was employed as a dataset for exploring hate-speech detection. The text was categorized into hate speech, offensive language, or neither. It's crucial to emphasize that, given the study's nature, this dataset includes text that may be perceived as racist, sexist, homophobic, or generally offensive.
    - https://github.com/zeeraktalat/hatespeech/blob/master/NAACL_SRW_2016.csv
- For Offensiveness category we will use the following datasets:
    - Hate Speech and Offensive Language Dataset (kaggle.com)

The following tables show some examples of the dataset and its features:

Table 2.1

	count	hate_speech	offensive_language	neither	class	tweet
0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't complain about cleaning up your house. & as a man you should always take the trash out...
1	3	1	2	0	1	" these garbage women are only after money "
2	3	0	1	2	2	" momma said no pussy cats inside my doghouse "
3	3	2	1	0	1	"@ChandlerParsons: How much Do I hate these black idiot communities!!!"

For Table 2.1 (Hate Speech and Offensive Language Dataset | Kaggle) :

The "**count**" column represents the overall frequency of each tweet in the dataset, while "**hate_speech**" and "**offensive_language**" columns indicate the counts of hate speech and offensive language instances within each tweet, respectively. The "**neither**" column does not fall into either category. The "**class**" column assigns a label to the tweets, likely reflecting the level of hate speech or offensiveness. The "**tweet**" column contains the actual text content of each tweet, forming the basis for the analysis of hate speech and offensive language in the dataset.

Table 2.2

	Tweet ID	Category
1	560595245814267905	none
2	576612926838046720	none
3	575089106029428737	racism
4	569627338050117632	racism
5	572348106202750976	sexism
6	572319306387599360	sexism

For Table 2.2 (hatespeech/NAACL_SRW_2016.csv at master · zeeraktalat/hatespeech · GitHub) :

The table contains tweet IDs and their corresponding categories, with each tweet classified into one of the categories: "racism," "sexism," or "none." The tweet IDs serve as unique identifiers for individual tweets within the dataset. The categories in the second column offer insights into the content of each tweet, facilitating a classification based on racism, sexism, or none.

2.1.2 Model (Aspect based hate speech detection model)

The project will result in an advanced aspect-based hate speech detection model. Utilizing sophisticated natural language processing (NLP) techniques, this model will categorize different aspects of hate speech, such as racism, sexism, and general offensiveness. This approach enables a detailed understanding of diverse hate speech content, contributing to a safer online environment with targeted content moderation strategies.

2.1.3 Web application (streamlit)

The web application serves as a practical implementation of the hate speech detection model. This application will provide an accessible interface for users to interact with the hate speech detection model, allowing them to input text or and receive analysis and insights. The web application aims to democratize access to hate speech detection tools and promote a more informed and responsible discourse.

2.2 Project Tasks

In this project, we propose the following tasks, for an input text:

Task 1: Natural Language Processing:

First and most crucial, we will get rid of stop words in the datasets because they are not useful for the purpose of this task. After preprocessing, we transform the text obtained from audio into a format that captures the meaning and context effectively. We achieve this through NLP, specifically focusing on two essential aspects: semantic representation and contextual information. Below, we explain each of these aspects.

Semantic Representation:

Objective: Capture the meaning and inherent relationships between words in the text.

Benefit: Enhances the system's understanding of the underlying semantics, allowing it to recognize the nuanced meanings associated with different words.

Contextual Information Extraction:

Objective: Retain contextual information within the text to understand how words are used in specific situations.

Benefit: Enables the system to differentiate between different uses of the same word based on context, leading to more accurate and context-aware hate speech detection.

This process transforms the text into tokens and embeddings, representing the intricate relationships and contextual nuances in natural language. These embeddings are crucial for the following tasks which are the backbone of our results.

Task 2. Hate Category Extraction:

In this task, we will investigate the correct category which in the text falls into. E.g.; if a text describes a comment that is racist, the category should be “Racism”. We define the following set of categories:

- **Category 1**. Racism: Any comment that brings discomfort and hatred towards a race or ethnicity.
- **Category 2.** Sexism: Any comment that shows and assumes that one gender is more superior to the other.

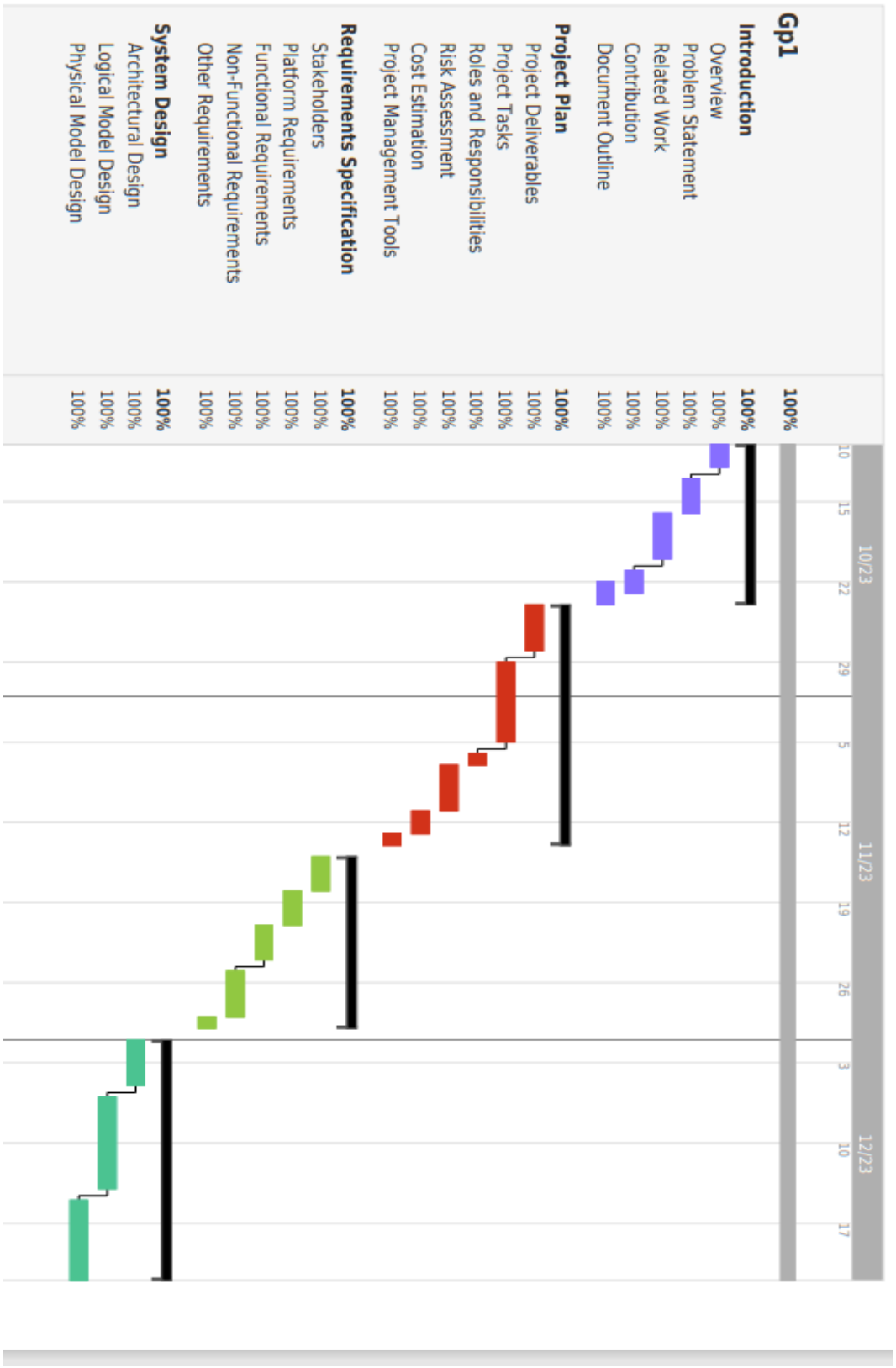
- **Category 3.** Offensiveness: Any comment that is likely to cause displeasure or anger. It can include language that is disrespectful, hurtful, or inappropriate.
- **Category 4.** None: Indicates that the provided sentence is hate-speech free.

Introducing the Gantt chart of our project which provides a visual roadmap that highlights key milestones and tasks over time. This timeline graphic offers a comprehensive overview of the project's schedule, illustrating the sequence and duration of each activity. It's important to note that this Gantt chart specifically outlines the plan for Part 1 of our project. Part 2 will be addressed separately in its own timeline.

Table 2.3 (Color Schema based on task to match Figure 2.1)

Task	Date Started	Date Ended	Duration (in days)	Weeks per chapter (2.6 = 2week6day)
Overview	2023-10-10	2023-10-11	2	2
Problem Statement	2023-10-13	2023-10-15	3	
Related Work	2023-10-16	2023-10-19	4	
Contribution	2023-10-21	2023-10-22	2	
Document Outline	2023-10-22	2023-10-23	2	
Project Deliverables	2023-10-24	2023-10-27	4	2.4
Project Tasks	2023-10-29	2023-11-04	7	
Roles and Responsibilities	2023-11-06	2023-11-06	1	
Risk Assessment	2023-11-07	2023-11-10	4	
Cost Estimation	2023-11-11	2023-11-12	2	
Project Management Tools	2023-11-13	2023-11-13	1	2.1
Stakeholders	2023-11-15	2023-11-17	3	
Platform Requirements	2023-11-18	2023-11-20	3	
Functional Requirements	2023-11-21	2023-11-23	3	
Non-Functional Requirements	2023-11-25	2023-11-28	4	
Other Requirements	2023-11-29	2023-11-29	1	
Architectural Design	2023-12-01	2023-12-04	4	2.5
Logical Model Design	2023-12-06	2023-12-13	8	
Physical Model Design	2023-12-15	2023-12-21	7	
Total				9.3

Figure 2.1


Gp1
100%
Introduction
100%
Overview
100%
Problem Statement
100%
Related Work
100%
Contribution
100%
Document Outline
100%
Project Plan
100%
Project Deliverables
100%
Project Tasks
100%
Roles and Responsibilities
100%
Risk Assessment
100%
Cost Estimation
100%
Project Management Tools
100%
Requirements Specification
100%
Stakeholders
100%
Platform Requirements
100%
Functional Requirements
100%
Non-Functional Requirements
100%
Other Requirements
100%
System Design
100%
Architectural Design
100%
Logical Model Design
100%
Physical Model Design
100%
10
15
22
29
5
12
19
26
3
10
17
10/23
11/23
12/23

2.3 Roles and Responsibilities

Equally distributed between the members of the group.

2.4 Risk Assessment

- **Data Accessibility**

  Challenge: No issues in finding and accessing datasets.

  Relevance: Critical for model training.

  Solution: Maintain consistent efforts in data collection and stay updated on relevant datasets.

- **Challenges in Hate Speech Detection Accuracy**
  Probability: Medium

  Impact: High; low accuracy means our model is not detecting hate speech and is classifying wrongfully.

  Solution: Utilizing tools for interpreting machine learning models contribute to ongoing improvement and fine-tuning.

- **Difficulty in Aspect-Based Categorization**
  Probability: Medium

Impact: High; It is necessary for the true value of the tweet to have the highest classification probability among the other values. (ex, picking racism over sexism when it is not a sexism case.)

Solution: Utilize methods from natural language processing, regularly evaluate performance, and fine-tune categorization algorithms.

2.5 Cost Estimation

Our project operates on an efficient model that completely removes the need for external funding. We are committed to use open-source software and technologies to focus on the goals of our project and satisfy our target audience's needs.

2.6 Project Management Tools

Google docs: Used for project report and brainstorming between the team members.

TeamGantt: Used for drawing our timeline graph.

Draw.io: Used for drawing project architecture/designs.

Chapter 3
Requirements Specification

This chapter outlines the essential components and requirements necessary for the successful development and deployment of our hate speech detection system. In understanding the diverse needs and expectations of stakeholders, defining the platform on which the system operates, and specifying both functional and non-functional requirements, we lay the groundwork for a robust and effective solution.

3.1 Stakeholders

Table 3.1

Stakeholder	**Interaction with the System**	**Importance of Role**
Individual Users	Engage in online conversations, utilize the web app for real-time speech-to-text conversion, receive alerts and visual cues for hate speech detection.	Vital contributors to the system's effectiveness. Their safety and well-being during online interactions depend on the system's ability to accurately detect and categorize hate speech.
Organizations (News Channels, NGOs, Universities)	Use the app to derive meaningful insights from hate speech data. This can be applied to raise awareness, conduct studies, or take legal and safety actions based on the patterns identified.	Contribute to the broader impact of the system. Insights derived by organizations can lead to societal awareness, policy changes, or legal actions against hate speech, addressing the issue at a larger scale.
Governments, Private companies	Utilize the “firewall” feature to automatically detect and handle offensive hate speech in incoming comments or reviews.	Enhance online safety, protect citizens/users, and maintain the integrity of public and private online spaces.

3.2 Platform Requirements

Based on our experience in creating and training machine learning models, we assume the following requirements will run the application without any issues whatsoever on both server/client side.

Table 3.2

Hardware Requirement	**Recommendation**	**Reasoning**
Processor	Multi-core processor	Handling concurrent requests
RAM	8GB or higher	Efficient data processing
Storage	SSD	Faster data access

Table 3.3

Software Requirement	**Recommendation**
Operating System	Windows Server
Web Server	Apache Nginx
Database	SparQLL GraphDB MongoDB
Server-Side Scripting Language	Python

3.3 Functional Requirements

Essential Requirements table:

Table 3.4

Description	**Input**	**Process**	**Output**	**Constraints**
Hate Speech Classification	Text from speech-to-text conversion	Analyze text for hate speech categories and severity	Hate speech categories, severity level	Accuracy of classification, speed of analysis
User Alerts	Hate speech detection results	Display alerts to the user during conversations	Visual alerts, notifications	Should not interrupt natural conversation flow
Data Privacy	User interactions, reported incidents	Ensure user privacy and data security	Privacy policy adherence, secure storage	Compliance with data protection laws
Reporting Feature	Reporting Feature	Detected hate speech incidents	User-initiated reporting of hate speech	Confirmation of report submission

Recommended Requirements table:

Table 3.5

Description	**Input**	**Process**	**Output**	**Constraints**
Real-time Speech-to-Text	Audio input from the user during conversations	Convert speech to text in real-time	Transcribed text	Must support a variety of accents, background noise
User Settings	User Settings	User preferences	Customize app settings, e.g., sensitivity	Updated app behavior based on preferences
User Education	User Education	Hate speech detection results	Provide information about hate speech, its impact, and prevention	Educational content, resources

3.4 Non-Functional Requirements

The application prioritizes moderate to high performance, with a focus on efficient text input. While real-time speech-to-text conversion is a valuable feature, users primarily input data through text transcripts or sentences. In both cases, the application aims to minimize the time needed for hate speech detection. Meeting this performance target remains essential for user satisfaction and system reliability. The relationship between performance and storage limits is intertwined, affecting the amount of data that necessitates secure storage. In this context, the system should proficiently handle and securely store a minimum of 100,000 user-generated comments and their associated metadata. Storage optimization significantly influences the application's overall performance, highlighting the importance of a well-optimized system.

Maintaining a high level of **code quality** is imperative for the long-term sustainability of the project. Adhering to established coding standards and best practices ensures readability and maintainability, impacting not only the development process but also the system's reliability and scalability. A clean and well-organized codebase facilitates easier troubleshooting and future enhancements, directly contributing to the app's overall quality.

Comprehensive documentation, including user manuals and technical documentation, is vital for the usability and maintainability of the application. Clear documentation not only aids users in understanding and utilizing the app effectively but also supports developers in maintaining and upgrading the system over time. This impacts the overall user experience and reliability of the application.

Accessibility is a fundamental aspect of the user interface, ensuring that all users can interact with the app effortlessly. The relationship between accessibility and user interface ease-of-use is evident, as designing an interface that adheres to basic accessibility guidelines contributes to a user-friendly experience.The user interface should consider basic accessibility guidelines to ensure that users with disabilities can use the app effectively.

Security measures play a crucial role in safeguarding user data. Implementing basic encryption for user data is directly related to the reliability of the system, assuring users that their information is protected. Additionally, the periodic review of hate speech detection algorithms for security enhances the overall trustworthiness of the application.

Reliability is a key factor influenced by various aspects, including performance, code quality, and security. Achieving a system uptime of at least 98% is contingent on the robustness of the underlying infrastructure, the efficiency of hate speech detection algorithms, and the overall quality of the codebase.

Scalability is closely tied to both performance and user load. The hate speech detection system's ability to handle a concurrent user load of 1,000 users without significant degradation in performance directly relates to the overall scalability of the application. An efficient and scalable system can accommodate growing user demands without compromising performance.

Portability, the compatibility of the app with major operating systems, enhances its reach and user base. This factor is intertwined with user interface ease-of-use, as a consistent and user-friendly interface across different platforms contributes to a seamless user experience.

In conclusion, Performance impacts storage limits, code quality affects reliability and scalability, accessibility is tied to the user interface, and security measures contribute to overall system trustworthiness.

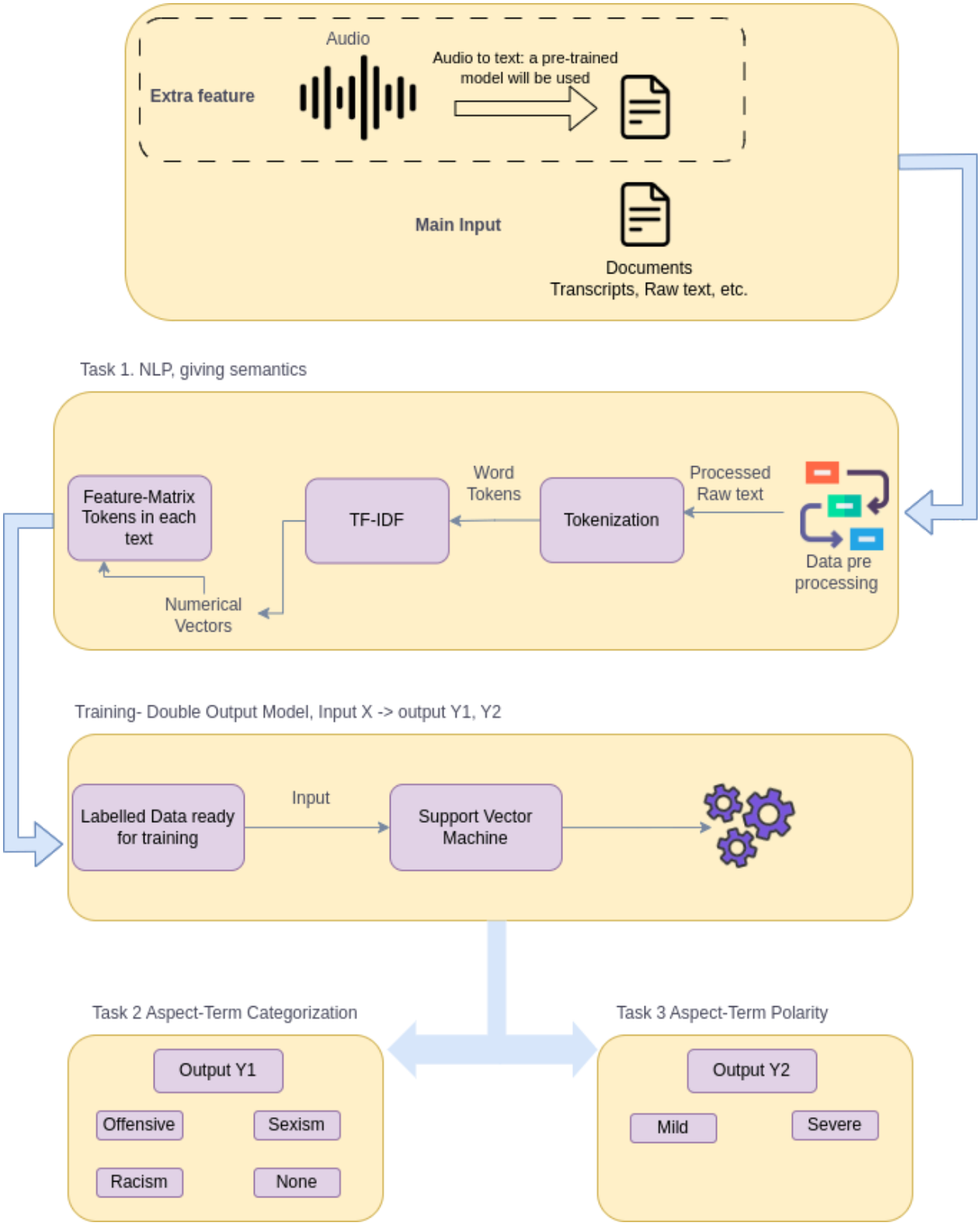
Chapter 4
System Design

This chapter explores the design of our hate speech detection system. A thoughtful design is crucial for turning our identified requirements into a strong and scalable solution. In this chapter, we thoroughly examine the architectural and logical aspects of the system, and we use visual aids like use case diagrams, deployment diagrams, sequence diagrams, and a physical model design to illustrate these aspects.

4.1 Architectural Design:

The following diagram explains the conditions and actions that have to be met for the project to be successful, showing a step by step path between the different components from input to output.

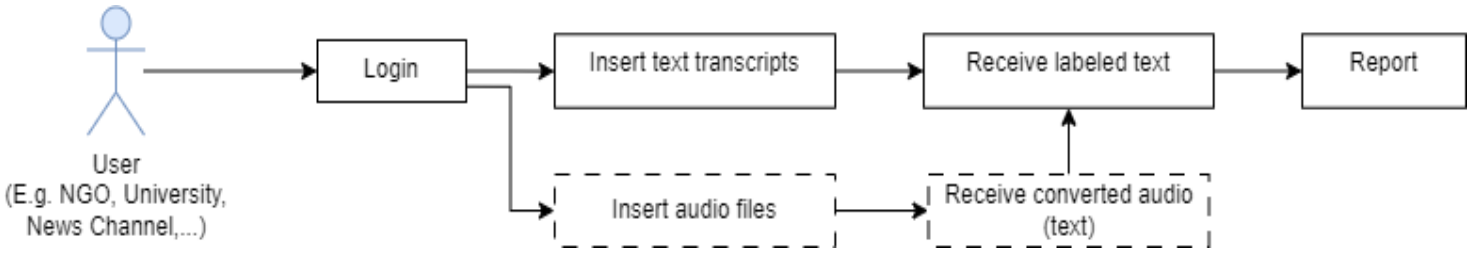
Figure 4.1 Architecture Design



4.2 Logical Model Design

In this section, we will be aiming to establish a robust and coherent blueprint for the internal workings of the software using several diagrams such as use case diagrams, sequence and deployment diagrams.

Figure 4.2 Use Case Diagram



The diagram above shows the main steps of how a user interacts with our model through the web application. The procedure starts with user login and authentication, then the user inserts text transcripts, which are needed to be classified as one of the categories of hate speech in our model. After that, the user receives a copy of the classified and labeled text and can continue to report hate speech if there is any. The diagram also shows an extra feature, which is inserting audio files instead of text transcripts which can then be converted into text to be classified.

Figure 4.3

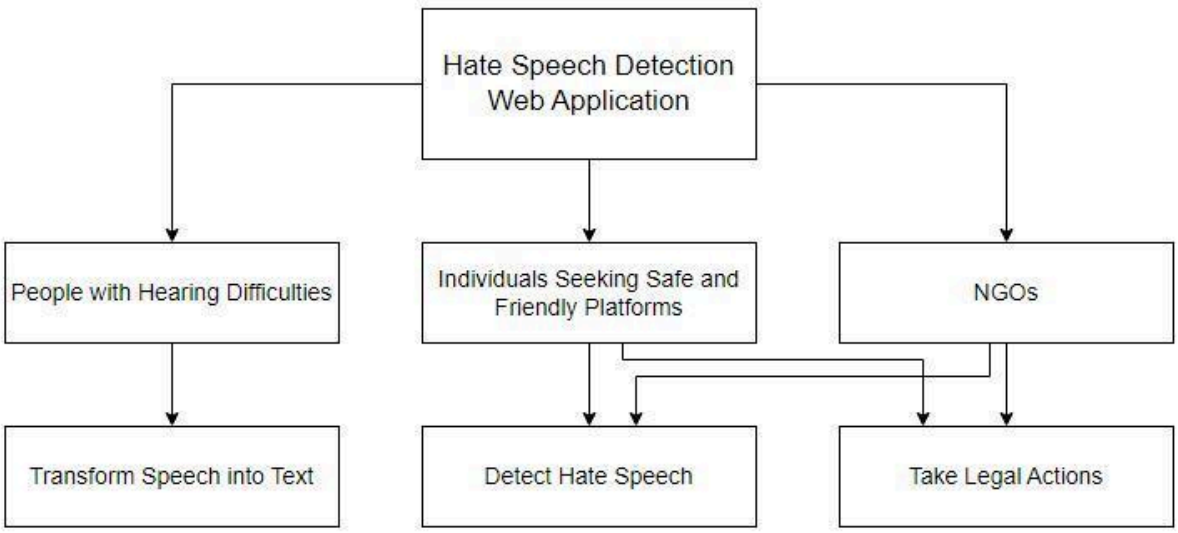


Figure 4.4 Deployment Diagram

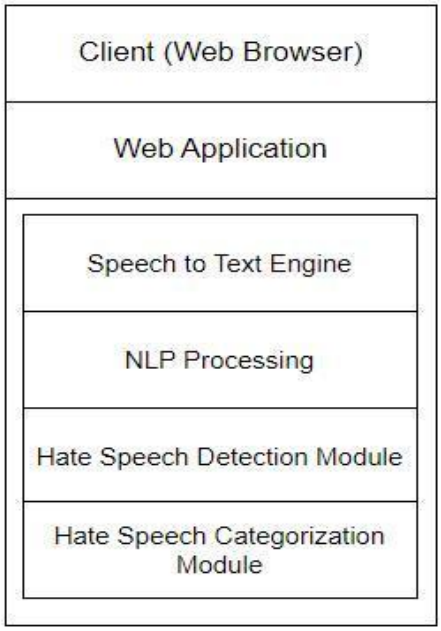
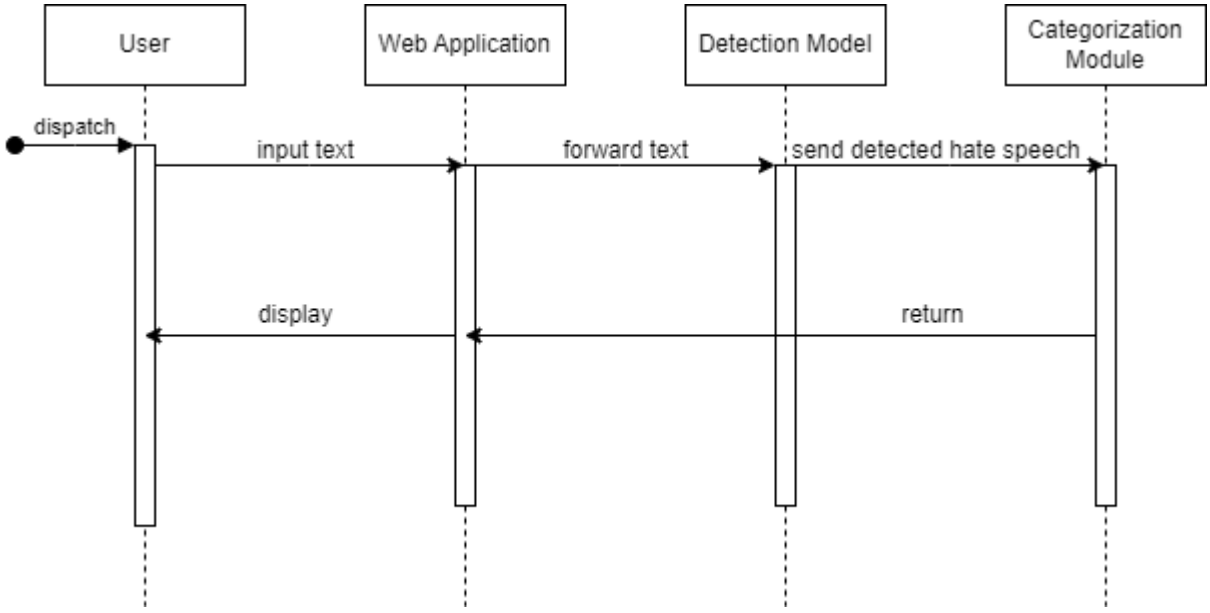


Figure 4.5 Sequence Diagram.



The sequence diagram above shows the flow of the data from the user to the categorization module and back to the user. The user inserts text transcripts to the web application, then the application forwards them to the hate speech detection

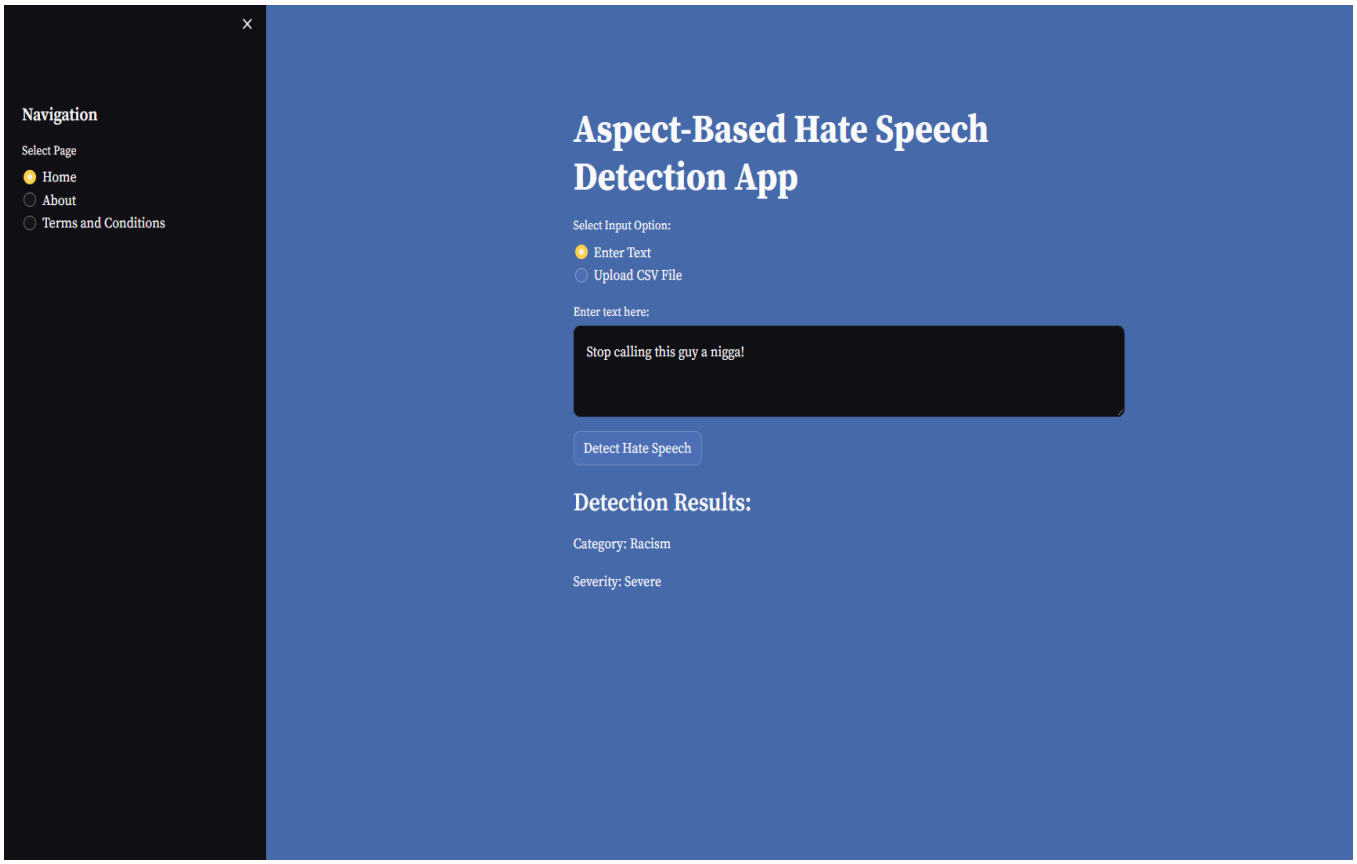
module then to the categorization module to be categorized, then the output which is the labeled and categorized texts are returned to the user.

4.3 Physical Model Design

For the interface design, we are planning to use Streamlit.[1] Streamlit is an open-source python library that helps in creating web applications.

System physical design may include the following as needed:

Figure 4.6 Pilot UI Design



[1] https://streamlit.io/

- Database design

Chapter 5
Data Preprocessing

~~This chapter is needed for data driven projects.~~

~~This chapter should describe issues related to the data ETL pipeline, including data extraction, data engineering, feature engineering, and the loading processes. The code functions related to this chapter should be provided in the Appendix. You need to refer to the line of codes while you describe your data pipeline in the following sections.~~

In this chapter, we will delve into the critical phase of data preprocessing, which forms the backbone of our hate speech detection application. We will discuss how we collected the data providing a sample of it along with the metadata. We will also explain the main data preprocessing techniques we used to ensure the data is clean whilst providing examples of the data before and after cleaning. Another section of this chapter will cover the feature engineering part of our project and explain how we stored the prepared data.

5.1 Data Collection and Description

~~In this section, you need to provide a detailed description of the data source and data collection process. You need to provide the metadata of your data and show a sample of the data.~~

Data source: The data for this project was collected from different datasets we found on Kaggle, Hugging Face and previous work on hate speech detection.

Data collection: The dataset we are using is a collection of the different datasets. We merged datasets and collected tweets and manually annotated some to make sure the data is balanced for efficiency.

Data description:

- Our dataset consists of 7 columns:
    - tweet: contains the tweet text
    - sexist: 1 if sexist, 0 if not sexist
    - racist: 1 if racist, 0 if not racist
    - offensive: 1 if offensive, 0 if not offensive
    - none: 1 if text has no hate speech, 0 if it has
    - class: 0 if none, 1 if sexist, 2 if racist, 3 if offensive

- Here are some examples from the dataset we created:

tweet	sexist	racist	offensive	none	class
wait your black your life dont matter	0	1	0	0	2
But first.... That bitch better get her ass in the kitchen and make me a sammich.	1	0	0	0	1
You hate football your a faggot	0	0	1	0	3
I sho do miss my Uncle Tom	0	0	0	1	0

5.2 Data Profiling and Engineering

~~This section should show a summary of the data quality issues and data cleaning. You should provide a clear justification of the techniques used. You need to provide examples of sample data before and after the data-cleaning process.~~

In this section, we discuss the data quality issues detected during the initial data profiling phase, as well as the approaches utilized to clean and preprocess the dataset. We provide justification for the approaches we picked and demonstrate their effectiveness with examples of data before and after cleaning.

Data Quality Issues:

We identified the following data quality issues:

- **Entities and Special Characters:** Presence of usernames, URLs, emojis, and special characters that do not contribute to the semantics of the text could affect model training.
- **Inconsistent Text Formatting:** Variations in text formatting, such as inconsistent use of punctuation, could impact the text processing pipeline.

Data Cleaning Techniques:

To address these issues, we applied several data cleaning and preprocessing techniques:

- **Removal of Entities and Special Characters:**

- **Justification**: Usernames, URLs, and special characters can be considered noise as they do not add semantic value.
- **Technique**: We defined a function to remove usernames, URLs, and special characters from the text.

- **Tokenization (BERT Tokenizer):**
  - **Justification:** Tokenization converts text into a format suitable for model input.
  - **Technique:** We used the pre-trained BERT tokenizer called 'bert-base-uncased' to convert all the text to lower case then tokenize the text.

- **Padding and Masking:**
  - **Justification:** Padding ensures uniform input length, and masking helps the model differentiate between actual data and padded elements.
  - **Technique:** The BERT tokenizer automatically handles padding and masking as part of its tokenization process.

- **Conversion to PyTorch Data Types:**
  - **Justification:** PyTorch requires specific data types for tensors to facilitate training and inference.
  - **Technique:** The tokenized and preprocessed data were already in PyTorch tensor format due to the tokenizer settings.

Here are some examples from the dataset we used before and after cleaning:

Before	After
@Ncoleycole y u not on the monkey bars too lol	y u not on the monkey bars too lol
RT @tbhjuststop: my score on flappy bird is higher than the broncos score right now	my score on flappy bird is higher than the broncos score right now
black peoples call it hood bc they don&#65533;t know how too spell neighbor	black peoples call it hood bc they don t know how too spell neighbor

5.3 Feature Engineering

This section should show the feature preparation process, including feature extraction, feature rescaling, feature selection, and dimensionality reduction. You should provide a clear justification of the techniques used. You need to summarize the extracted features and provide examples of the prepared features.

5.4 Data Loading

This section should summarize the loading process of the prepared data in the data storage.

Chapter 6

Implementation

6.1 General Implementation Description

Programming Languages, Tools, and APIs Used

- **Python:** Chosen for its versatility and extensive libraries for machine learning and natural language processing.

- **Libraries:**
    - **numpy:** Utilized for numerical computing and array manipulation.
    - **pandas:** Used for data manipulation and preprocessing.
    - **os:** Facilitates interaction with the operating system, including file path handling.
    - **re:** Employed for regular expressions in text preprocessing.
    - **PyTorch:** Selected for building and training the neural network due to its flexibility and support for GPU acceleration.
    - **joblib:** Utilized for model persistence, enabling saving and loading Python objects.
    - **Transformers (Hugging Face):** Chosen for BERT model implementation and tokenization, leveraging state-of-the-art NLP models.
    - **matplotlib:** Used for data visualization and plotting to support exploratory data analysis.
    - **seaborn:** Complements matplotlib for advanced data visualization.
    - **scikit-learn:** Provides tools for data splitting, preprocessing utilities, and evaluation metrics.

- **Streamlit:** Used to create the web application interface, allowing users to input text and see the results of hate speech detection.
- **Google Sheets and Docs:** Utilized for report creation and dataset management.

Overview of Implementation:

- **Classes and Scripts:** The project consists of several scripts and classes, each serving a specific purpose in the data pipeline and model training/testing phases. The main scripts include data preprocessing, model training, evaluation, and web application deployment.
- **Lines of Code:** The project comprises approximately 2000 lines of code, structured across multiple scripts.
- **Coding Conventions:** We adhered to PEP 8 coding conventions for Python to ensure readability and maintainability. This includes consistent indentation, meaningful variable names, and modular code design.

6.2 Pipeline Implementation Description

Data Pipeline Structure and Integration:

The data pipeline involves several key stages:

1. **Data Collection and Preprocessing:**
    - **Loading Data:** The dataset is loaded from CSV files using pandas.
    - **Text Cleaning:** Regular expressions (re) are used to remove unwanted characters, URLs, and specific patterns like 'RT' (retweet notation).
    - **Tokenization:** Sentences are tokenized using the BERT tokenizer from the Hugging Face Transformers library.
2. **Model Training:**
    - **Model Initialization:** The BERT model is initialized for sequence classification using PyTorch.
    - **Training Loop:** The model is trained over multiple epochs, with gradient computation and optimization performed using AdamW optimizer and a learning rate scheduler.
    - **Evaluation:** Validation is performed after each epoch to monitor performance and prevent overfitting.
3. **Model Testing:**
    - **Data Preparation:** Test data is encoded and prepared using the same tokenization and tensor conversion process as the training data.
    - **Prediction Loop:** The model predicts labels for the test set in evaluation mode to prevent gradient computation.
    - **Metrics Calculation:** Predictions are evaluated using accuracy, Matthews Correlation Coefficient (MCC), and classification reports from scikit-learn.
4. **Deployment:**
    - **Streamlit Application:** A user-friendly interface is created using Streamlit to allow users to input text and see the detection results.

APIs and Technologies:

- **Hugging Face Transformers API:** Provides pre-trained BERT models and tokenizers.
- **PyTorch:** Offers the neural network framework for model development.
- **scikit-learn:** Supplies tools for data splitting and evaluation metrics.

6.3 Model Implementation

Model Choice and Rationale:

- **BERT (Bidirectional Encoder Representations from Transformers):** Selected for its superior performance in NLP tasks. BERT's bidirectional approach allows it to capture context from both directions, making it highly effective for tasks like text classification.

Model Development and Hyperparameter Tuning:

Throughout our project, we have implemented and tested several different models, ranging from support vector machines to different types of neural networks etc…

Classification Model	Online Accuracy	Our Accuracy	
Naive - Bayes			
Support Vector Machines (SVM)	76%	79%	
SVM hyperparameters	- -	85%	
Recurrent Neural Networks (RNN)	77%	77%	
Bi-Directional RNN	79%	80%	
LSTM	74%	76%	
12-Layer Bert	–	94%	

- **Initialization:** The BERT model is initialized with a classification head on top, suitable for binary classification.
- **Training Setup:**
    - **Optimizer:** AdamW is used for its efficiency in handling large-scale datasets and complex models.
    - **Learning Rate Scheduler:** Adjusts the learning rate during training to enhance convergence.
- **Hyperparameters:** Tuned parameters include learning rate, batch size, and number of epochs, determined through experimentation and cross-validation.

6.4 Additional Implementation Details

Critical Algorithms and Techniques:

- **Text Preprocessing:** Custom function using regular expressions to clean and standardize text data.

  def strip_all_entities(text):

  return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)|(RT)", " ", text).split())

  Data Encoding:

- **BERT Encoding Function:**

  def bert_encode(data, max_len):

  input_ids = []

  attention_masks = []

  for sentence in data:

  encoded = tokenizer.encode_plus(

  sentence,

  add_special_tokens=True,

  max_length=max_len,

```
        padding='max_length',
        truncation=True,
        return_attention_mask=True
    )
    input_ids.append(encoded['input_ids'])
    attention_masks.append(encoded['attention_mask'])
return np.array(input_ids), np.array(attention_masks)
```

Training Loop:

- **Main Idea: Iterate over epochs and batches, compute loss, backpropagate, and update model parameters.**
- **Pseudo-code:**

```
for epoch in range(epochs):
    model.train()
    total_loss = 0
    for batch in train_dataloader:
        b_input_ids, b_input_mask, b_labels = tuple(t.to(device) for t in batch)
        model.zero_grad()
        outputs = model(b_input_ids, attention_mask=b_input_mask, labels=b_labels)
        loss = outputs[0]
        loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
        optimizer.step()
        scheduler.step()
```

total_loss += loss.item()

avg_train_loss = total_loss / len(train_dataloader)

Implemented Features:

- Data Cleaning and Preprocessing: Implemented.
- Model Training and Evaluation: Implemented.
- Web Application for Prediction: Implemented.

Deferred Features:

- Real-time Data Stream Integration: Deferred.
- Advanced Hyperparameter Tuning: Deferred for future iterations.

Chapter 7
Testing

7.1 Testing Approach

Discuss the experiment design, including:

- Evaluation measures
  The three evaluation measures used were:

  1-**Accuracy** :was computed to understand the overall effectiveness of the model in correctly predicting the class labels.

  2-**Matthew's Correlation Coefficient (MCC): it** provided a balanced measure that can effectively handle class imbalances.

  3-**Precision, Recall, and F1-Score :** calculated for each class to scale the model and to identify each class correctly.

- Model testing approach

  We used 'BertForSequenceClassification' , it's a a pretrained BERT model with a single linear classification layer on top, trained with binary labels and evaluated after each epoch using the validation set to tune the model parameters effectively.

- Data splitting technique used.

  The dataset was split into 50% training, 25% validation, and 25% testing using train_test_split from sklearn,

- Pipeline testing approach.

  The pipeline testing covered every phase of the pipeline for handling and processing data, beginning with batch preparation, tokenization, preprocessing, and data loading.

  Data Loading and Preprocessing: Verified that the preprocessing operations (such as removing unwanted characters from tweets) were applied consistently throughout the dataset and that the data had loaded accurately from the source.

  Tokenization: To make sure the BERT tokenizer reliably transforms text into the right format (tokens) needed by the model, it was put to the test. For the input data to remain intact, this step was essential.

  Batch Preparation: The process of creating data batches was tested to ensure that the right combination of inputs, masks, and labels were used to form the batches, and that the batches were compatible with the input requirements of the model.

- Complete system testing approach

  System performance was regularly monitored for any potential issues during training phases, which are :

  Integration testing :it was carried out to make sure that data loading, preprocessing, model training, and prediction all function together. This involved making sure that the output from one stage correctly feeding into the next and testing the data flow through the pipeline.

  Performance Testing: During the training phase, the system's performance was closely observed, with a particular emphasis on memory usage and processing speed. This was essential for maximizing the use of resources, particularly during the resource-intensive BERT model's training.

  Testing for Scalability: We wanted to see how well the system handles growing data volumes. In order to do this, the dataset size had to be gradually increased by using more than 1 dataset or adding more data,response was tracked, especially how long training and prediction took.

7.2 Testing Results

- The BERT model achieved an accuracy of 94% on the test set, indicating strong predictive performance.
- The Matthew's Correlation Coefficient (MCC) was 0.849, showing good quality predictions across both classes.
- Precision and recall scores were high, with an F1-score of 0.96 for class 0 “non-hate speech”and 0.89 for class 1 “hate speech”.

  **The model's training loss decreased significantly across epochs, from 0.40 in the first epoch to 0.05 in the final epoch, showing effective learning rate and optimization.

  **Validation accuracy improved from 91% in the first epoch to 96% in the final epoch, which reflects reliability in classifying unseen data correctly.

7.3 Discussion (optional)

Discuss any insights and test results that have impacted the design or implementation of the system.

1. ~~Model Generalization: The training and validation results not only demonstrated that the model was not overfitting but also highlighted its ability to generalize well to unseen data. The incremental improvements in both training loss and validation accuracy across epochs affirm that the model was learning general patterns rather than memorizing the training data, which is crucial for its deployment in dynamic real-world environments.~~
2. ~~Balanced Performance Across Classes: The high Matthew's Correlation Coefficient (MCC) score is particularly significant in the context of hate speech detection. MCC provided a more truthful measure of the binary classification performance, especially useful when the classes are imbalanced. This balanced performance is essential for ensuring that the model responds with high sensitivity to instances of hate speech while maintaining a low rate of false positives, which are critical in avoiding the censorship of benign content.~~

3. ~~Efficient Handling of Diverse Data: The robustness of the preprocessing and tokenization pipeline was validated through the use of mock data, which included extreme and edge cases. This ensured that the model's performance remains stable regardless of variations in data input, which is typical in social media settings where new slang, languages, and symbols frequently emerge.~~

Chapter 8
Conclusions and Future Work

~~Summarize any results achieved in this project and discuss how you intend to extend the project in the future.~~

8.1 Conclusions

In a world where digital communication is becoming one of the main ways of human interaction, the need for maintaining a safe and friendly environment is increasing. This project successfully developed an Aspect-Based Hate Speech Detection application that detects hate speech and categorizes it into four main categories: racism, sexism, offensive, none. A pre-trained BERT model was utilized for the categorization and the model was deployed using Streamlit to create a web application.

Key results from the project include:

- Model Performance: —----------
- User Interface: The Streamlit-based web application provided a user-friendly interface for users to input text and receive hate speech classification.

8.2 Future Work

While the project has achieved its initial goals, several avenues for future work can enhance its capabilities and expand its applications:

- Integration of Audio-to-Text for Hate Speech Detection:
    - We would like to expand the current text-based detection system to include audio inputs by including voice-to-text technologies to transcribe audio content into text that can then be examined by existing hate speech recognition models. This update would enable the application to detect hate speech in audio forms, increasing its versatility and applicability to a wider range of media types.
- Real-Time Detection:
    - We would love to create a real-time hate speech identification system by minimizing model inference time. This update would allow the program to be utilized in real-time monitoring scenarios, such as live broadcasts or social media streams, with rapid feedback and intervention options.

- Firewall Security Layer for Media Platforms:
    - We are interested in deploying the hate speech detection system as a security layer for news channels and social media platforms by collaborating with them to integrate the detection system, as well as developing automated moderation tools that can flag or block hate speech content. This application would assist media platforms in preventing the spread of hate speech, improving the safety and friendliness of online environments.
- Text Summarization Web Extension:
    - We intend to construct a web browser extension that would help summarize articles or text and highlight or emphasize hurtful words and hate speech. This would help save researchers' time and effort as they read the summary and focus on negative comments if that is their objective.
- Continuous Model Improvement:
    - We hope to constantly enhance the model's robustness and accuracy by adding more varied and nuanced examples of hate speech to the training dataset and by utilizing user feedback to retrain and fine-tune the models. By keeping the models current with changing linguistic conventions and new types of hate speech, this method keeps the models effective over time and improves their capacity to recognize and reduce harmful content.

Appendix A
Users' Manual

Provide a description (textual and pictorial) of how the system can be installed and used and how the errors messages should be interpreted.

Installation Guide

Installation Steps:

1.Install Streamlit:

-Ensure you have Python installed, then install Streamlit using:

pip install streamlit

2. Download the Application Code:

-Clone or download the code from your repository:
https://github.com/Mo-Malalha/AB-Hate-Speech-GP2.git

3.Run the Application:

-Navigate to the application directory and run:

streamlit run "W:/Anaconda/Hate Speech/HateSpeechTest.py"

-This command will start the server and open the application in your default web browser.

Using the Application

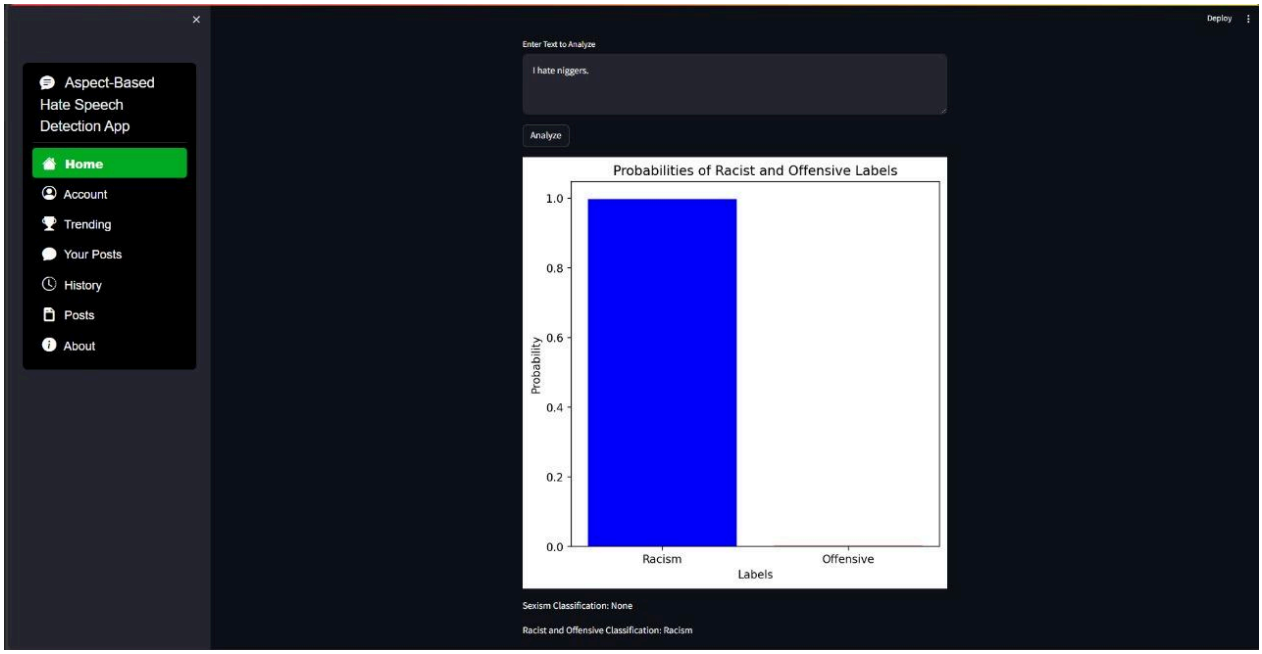
(screenshots are provided below each step)

Navigating the Interface:

A.Home Page→ Text Input for Analysis:

- Users can directly input text into the provided text box for hate speech analysis.
- After entering the text, clicking the "Analyze" button will classify the text and display the probabilities of various hate speech categories(Sexism , Offensive ,or Racism).

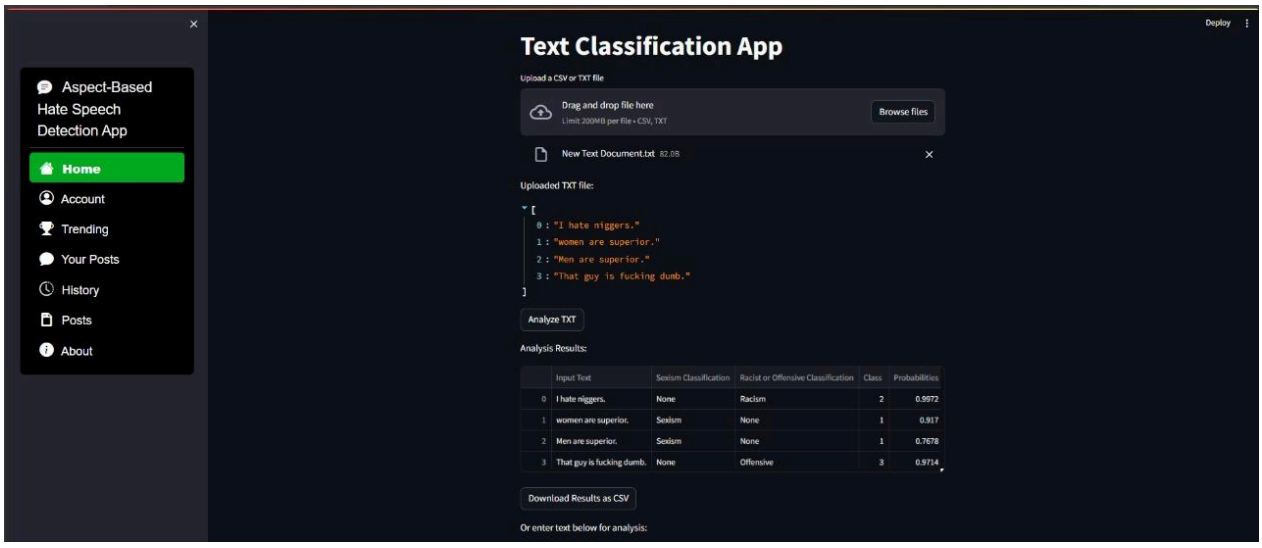
Example:



The result shows a bar chart representation of the classification probabilities for categories like "Racism" or "Offensive".

2. Uploading a Text File for Bulk Analysis :
    - Users can also upload a text file containing multiple entries for batch analysis.
    - The application processes the uploaded file and displays the classification results for each line of text directly in the interface.
    - There is an option to download these results in a CSV format for further analysis or reporting.

Example:



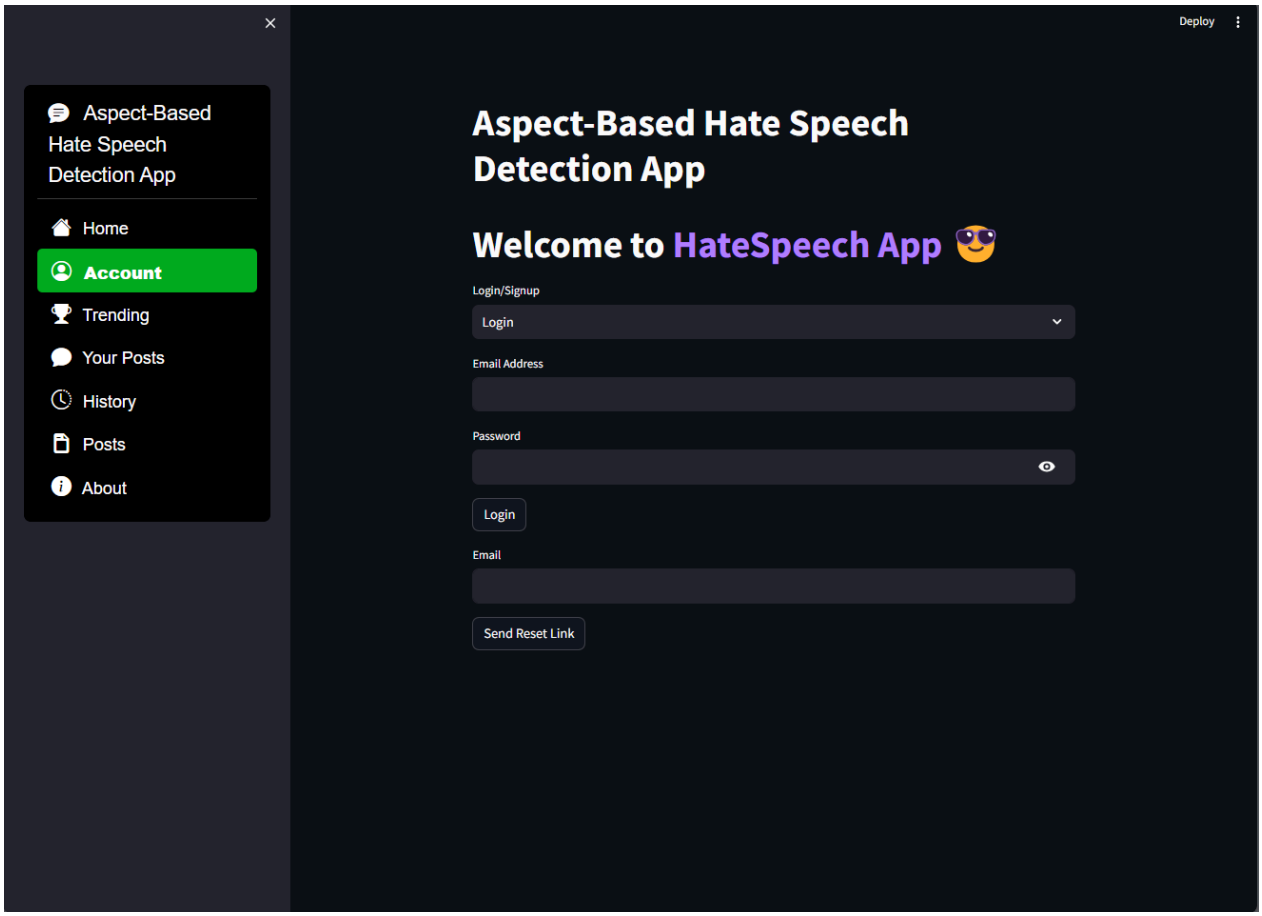
3. Exporting Results to CSV:
    - After analyzing text through the file upload feature, users can download the results as a CSV file.
    - The CSV file includes detailed columns such as input text, classification type, and probability scores, making it easy to review and utilize the data outside the application.

Example:

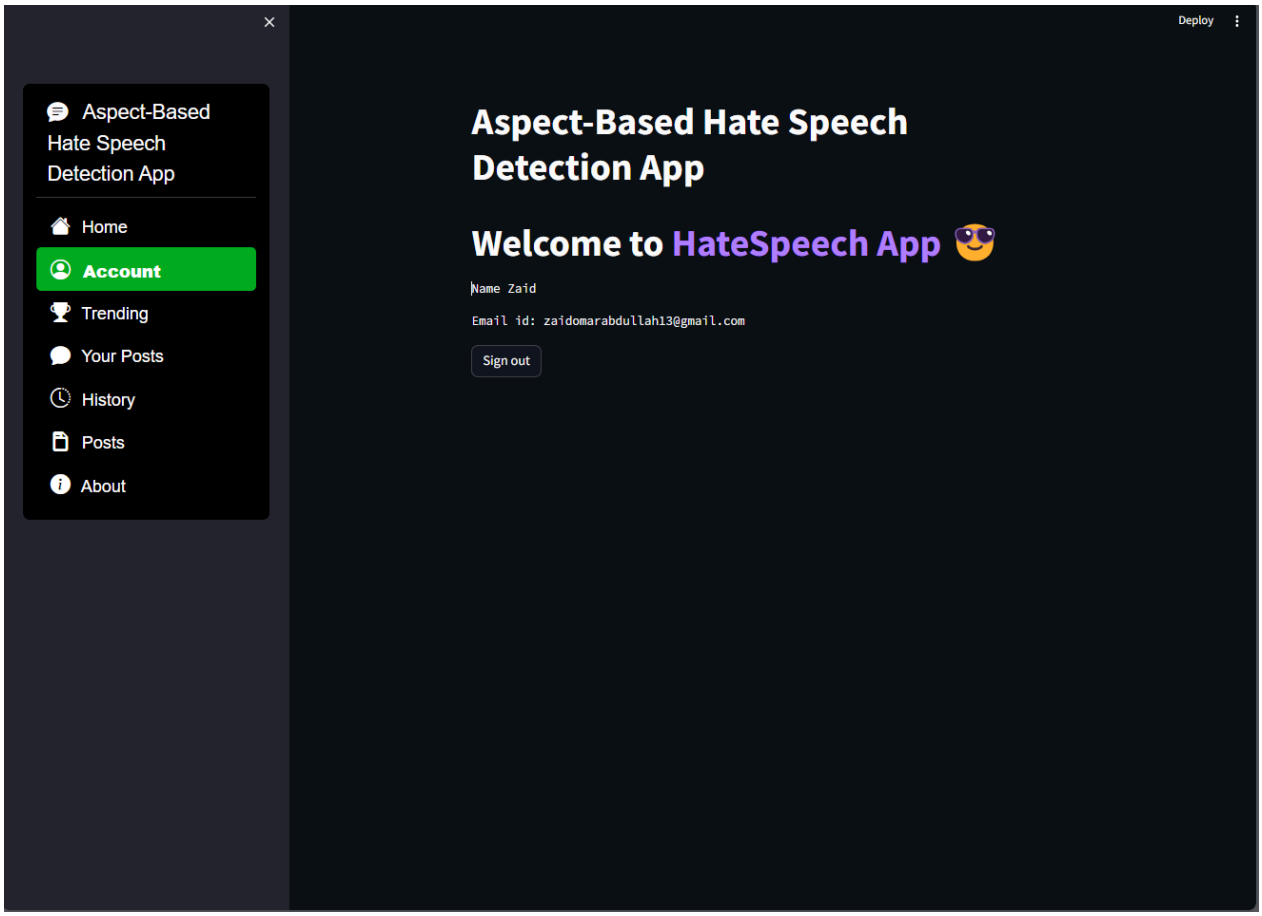
	A	B	C	D	E
1	Input Text	Sexism Classification	Racist or Offensive Classification	Class	Probabilities
2	I hate niggers.	None	Racism	2	0.997226417
3	women are superior.	Sexism	None	1	0.916957438
4	Men are superior.	Sexism	None	1	0.767817259
5	That guy is fucking dumb.	None	Offensive	3	0.971394241

B.Account: Sign in or create a new account to personalize your experience and to be able to post your thoughts. Users can also reset passwords here.

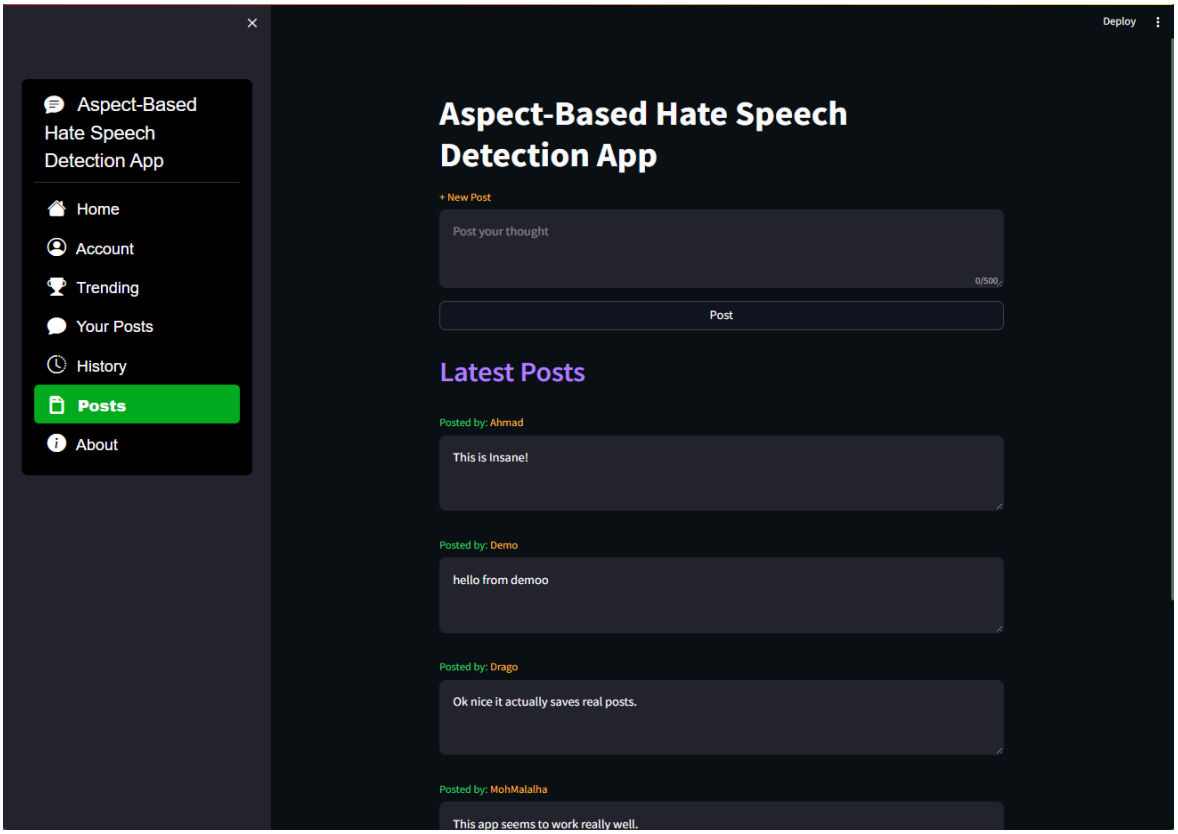
Before Sign In:



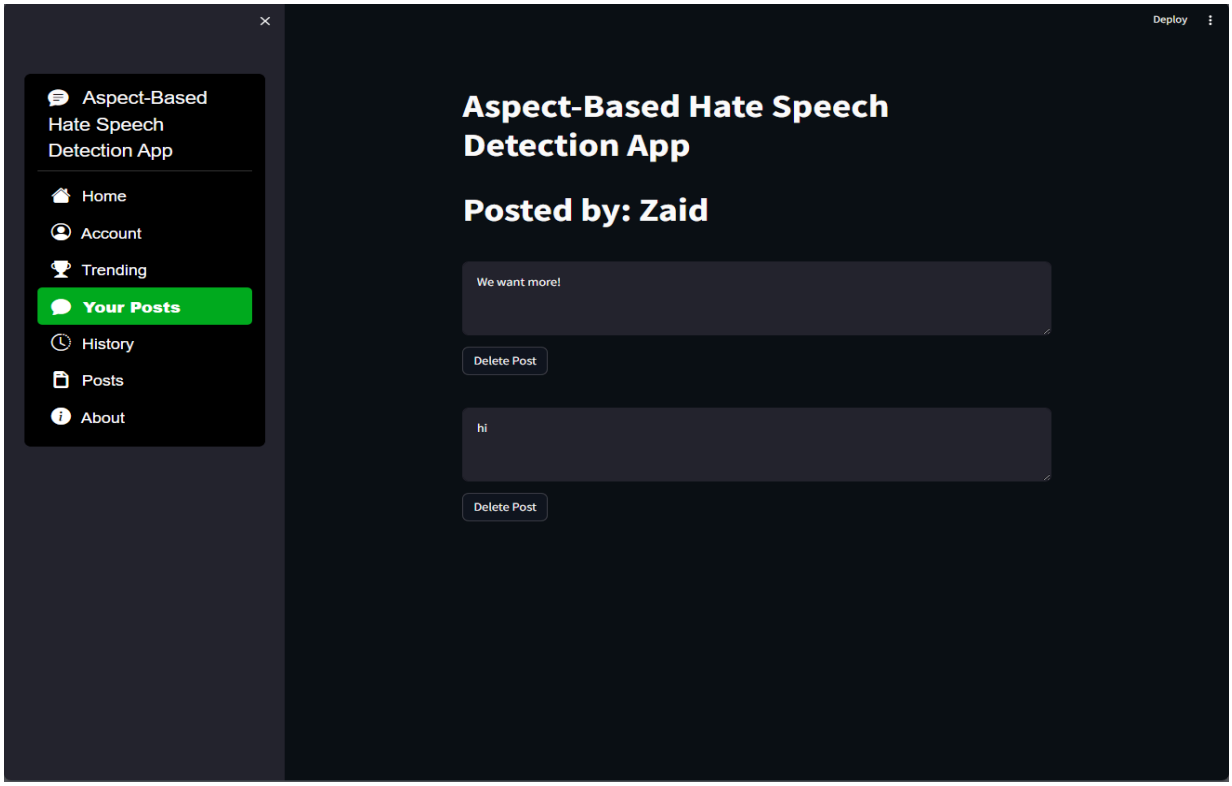
After Sign In:



C.Posts: Post your thoughts and see what others are sharing.



D.Your Posts: View and manage your posts. Users can delete their posts from this section.



E.History: Check your recent activity and previously analyzed texts.

F.About: Learn more about the application and its purpose.

Error Messages and Troubleshooting

Here are the common problems you might face and a solution to each one of them:

1. Failure to Start Streamlit:
    - Error Message: "streamlit command not found"
    - Solution: Make sure Streamlit is installed correctly using pip install streamlit. Check your Python environment if necessary.
2. Application Doesn't Load:
    - Error Message: "Error running app"

- Solution: Ensure all required libraries are installed. Check for any errors in the terminal that indicate missing files or permissions.

3. Connection Issues:
    - Error Message: "Connection Error" on login or post submission.
    - Solution: Verify network connectivity. Ensure that Firebase services are not blocked by a firewall or network policy.

Appendix B
Document Changes

~~Discuss here any changes you have made to the document you have submitted in Project 1 and the reasons behind the changes.~~

The idea of classifying the polarity of hate speech into severe or mild was abandoned as we think that the presence of hate speech is enough for us to take action as the polarity won't affect our decision in reporting; therefore, any sections related to classifying hate speech polarity as mild or severe were removed from the document.

Instead of using SVM and LSTM for classifying hate speech into our four categories, we decided to use pre-trained BERT models as it outperformed the previous models by a lot, providing better results and accuracy.

Appendix C
Code Documentation

List the code documentation. Such a documentation can usually be automatically generated using tools like JavaDoc and Doxygen.

References

Use any referencing style/standard and be consistent. Consider the IEEE citation style. IEEE citation style includes in-text citations, numbered in square brackets, which refer to the full citation listed in the reference list at the end of the paper. The reference list is organized numerically, not alphabetically.

[1] Zeerak Waseem and Dirk Hovy. 2016. Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93, San Diego, California. Association for Computational Linguistics.

[2] cj adams, Jeffrey Sorensen, Julia Elliott, Lucas Dixon, Mark McDonald, nithum, Will Cukierski. (2017). 2018 Toxic Comment Classification Challenge | Kaggle. (Kaggle competition and we will compare with the top-voted entries).

[3] Automated Hate Speech Detection and the Problem of Offensive Language Thomas Davidson, Dana Warmsley, Michael Macy, Ingmar Weber

[1703.04009] Automated Hate Speech Detection and the Problem of Offensive Language (arxiv.org)

[4] Exploring Hate Speech Detection in Multimodal Publications ***Raul Gomez, Jaume Gibert, Lluis Gomez, Dimosthenis Karatzas***; Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2020, pp. 1470-1478

[5] View of Islamophobia Content Detection Using Natural Language Processing (jcbi.org) ,Abdul Jaleel, Mehmoon Anwar, Farooq Ali, Raza Mukhtar, Muhammad Farooq.