Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Andreas Aarrestad
Santhosh Shanmugam

# An Applied Approach to Machine Learning for Hate Speech Management on Social Media Platforms

Master's thesis in Communication Technology
Supervisor: Sule Yildirim Yayilgan
Co-supervisor: Sarang Shaikh
June 2023

NTNU
Norwegian University of
Science and Technology

Andreas Aarrestad
Santhosh Shanmugam

# An Applied Approach to Machine Learning for Hate Speech Management on Social Media Platforms

Master's thesis in Communication Technology
Supervisor: Sule Yildirim Yayilgan
Co-supervisor: Sarang Shaikh
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

# An Applied Approach to Machine Learning for Hate Speech Management on Social Media Platforms

**Andreas Aarrestad and**
**Santhosh Shanmugam**

| | |
|---|---|
| Submission date: | June 2023 |
| Main supervisor: | Sule Yildirim Yayilgan, NTNU |
| Co-supervisor: | Sarang Shaikh, NTNU |

Norwegian University of Science and Technology
Department of Information Security and Communication Technology

| | |
|---|---|
| **Title:** | An Applied Approach to Machine Learning for Hate Speech Management on Social Media Platforms |
| **Students:** | Andreas Aarrestad and Santhosh Shanmugam |

**Problem description:**

This thesis aims to analyze the practical usage of internal resources on social media platforms to detect and mitigate hateful content. Hate speech is a complex and nuanced issue that is influenced by a variety of contextual factors, such as the social norms of the platform's user base, the speaker's language, and the relationship between the speaker and the target of the hate speech. Existing research has paid less attention to contextual information, instead focusing on developing generalized models that are applicable across multiple platforms and use cases. A hate speech detection model based solely on text might be limited in identifying unique hateful content as a significant portion of hate speech, it is distinct and it lacks common characteristics. Consequently, such a model may not meet the required classification performance for effective deployment on a platform, leading to imprecise mitigation strategies.

The deployment of a system on a social media platform that detects hate speech involves making a set of careful considerations. The system must ensure scalability, robustness against adversarial attacks, and overall cost-effectiveness while navigating various legal and ethical considerations such as protection of free speech and compliance with differing hate speech laws across jurisdictions. Deploying such a system also presents the challenge of integrating the detection model into an adaptable architecture to accommodate specific use cases, demanding input and engagement from domain experts.

By conducting this research, we will leverage contextual data to detect and mitigate hateful content on social media platforms while exploring potential use cases for the proposed detection model and ensuring that it meets relevant system requirements.

| | |
|---|---|
| **Approved on:** | 2023-02-21 |
| **Main supervisor:** | Sule Yildirim Yayilgan, NTNU |
| **Co-supervisor:** | Sarang Shaikh, NTNU |

# Abstract

The rise of hate speech on social media platforms has led to a growing need for effective automatic hate speech management systems that can identify and mitigate hateful content. While significant progress has been made with recent advancements in natural language processing, there is a need for continuous system improvement in response to government regulations, new forms of hate speech, and evolving user behavior. The confidential nature of proprietary algorithms used by social media companies complicates these efforts, as it often restricts the depth of research insights and hinders the open sharing of advancements. Simultaneously, contemporary publically available research has primarily been focused on improving the detection algorithms in simplified environments, often neglecting to consider the broader complexities, such as incorporating contextual information and adapting the supporting architecture surrounding the detection model. In response, this thesis analyzes strategies for developing a system using machine learning techniques to optimize adaptability, scalability, robustness against adversarial attacks, transparency, legal compliance, and auditability in addition to solely the performance. We offer a multifaceted approach to developing a hate speech management system, exploring a variety of strategies drawn from an extensive literature review and hands-on experimentation. By conducting a comprehensive literature review, the thesis formulates strategies regarding the standardization of input to ensure robustness, leveraging adversarial examples during language model fine-tuning, and employing persistent monitoring using XAI techniques. The thesis emphasizes the advantages of utilizing GPT-based models for hate speech annotation, achieving performance levels close to human annotation while significantly reducing time and cost. Moreover, we underscore the benefits of incorporating contextual information as features of a hate speech detection model. Additionally, the thesis highlights the advantages of prioritizing uniqueness and uncertainty when selecting samples for sequential fine-tuning of the model, improving the performance compared to random sampling. Finally, we introduce a triage strategy that adaptively classifies instances using models of varying complexity, depending on the inherent characteristics of each instance. Finally, the thesis integrates all these strategies into a cohesive system architecture.

# Sammendrag

Fremveksten av hatytringer på sosiale medier har ført til et økende behov for effektive automatiske systemer som kan identifisere og fatte tiltak mot hatefullt innhold. Selv om det er gjort betydelige fremskritt innen språkbehandling, er det et stort behov for videre utvikling av systemer som addresserer statlige reguleringer, nye former for hatytringer, og endrende brukeratferd. Den konfidensielle karakteren til proprietære algoritmer som brukes av sosiale medier kompliserer denne innsatsen, siden den ofte begrenser dybden av forskningsinnsikt og hindrer åpen deling av fremskritt. Samtidig har offentlig tilgjengelig forskning først og hatt et fokus på å forbedre deteksjonsalgoritmene, og ofte unnlatt å vurdere de bredere kompleksitetene, for eksempel å inkludere kontekstuell informasjon og tilpasse støttearkitekturen til deteksjonsmodellen. Som svar foreslår denne oppgaven strategier for å utvikle et system ved hjelp av maskinlæringsteknikker for å optimalisere tilpasningsevne, skalerbarhet, robusthet mot motstandsangrep, åpenhet, juridisk etterlevelse og reviderbarhet i tillegg til kun ytelse. Vi tilbyr en mangefasettert tilnærming til å utvikle et system for hatefulle ytringer, og utforsker en rekke strategier hentet fra en gjennomgang av litteratur og ved praktisk eksperimentering. Oppgaven foreslår en rekke strategier innen standardisering av input for å sikre robusthet, finjustering av språkmodeller og bruk av vedvarende overvåking med XAI-teknikker fra litteratur. Oppgaven understreker fordelene ved å bruke GPT-baserte modeller mot hatytringer, og viser til ytelsesnivåer nær menneskelig nivå samtidig som modellene reduserer tid og kostnad betydelig. Videre understreker oppgaven fordelene ved å gi modeller kontekstuell informasjon for å kunne oppdage hatefulle ytringer. I tillegg fremhever oppgaven fordelene ved å prioritere unikhet og usikkerhet ved valg av sampling for sekvensiell finjustering av modellen, noe som forbedrer ytelsen sammenlignet med tilfeldig sampling. Oppgaven foreslår også en prioriteringsstrategi som adaptivt klassifiserer instanser ved hjelp av modeller med varierende kompleksitet, avhengig av de iboende egenskapene til hver instans. Til slutt integreres alle disse strategiene i en sammenhengende systemarkitektur.

# Preface

The research presented in this thesis was conducted at the Department of Information Security and Communication Technology under the supervision of Dr. Sule Yildirim Yayilgan and Sarang Shaikh. It represents the culmination of our integrated five-year Master's program at the Norwegian University of Science and Technology (NTNU).

From the first year of university, we have always found ourselves working together on projects, learning how to play off each other's strengths and balance out our weaknesses. Yet, this thesis was a whole new ball game; it was bigger and more demanding than anything we had tackled before. Although we had a background in data science, natural language processing and linguistics were new ground for us, resulting in a steep learning curve. Despite the challenges, our research brings significant and useful contributions to the field.

We are grateful to our supervisors for their consistent support and guidance throughout this research process. We also want to acknowledge the SOCYTI research project for inspiring our study of hate speech and its prevention. We wish the project continued success.

Finally, we would like to express our gratitude to our families and friends for their unwavering support and encouragement. We especially thank our dear friend Olav Førland for dedicating his time proofreading our thesis.

Andreas Aarrestad and Santhosh Shanmugam
*Trondheim, 2023*

# Contents

# List of Figures

# List of Tables

# List of Acronyms

**ADL** Anti-Defamation League.

**AHP** Analytical Hierarchy Process.

**AI** Artificial Intelligence.

**ALBERT** A Lite BERT.

**API** Application Programming Interface.

**AUC ROC** Area Under the Curve of the Receiver Operating Characteristic.

**BERT** Bidirectional Encoder Representations from Transformers.

**BoW** Bag of Words.

**CD** Continuous Delivery.

**CNN** Convolutional Neural Network.

**CSR** Corporate Social Responsibility.

**DistilBERT** Distilled BERT.

**DOE** Design of Experiments.

**EDA** Exploratory Data Analysis.

**FNR** False Negative Rate.

**FPR** False Positive Rate.

**GDPR** General Data Protection Regulation.

**GPT** Generative Pre-Trained Transformers.

**IAA** Inter Annotator Agreement.

**IID** Independent and Identically Distributed.

**LR** Logistic Regression.

**LSTM** Long Short-Term Memory.

**MCDM** Multi-Criteria Decision-Making.

**MLM** Masked Language Model.

**MLP** Multilayer Perceptron.

**NER** Named Entity Recognition.

**NLP** Natural Language Processing.

**NLTK** Natural Language Toolkit.

**NPV** Negative Predictive Value.

**NSP** Next Sentence Prediction.

**NTNU** Norwegian University of Science and Technology.

**OLS** Ordinary Least Squares.

**POS** Part-of-Speech Tagging.

**PPCR** Predicted Positive Condition Rate.

**PPV** Positive Predictive Value.

**RE** Requirements Engineering.

**RNN** Recurrent Neural Network.

**RoBERTa** Robustly Optimized BERT.

**ROC** Receiver Operating Characteristic Curve.

**SD** Standard Deviation.

**Sklearn** Scikit-learn.

**SST** Stanford Sentiment Treebank.

**SVM** Support Vector Machine.

| | |
|---|---|
| **TF-IDF** | Term Frequency-Inverse Document Frequency. |
| **TNR** | True Negative Rate. |
| **TPE** | Tree-Structured Parzen Estimator. |
| **TPR** | True Positive Rate. |
| **TSAR** | Text Scheme Adversary Recognition. |
| **UN** | United Nations. |
| **URL** | Uniform Resource Locator. |
| **XAI** | Explainable AI. |

Chapter 1

# Introduction

This chapter provides a concise introduction to our research, presenting its motivation, goal, scope, and the research questions that will be addressed. It highlights the significance of our work and the potential contributions it may have and gives a brief overview of the following thesis structure.

## 1.1 Motivation

The online world has become a space for free expression and communication for people worldwide. However, the anonymity and ease of access to these platforms have also led to a rise in hateful speech, which in turn causes harm to individuals [75], incurs a loss of revenue for the platform [19], exacerbates existing social divisions [88], and even incites real-world violence [36]. Combating hateful speech has become an investment priority for social media companies, a major target for regulatory action, and a central focus for efforts in natural language research.

Given the vast amount of social media posts, relying solely on manual solutions like human moderation can lack scalability, be time-consuming, and less effective [31]. This highlights the necessity for automated hate speech management systems. A dedicated system optimizes resources by reducing the burden on human moderators and ensures a standardized approach to managing hate speech. Following the system's predefined rules, guidelines, and policies can address all instances of hate speech objectively and uniformly. In contrast, relying solely on human intervention may result in inconsistencies, and subjective judgments as individual moderators may have different interpretations and personal biases [74]. While non-system approaches, such as educational campaigns or legal frameworks, are crucial in combating hate speech, dedicated systems complement these efforts by offering scalable and consistent approaches to identifying, monitoring, and responding to hate speech.

Recent research on hate speech has mainly focused on improvements in performance rather than considering strategies for the practical implementation of a hate speech management system. This emphasizes the importance of conducting research that focuses on strategies that are both effective and applicable in real-world settings [57]. For social media companies, a pragmatic approach is required where the potential gains of new strategies need to be weighed against the various side effects that come with the strategy. For instance, the added value of using a machine learning model that optimizes performance might come at the cost of higher computational power. In this context, there is value in developing strategies that aid social media companies in efficiently navigating the complex task of combating hate speech while maintaining operational efficiency. While social media platforms may have developed systems to detect and take action against hate speech continuously, the lack of transparency regarding the inner workings of social media companies hampers the collective progress in combating hate speech effectively. Sharing best practices and lessons learned across the industry is valuable. Not only may social media companies benefit from this research, but other actors who are considering the use of automated systems to manage hate speech may also benefit. Throughout this thesis, we have collaborated with the SOCYTI project[1], a research initiative dedicated to preventing violence-inducing behavior in the social-cyber space of local communities.

System properties in regard to a system that manages hate speech refer to the inherent characteristics and features of the system that ensure effective identification, monitoring, and mitigation of hate speech. Amongst others, these properties can include performance, scalability, adaptability to evolving language patterns, and robustness against possible adversarial attacks. By incorporating such system properties into the research process, the development of new strategies for hate speech management systems can be better aligned with the real-world needs of these companies. These strategies should optimize not only the technical system properties of developing and deploying models for hate speech management systems but also the economic factors that play a significant role in deciding which aspects to implement in the final system. One of these strategies is to integrate contextual information into the detection systems. Social media platforms collect extensive information about the user's demographics and their usage patterns. This metadata could be leveraged to enhance the systems social media companies utilize to manage hate speech by optimizing system properties [43]. Similarly, other strategies should be developed as well.

In this manner, this thesis adopts a novel approach by focusing on contributing to the understanding of what system properties a hate speech management system on a social media company should include, as well as propose strategies to optimize them. To the best of our knowledge, no such focus has been directly pursued in the research field prior to this point. The final objective is establishing a generalized

[1]For more information on the SOCYTI project, visit: https://www.bigdata.vestforsk.no/ongoing/socyti

system architecture with these strategies. Consequently, to achieve this objective, we will commence by identifying the operational system properties that impact the practicality of detection systems. This thesis builds upon and extends the research that was conducted in the preceding specialization project [105] regarding the integration of contextual information. Consequently, there is a focus on the use of contextual information. Hence, we will assess the effects of incorporating contextual information to determine its feasibility as a viable strategy to optimize a subset of the identified properties. However, as we have extended this initial scope we will identify and analyze additional strategies that together consider all the identified system properties. Finally, we will integrate these various strategies into a unified system architecture while ensuring that system properties are still optimized and implemented cost-efficiently.

## 1.2 Problem Specification

This section presents the overall goal, the research questions, as well as the scope of the thesis. In preparing for this study, a preliminary literature review was carried out in accordance with the goal of the specialization project. Here, we identified a number of gaps in the literature which informed the development of the current goal and the research questions. This process ensures that our research will contribute to new insights in the research field and that our results are industrially applicable.

### 1.2.1 Goal

The goal of the thesis is to identify and analyze various strategies for establishing a system architecture that effectively optimizes system properties in a hate speech management system. This system architecture will illustrate how social media companies can develop generalizable hate speech management systems in which strategies that optimize different system properties can be added or removed.

### 1.2.2 Research Questions

The research questions depict the focus of our research, while also dividing the goal into more concrete and manageable tasks.

**RQ1** *What are the strategies that optimize system properties of a hate speech management system, with particular attention to the incorporation of contextual information as a potential strategy?*

To formulate effective strategies for a hate speech management system, it is crucial to optimize such a system's properties. This can involve ensuring consistent performance in real-world scenarios, defending against adversarial attacks, adapting to evolving hate speech patterns, complying with legal regulations, and providing transparency in decision-making.

Our preceding specialization project focused on the impact of integrating contextual information as features to a machine learning model tasked with detecting hate speech. While our research aims to build upon our previous specialization project by emphasizing the real-world applicability of a hate speech management system, adding contextual information for hate speech detection remains a pertinent strategy to optimize essential system properties of such a system [43]. We believe integrating contextual information will benefit a hate speech management system, resulting in insight into the nuanced characteristics and intentions embedded within the speech.

While integrating contextual information is a noteworthy strategy, it is essential to explore a diverse array of techniques and solutions to optimize system properties comprehensively. By analyzing the impact of such strategies, including the integration of context, we can gain insight into their effectiveness in managing hate speech while considering system properties such as computational cost, real-time performance, scalability, and data handling capabilities.

**RQ2** *What is a system architecture that integrates strategies optimizing system properties, while ensuring that all properties are still optimized to the same degree?*

When exploring strategies that optimize system properties of a hate speech management system, including integrating contextual information as a potential strategy, it is crucial to consider the potential interplay and impact of these strategies on one another. For example, the inclusion of context in the detection may hinder the implementation of a monitoring strategy. Therefore, it becomes imperative to devise an architecture that effectively integrates these strategies while preserving their individual functionalities. It is important to note that our primary focus lies in the theoretical establishment and conceptualization of this architecture. While we acknowledge the potential practical implementation of the proposed system, our research primarily centers on the design and theoretical aspects. This process entails grouping comparable strategies together and determining their optimal implementation sequence. The identified strategies will ultimately form the system architecture of a crucial aspect of the hate speech management system, ensuring that system properties are effectively optimized. Further details can be found in subsection 1.2.4, which covers the scope of the topic.

### 1.2.3 Subgoals

To achieve the overall goal and address the research questions, it is practical to formulate some intermediate steps. We have therefore outlined the following subgoals:

1. Identify core system properties of a hate speech management system and postulate potential strategies to optimize these properties.
2. Develop a dataset to evaluate the quantitative aspects of the postulated strategies.
3. Develop machine learning classifiers aimed at identifying hate speech for usage in quantitative experiments.
4. Conduct quantitative experiments for the strategies where the literature was insufficient.
5. Formulate and concretize the strategies, drawing on the qualitative findings from the literature review and the quantitative experiments conducted.

### 1.2.4 Scope

A hate speech management system comprises an algorithmic framework designed to detect and mitigate instances of hate speech by analyzing collected data, assessing their potential harm in accordance with policies, and implementing appropriate measures to promote a safe and inclusive online environment. Hate speech management encompasses a broader range of actions after detection, such as moderation, user sanctions, and content removal. Additionally, the method by which data is transmitted from a social media platform to the architecture responsible for detecting hate speech is also worth considering. This can include aspects such as batch processing, periodic updates, or real-time processing. We will delve further into hate speech management systems in section 2.3 in the background chapter.

The focus of this research is exclusively on the part of the system responsible for processing an abstracted input data stream and generating a labeled output stream, as illustrated in Figure 1.1. Our thesis specifically excludes the discussion of the mechanisms involved in how the input data stream is retrieved, as well as the subsequent handling of the output stream. Throughout the remainder of this thesis, we will refer to this particular part of the system as the *detection architecture* of a hate speech management system. It is crucial to emphasize that the detection architecture not only comprises a detection module but also encompasses various other modules that address different aspects of the architecture (e.g., adaptive learning, sampling, and maintenance). This thesis will therefore focus on strategies to establish these modules. In this manner, we will optimize the identified system properties of this part of a hate speech management system by suggesting appropriate strategies and finally combining the strategies in a detection architecture. We will utilize a dataset to simulate the input data stream in order to evaluate some of the identified strategies. Still, there will be no focus on how each data point would have been retrieved from the platform.

By directing our attention solely toward the detection architecture, our research ensures independence from other elements of the hate speech management system. As a result, our findings become more generalized and adaptable. In doing this research, we will also take a social media company's perspective in order to better tailor a detection architecture of a hate speech management system.

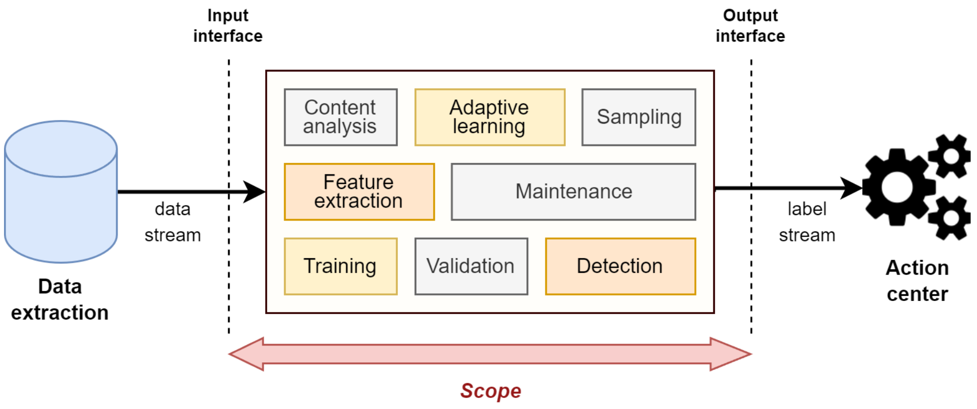


**Figure 1.1:** The scope of the thesis. We will propose an architecture for hate speech management on social media, scoped to not delve into how the data is sourced or what actions will be taken with the predicted labels.

The overarching research done in this thesis will be platform agnostic, meaning that it can be applied to any social media platform. However, the basis for our trials is a dataset derived from Twitter, which serves as a benchmark platform representing the environments to which we aim to generalize our developed strategies and insights. It should also be noted that the dataset contains only tweets written in English, as accounting for multilingual tweets is not a focus of this thesis. We will also solely distinguish between non-hateful and hateful speech and thus only utilize binary classification.

## 1.3 Research Approach

We employed a pragmatic mixed approach, drawing on inductive and deductive reasoning. From a deductive standpoint, we sought to investigate the effect of integrating contextual information as features to a machine learning model tasked with hate speech detection, following the preliminary research from the specialization project. Simultaneously, we identified effective strategies and evaluation methods for a realistic hate speech detection system through an inductive lens. By combining both approaches, we established causal relationships while contextualizing our findings.

To establish a solid basis for investigation, we utilized a content analysis approach in which we systematically analyzed relevant literature, presented in a later chapter. Such an approach helped us to interpret the findings in relation to previous research and identify inconsistencies or discrepancies. Additionally, the content analysis of relevant literature helped us explore potential explanations for the patterns and themes identified in the analysis.

In the cases where relevant literature was insufficient, we followed an experimental approach. Literature can be insufficient when it is contradicting, inconclusive or irrelevant in nature. For example, conflicting literature on context integration makes it challenging to provide a definitive answer on how it should be implemented. In these instances, it was not feasible to draw conclusive findings from the content analysis. Consequently, by both leveraging our accumulated knowledge from previous experiences and conducting new research experimentally, we were able to adapt the process. This adaptability enabled us to incorporate newfound insights and adjust our approach accordingly.

## 1.4 Research Design

Upon analyzing the progressive formulations of our research questions, it became apparent that a variation of a sequential design was the most appropriate. The inter-dependencies among the questions necessitated a step-by-step approach to obtain reliable results. The first question served as the foundation for the second research question. By exploring the impact of contextual information as features and identifying similar strategies, we could gain insights into the underlying mechanisms and dynamics of the system properties. Without a comprehensive understanding of the system properties and corresponding strategies, it would be challenging to develop a system architecture.

For each research question, we utilized a mixed-methods research design that integrated both quantitative and qualitative research methods in addition to the overall sequential design. A comprehensive explanation of these methods will be presented in the methodology chapter. The content analysis approach employed qualitative methods, while the experimental approach used quantitative methods. By combining these two designs, a more comprehensive analysis of the different strategies could be conducted to address the multifaceted challenges of hate speech management. While qualitative methods provide a broader analysis and generate a wider range of strategies without relying heavily on numerical data or actual implementation, quantitative experiments, when applied to specific aspects such as the strategy of utilizing contextual information, can offer more specific insights. While we will delve into the detailed methods employed in the methodology chapter, we present a roadmap of our research in Figure 1.2 to provide a more comprehensive understanding before presenting the findings from our literature review, which we will present before the methodology chapter. This decision was made, as the literature review was instrumental in choosing the methods used.

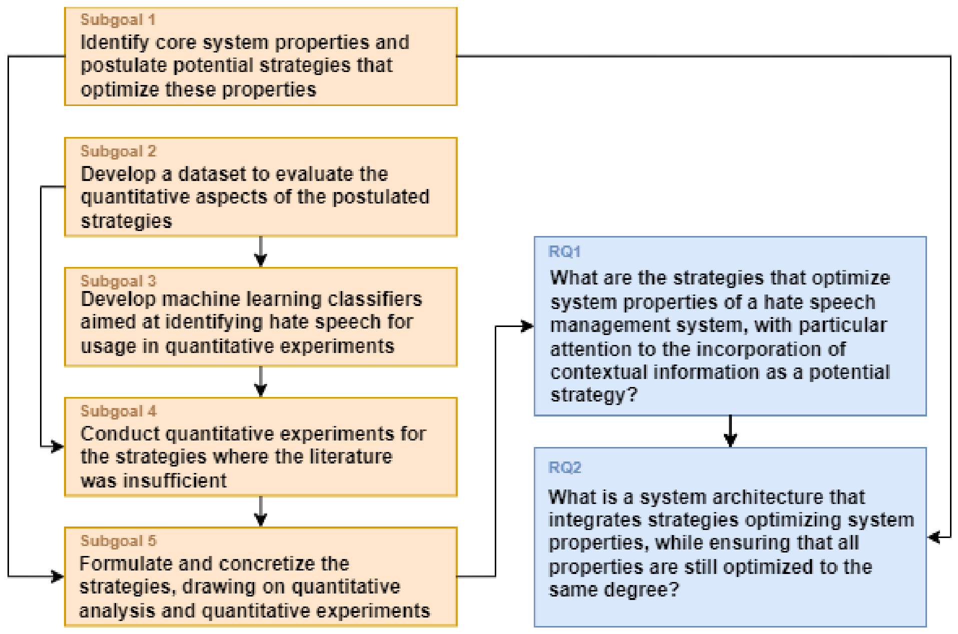

Subgoal 1
Identify core system properties and postulate potential strategies that optimize these properties
Subgoal 2
Develop a dataset to evaluate the quantitative aspects of the postulated strategies
Subgoal 3
Develop machine learning classifiers aimed at identifying hate speech for usage in quantitative experiments
Subgoal 4
Conduct quantitative experiments for the strategies where the literature was insufficient
Subgoal 5
Formulate and concretize the strategies, drawing on quantitative analysis and quantitative experiments
RQ1
What are the strategies that optimize system properties of a hate speech management system, with particular attention to the incorporation of contextual information as a potential strategy?
RQ2
What is a system architecture that integrates strategies optimizing system properties, while ensuring that all properties are still optimized to the same degree?

**Figure 1.2:** Roadmap illustrating the sequential progression of the subgoals and the research questions in the study.

It is important to highlight that in order to gain insights into the implementation of a hate speech management system that is to be used by a social media company, we had to rely on informed assumptions derived from the research community. Direct literature on how social media platforms specifically implement such systems is not readily available, as these companies tend to keep such information confidential for various reasons, which will be further discussed in the upcoming chapter 2.3.2.

After identifying both system properties and additional strategies, which are presented in the methodology chapter, we evaluated each identified strategy using both the findings from the literature review and quantitative experiments in order to answer the first research question. For instance, to investigate the implications of using contextual information as features, the process involved obtaining a labeled dataset that incorporated contextual information, utilizing the dataset to train a suitable machine learning model, and then assessing the impact on the model compared to a baseline that did not incorporate contextual information.

Thereafter, to answer our second research question, we used both the knowledge about system properties gained from our literature review and the formulated strategies in order to develop system components of the final system architecture. By combining the established system components and deciding the sequence of implementation, we developed a generalizable system architecture.

## 1.5 Outline of Thesis

The thesis is divided into hierarchical chapters, sections, and subsections, each identified in the CHAPTER.SECTION.SUBSECTION format to allow for easy navigation and reference. The thesis is structured as follows:

**Chapter 2 - Background** provides the background theory and preliminaries required to understand the rest of our work. This includes a description of what qualifies as hate speech, as well as the motivations of industry actors to eradicate it. We will then delve into what constitutes a hate speech management system. Additionally, we will provide an overview of natural language processing and relevant computer science theory. Subsequently, we will delve into the current state-of-the-art approaches and necessary prerequisites for understanding the utilization and evaluation of a natural language model.

**Chapter 3 - Literature Review** presents a review of existing literature on hate speech management, conducted preliminarily and continuously throughout the research process. It focuses on the challenges of hate speech detection and dataset acquisition, the use of context as features, and system properties for real-world applicability in the development and deployment of hate speech management systems.

**Chapter 4 - Methodology** presents the methods employed in our research. This includes methods for prioritizing the system properties identified from the findings of our initial literature review. We then describe the process of identifying and evaluating strategies that optimize the system properties. In doing this, we also present the quantitative experiments conducted to evaluate some of the identified strategies. Amongst others, this included analyzing the integration of contextual information. Specifically, we present the novel process for accurately and cost-effectively labeling the dataset used, followed by the necessary preprocessing and feature engineering, and conclude with the model architecture and training. Finally, we present the methods to combine the formulated strategies into a system architecture.

**Chapter 5 - Results and Discussion** presents and discusses the identified strategies to develop and deploy hate speech management systems that meet the identified system properties. This includes an overview of the different identified strategies of the hate speech management system, as well as results from the quantitative analyses.

All the strategies will be consolidated into a single, cohesive system architecture. The chapter will also include a discussion of the strengths and limitations of the proposed strategies.

**Chapter 6 - Conclusion** will summarize the contributions of the thesis, outline the significance of our findings for the industry and society, and discuss potential future directions for the research.

# Chapter 2 Background

This chapter presents an overview of the background knowledge needed in order to understand the domain and the task at hand. By clarifying the preliminary aspects of the research, this section aims to establish the context for the subsequent analyses and findings.

## 2.1 Hate Speech

Hate speech is a pervasive and complex phenomenon that usually refers to various forms of speech that expresses hate against an individual or group based on their inherent characteristics. It often causes significant emotional distress and psychological harm to its targets. It creates an environment of fear, anxiety, and insecurity, contributing to a hostile social climate. Combating it requires investments in education, awareness, legal frameworks, and a societal commitment to promote diversity.

### 2.1.1 Prevalence and Dynamics in Social Media

In recent years, the landscape of hate speech has significantly shifted because of the emergence of social media. With billions of interconnected users, the digital world has provided users with powerful channels for communication and expression but simultaneously created a fertile ground for hate to spread in an instantaneous and wide-reaching fashion.

#### Scale

The amount of online hate speech has increased exponentially with the rise of social media platforms. In the first quarter of 2023, Meta removed 11 million pieces of content from their Facebook platform because of violations of their hate speech policies, up from 10.6 million in the proceeding quarter according to the most recent community standards enforcement report [45]. Teens are especially exposed to hate speech online; one study from 2021 found that over 70% of respondents between the ages of 18 and 25 have witnessed online hate speech in the last three months [63].

### Targets

Hate speech often targets individuals or groups based on attributes such as their race, religion, ethnicity, heritage, gender identity, or sexual orientation. Particularly, marginalized groups are often disproportionately the targets of such discourse. As seen in the net harassment in Figure 2.1, all the marginalized groups face a significant amount of harassment online. Individuals in the LGBTQ+ community (lesbian, gay, bisexual, transgender, queer/questioning, and others) report especially high harassment rates. These extreme figures highlight the urgent need for further investment into research which can be applied to real-life systems, as protecting marginalized groups should be a top priority in diverse societies.

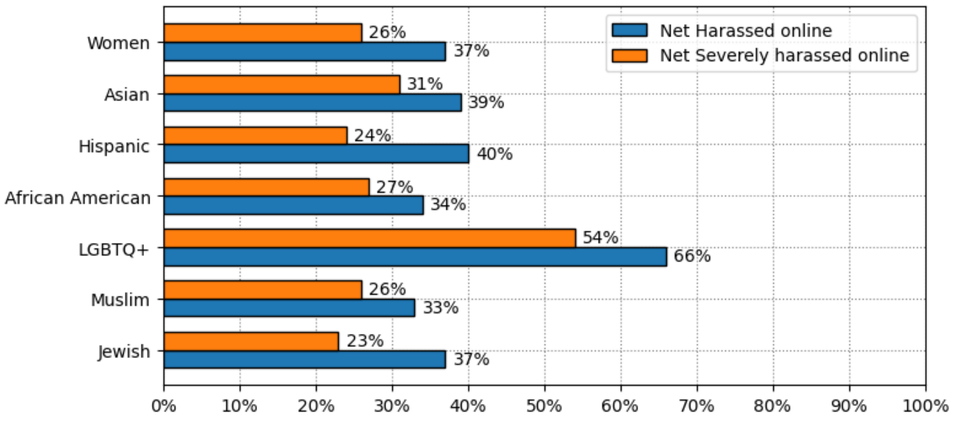


**Figure 2.1:** Net harassment by marginalized group in the United States in the 2022 *Online Hate and Harassment* survey by the ADL[1] [75]. Severe harassment includes experiencing physical threats, sustained harassment, stalking, sexual harassment, doxing[2], and swatting[3].

### Amplification Factors

The inherent nature of digital space and the incentives of social media companies give rise to a range of amplification factors that makes hate speech persist easily on the platforms. First, anonymity can often embolden users to direct hate toward others without facing personal consequences. Additionally, hate can be directed

[1]The Anti-Defamation League (ADL) is an international anti-hate organization.

[2]Doxing is searching for and publishing personally identifiable information about an individual, usually for reasons such as exposure, blackmail, or public shaming.

[3]Swatting is a harassment technique that involves making false reports to emergency services, usually with the goal of tricking law enforcement to respond to a target individual's home address.

toward unfamiliar users, potentially reducing empathy and facilitating the expression of hateful sentiments.

Most social media platforms operate with an advertising-based[4] revenue model, and they are thus paid in relation to how much engagement they get from users. As such, it is in their interest to design their recommendation algorithms[5] such that they maximize user engagement, which often translates into showing users content similar to what they have interacted with earlier on. These algorithms prioritize content that aligns with a user's beliefs and preferences, reinforcing their biases and fostering an environment conducive to hate speech [18]. This can lead to the radicalization of individuals, the proliferation of discriminatory ideologies, and the formation of echo chambers that perpetuate and amplify hate speech within a community [88]. In this manner, hate speech can potentially incite violence against targeted individuals or groups. It can encourage individuals who hold extreme views to act upon their prejudices, leading to physical attacks, harassment, or even acts of terrorism [36].

#### Relationship with Real-World Incidents

Online hate speech has been linked to real-world violence in numerous instances. According to a study conducted in 2019 by academics from Cardiff University's HateLab, it was discovered that a rise in hate speech on social media platforms could directly contribute to an increase in real-world crimes targeting minority groups [100]. The researchers analyzed crime data from London alongside Twitter data and observed a correlation between the number of "hate tweets" originating from a specific location and an escalation in racially and religiously aggravated crimes in that area. Similarly, former U.S. President Donald Trump's frequent use of hate speech and divisive rhetoric has been linked to a rise in hate crimes and a normalization of discriminatory beliefs and behaviors [50].

The intensity of hate speech tends to escalate during significant global events. According to a study conducted by the AI security company Light, there was a significant 900% surge in hate speech tweets targeting Chinese individuals and China between December 2019 and March 2020, coinciding with the beginning of the COVID-19 pandemic [33].

### 2.1.2 Platforms' Incentives for Action

Though there is an ethical responsibility of platforms to protect their users from hateful content, it's important to recognize that these companies might not prioritize

[4]An advertising-based revenue model is a common revenue model for companies offering digital services where the users do not pay to use the service, but are instead exposed to ads by third-party advertisers which pay for exposure.

[5]Social media companies use recommendation algorithms to automate the selection, ranking, and presentation of content for a specific user on the platform.

ethical considerations in decision-making. However, there are still legal, financial, and reputational factors that incentivize social media companies to remove hateful content from their platforms.

#### Reducing Risk of Litigation

Some jurisdictions have introduced legislation to regulate the responsibility of social media platforms to take appropriate measures to remove hateful content. The content moderation provisions of the Digital Services Act of the European Union may, for example, require companies to remove what is deemed hateful content [19]. Failure to adhere to government regulations might result in the company paying large fines or criminal charges to executives, depending on the severity of the non-compliance and the jurisdiction.

#### Improving Public Image

Advertisers are concerned with the ethical values of the platforms they display their advertisements [76]. Therefore, having good relations with advertisers is important to maintain steady ad revenue streams. Additionally, CSR[6] has become an increasingly important strategy where companies' commitment to social issues can help attract investors and strategic partnerships.

#### Increasing User Growth

User satisfaction is important as, needless to say, users do not like being harassed. Therefore, addressing hateful speech and providing users with inclusive social spaces can increase user retention rates and make the user base more diverse, leading to higher user growth rates for the platform.

## 2.2 Twitter

Twitter is an American social media platform that provides users with the ability to share content, engage in conversations, and connect with others in a real-time global context. By default, the content on Twitter is public, allowing the platform's over 350 million users to view a vast majority of the approximately 6,000 posts posted per second. Central to the platform are tweets, which are multi-modal posts containing text, limited to 280 characters, as well as images, videos, URLs, and hashtags. Tweets are commonly accessed through a user's feed, a personalized and dynamic stream of posts designed to maximize user engagement.

[6]Corporate social responsibility is a strategic business model in which companies adopt sustainable practices that provide risk management and value creation by taking accountability for their impact on society.

Twitter serves as a platform for governments, corporations, and individuals to make official announcements and communicate with others. Influencers, brands, and activist movements leverage its extensive global reach for advertising and influencing the public, while individuals utilize it to share opinions, experiences, and creative works. This versatility positions Twitter as a significant platform for studying and understanding online public discourse.

### 2.2.1 Interaction Model

The platform facilitates various interactions among users, between users and tweets, as well as between tweets themselves. Given the vast range of interactions available, we will only outline the ones that influence our thesis. These are typically those that establish relationships or are quantifiable statistics present in the metadata. An overview of the interactions can be seen in Figure 2.2.

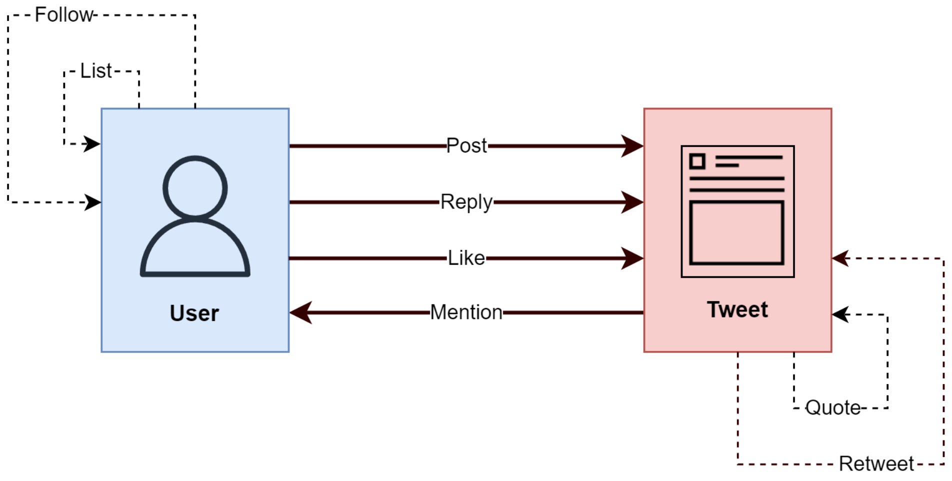


**Figure 2.2:** Schema graph for users and tweets in Twitter's interaction model.

#### Interactions Among Users

Apart from private messaging, users interact with each other mainly through the interactions "following" and "listing". These interactions enable the user to personalize their content consumption on the platform, as following and listing will enable the user to decide what content is displayed.

Follow: Subscribes to another user's future tweets, making them visible in the feed.

List: Adds another user to a customized list, allowing the user's future tweets to be viewed in a dedicated feed.

#### Interactions Between Users and Tweets

Users interact with tweets through engagement actions such as posting, liking, replying, and mentioning.

| | |
|---|---|
| Post: | Publishes a tweet that is shared with users that have followed or listed the user. |
| Like: | Showing appreciation for a tweet by liking it. Likes are anonymous, but the number of likes is visible on a tweet. |
| Mention: | Adds a clickable reference to another user's account in a tweet to notify or credit them, allowing for direct attribution. |
| Reply: | A comment is added below a tweet and is threaded to maintain conversation context. The number of replies is stored as a metric. |

#### Interactions Between Tweets

Similarly, tweets interact with other tweets. In this manner, tweets can be instantiated from existing ones.

| | |
|---|---|
| Retweet: | When someone shares another person's tweet, it allows the retweeter's followers to see it on their own timeline, giving it visibility under the retweeter's name. |
| Quote: | Enables users to share another person's tweet while adding their own commentary or context, allowing it to be seen by their followers along with their additional insights. |

### 2.2.2 Components of a Conversation Chain

A Twitter conversation chain typically refers to a series of connected tweets part of a common thread. This occurs when users reply, retweet, or quote a tweet. Regarding retweets and quotes, the original tweet is often referred to as the parent tweet, and a response tweet is considered a child tweet. While a parent tweet can have multiple child tweets, a child tweet can only have one parent tweet. Similarly, the user who posted the parent tweet can be referred to as the parent author, and the user who responded can be referred to as the child author.

## 2.3 Hate Speech Management Systems

Social media companies have employed various strategies to manage hate speech on their platforms. The management of hate speech encompasses continuously identifying and responding to instances of hate speech. Although a standardized naming convention for the various aspects of a hate speech management system is currently unavailable, our research has identified three distinct elements [83]. The first element involves defining community guidelines and determining how posts on the platform are directed to a detection architecture. This usually involves user response mechanisms (e.g., reporting, flagging) and real-time processing approaches. The second element, which we refer to as the detection architecture as outlined in subsection 1.2.4, encompasses the detection, which includes both human moderation and automated tools for detection, and the associated support architecture in place to optimize relevant system properties of detection. The final element involves taking appropriate actions against speech identified as hate speech. Collectively, these stages form a comprehensive hate speech management system as depicted in Figure 2.3.

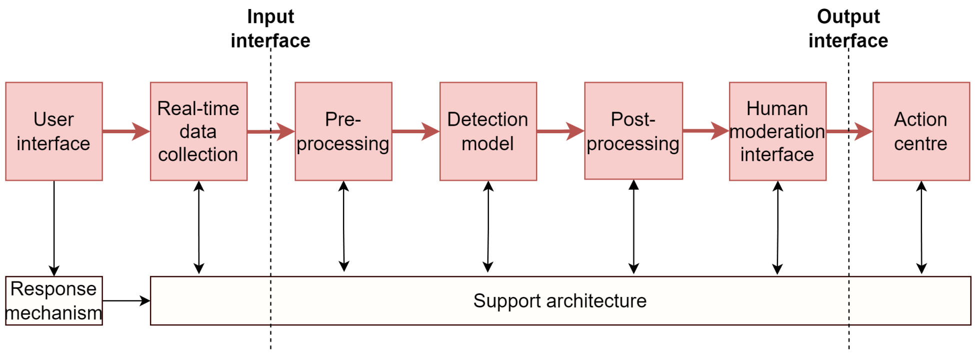


**Figure 2.3:** Abstract hate speech management system architecture. Our scope focuses on the component between the input and output interfaces.

## 2.3.1 System Components

Before the actual detection of hate speech, most social media companies establish community guidelines that outline acceptable behavior and content standards. These guidelines explicitly prohibit hate speech, with the company itself responsible for defining the specific parameters that classify hate speech. For example, Twitter has established criteria to identify and address hateful conduct, including but not limited to hateful references, slurs, and dehumanization, as outlined in their policy on hateful conduct [28]. Some companies have also established partnerships with advocacy groups, experts, and civil society organizations to refine their guidelines. The guidelines serve as a basis for determining what content should be removed or restricted.

After establishing community guidelines posts must be sent to the detection architecture. There are several approaches to accomplish this. Social media platforms encourage their users to report content that violates their community guidelines, leading to the direct transfer of reported posts to the detection architecture. However, such an approach is reactive, requiring users to encounter hateful content before any action can be taken against the offending post. Ullmann [91] describes such approaches as unethical as the posts will still cause the recipients psychological harm before being taken down. Therefore, the majority of social media companies have implemented some form of processing of all posts. The decision of whether to perform this processing in real-time or periodically using batch processing is therefore an important aspect.

Traditionally the detection architecture consisted of human moderators. Social media platforms most often employ teams of moderators who review reported or flagged content to assess its compliance with the guidelines. Moderators evaluate factors such as context, intent, potential harm, cultural nuances, and severity of the offense to determine whether a particular piece of content qualifies as hate speech. Due to the complex nature of hate speech, moderators employ multiple categories to classify the various types of hate speech (e.g. aggressive, harmful, satirical), enabling the company to take targeted actions accordingly.

In recent years, social media companies have utilized automated tools to aid in the detection architecture of the hate speech management system in addition to human moderation. These tools employ various state-of-the-art machine learning technologies to analyze content and identify instances of policy violations. Automated systems can flag suspicious content for further human review or even remove it directly based on predefined rules. However, due to the complexities of hate speech, these automated tools are not always foolproof and require continuous refinement and human oversight. This human oversight with multiple moderators frequently leads to significant cost implications for social media companies [84]. This matter will be further elaborated on in the literature review.

Once an instance of hate speech is detected and its classification is confirmed through human oversight, the appropriate action is determined based on its severity in relation to the violated guidelines. This constitutes the last system component and can include making a post less visible, removing a post from the entire platform, or even banning a user.

### 2.3.2 System Evolution and the Need for Research

Although hate speech management systems exist today, new forms of hate speech, emerging trends, and evolving user behavior require continuous improvement. As these factors evolve, it is essential to recognize that previous community guidelines may become outdated, necessitating the implementation of new ones. In addition, automated systems often struggle with the context, sarcasm, and cultural nuances of the evolving language, leading to errors in classification. The existing detection architecture that utilizes these automated systems may require the implementation of novel strategies to adapt to evolving inputs or new restrictions imposed by authorities.

As new types of hate speech emerge, there may also be a need to introduce new measures or actions to address them effectively. With this in mind, there is a constant need for ongoing research to enable the development and implementation of an up-to-date hate speech management system.

Regarding the availability of such research, while social media companies have made efforts to combat hate speech, they may not publicly disclose all the details of their management systems. Companies often guard their proprietary technologies and algorithms to maintain a competitive edge. Consequently, there is no current standardized system for managing hate speech. Companies are also dependent on continuous research in this field in order to receive external feedback and knowledge. Therefore, we will focus our research on enhancing accessibility to pertinent information and advancing the field. This thesis exclusively concentrates on the detection architecture, disregarding components such as community guidelines, data transfer to the detection architecture, and the subsequent course of action post-detection.

## 2.4 Machine Learning

Machine learning is a sub-field of artificial intelligence (AI) concerned with leveraging data to develop methods that make machines capable of learning patterns and making predictions without being programmed explicitly. In this subsection, we aim to navigate the domain of machine learning in the context of binary hate speech detection and present relevant concepts.

### 2.4.1 Statistical Classification

Statistical classification is a branch of machine learning that aims to enable models to categorize instances in a dataset into predefined classes by learning statistical patterns in their characteristics. These characteristics are often referred to as features and are represented in a numerical feature vector. An algorithm that implements a classification scheme is commonly referred to as a classifier and produces an output prediction given an input.

### 2.4.2 Supervised Learning

Machine learning is generally categorized into three paradigms: supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, the feature vectors are accompanied by corresponding ground truth labels which are provided to the classification models. The feature vectors are the inputs, and the labels are the output that the model tries to predict. Hate speech detection usually demands a supervised approach to be performative, as hate speech is a subjective concept, and a machine would require the judgment of a human to determine what is hate speech and what is not.

Moving into a mathematical framework for supervised learning, we define $X$ and $Y$ as the set of feature vectors and label vectors in a dataset, respectively. $\mathbf{x}_i \in X$ and $y_i \in Y$ represent samples from i-th elements in these sets. The hypothesis function $f : X \to Y$ maps the feature vectors from $X$ to labels in $Y$. In the context of machine learning, a model tries to optimize this function by approximating the underlying relationship between the input data and the desired outputs. Given a set of $n$ input-output pairs and a hypothesis function $f$ parameterized by a parameter vector $\mathbf{w}$, we define the objective function $E(\mathbf{w})$ as the cumulative sum of a loss function $L$ over all samples $n$ as seen in equation 2.1. The objective function, therefore, serves to quantify the performance of the model given the parameter $\mathbf{w}$.

$$E(\mathbf{w}) = \sum_{i=1}^{n} L(f(\mathbf{x}_i; \mathbf{w}), y_i) \tag{2.1}$$

The loss function $L$ quantifies the discrepancy between the prediction of the model, $f(\mathbf{x_i}; \mathbf{w})$, and the true value, $y_i$. In binary classification problems, it is often given by the cross-entropy loss as defined in equation 2.2. This essentially measures how efficient the predictions are in terms of encoding information about the true outcomes when assuming a different distribution. Here, $p_i$ represents the model's predicted probability of a particular class for the i-th instance.

$$L_{CE}(p_i, y_i) = -y_i \cdot \log(p_i) - (1 - y_i) \cdot \log(1 - p_i) \tag{2.2}$$

The aim of the supervised learning scheme is to find the optimal parameter values $\mathbf{w}^*$ that would minimize the objective function $E(\mathbf{w})$ as denoted in equation 2.3. This process of finding the optimal parameter values is commonly referred to as training and usually involves applying optimization techniques that iteratively adjust the parameter vector $\mathbf{w}$ based on the inputs $\mathbf{x}$ and corresponding target labels $\mathbf{y}$.

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} E(\mathbf{w}) \tag{2.3}$$

If the classifier is not able to capture the underlying patterns in the data, it's said to be underfitting. If the classifier performs well but generalizes poorly to unseen instances, it's said to be overfitting. Generalization refers to the ability of a trained model to accurately perform on unseen or new data that it has not been previously exposed to during training.

### 2.4.3 Performance Metrics for Supervised Classification Models

Several metrics are used to measure the performance of classification models, each offering different insights into a model's effectiveness. We have selected metrics that we found to best align with the characteristics of our data and the experimental design we employed.

#### Classification Outcomes

Our task of hate speech detection is a binary task, and we are therefore concerned with binary classification. In a supervised binary classification task, we encounter four potential outcomes when comparing a model's prediction $\hat{\mathbf{y}}$ to the true label $\mathbf{y}$. These outcomes can be organized in what is called a confusion matrix, illustrated in Table 2.1. When applying the model to a set of data, it is conventional to use the same notation (TP, FN, FP, TN) to refer to the counts representing the number of each of these outcomes.

| | | **Predicted Condition** | |
|---|---|---|---|
| | | Predicted Hate<br>$\hat{y} = 1$ | Predicted Non-Hate<br>$\hat{y} = 0$ |
| **Actual Condition** | Hate<br>$y = 1$ | True Positive (TP)<br>Hit | False Negative (FN)<br>Type II Error |
| | Non-Hate<br>$y = 0$ | False Positive (FP)<br>Type I Error | True Negative (TN)<br>Correct Rejection |

**Table 2.1:** 2x2 confusion matrix for a binary hate speech classification problem.

#### Fundamental Metrics

From the outcomes mentioned, we can further define four basic metrics, as seen in equation 2.4, that are often used in binary classification tasks: Positive Predictive Value (PPV), Negative Predictive Value (NPV), True Positive Rate (TPR) and True Negative Rate (TNR).

$$\begin{aligned} PPV &= \tfrac{TP}{TP+FP} \qquad & NPV &= \tfrac{TN}{TN+FN} \\ TPR &= \tfrac{TP}{TP+FN} \qquad & TNR &= \tfrac{TN}{TN+FP} \end{aligned} \tag{2.4}$$

#### Precision and Recall

Precision and recall are conventional metrics used in supervised classification problems that provide complementary perspectives on model performance. Precision quantifies the model's ability to detect relevant instances, while recall quantifies its ability to avoid missing instances.

$$\begin{aligned} \text{Precision} &= w_{\text{positive}} \cdot \text{PPV} + w_{\text{negative}} \cdot \text{NPV} \\ \text{Recall} &= w_{\text{positive}} \cdot \text{TPR} + w_{\text{negative}} \cdot \text{TNR} \end{aligned} \tag{2.5}$$

Conventionally, in binary classification problems, precision and recall are evaluated considering the positive class, which is typically the class of interest. This approach, often referred to as *binary*, treats precision as the Positive Predictive Value (PPV) and recall as the True Positive Rate (TPR). However, in cases of severe class imbalance in the dataset, it may be more appropriate to use the *weighted* forms of these metrics, as defined in equation 2.5. With this approach, the metrics are computed separately for each class, and the weighted sum of the metrics is calculated to provide an unbiased evaluation of the model's performance. We denote $w_{\text{positive}}$ and $w_{\text{negative}}$ as the proportion of instances that belong to the positive and negative class, respectively, in the dataset. It is essential to note that the sum of $w_{\text{positive}}$ and $w_{\text{negative}}$ always equals 1, representing the entirety of the dataset.

#### $\mathbf{F_1}$ Score

It is common to balance both precision and recall by computing the $F_1$ score, which represents the harmonic mean of the two metrics. A high $F_1$ score indicates that the model performs well in terms of both precision and recall. In other words, the model accurately distinguishes hateful instances from others while capturing a significant portion of the hateful instances. The weighted $F_1$ score is defined in equation 2.6.

$$F_1 = w_{\text{positive}} \cdot \frac{2 \cdot \text{PPV} \cdot \text{TPR}}{\text{PPV} + \text{TPR}} + w_{\text{negative}} \cdot \frac{2 \cdot \text{NPV} \cdot \text{TNR}}{\text{NPV} + \text{TNR}} \tag{2.6}$$

#### ROC AUC

The Receiver Operating Characteristic (ROC) curve depicts the TPR plotted against the false positive rate (FPR) at various classification thresholds, and serves as a graphical representation of the inherent trade-off between TPR and FPR. The area under this curve (AUC-ROC) is the integral of this curve. While not as interpretable, it serves as a powerful metric as it can measure performance independent of the threshold and the class balance.

### 2.4.4 Neural Networks

Neural networks are a broad term that refers to any computational system made up of interconnected layers of units called neurons. Just like the brain that serves as its inspiration, a neural network processes data by passing it through interconnected layers. The most basic version of a neural network is a Multilayer Perceptron (MLP) where the network forms a directed graph and each neuron computes a weighted sum of its inputs, adds a bias term, and applies a non-linear transformation to the result.

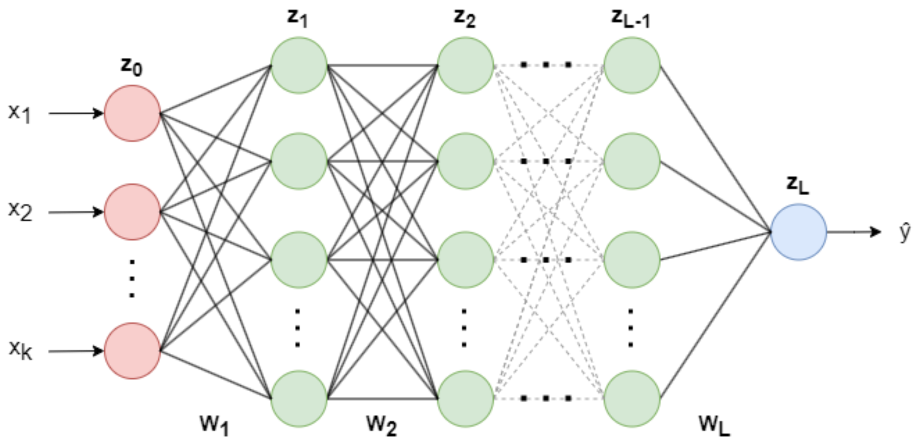


**Figure 2.4:** Architecture of a multilayer perceptron.

Within the context of the supervised learning framework as defined in subsection 2.4.2, the parameters $\mathbf{w}$ of an MLP network consist of the weights of the links between neurons and the bias of each neuron. This can be denoted as $\mathbf{w} = \{\mathbf{W}_1, \mathbf{b}_1, ..., \mathbf{W}_L, \mathbf{b}_L\}$ where $\mathbf{W}_l$ and $\mathbf{b}_l$ represent the weight matrix of the links and the bias vector for the $l$-th layer $z_l$, respectively. The hypothesis function $f$ is the forward pass as denoted in equation 2.7, where $g_l$ is the activation function applied after layer $l$, implementing non-linearity into the model and enhancing its ability to capture complex patterns in the data. In a supervised classification problem, the weights and biases are updated as the network is trained using a large number of input-output pairs. In the context of hate speech detection on a social media platform, a single input to the model, such as a text string or post, is referred to as an *instance*, and the output would be a binary label indicating whether the instance contains hate speech or not (i.e., hate or non-hate).

$$\begin{aligned} \mathbf{z}_0 &= \mathbf{x} \\ \mathbf{z}_l &= g_l(\mathbf{W}_l \mathbf{z}_{l-1} + \mathbf{b}_l) \quad \text{for } l = 1, ..., L \\ \hat{y} &= \mathbf{z}_L \end{aligned} \tag{2.7}$$

### 2.4.5 Deep Learning

Deep learning expands upon the foundational principles of neural networks through the incorporation of numerous layers, augmenting the system's capacity and enhancing its capacity for generalization across intricate patterns. Such deep neural networks facilitate representation learning, such that manual feature extraction is not needed as the layers learn a more abstract representation of the data.

### 2.4.6 Hyperparameters

Model parameters that are not learned during training, but instead set explicitly, are commonly referred to as hyperparameters. Selecting appropriate hyperparameters is essential for optimizing the performance of a machine learning model as it can have a significant impact on the model's ability to generalize and avoid overfitting. Tuning hyperparameters is essential to strike the right balance between performance and generalization, leading to a model that performs optimally on unseen data. It often involves a trial-and-error process, where different values are tested to find the best configuration.

### 2.4.7 Transfer Learning

Transfer learning is a technique in deep learning where one is concerned with applying the knowledge learned during a source task to a new and related target task. The training of the source model, commonly referred to as pre-training, is often done on a generalized task with large amounts of data. When re-purposing the pre-trained model, the parameters of the lower layers in the neural network are typically frozen, and the model is trained on a target dataset, a process commonly referred to as fine-tuning, as depicted in Figure 2.5.

### 2.4.8 Lifecycle Framework

With machine learning techniques now being implemented in safety-critical applications, insufficient performance or inaccuracies in machine learning algorithms can in these applications lead to irreversible system malfunction. The usage of machine learning techniques in such critical systems has therefore evolved into a cycle of multiple stages in order to achieve higher levels of assurance. This is an intricate and iterative process, commencing with the acquisition of data for training a machine learning model, and culminating in the integration of said model into the system's operational framework.

The following stages are presented by Ashmore et al. [4], who describe the methods available from the literature for ensuring high levels of assurance in a framework which is coined the *machine learning lifecycle*:

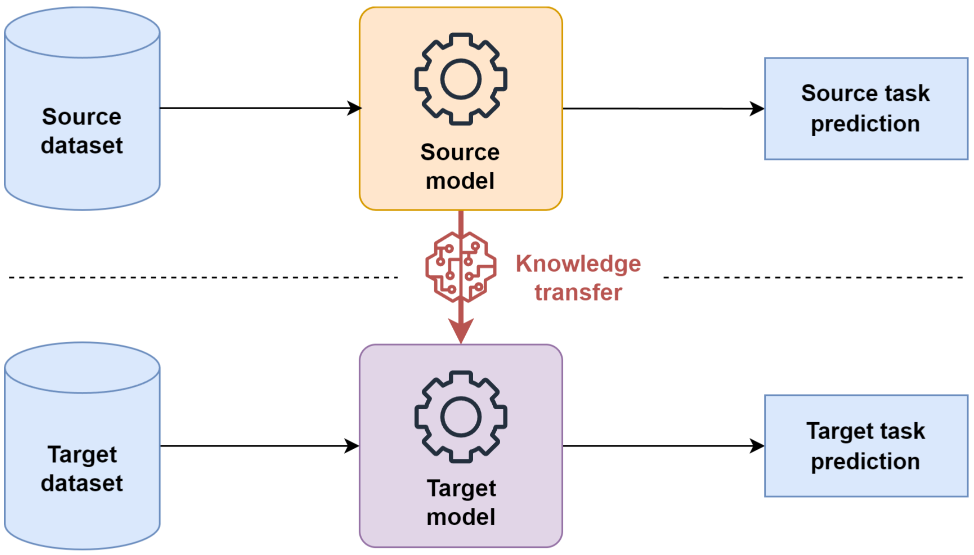


**Figure 2.5:** Generalized transfer learning workflow where knowledge and representations, often via learned model parameters, are transferred from a source model to a target model.

1. Data Management
2. Model Learning
3. Model Verification
4. Model Deployment

The first three stages concern the creation and maintenance of the machine learning model, while the last stage concerns the deployment of the developed model within an operational system. In this manner, the machine learning lifecycle can represent the detection architecture while the operational system in which the architecture will be deployed is the hate speech management system.

The data management stage comprises data collection, augmentation, preprocessing, and analysis. The result of this process is a training dataset and a verification dataset.

The subsequent model learning stage includes choosing an adequate model through training. This entails training models on the obtained training dataset with a corresponding loss function, followed by adequate hyperparameter tuning. Following the initial training phase, the model may also undergo multiple fine-tuning iterations. The model that achieves the best results, may continue to the next stage. If satisfactory levels of performance are never met, the data management stage must be re-conducted.

At the model verification stage, the primary focus revolves around the task of guaranteeing that the optimal performance of the trained model generalizes well. The previously obtained verification dataset is used for this task before it computes the error of generalization. If this error exceeds a predefined threshold, the process must revert back to either the model learning stage or the data management stage. While this primarily pertains to the model's performance, other system properties can also be examined in a similar manner during this stage. This stage ultimately produces a verified model.

Finally, the model deployment stage comprises the integration of the verified machine learning model with other components of the entire system. This also involves the continuous monitoring of the model and its updating through offline maintenance or continuous sampling to enhance the datasets used for fine-tuning. Such approaches will be further reviewed in the subsequent literature review chapter. This stage results in the successful deployment and operation of a fully functional system.

## 2.5 Natural Language Processing

Natural Language Processing (NLP) is a subfield of AI focused on giving machines the ability to understand natural language[7]. Amongst other areas of use, this involves developing computational models that enable computers to understand, analyze and generate natural language. Machine learning techniques have significantly advanced NLP, enabling machines to analyze and generate language in a more efficient manner.

As different stakeholders have expressed their interest in NLP, the field itself has evolved into multiple subfields. Search engines make use of NLP to retrieve relevant information from large amounts of unstructured text data. More recently, chatbots and virtual assistants that can understand and respond to human language have been developed. In larger systems, NLP is used to understand and draw conclusions from large amounts of natural language.

A combination of these subfields, amongst others, is relevant in the case of hate speech detection. The continuous improvement of NLP models through progress in the field of machine learning and data analysis empowers hate speech detection systems to adapt and evolve, consequently keeping up with the ever-changing nature of language and the emergence of new forms of hate speech.

[7]Natural language is the form of communication that humans use to convey meaning through spoken or written words.

### 2.5.1 Language Representation

In order to make machines able to process natural language, we need to provide the model with a format in which it can understand. Although humans are capable of understanding language in its raw and nuanced form, machines need a more structured and numerical representation in order to process it effectively. This process can be broken down into two sub-processes; tokenization and vectorization.

This process becomes even more crucial in the domain of hate speech detection, given the inherently non-generalizable nature of the online text, characterized by factors such as increased grammatical errors, abbreviations, emojis, and the substitution of letters with numbers.

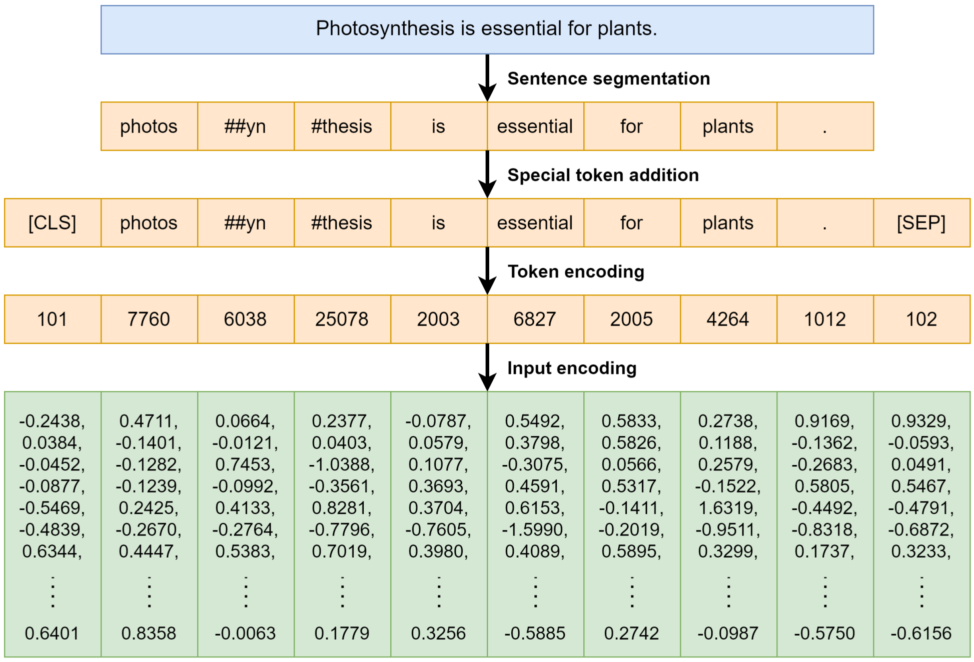


**Figure 2.6:** Simplified input representation workflow in NLP, showing how a sentence is represented as a series of embedding vectors.

#### Tokenization

Tokenization is the process of breaking down text into smaller units, commonly referred to as tokens. Often, these tokens are words. The tokenizer module is equipped with a fixed vocabulary to which it will map strings. In the case of unknown, highly complex compound words or morphological variations[8], the tokenization process may result in a single word being segmented into multiple tokens, as depicted at the beginning of Figure 2.6. Here, specialized tokens are added as well to convey structure and aid information processing; the *CLS* token summarizes the input sequence for classification tasks, while the *SEP* token separates different segments within a single input sequence. This approach helps handle the diversity and complexity of human language. Each of these tokens is further represented by an integer identifier.

#### Vectorization

For a machine to understand the relationships between words, it is not enough to represent a text as a string of integers. The model needs a way to group related concepts or linguistic structures together. Therefore, the tokens will be encoded into vectors called embeddings, which capture the semantic and syntactic relationships between tokens based on the order and context in which they appear. This representation enables a neural network to learn and update the embeddings by being fed large amounts of natural language data.

### 2.5.2 Transformer

The transformer is a groundbreaking deep learning architecture. The transformer takes an input sequence and performs a series of operations to generate an output sequence that encompasses richer and more comprehensive information derived from the input. An important aspect of these operations is a novel self-attention mechanism that enables the transformer to effectively capture complex dependencies in the data, regardless of the sequential order, enabling it to handle sequential data in a parallelized fashion [93]. Outlining the complete inner workings of the Transformer architecture is outside the scope of our research, but we want to mention that it consists of transformer blocks, as depicted in Figure 2.7. By stacking multiple transformer blocks on top of each other, information flows through the model, and each block refines the representations learned from the previous block. This multi-layered architecture enables a model to capture hierarchical dependencies and extract high-level features from the input sequence. These factors have contributed to the transformer's success and its ability to achieve state-of-the-art performance across a variety of NLP tasks.

Transfer learning with transformer-based models has proven to be highly effective. In the context of NLP and transformers, this involves pre-training a transformer-based model on a large corpus of unlabeled text data in a semi-supervised manner.

[8]In the context of linguistics, a morphological variation refers to the change in the form of a word to express different meanings or grammatical features such as tense, case, gender, number, or mood.

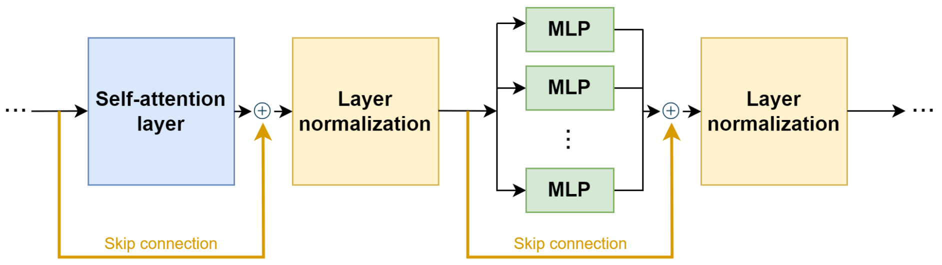


**Figure 2.7:** Architecture of a single layer, commonly referred to as a *transformer block*, within the encoder component of a Transformer model.

### 2.5.3 BERT

In 2018 researchers at Google leveraged the findings of the preceding year's research that introduced the transformer to publish Bidirectional Encoder Representations from Transformers (BERT), a revolutionizing NLP model with bidirectional training and word representations [15].

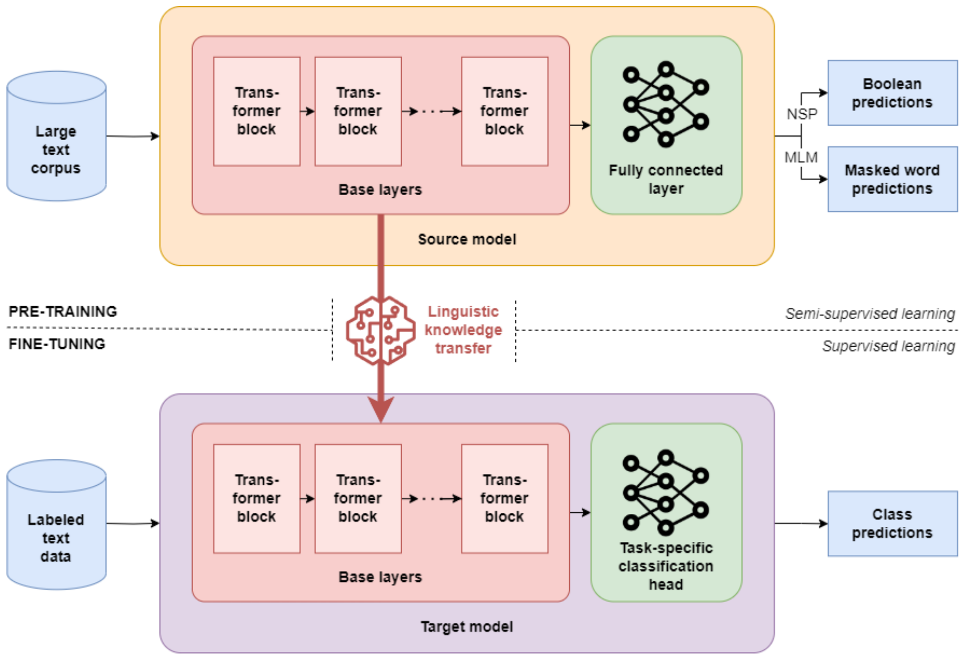


**Figure 2.8:** Transfer learning workflow for BERT. The knowledge transfer is done by re-using the pre-trained base layers from the source model that contain abstract linguistic and worldly knowledge.

BERT functions as a pre-trained generalized framework that can be further customized through fine-tuning to enable its application to an array of NLP tasks, thus enabling it to attain a remarkable level of performance across multiple domains. BERT's impressive performance stems from the transformer architecture and the Masked Language Model (MLM) pre-training. MLM is a training method in which some words in a sentence are randomly masked or replaced with a specialized masking token with the objective of predicting the original words based on the surrounding context. This technique allows BERT to learn the relationships between words and their contexts in a bidirectional manner, meaning that it can understand the meaning of a word based on both the preceding and succeeding words. Additionally, BERT utilizes the Next Sentence Prediction (NSP) technique, where it learns to determine whether a pair of randomly sampled sentences occur consecutively in the original text or if they are selected from different sections of the text. BERT is able to pre-train bidirectional representations from unlabeled text by accounting for the preceding and the following context [15]. This enables BERT to be fine-tuned for various tasks, such as hate speech detection, without the need for a task-specific framework. After pre-training, a classification head is added to the model, facilitating its adaptation for specific tasks like hate speech detection. Figure 2.8 illustrates both the pre-training and fine-tuning processes.

### 2.5.4 GPT

In the course of writing this thesis, OpenAI unveiled their ChatGPT chatbot which swiftly captured the world's attention. The chatbot utilized GPT-3.5, an advanced GPT framework known as the Generative Pre-trained Transformer 3, consisting of 175 billion parameters [9]. GPT-3.5 is part of a broader family of large language models called Generative Pre-Trained Transformers (GPT). The GPT family represents a breakthrough in natural language processing, leveraging transformer-based architectures and extensive pre-training on large-scale datasets to generate coherent and contextually relevant text. The large parameter count of GPT-3.5 frameworks also contributes to their ability to capture intricate patterns, syntactic structures, and semantic relationships in human language. The ChatGPT model was initially trained on a massive corpus of publicly available text data from the internet. Similar to BERT, it takes advantage of the transformer architecture but makes use of multi-task unsupervised learning in pre-training. Consequently, the input text returns the specific NLP task the model has to perform, making the GPT frameworks useful in instruction-based models. By providing the model with specific instructions or prompts, developers can guide the model to perform desired tasks or generate outputs according to specific requirements. This instruction-based approach allows for fine-grained control and customization of the model's behavior. One notable capability of GPT is its ability to perform few-shot demonstrations, which involves training the GPT-based model on a limited amount of example data to perform a specific task. While GPT frameworks excel in pre-training on a vast corpus of data, they can also be fine-tuned on narrower, more specific tasks with a few examples. Using these capabilities, developers can create specialized models for specific applications or domains using only a handful of example data points.

# Chapter 3 Literature Review

In this chapter, we will present the findings derived from our initial literature review. Firstly, it will present some valuable findings about current hate speech management systems. Subsequently, it will offer a review of the current state-of-the-art hate speech detection models, encompassing a comparative analysis of various models. Additionally, we will explore the challenges associated with identifying hate speech, investigate the utilization of contextual information in hate speech detection, examine the issues related to hate speech datasets, and finally explore the system properties of a hate speech management system. The literature review done in the specialization project [105] preceding this thesis still holds. While certain sections in this thesis draw from the previous project, they have mostly been modified and expanded to better align with the specific focus of this thesis.

## 3.1 Current Hate Speech Management Systems for Social Media Platforms

As mentioned in the previous chapter social media companies utilize various strategies in establishing their hate speech management systems. Information about these specific hate speech management systems has traditionally been kept private both due to competitive advantages and privacy issues. There is, however, some publicly available information about community guidelines for such systems and the actions taken after detection. In April 2023, Twitter implemented new community guidelines and enforcement measures [28]. Twitter now prioritizes the significance of free speech by predominantly reducing the visibility of content rather than outright removing it from the platform. Instances that warrant removal or user suspensions are considered more intricate and undergo a comprehensive examination. This indicates the presence of user reporting mechanisms and human moderation. In addition to this, Twitter made parts of its code database publicly available[1]. In this repository, we can find the training codes for the Trust and Safety models, and specifically the training codes

[1]https://github.com/twitter/the-algorithm

for the models that detect toxic tweets and abusive content. Therefore, it is apparent that Twitter also makes use of automated systems for detection. Furthermore, they emphasize the value of context and how a number of factors, including the severity of the textual content, are taken into account before action is issued [84]. Hence, it is also apparent that there is a need for a detection architecture that is able to analyze more than just the textual content while also being able to pick up on continuously evolving language. The publicly available training codes for the Trust and Safety models indicate the use of both BERT and other customized language models. In order to comprehend the rationale behind this decision, as well as other decisions made in the training codes, it is necessary to conduct a review of the current components pertaining to the detection of hate speech.

## 3.2 Models Used for Hate Speech Detection

The field of hate speech detection has been rapidly evolving over the last few years. Initially, the employed tactics included simpler supervised learning approaches such as logistic regression (LR)[2] or support vector machines (SVMs), which relied on lexical resources like the Bag of Words (BoW)[3] and term frequency-inverse document frequency (TF-IDF)[4]. SVMs are popular classification models that identify the hyperplane that best separates the data points into different classes. With this technique, the model is much less prone to overfitting as it tries to maximize the distance to the nearest training data point, or support vectors, of each class. However, hate speech detection often involves more intricate and nonlinear patterns in the data, which can limit the ability of SVMs to capture complex relationships effectively. SVMs can also be computationally intensive when dealing with larger datasets which is often the case for hate speech datasets.

In recent years, research concerning the use of deep learning to detect hate speech has emerged and has yielded state-of-the-art results [7]. By training on large amounts of labeled data, deep learning frameworks can learn to identify hate speech in text accurately. Several of these studies have explored the use of recurrent neural networks (RNNs), such as Long Short-Term Memory (LSTM), and convolutional neural networks (CNNs). These frameworks are particularly effective in capturing sequential dependencies in text data, making them well-suited for tasks like language modeling, speech recognition, and machine translation. RNNs can process input sequences of variable length and learn to generate meaningful representations of text, capturing contextual information and improving the quality of NLP tasks. However,

[2]LR is a statistical model that is suitable for binary classification as it can be interpreted as a probability for an input belonging to a certain class.

[3]BoW is a simple representation of the presence of specific words that maps a word to its frequency.

[4]TF-IDF returns a weight representing the number of occurrences of a term in a single document, scaled by the inverse of the number of documents it appears in.

these frameworks still require large amounts of training data and computing power, which can be costly and time-consuming. In addition, the lexical detection methods the deep learning frameworks used still tended to have low precision as they were dependent on specific words in their categorization [14].

### 3.2.1 State-of-the-Art Models

Since the introduction of transformers, recent studies on hate speech detection have indicated that transformer-based embedding methods fine-tuned for hate speech detection outperform the previous state-of-the-art approaches [42]. Today, there are various models making use of the transformer. Mutanga, Naicker, and Olugbara [49] showed that the DistilBERT transformer-based model outperforms other transformer-based models and an attention-based LSTM in detecting hate speech, while also allowing parallelization. Additionally, as transformer-based models utilize transfer learning, the size of the training data necessary is greatly reduced. DistilBERT is, as the name suggests, a distilled version of BERT introduced in 2019 by researchers from Hugging Face and the University of Ottawa [64]. It retains 97% of the original BERT model's performance using only half the number of parameters [64]. The training time, therefore, is greatly reduced, while the parallelization makes it possible to learn effective trends toward the classification of hate speech in resource-constrained environments.

Similar to DistilBERT, other methods have been presented to improve BERT and have consequently also been fine-tuned for hate speech detection. RoBERTa, a Robustly optimized BERT approach, was introduced by Facebook earlier in the same year as DistilBERT [39]. Rather than focusing on reducing the inference time, the researchers have focused on increasing its prediction metrics by re-training the BERT model using more data and computing power. Consequently, it outperforms both DistilBERT and BERT.

ALBERT (A Lite BERT) is another transformer-based framework and NLP model that was introduced by Lan et al. in 2019 [35]. ALBERT is designed to be a more parameter-efficient variant of BERT. This is done through a technique named cross-layer parameter sharing, i.e., sharing parameters between layers and reducing the number of parameters overall. The original BERT model consists of 110 million parameters (BERT-base) or 340 million parameters (BERT-large). ALBERT, on the other hand, achieves a reduction of 18x to 57x in parameter count compared to BERT. For example, ALBERT-base may have around 12 million parameters, while ALBERT-large could have around 18 million parameters. However, this requires longer training times to ensure that the shared parameters are learned correctly, which further increases the training time and computational cost. Despite this, studies have proven that ALBERT performs as well as, and in some cases better than, all the mentioned models [101] in detecting hate speech. However, this is observed only when the parameter count is increased to a value that closely aligns with the number of parameters in BERT.

In recent years, the emphasis has shifted towards improving not just the size but also the training efficiency. XLNet is an extension of BERT that is trained using a permutation language modeling task rather than the MLM task [102]. In this task, the XLNet-based model is trained to predict a sequence of words by considering all possible permutations of the words in the sequence. This enables XLNet to capture long-range dependencies between words and to model interactions between them in a more effective way than BERT while maintaining the performance in hate speech detection [48]. Similarly, ELECTRA is a pre-training framework that improves the efficiency and effectiveness of training by using a modified version of the generator-discriminator architecture [11]. Instead of MLM, ELECTRA trains a generator model to replace some of the words in the input sentence with plausible alternatives and a discriminator model to determine if each word in the modified sentence is original or replaced. Malik et al. [42] show that while BERT and ALBERT perform slightly better than ELECTRA on most hate speech benchmark datasets, ELECTRA achieves among the best classification accuracy while being sufficiently computationally efficient.

### 3.2.2 Comparison of Transformer-Based Models

Based on the above-mentioned findings, we will now compare the described pre-trained transformer-based models based on key attributes that hold significant value for social media companies. Specifically, we will assess the models in terms of their size, inference time, and performance by referencing their corresponding papers.

By using pre-trained transformer-based models a social media company is able to leverage their contextual understanding and transfer learning capabilities to effectively capture complex patterns and nuances within the limited data, yielding more accurate and reliable results. Furthermore, transformer-based models exhibit lower computational demands compared to directly employing RNNs, CNNs, or SVMs, which will reduce costs. Hence, the careful selection of an optimal pre-trained transformer-based model that strikes a balance between performance and cost is of relevance for a social media company aiming to effectively manage hate speech on its platform while maintaining cost efficiency.

As there exist multiple versions of each of the above-mentioned pre-trained models, we opt to utilize the base versions as it provides a reliable representation of the model's characteristics while being moderately sized. Additionally, it should be noted that we compare the models in their original form, without fine-tuning them for hate speech detection or incorporating a classification layer. It is possible that the outcomes may differ when considering this fact, but as the same classification layer and the fine-tuning process would have been applied to all the models mentioned, any changes in the sizes would have been similar for all models.

The above-mentioned studies [101] [42] [48] [49], indicate that the transformer-based models of our comparison exhibit similar performance when used for hate speech detection. However, as they have not made use of the same dataset, these performance scores are not comparable. Consequently, our comparison includes model sizes and their corresponding performance on the GLUE benchmarking tasks as reported by the authors of the models. The GLUE benchmarking tasks are a collection of various NLP tasks that have been widely utilized to assess the effectiveness of NLP models and the benchmark score represents an average performance on all these tasks [94]. While the majority of tasks in the GLUE collection are not directly relevant to detecting hate speech, the performance in these types of tasks can still affect the ability to detect hate speech. Therefore, we include the benchmark average scores for these tasks. In order to further understand the hate speech classification capabilities of the models, we also specifically include the results of the Stanford Sentiment Treebank (SST-2) task. This particular task is focused on binary classification and aims to differentiate a sentence's sentiment as either positive or negative.

| **Model** | **Parameters** | **Size**[1] | **GLUE Benchmark Score** | **SST-2 Score** |
|---|---|---|---|---|
| $\text{BERT}_{\text{BASE}}$ | 110M | 440 MB | 79.6% | 93.5% |
| XLNet | 110M | 440 MB | 88.4% | 93.35% |
| $\text{RoBERTa}_{\text{BASE}}$ | 125M | 477 MB | 86.35% | 94.8% |
| $\text{DistilBERT}_{\text{BASE}}$ | 66M | 252 MB | 77% | 91.3% |
| $\text{ALBERT}_{\text{BASE}}$ | 12M | 46 MB | 80.1% | 90.3% |
| $\text{ELECTRA}_{\text{BASE}}$ | 110M | 440 MB | 83.5% | 93.4% |

[1] Approximate size when implemented in the PyTorch deep learning library

**Table 3.1:** Comparison of transformer-based pre-trained NLP models evaluated on hate speech detection tasks.

### Model Size and Performance

Considering the vast number of users and the high frequency of tweets on Twitter, it is reasonable to assume that the memory usage of each of the compared models is comfortably below Twitter's available resources. However, the platform might run on devices with different storage capabilities and multiple instances of the model might be necessary, as we will describe in later subsections. Therefore, it will still be ideal to choose a model that requires a minimal amount of resources while maintaining high performance. The size of a hate speech detection model has a significant impact on the cost of running a hate speech management system. Generally, larger models require more computational resources and memory, which results in increased hardware costs. If a GPT-3-based model is used in its entirety, it utilizes 175 billion parameters, requiring around 660GB to store [9]. Employing expansive generalized models, such as GPT-3, for the objective of detecting hate speech may thus be excessively comprehensive and computationally demanding.

Using BERT, XLNet, and ELECTRA requires about 440MB of storage, while RoBERTa requires around 477MB. Considering its high performance and smaller size, we concluded that XLNet was a better choice than RoBERTa. As stated earlier, DistilBERT focused on reducing the number of parameters [64], and consequently, the model is reduced even more in size. However, although smaller in size, DistilBERT runs the risk of lower performance. ALBERT uses the least amount of parameters and consequently the least memory usage. However, as can be observed in Table 3.1, the base version of ALBERT will have lower performance than the rest. Consequently, in regard to performance and size XLNet is seemingly an optimal choice.

### Inference Duration in Relation to Model Size

Apart from the size, the inference time is also a critical factor for a social media company as it induces a cost. The inference time is closely linked to the model architecture used. Assuming identical hardware and software specifications, BERT and XLNet will have the lengthiest inference times, with BERT being slightly faster than XLNet. As reported by the authors of the ELECTRA paper, ELECTRA is known to have faster inference times compared to BERT and XLNet while achieving similar or better performance, hence a more cost-efficient training procedure [11]. The DistilBERT authors have reported even lower inference times due to their distilled architecture [64]. Specifically, they report that the inference time on one of the GLUE downstream tasks was only 60% of the time required by the base version of BERT. But as stated, DistilBERT's lower performance will not be ideal. Hence, ELECTRA will therefore be the optimal choice in terms of inference times.

On the other hand, considering the requirements set forth by the Digital Services Act which mandate the timely removal of hate speech from social media platforms, it is advantageous to utilize multiple deployed models. This approach, which will be described in later sections, ensures greater efficiency in meeting time constraints and effectively addressing the issue of hate speech. As the increased cost of longer inference times can be justified by using multiple deployed models, all the aforementioned models would be suitable for real-time detection. Therefore ELECTRA, renowned for its impressive inference speeds, may not necessarily be the optimal choice, as faster inference speeds alone are not crucial. An optimal model choice for a social media company is therefore one that balances performance, size, and inference time. In our comparison, XLNet is such a model.

#### Challenges Regarding Choice of Model

Although the current state-of-the-art hate speech detection models reach F1-scores above 93%, there exist over-fitting and sampling issues [2]. Agrawal and Awekar [1] have proven this in their research when using a deep learning model trained on tweets to detect cyberbullying on data from Wikipedia and Formspring. In doing so, they found that transfer learning from Twitter to the two other domains achieves less than 10% $F_1$-score. Similarly, Dadvar and Eckert [13] perform transfer learning from a model trained on tweets to a dataset with YouTube comments, retrieving an $F_1$-score of 15%. However, in a realistic setting, there is not necessarily a criterion that the performance of a model trained on a specific platform is maintained when making use of it on other platforms. To counter this argument, Gröndahl et al. [25] confirms in their study of several state-of-the-art hate speech detection models that the performance degradation is also prevalent when performing transfer learning on a model trained on one dataset of tweets to two other similarly labeled datasets of tweets. Specifically, they experienced a drop from the original 93% $F_1$-score to 33% and 47%. The research done by Gröndahl et al. [25] argues that the "model architecture is less important than the type of data and labeling criteria" being used. Hence, as research utilizes datasets that lack complete comparability, it often becomes challenging to arrive at a definite conclusion regarding the most suitable model to employ.

## 3.3 Intricacies of Identifying Hate Speech

Kovacs et al. [31] discuss why automatic hate speech detection on social media is difficult. First of all, there are challenges concerning the text itself. For instance, words can be obfuscated (both in an intentional attempt and due to misspellings), expressions can have different denotations (e.g., slurs), and the interpretations can change over time. On top of that, implicit hate speech is often masked as not being hateful at all. For instance, disguised hate speech often is seemingly innocuous by using pseudo-intellectual arguments or humor/satire, but in reality, it serves to justify discrimination and perpetuate harmful stereotypes as seen in Table 3.2.

The definition of hate speech itself also poses a challenge as it often varies and remains open to interpretation. Various definitions have been proposed, all of them subject to the cultural norms of the geography in which they are created [74]. In addition, hate speech is commonly grouped under broader terms like *abusive language*, *offensive language*, or *toxicity* [61]. Consequently, models have a tendency to recognize patterns that prioritize the more frequently appearing categories, such as *insult*, and demonstrate lower proficiency in identifying hate speech [22].

Another important aspect of hate speech is that it follows a long tail distribution

| **Hate Speech Type** | **Examples** | **Notes** |
|---|---|---|
| **Explicit** | | |
| Racial slurs | "Wop" | Denotes racial hatred |
| Homophobic insults | "Faggot" | Targets individuals based on sexual orientation |
| **Implicit** | | |
| microaggressions | "You're so articulate for a black person" | Disguised as compliments, but reinforce stereotypes |
| dog whistles | "Urban crime" | Coded language used to express prejudice |
| **Disguised** | | |
| pseudo-intellectual arguments (e.g., scientific racism) | "Biologically white people are more evolved" | Use of pseudo-science to justify discrimination |
| humor/satire (e.g., racist jokes) | "A lot of dads in the hood likes to go and get milk" | Cloaked as humor, but perpetuates stereotypes |

**Table 3.2:** Taxonomy of hate speech, categorized into explicit, implicit, and disguised types, along with their specific characteristics and implications.

in regard to uniqueness. Zhang and Luo [103] discuss how a large portion of hate speech is unique and does not share the same characteristics of other hate speech, such as using commonly used hateful words and compositions. Models without the means of translating linguistic sequences to logical arguments would in such cases be unable to establish common traits that make unique hateful content identifiable. They end their paper by discussing possible future work where they suggest the inclusion of contextual information to achieve higher accuracy in the detection of tweets from the long tail.

## 3.4 Usage of Context in Hate Speech Detection

The concept of leveraging context for deeper insights is not a novel notion. Nonetheless, making use of the available context is no easy task. Markov and Daelemans [43] prove that the performance of a transformer-based pre-trained model significantly improves by adding relevant annotated context. This is in contrast to previous research done by both Pavlopoulos in 2020 [59] and later Menini in 2021 [44]. In both papers, they conclude that while context notably affects the annotation process, as fewer tweets were annotated as abusive when the context was provided to annotators, it did not yield any improvements in performance. Markov and Daelemans, on the other hand, point out that the shortcomings of their peers were due to the lack of relevant contextual information. Their peers' work had included only previous comments or posts which proved not to be of significant value. The challenge is thus discovering relevant context that can provide the machine learning models with the necessary data to create trends towards the classification when the tweet alone is not sufficient.

To discover relevant context, testing with different combinations of the available contextual features is necessary. Unsvåg does exactly this in her Master's thesis [92]. Unfortunately, the thesis did not yield any decisive results and concluded that only the "Network" feature (i.e., a combination of the features "number of followers" and "number of friends") caused some improvement to the classifier's performance. Unsvåg points out that a limitation of her research was the Twitter API itself (specifically Tweepy, a Python library for accessing the Twitter API) as it did not make all the user features publicly available. The features that were available through Tweepy proved mostly not to be of value. In particular, Unsvåg makes use of non-textual features that are represented as counts or booleans (e.g., gender, number of favorites, whether the geographical position is enabled or not, and number of followers and friends) which proved to be inconsequential in influencing the accuracy of the machine learning model. Features containing information about the parent tweet of a reply, the user description, or the actual geographical location of a tweet, as depicted in Figure 3.1, were not directly made available by Tweepy at the time Unsvåg did her work. However, as such features contain more information than counts and booleans, we believe making use of such information will have a positive impact on the performance of the model.

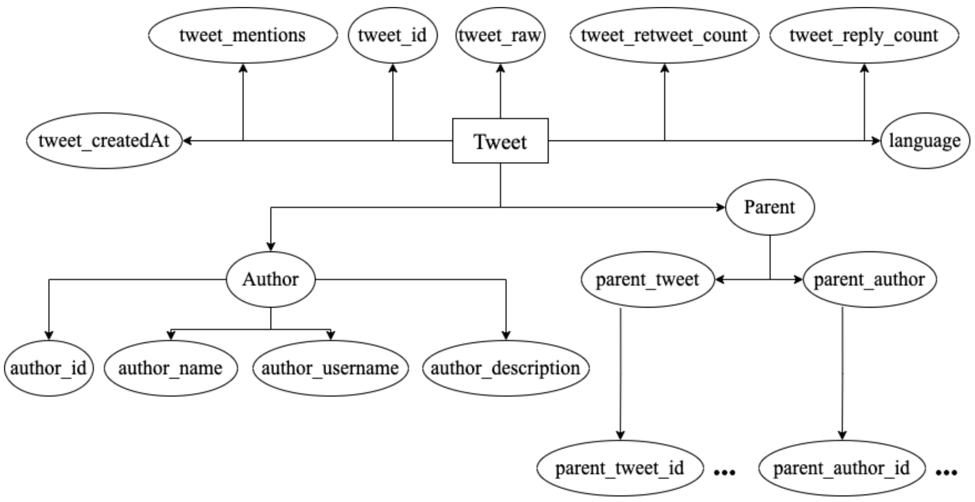


**Figure 3.1:** Hierarchical breakdown of properties of a Tweet object which can be regarded as contextual information in hate speech classification.

## 3.5 Hate Speech Datasets

To analyze the added value of contextual information in the classification problem, the dataset will have to include relevant context while also being labeled using a fair and unbiased annotation scheme. Initially, the majority of datasets categorized content as either hateful or non-hateful using binary classification. As hate speech has become more prevalent the need to have more detailed categories has emerged, resulting in more nuanced labels (e.g., offensive, racist, sexist). Regardless of the number of labels, most datasets exhibit an uneven distribution of non-hateful and hateful speech, as abusive tweets are relatively uncommon (typically ranging from 0.1% to 3%, depending on the label) [23]. This disproportion ultimately leads to inferior classification performance due to the limited training opportunity on specific features.

### 3.5.1 Challenges of Annotation

The challenges of annotation are also a concern. The challenge of annotating hate speech is mainly due to the obscurity of the definition of hate speech itself, as mentioned above. A social science study performed by Brown [8] states that "hate speech is now often used as an umbrella label for all sorts of hateful/insulting/abusive content", and that the annotators could incorrectly label hate speech through their own biases if they are not provided with a precise definition.

Davidson et al. [14] outline that hate speech is a subtype of the broader sphere of offensive language and that consequently these terms are often used interchangeably. Hence, they had the annotators categorize the instances as either "hate speech", "offensive speech", or "neither offensive nor hate speech". As a result, they found a bias in the annotation; racist or homophobic instances were mostly categorized as hate speech, and sexist instances were mostly categorized as offensive. Furthermore, instances that were earlier labeled as hate speech in research done by Waseem et al. [97] only gained the label offensive but not hate speech. Waseem, Davidson, et al. [96] then collaborated on identifying the sources of the annotation confusion. In their research, they identified that there were misunderstandings about whether there existed abusive language directed toward some generalized group as opposed to a specific individual.

Research done by Talat et al. in 2022 [74] implies that socially biased systems are still a concern within NLP. As an example, in the labeling of hate speech datasets, the annotators must make assumptions regarding the intent of the author. Hence, there might exist contradicting labels within a dataset after a round of annotation. The Inter-Annotator Agreement (IAA) is frequently used to assess the consistency of labels within a dataset. When acceptable IAAs have been reached the majority label is used to acquire a ground truth. However, the term "majority" may not be as definitive as desired in this context. Frequently, researchers depend on one single label as the ground truth, hence there is a risk of neglecting the lack of consensus, fluctuations, and subjectivity that may arise from the obtained ground truth [58]. In this manner, the majority label reached using the IAA is often affected by the subjectivity of an annotator.

### 3.5.2 Challenges of Large-Scale Crowdsourcing

The work of Founta et al. [23] focused mainly on labeling large datasets effectively while keeping the IAA low. This was done by introducing a novel (at the time of publishing) methodology to detect and mitigate the obscurity of label assignments for the annotators. With this methodology, they produced a dataset of 80,000 English tweets while minimizing the cost and, in their opinion, maintaining the quality of the annotation. To address the possible lack of annotation of minority classes they designed a boosted random sampling technique, in which text analysis and preliminary crowdsourcing rounds were used to design a model that can pre-select tweets to fit into the chosen minority classes. Then the boosted set and randomly sampled set are mixed to create a more balanced dataset. Founta et al. [23] point out that these preliminary rounds also could reduce the ambiguity of the labels. The labeling itself used between 5 and 20 annotators per tweet, who were all paid according to a specifically designed payment plan.

The examples depicted in Table 3.3 show the range of what is labeled as hateful in the dataset of Founta et al. [23]. It is evident that the label of hate speech has not been consistently applied. Awal et al. [6] created a framework to analyze the annotation inconsistency in multiple hate speech datasets. The authors discovered that the Founta et al. dataset contained numerous instances of duplicate tweets. Furthermore, some of these tweets were labeled with opposing labels, which added to the dataset's inconsistency. In addition, Founta et al. do not mention anything about their regard to the crowd workers' subject matter expertise or social and cultural backgrounds. If not adequately taken into consideration, the lack of subject matter expertise and cultural backgrounds can affect the labeling. In the Founta et al. dataset this is especially evident in the tweets regarding politics, as tweets discussing Donald Trump are often labeled as hateful regardless of their sentiment. The human annotators' beliefs and identities ultimately bias the labeling [65].

There is also reason to believe that the questionable hate speech labeling is due to a vague definition of hate speech [6]. According to Founta et al., hate speech is defined as "the language used to express hatred towards a targeted individual or group, or is intended to be derogatory, to humiliate, or to insult the members of the group, on the basis of attributes such as race, religion, ethnic origin, sexual orientation, disability, or gender". While this definition aligns with the one commonly

| Tweet | Parent Tweet |
|---|---|
| One guy was hoping Trump kicks out all Mexicans and builds the wall when he himself is Hispanic. People hurt my brain. | I need to stop putting my opinions on Facebook. Any opposition I get seem to be deep in their own ultra conservative holes. |
| @JohnTrumpFanKJV @complxgrl GOD sent TRUMP to help us and to save our Country! And We ELECTED TRUMP! GOD Bless DONALD Trump | The Russians did not cause Donald Trump to win the Presidency. God caused Donald Trump to win the Presidency! |
| @dosnostalgic @YouTube "This video is blocked in your country"... I'm in Canada, this regional-based web media thing is stupid. | I do love the Money for Nothing (1985) music video, which is what eventually lead to ReBoot https://t.co/9xCPGjYt0T |

**Table 3.3:** Subset of incorrectly labeled positives in the Founta et al. dataset, representing instances misclassified as hate speech.

used in legal matters, Assimakopoulos et al. [5] argue that crowdsourced annotation schemes should avoid using the label "hate speech" directly. This is because, as previously stated, different annotators may have varying thresholds for determining what qualifies as hatred and what does not. In their work, they present a novel multi-layer annotation scheme, wherein they pose a series of progressively detailed yes-no inquiries. Given the answers of the annotators, the interested party will then be able to set their own criteria for hate speech (i.e., a combination of answers that define what should be labeled hate speech). For this reason, the interested party will be able to indirectly control annotator disagreement in some cases and adapt what constitutes hate speech. They emphasize that such a scheme still does not eradicate the disagreements as there still exists ambiguity in the text, yet it does result in a notable reduction.

### 3.5.3 Machine-Generated Datasets

Lately, there has been a shift towards generating data points rather than merely collecting and labeling them, aimed at addressing the issue of imbalance, as well as removing the need for labeling. This approach enables the design of targeted data points that effectively train the model for specific use cases. Hartvigsen et al. [27] address this by introducing Toxigen, an extensive machine-generated dataset specifically designed for detecting adversarial and implicit hate speech. In doing so, they tailor hate speech to specific demographic groups and employ distinct implicit formulations of hate speech, thus showcasing the customization aspect of machine generation. To ensure the quality of the dataset, they perform a human evaluation on a challenging subset of ToxiGen, revealing that annotators face difficulties in discerning between machine-generated text and human-written language. Finally, they demonstrate the dataset's use cases by illustrating that finetuning a toxicity classifier on Toxigen significantly enhances its performance on human-written data.

### 3.5.4 Using GPT Models for Data Labeling

As mentioned in the previous chapter, GPT models have obtained impressive few-shot performance on various NLP tasks. Following the debut of ChatGPT, there has been an emerging interest in employing GPT frameworks for data labeling [95] [16].

It is natural to question why GPT frameworks cannot be utilized directly for inference. Due to their substantial size, GPT frameworks are prone to high latency, making them impractical for large-scale inference in NLP tasks. This means that utilizing such a framework for processing a considerable amount of data may result in significant delays, limiting their practicality for such applications. Furthermore, a GPT framework, in its entirety, is not easily accessible to the public (at the time of writing this thesis). Additionally, each GPT-based model's API cost rises proportionally to the number of instances they process during inference. Therefore, it is more practical to utilize this powerful tool to create a ground truth for a smaller, more cost-effective model.

Attempting to generate synthetic data points for an entire dataset using a GPT-based model will lead to subpar classification performance compared to using real data points [46], rendering it impractical. However, a GPT-based model can be used to label real data points, which requires a manageable amount of resources and cost. Employing such a model to label training data for smaller, in-house models that are deployed for inference results in a significantly lower cost (ranging from 50% to 95% less) compared to having humans label the data [95]. In experiments performed by Wang et al., the smaller in-house models trained with labels generated by a GPT-3-based model often outperform the raw GPT-3-based model in terms of accuracy. They also prove that there is a theoretical upper-bound to the error rate of the models trained on the labels generated by the GPT-3-based model, i.e., the error rate of the model trained using labels generated by the GPT-3-based model can be lower than that of the GPT-3-model itself.

In this manner, the labeling evades the bias and inconsistencies of human moderators, however as GPT-3-based models are trained on human-made content, there is a risk that the model also has a bias. To ensure adequate quality labels, Wang et al. propose an active labeling strategy in which human annotators relabel data labeled by GPT-3 that has the lowest confidence scores. This is possible, as GPT-3 also returns the log probabilities for each token together with the generated text that indicates the uncertainty of the proposed label. By defining a threshold logit, human moderators are able to manually relabel the labels that have a substantially high logit uncertainty.

## 3.6 System Properties for Feasibility and Practicality

In this section, we will explore some of the system properties that are important to a hate speech management system. Understanding these properties is crucial for the development and implementation of effective strategies that are necessary for a hate speech management system. By examining the key characteristics and components of such systems, we can gain insights into how they operate and identify potential challenges and opportunities for improvement.

### 3.6.1 Legal Compliance and Data Efficiency

Hate speech detection is a priority for social media platforms due to recent developments in regulation on accountability of content moderation. As stated in the previous chapter, landmark EU legislation approved by the European Council in October 2022, the Digital Services Act, will from the year 2024 upgrade the rules on content regulation on social media platforms [19]. This act might, among other requirements, compel designated social media companies to enforce hate speech frameworks and promptly remove any content that is deemed hateful from their platforms within a specific time constraint. To account for this time constraint, the system must be able to process a high intensity of tweets. Twitter records an average of 5,700 tweets per second on their platform [85]. On a few occasions, the platform has reached almost 10,000 tweets per second. Therefore, a necessary system requirement is the capability to handle such a significant volume of data.

A system using a machine learning model will also have to take into account the ethics and privacy demands of the geographical area it is deployed. Most prevalent are the regulations to protect personal data such as the General Data Protection Regulation in the European Union (GDPR). Under GDPR, personal data must be processed lawfully, fairly, and transparently [21]. In the case of a hate speech detection system, it must have a "lawful basis for processing personal data", such as getting the individual's consent [90]. The system must also provide clear information about the data processing activities to the individuals whose data is being processed. There also exists various laws against discrimination on any platform. Amongst others, these include discrimination based on age, race, sex, and disability [41]. Hence, the individuals should be able to obtain human intervention in the decision-making process, be able to express their point of view, and be able to challenge the decision made by a hate speech management system. The laws, however, differ significantly across various countries and the system must consequently be adjusted depending on the location it is deployed.

### 3.6.2 Robustness Against Adversarial Attacks

The research done by Gröndahl et al. [25], mentioned above, also touches upon the problem of adversarial attacks and the challenges of the protection mechanisms against these types of attacks. They show that various hate speech detection models are vulnerable to attacks such as changing word boundaries, typo insertions, or the addition of harmless words. Similarly, Oak finds that changing the text slightly degrades the performance of a Random Forest[5] hate speech classifier by 20% [52]. In this manner, an adversary might prevent the detection of a hateful user or force the model to classify normal users as hateful [24]. As a consequence, the hate speech management system's integrity will be called into question. The general public's trust in the platform might then diminish and more may become hesitant to engage with or use the platform. Previous research has achieved some progress in protecting against these types of attacks by incorporating adversarial examples, e.g., the examples depicted in Table 3.4, in the training of the model.

| **Original Sentence** | **Adversarial Example** | **Classification** |
|---|---|---|
| go back to where you came from these fucking immigrants are destroying america<br>Score: **95% Toxic** | **Split**: go back to where you came from these fu cking im migrants are de stroying america | **59% Non-Toxic** |
| | **Merge**: go back towhere you came from these fucking immigrantsare destroying america | **54% Non-Toxic** |
| | **Drop (Mid)**: go back to where you came from these fuckin immgrants are destroying america | **63% Non-Toxic** |
| | **Drop (Last)**: go back to where you came from these fuckin immigrant are destroyin america | **52% Non-Toxic** |

**Table 3.4:** Effect of adversarial perturbation through sub-word level manipulation on a classifier's toxicity score. The table is adapted from *Adversarial Examples for Hate Speech Classifiers* by Oak [52].

However, as per the research of Gröndahl et al. [25], "adversarial training does not completely mitigate the attacks" and does not stand the test of time as it only protects against specific types of attacks. In a realistic setting, when models are deployed

[5]A random forest classifier is a machine learning algorithm that combines the predictions of multiple decision trees to make its predictions.

and updated with new non-adversarial data samples, adversarial training will not be sufficient. Omar and Mohaisen study the temporal impact of adversarial training on naturally-evolving language models [53]. While they confirm that adversarial training does significantly improve performance on the specific attacks that are included as examples in the training, they also confirm the problem of adapting to new attacks. Models trained on the same dataset attain high accuracy in their predictions but perform badly when evaluated with other datasets, even after training models with adversarial examples. Hence, they find that adversarial training is task-dependent and dataset-dependent.

Moh et al. [47] propose four different preprocessing defense techniques that effectively prevent certain types of adversarial attacks, reducing the need for adversarial training. The techniques together constitute a framework for detecting and reacting to specific adversarial attacks, Text Scheme Adversary Recognition (TSAR). The defense techniques employed in the TSAR framework capitalize on the attackers' objective to retain the post's readability while evading detection mechanisms. Attackers seek to convey a message while simultaneously avoiding detection. Specifically, the TSAR framework protects against the removal of whitespaces, intentional typographical errors, and the "Love" attack. The love attack involves strategically inserting a word with an extremely positive sentiment, thereby tricking the hate speech model into failing to detect the underlying hate speech within the post. The techniques employed within the TSAR framework effectively mitigate these types of attacks by implementing diverse methods to standardize the input. This includes separating composite words, removing words that violate the grammar of a sentence (if it is not classified as negative by a sentiment analyzer), and building a dictionary of misspelled words for correction. Finally, the training is done on the cleaned data. Though such a framework reduces the need for training on adversarial examples, the problem of adaptability still remains as the defense techniques only protect against specific types of attacks.

### 3.6.3 Transparency and Auditability

Transparency encompasses the essential aspects of interpretability and explainability. In regards to a machine learning system, interpretability refers to the ability to understand and make sense of the internal workings of a machine learning model or system. Explainability, on the other hand, focuses on providing understandable explanations for the outputs or decisions made by a machine learning model. Being able to interpret the results of a model in terms that are understandable in a business context is often crucial in the process of selecting a model. In fact, this ability can even be more important than performance considerations [55]. In the context of the transformer architecture, interpretability, and explainability can be challenging due to its complex structure and attention mechanisms. Their inner workings are often less transparent compared to simpler models like decision trees or linear regression. Decision Trees are therefore often used in practice as their transparent branching logic can be used for auditing [26]. However, the attention mechanism in the transformer architecture can provide insights into which parts of the input text are most significant for making predictions. Specifically, the attention mechanism allows the model to assign different weights or importance to different positions or tokens in the input sequence. This impacts both explainability and interoperability, as analysts can gain insights into the model's decision-making process and provide human-friendly explanations based on attention patterns. It's important to note that while the attention mechanism provides valuable interpretability and explainability, it may not be sufficient on its own for complete transparency [93].

In the case of hate speech detection, the need to understand why the model classifies an instance as hate speech will provide an administrator the ability to better fine-tune the model. Alongside the measures aimed at combating illegal content online, the Digital Services Act emphasizes the need for transparency measures and effective user safeguards [20]. These include provisions for users to challenge content moderation decisions made by platforms. Being transparent about the reasons for flagging can help educate users about what constitutes hate speech and encourage them to be more mindful of their language. Additionally, providing a reason for the flagging can also help build trust between the social media company and its users, by demonstrating a commitment to promoting a safe and respectful online environment.

The Alan Turing Institute highlights the ethical factors and governance processes that should be considered when deploying AI to the public [37]. Amongst others, they usher the need for human responsibility throughout the whole AI project delivery chain. They point out that as machine learning models leverage existing data to make decisions they can be susceptible to inheriting latent biases that already exist within the data. When it comes to detecting hate speech, the datasets utilized are not only imbalanced in terms of the degree of being hateful, but they can also exhibit biases related to factors such as race, gender, and so on. These biases can be exploited by the machine learning model. Consequently, having the ability to comprehend the reasoning behind a model's decision is advantageous for both the user and the company.

Due to the recent demand for transparent models, an emerging field is eXplainable AI (XAI). The field focuses on creating AI systems that are transparent and in which the model's output is explainable with human language [17]. Various methods have been proposed to better understand a model's decision. Analyzing the value of the different features given as input to a model is one such method. However, language models typically operate on sentence embeddings, which represent the semantic meaning of the entire sentence rather than individual features. Due to the nature of sentence embeddings, dissecting and attributing importance to specific features within a sentence becomes less straightforward. Traditional feature importance analysis techniques, such as feature weights or gradients [72], may not directly apply in this context. Instead, alternative approaches must be explored to unravel the decision-making process of language models. Hendricks et al. [29] devised a model capable of generating both a predicted label and an accompanying explanation justifying its appropriateness specifically for the classification task of visual recognition. Such methods will increase the transparency as the model becomes more interpretable.

### 3.6.4 Maintainability and Consistent Performance

Monitoring can be used to ensure consistency in the model's behavior after deployment to a platform. This includes monitoring the evolving input data, possible classification bias, and overall performance. Sculley et al. [67] address the issue of hidden technical debt in machine learning systems, emphasizing the significant and ongoing maintenance costs that these systems can accrue when deployed in real-world scenarios. In their paper, they highlight that machine learning systems, which rely on external data, may be susceptible to feedback loops and correction cascades. Feedback loops occur when a machine learning model makes decisions that affect the data it receives, and these changes further influence the model's output. This can result in a reinforcing cycle that magnifies any biases or errors in the system, leading to unreliable results. Correction cascades can occur when a machine learning model makes a mistake and attempts to correct it, but these corrections lead to further errors and require more corrections, leading to a snowball effect that can be difficult to control. This highlights the importance of ensuring that machine learning systems are designed with appropriate safeguards to prevent these types of feedback loops and correction cascades from occurring. By monitoring its performance in a production environment, we can also deploy such safeguards in real-time. In practice, this is done by collecting data in multiple stages of the lifecycle, so for both the upstream and the downstream data analysis. Upstream data analysis in the context of hate speech detection refers to the analysis of data collection and necessary preprocessing. Likewise, downstream data analysis refers to the analysis of the training, evaluation, and deployment of the model. In the case of hate speech, it is important to analyze the data against known features that could bias the model, such as race, gender, age, income groups, etc., to ensure model fairness.

### 3.6.5 Adaptability, Availability, and Scalability

As stated earlier, hate speech is a dynamic area of speech that is constantly evolving, i.e., the rate of the linguistic shift occurring in online social media is high [30]. This is not ordinarily accounted for after training a model once. With this in mind, it is essential to fine-tune the model continuously (i.e., sequential fine-tuning) to ensure consistent performance over time. In addition, the previously mentioned process of feedback loops, i.e., incorporating feedback from the model's output back into its input, can be used intentionally. As long as the loops are designed carefully, such a process allows the model to be adjusted to better fit the data and produce more accurate predictions [89].

In addition, the evaluation of the trained model will also have to reflect the model's adaptability to linguistic shifts. A standard for evaluating the trained model is, as we will demonstrate in the subsequent chapter, by evaluating with a subset of the initial dataset. Such a standard for evaluation assumes that the train set and the data encountered when the model is deployed are both independent and identically distributed (IID) [3]. Failure to implement measures to increase the likelihood of the IID assumption can lead to over-fitted models lacking robustness, as demonstrated by Wiegand et al. [99] and Rahman et al. [62]. However, in the case of hate speech detection in which the ratio between hate and non-hate is imbalanced, a sampling strategy to balance the classes might be needed [70]. Datasets of this type present a significant challenge to classifiers, as accurately classifying the minority samples becomes problematic due to their small number. Ordinarily, oversampling techniques, such as randomly duplicating instances from the minority class until it is balanced with the majority class, are used [98]. However, this results in additional preprocessing steps which might increase the latency of the system.

Regardless, to ensure that the model is trained on a dataset that has a distribution that is both independent and identical to the distribution of the data it will encounter during deployment while maintaining enough instances for both classes, it is essential to account for this in the system. In regards to software engineering, accounting for such aspects is the field of continuous delivery (CD). CD is a widely used technique in software engineering that aims to streamline the software development process. This approach involves creating an automated pipeline for building, testing, and deploying software changes, which helps to accelerate the development cycle and improve the overall efficiency of the workflow.

Sato et al. [66] established a framework demonstrating how CD can be utilized in machine learning projects. They emphasize the distinction between offline models and online models. Offline learning, also known as batch learning, involves training the model on a fixed dataset before making predictions. In this approach, the model is trained on a fixed set of examples and cannot learn from new data as it becomes available. Online learning models employ algorithms and techniques that allow them to continuously enhance their performance with the influx of new data, constantly learning in a production environment. This is further proven in the work by Kading et al. [32], concluding that "continuous learning can be directly achieved by continuous fine-tuning". This is necessary in environments where data arrives continuously, as is the case in hate speech detection.

Additionally for hate speech detection, an up–to–date model is necessary every time leaving no room for waiting until the model is fully adjusted. Hence there is a need for mechanisms to increase the availability of the model. Sato et al. propose the utilization of multiple models, with the incoming data being distributed among them. They mention that a model can be deployed in multiple ways. In some cases, it is possible to have multiple models performing the same task, while other cases might necessitate models performing different tasks. As an example, one could train separate models to be specialized in different types of hate speech (e.g., sexism, racism). In this manner, the scalability of the system can also be achieved. In evaluating each model they also emphasize the need to evaluate model performance against multiple data slices to uncover biases that may exist due to imbalanced training data. While overall test and validation results may appear satisfactory, examining performance across various data subsets is necessary to ensure a fair representation of the real-world distribution.

### 3.6.6 Effects of Detection Cost

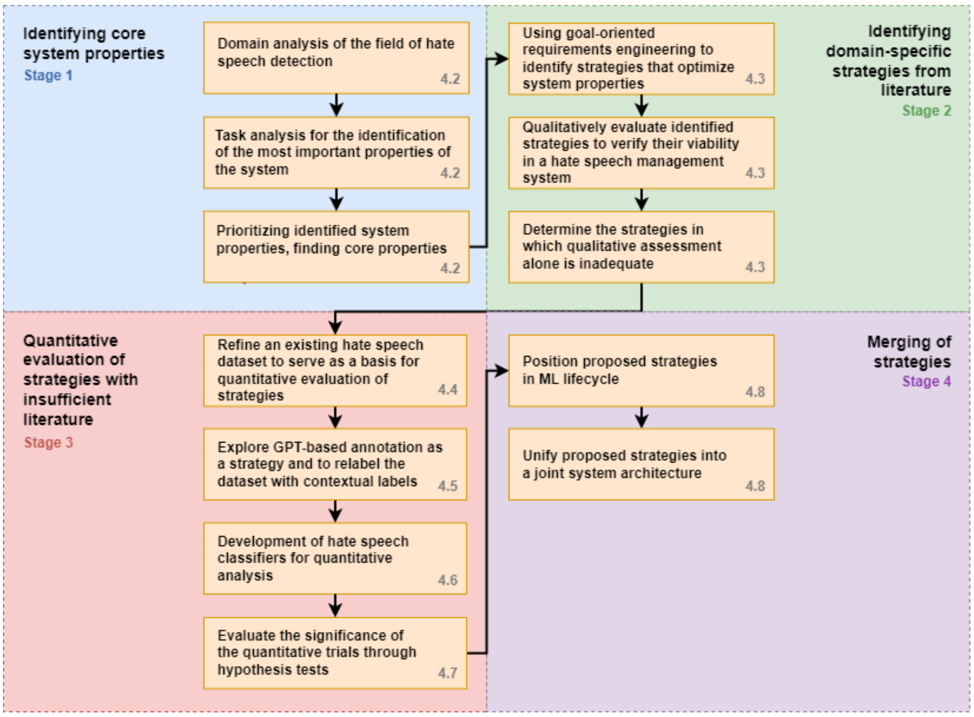
The model training stage incurs a significant economic cost due to the computational resources required for the training procedure [69]. In a study analyzing the cost of NLP models, BERT's full training procedure resulted in a cost anywhere between \$50K and \$1.6M in cloud computing resources depending on the chosen model size. The training set size, the number of parameters in the model, and the number of operations are all factors that impact this cost. Not only is cost a concern but also the impact computationally intensive training has on the environment. Strubell et al.'s research on the environmental impact of training machine learning models [71] found that a single training cycle utilizing a neural framework emits an amount of $CO_2$ equal to what four average cars emit over their entire lifetimes. The authors emphasize the importance of researchers being aware of the environmental impact of model training and advocate for the community to prioritize computationally efficient hardware and algorithms.

Due to the significant associated expenses the utilization of machine learning remains limited, thereby hindering its optimal deployment. Parker and Ruths, active researchers in the field of hate speech detection for the past decade, point out that "there is a profound disconnect between the computer science research community and other stakeholder groups " [57]. Their research suggests that "outside of computer science, there is virtually no discussion of automated hate speech detection as a tool for mitigating hate speech or for any other use. Despite this, they state that there is still a demand for a multi-stakeholder, holistic solution. The current emphasis on improving the performance of hate speech detection models may have led to a relative neglect of their practical applications for stakeholders. As mentioned in our background, most social media platforms have established their own definitions, guidelines, and policies to tackle hate speech. As a consequence, many social media platforms still heavily rely on user moderation [91]. Twitter makes use of global review teams that are continuously trained on the latest hate speech trends and variations [84]. Therefore, the issue of scalability persists as manual labor poses a significant obstacle to efficient moderation processes.

Chapter 4

# Methodology

In this chapter, we present the methods and design choices we employed to address the subgoals and the research questions by using the research approach and design described in the introduction. Figure 4.1 presents a detailed roadmap of the methodology, grouping the methods utilized in stages that reflect their related nature in the chapter.



Identifying core system properties
Stage 1
Domain analysis of the field of hate speech detection 4.2
Task analysis for the identification of the most important properties of the system 4.2
Prioritizing identified system properties, finding core properties 4.2
Using goal-oriented requirements engineering to identify strategies that optimize system properties 4.3
Qualitatively evaluate identified strategies to verify their viability in a hate speech management system 4.3
Determine the strategies in which qualitative assessment alone is inadequate 4.3
Identifying domain-specific strategies from literature
Stage 2
Quantitative evaluation of strategies with insufficient literature
Stage 3
Refine an existing hate speech dataset to serve as a basis for quantitative evaluation of strategies 4.4
Explore GPT-based annotation as a strategy and to relabel the dataset with contextual labels 4.5
Development of hate speech classifiers for quantitative analysis 4.6
Evaluate the significance of the quantitative trials through hypothesis tests 4.7
Position proposed strategies in ML lifecycle 4.8
Unify proposed strategies into a joint system architecture 4.8
Merging of strategies
Stage 4

**Figure 4.1:** Detailed roadmap of the methodology, enumerating each step with its corresponding subsection. The arrows represent the sequential manner in which the methods are presented in this chapter, reflecting the progression of our thesis work.

In the upcoming sections, we will first describe the methodologies used to identify the core system properties of our scoped hate speech management system. Subsequently, we will outline the methods employed to identify domain-specific strategies aimed at optimizing these system properties, followed by a qualitative evaluation to verify their effectiveness in optimizing the desired system properties. Then, we will delve into the methodologies utilized to quantitatively evaluate the strategies where the qualitative evaluation was insufficient. This entailed the development of a dataset that included contextual information in which we also evaluated an annotation strategy, the development of models needed for the quantitative experiments, and finally the evaluation of two other strategies using these models and dataset. Lastly, we will explain the method used to integrate the formulated strategies into a generalized system architecture. In doing this, we will utilize the results from both the qualitative and quantitative evaluations of the identified strategies.

## 4.1 Computing Environment and Resources

Computationally-intensive processing tasks such as inference and hyperparameter optimization in the quantitative experiments were done on a 20.04 Ubuntu server with an AMD EPYC 7443 24-Core @ 2.85GHz CPU, 60 GB of RAM, and a 1 TB disk. For GPU acceleration, we used an NVIDIA GRID A100-10C vGPU with 10 GB VRAM and CUDA version 11.4.

We used a virtual Python 3.9.1 environment to ensure the reproducibility and consistency of our research. Exploratory development was performed using Jupyter Notebooks to facilitate interactive and dynamic modeling of data and allowed for rapid iteration and refinement of code. We leveraged the Natural Language Toolkit (NLTK) library [40] for various natural language processing tasks, the Scikit-learn library [81] for machine learning analysis, the PyTorch deep learning library [80] for deep learning tasks, and the Transformer [82] library to implement transformer models.

## 4.2 Identifying Core System Properties

In order to identify the most relevant system properties of the detection architecture of a hate speech management system and the corresponding strategies to optimize them, we used theory from requirements engineering (RE) which is the process of defining, documenting, and maintaining requirements [51]. In RE there are ordinarily two processes, namely gathering and implementation. Our research involved both processes as we both identified system properties of the detection architecture and later suggested strategies to optimize the identified properties. The methodology for identifying and suggesting strategies will be presented in the next section. This section will present the methodologies presented to gather system properties, i.e. obtain the system properties of the detection architecture of a hate speech management system.

While the theory of RE aids in determining system properties, it is essential to acknowledge that system requirements and system properties are separate and distinct concepts. System requirements are specific and measurable, while system properties represent essential and desirable attributes of the system. System requirements serve as a concrete manifestation of system properties to develop a particular system. Our research focuses on proposing a generalized system architecture with strategies to optimize system properties, rather than developing a specific system. Meeting these system properties to a precise degree requires defining system requirements, which falls outside the scope of this thesis.

In addition, since the aim of our research is to analyze strategies for this generalized architecture rather than produce an exact solution, we employed a traditional linear RE methodology [68] to identify relevant system properties. It is worth noting that the system properties of a hate speech management system are dynamic in nature and can change. A linear RE methodology will not account for evolving system properties but will suffice to establish the mentioned generalized architecture.

Before the gathering of system properties, we first conducted a domain analysis [104] in order to gain a more comprehensive understanding of the field. In doing so, we conducted research on various aspects of the field of hate speech detection and previous hate speech management systems. This included analysis of existing system information, organizational standards, government regulations, and the demands of stakeholders. These analyses are further elaborated on in our literature review.

To identify relevant system properties of the detection architecture of a hate speech management system using the information gathered from the domain analysis, we employed a task analysis approach. This involved examining the top-down task hierarchy of the detection architecture and categorizing the literature from the domain analysis into the different steps of this hierarchy. By dividing the information gathered from the domain analysis into such a hierarchy it was much easier to identify relevant information. This approach is well-suited for our case, given that the detection architecture of a hate speech management system involves a pipeline of several stages. Specifically, we employed the machine learning lifecycle definition proposed by Ashmore et al. [4] to retrieve a top-down task hierarchy of the detection architecture. As described in the background chapter, this hierarchy comprises the stages of data management, model learning, model verification, and model deployment. By adopting this approach, we were able to isolate different stages of the system and pinpoint properties related to different aspects of the specific stages. In practice, this involved documenting all the system properties of the different stages that were mentioned in the literature we reviewed.

The following system properties were mentioned: legal compliance, scalability, adaptability, customizability, robustness against adversarial attacks, cost, transparency, performance, maintainability, latency, data efficiency, availability, and auditability. In this context, customizability refers to the system's capacity to be tailored for additional functionality, while adaptability denotes the system's capability to adjust to changes in the distribution of incoming data. The performance property and adaptability property are also distinct in nature. The performance property encompasses the model's ability to classify any instances at a given time, while adaptability specifically focuses on the model's ability to classify new instances or handle unseen data. Additionally, it is important to note that performance encompasses a wide range of metrics employed to evaluate the classification capabilities of a system. In the subsequent sections, we will utilize more precise metrics that address specific facets within the overarching concept of performance.

Since identifying and suggesting strategies that optimize all of the documented system priorities would require an extensive amount of time, it became necessary to select the most crucial system properties to prioritize and focus on. To accomplish this, it was necessary to prioritize the documented system properties based on the insights gained from our comprehensive literature review. This allowed us to identify the properties that we considered most crucial in light of the acquired knowledge. Hence, we needed a multi-criteria decision-making (MCDM) technique, which allowed for the comparison and evaluation of different criteria based on their relative importance and impact.

We used the most cited MCDM technique, namely the analytical hierarchy process (AHP) [73]. This process involved conducting a pairwise comparison of each property based on their relative value and cost. The pairwise comparison approach is less sensitive to judgmental errors that are common in techniques using absolute assignments, as it involves redundancy. It is important to acknowledge that the comparison and prioritization process was conducted internally, making it susceptible to the influence of our subjective opinions. Ideally, the comparison should be carried out by multiple industry actors and an average should be used. Using this method, we assessed each property and assigned a numerical value indicating its level of importance relative to the compared properties.

Before prioritizing, we first put aside the system properties that constituted non-negotiable, hard requirements. Since these system properties need to be optimized regardless, prioritizing them becomes redundant. Consequently, legal compliance was not included in the prioritization, but handled with the necessary attention later on. Similarly, we opted for the perspective that a hate speech management system can be classified as either auditable or non-auditable. Hence, this property also resulted in a hard requirement and was consequently removed from the prioritization. The prioritization also did not explicitly include cost as a separate factor. This was because the cost could be viewed as a comprehensive property encompassing all the mentioned properties. In other words, it would always be optimal to reduce cost, but never at the expense of compromising compliance with a property. Consequently, when identifying and suggesting strategies to optimize the system properties in later sections, we did not directly consider the cost. However, when choosing the optimal strategies in the generalized system architecture among the multiple identified strategies, we have prioritized those that effectively optimize the desired property while minimizing costs. For instance, when choosing a sampling strategy for the fine-tuning of a hate speech detection model, we have specifically considered options that offer cost reduction without compromising the optimization of desired system properties.

| | **Scalability** | **Adaptability** | **Customizability** | **Robustness** | **Latency** | **Performance** | **Transparancy** | **Maintainability** | **Data Efficiency** | **Availability** |
|---|---|---|---|---|---|---|---|---|---|---|
| **Scalability** | 1.00 | 0.50 | 8.00 | 3.00 | 6.00 | 0.25 | 3.00 | 4.00 | 6.00 | 4.00 |
| **Adaptability** | 2.00 | 1.00 | 8.00 | 3.00 | 6.00 | 0.33 | 4.00 | 4.00 | 7.00 | 5.00 |
| **Customizability** | 0.13 | 0.13 | 1.00 | 0.2 | 0.50 | 0.11 | 0.25 | 0.20 | 0.33 | 0.33 |
| **Robustness** | 0.33 | 0.33 | 5.00 | 1.00 | 3.00 | 0.17 | 3.00 | 3.00 | 5.00 | 3.00 |
| **Latency** | 0.17 | 0.17 | 2.00 | 0.33 | 1.00 | 0.17 | 2.00 | 2.00 | 3.00 | 2.00 |
| **Performance** | 4.00 | 3.00 | 9.00 | 6.00 | 6.00 | 1.00 | 7.00 | 6.00 | 8.00 | 8.00 |
| **Transparency** | 0.33 | 0.25 | 4.00 | 0.33 | 0.50 | 0.14 | 1.00 | 2.00 | 3.00 | 2.00 |
| **Maintainability** | 0.25 | 0.25 | 5.00 | 0.33 | 0.50 | 0.17 | 0.50 | 1.00 | 0.50 | 0.50 |
| **Data Efficiency** | 0.17 | 0.14 | 3.00 | 0.33 | 0.33 | 0.13 | 0.33 | 2.00 | 1.00 | 0.50 |
| **Availability** | 0.25 | 0.20 | 3.00 | 0.33 | 0.50 | 0.13 | 0.50 | 2.00 | 2.00 | 1.00 |

**Table 4.1:** Paired relative importance of system properties of the detection architecture of a hate speech management system. Each value represents the relative importance of the row property over the column property.

As we were prioritizing ten properties, we used a scale of 1-10, where 1 indicated equal importance and 10 indicated high importance. Table 4.1 represents the relative importance of each row compared to its corresponding column, with the decimal number in each cell indicating the magnitude of importance of the row over the column. As previously stated, these numbers are derived through internal prioritization, where we assigned a value between 1 and 10 to each cell based on our understanding from the literature review of the detection architecture of a hate speech management system. As an example, scalability is prioritized as eight times more important than customizability.

After creating the matrix of pairwise comparisons, we used averaging over normalized columns to compute the eigenvalues of the matrix. Averaging over normalized columns involves calculating the mean value for each column in the matrix after scaling the values in each column to have a comparable range or distribution, ensuring that no single column dominates the overall average calculation. The retrieved vectors, namely the priority vectors, represent the relative value of each system property compared to the other system properties. Higher values indicate greater importance, while lower values indicate lesser importance. Table 4.2 reveals that performance was the most significant factor, followed by adaptability and scalability, while customizability was deemed to be the least important. We decided to mainly focus on the five most significant properties. As can be observed, this ended up being: performance, adaptability, scalability, robustness against adversarial attacks, and transparency. In addition to these, we also included the previously separated hard requirements: legal compliance and auditability.

| **Property** | **Eigenvalue** |
|---|---|
| Performance | 0.337 |
| Adaptability | 0.190 |
| Scalability | 0.154 |
| Robustness | 0.098 |
| Transparency | 0.098 |
| Availability | 0.087 |
| Maintainability | 0.078 |
| Data Efficiency | 0.062 |
| Latency | 0.056 |
| Customizability | 0.038 |

**Table 4.2:** System property eigenvalue matrix representing the overall relative importance of properties.

## 4.3 Postulating Domain-Specific Strategies

After selecting the relevant system properties, we utilized the findings from our literature review to identify possible strategies to optimize the selected system properties, thus addressing the first subgoal. By initially exploring various strategies, we could then evaluate them in order to identify the ones that are most suitable for inclusion in the generalizable detection architecture of a hate speech management system. This entailed making a note of all potential approaches to enhance the optimization of relevant system properties from the findings of our literature review. A special emphasis was placed on reviewing the literature on the use of contextual information in order to suggest a better strategy involving contextual information. Some strategies were retrieved directly from one source from the findings of the literature review and others were synthesized using multiple sources. In identifying and suggesting strategies that optimize the identified system properties, we used the theory of goal-oriented RE [34] to explore and analyze the strategies for each property. The system properties now served as the goals to be met, and in this manner, we analyzed the strategies required to optimize them. Goal-oriented RE offers the advantage of allowing for the verification that the identified strategies are aligned with the established goals, i.e., optimizes the system properties. Each strategy was therefore evaluated to verify that it optimized the desired system properties. In this manner, we are able to establish the strategies that should be included in the generalizable detection architecture.

The evaluation first entailed a qualitative evaluation based on other findings from the literature review that either supported or verified the impact of an identified strategy. In evaluating and verifying the identified strategies we tried to obtain multiple sources that confirmed the strategy's validity. In this manner, we considered the feasibility and effectiveness of all the identified strategies based on available data from the literature review. The following strategies were established based purely on the literature review and will solely be elaborated on qualitatively in the results chapter:

- Standardization of input to increase robustness
- Adversarial examples in the fine-tuning of the model
- Persistent monitoring using XAI techniques

Some of the identified strategies lacked sufficient literature-based support to defend their viability. Therefore, we concluded that complementary quantitative evaluation was necessary to assess the feasibility of these, as outlined in our research approach. As a result, we determined that the following potential strategies warranted quantitative experiments for proper evaluation:

- The integration of relevant contextual information
- A detection triage scheme that adaptively routes classifications from a lightweight language model to GPT and human moderation
- An active labeling scheme with GPT-based models and human moderators for continuous fine-tuning
- A sampling strategy that prioritizes uniqueness and uncertainty

The evaluation of each strategy through quantitative experiments necessitated the availability of a dataset. Therefore, in the subsequent sections, we will elaborate on our methodology for acquiring a suitable dataset while concurrently exploring the strategy of the GPT-driven labeling scheme. Then, we will outline the methodology used to leverage the obtained dataset in order to develop hate speech classifiers. These classifiers will subsequently be used to evaluate the strategy concerning contextual information and the proposed sampling strategy. The results obtained from all of these quantitative experiments will be used to evaluate the proposed triage scheme, which combines elements from both the GPT-driven labeling scheme and classification using a lightweight model. The formulated strategies from both the qualitative and quantitative evaluations and the corresponding optimized system properties are presented in the subsequent results chapter.

## 4.4 Refining Existing Hate Speech Dataset

The evaluation of strategies that demand quantifiable trials when qualitative analysis falls short depends upon the availability of an appropriate dataset. We opted to use an existing dataset for our research as we aimed to focus our thesis on other aspects of the field rather than data collection. As described in our literature review, the work of Founta et al. [23] culminated in a dataset consisting of 80,000 samples of tweets and their assigned labels from crowdsourcing. This dataset was provided to us at the beginning of our specialization project, and we were tasked with analyzing and drawing insights from it.

Furthermore, our literature review revealed, as described in subsection 3.5.2, that inconsistencies and poor labeling practices in the dataset had been previously identified. Despite attempts to obtain a better dataset, no suitable alternatives were available, leaving us with no choice but to use the available dataset. Despite the limitations of the dataset, it still includes a substantial amount of valuable content (e.g., instances of hateful content). Therefore, we opted to use this dataset as a basis for our research and refine it as needed to fit our research objectives. Figure 4.2 depicts the enhancements made to the Founta et al. dataset. Using an existing dataset, therefore, offers the following advantages:

1. Allowing us to allocate more time towards the primary goal of the thesis, rather than having to collect new samples.
2. Establishing a comparative baseline for the performance of the GPT-based labeling scheme.
3. Providing access to the unique tweet identifiers of hate speech-labeled tweets for data collection, ensuring the presence of reliable hate speech samples.

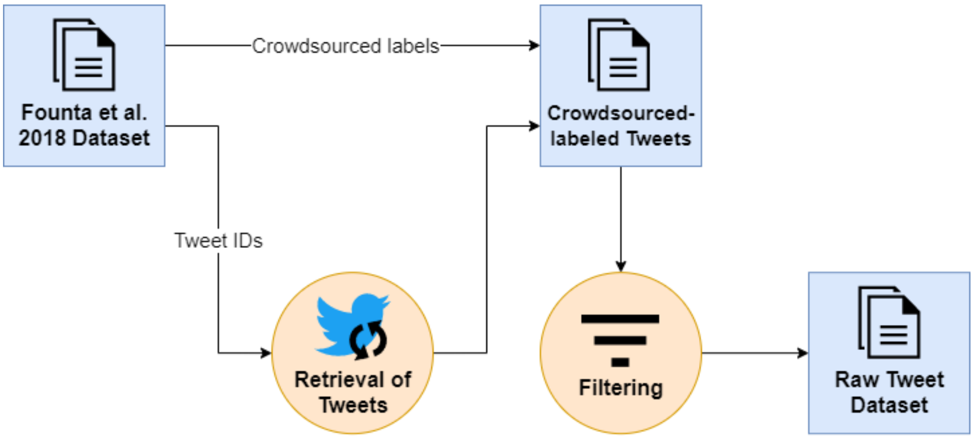


**Figure 4.2:** Workflow for developing augmented tweet dataset from the dataset provided by Founta et al.

### 4.4.1 Retrieving Contextual Information with the Twitter API

The Twitter API[1] allows developers to access and interact with Twitter data and functionality. Of particular relevance, the API provides means to programmatically retrieve tweet objects and their associated data simply by querying for their unique tweet identifier. This makes it possible to iterate over each tweet identifier in the Founta et al. dataset and gather the contextual information as defined in section 3.4 from the tweet object and the associated metadata. However, we ended up with a considerably reduced dataset when retrieving the tweet objects from the Twitter API, likely due to the considerable age of the dataset. Over time, much of the data may have been removed by the users themselves or by Twitter's hate speech filters. Nevertheless, the dataset remains sufficient enough for our purposes. Most importantly, it contains a high degree of the implicit hate speech described in Table 3.2, a pressing problem to address within this domain.

### 4.4.2 Filtering

In order to ensure a scientific approach to our analysis of adding contextual information, we employed a strict sampling and filtering process. Our objective was to prioritize cases that exhibited intricate nuances, thus potentially requiring contextual information for precise labeling. To accomplish this, we implemented a filtering process to exclude samples that, based on existing literature, were deemed less likely to benefit from contextual information.

[1]An Application Programming Interface (API) is a set of protocols, methods, and tools that facilitate communication and data exchange between software applications.

An important step in the filtering process was the removal of non-conversational tweets from the dataset. Previous literature regarding the integration of contextual information had not observed significant improvements when analyzing this specific category of tweets. Hence, the decision to remove them aimed to narrow down the sample to instances that were expected to be more focused and contextually rich. Therefore, we excluded standalone tweets and retweets, reducing the dataset size by a significant margin. Similarly, we eliminated quote tweets and replies whose parent tweets were not available, likely due to a violation of Twitter's content guidelines. Finally, we excluded samples that were previously labeled as spam by the original dataset annotators, as we frequently found these samples to be lacking in semantic value and/or conceptual significance.

The following summarizes the rules applied in the filtering procedure:

**FR1** Exclude non-reply or non-quote samples

**FR2** Remove conversational tweets that are lacking parent objects

**FR3** Remove samples that were classified as spam by the original annotators

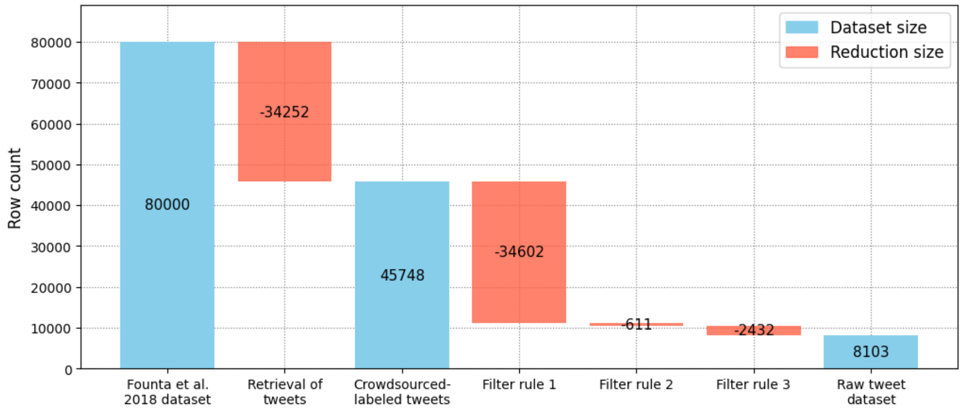


**Figure 4.3:** Waterfall plot demonstrating the effects of the size-reducing operations on the dataset.

As a result of the data retrieval and filtering process, we were left with 8103 raw samples, representing approximately 10% of the original Founta et al. dataset as seen in Figure 4.3. The hate class proportion increased from 4.54% to 5.31%, highlighting a persistent imbalance in the dataset. A feature summary of the dataset after filtering can be seen in Table A.1 in the Appendix.

Despite the considerable reduction in our dataset's size, we still consider it valuable for our experimental applications. Improved performance on this dataset could arguably lead to an increase in overall performance, albeit to a lesser extent.

Reducing the dataset to this specific focus allows us to more easily identify the potential impacts of our strategies, although this comes with the trade-off of the dataset potentially being less representative of the real data stream on a social media platform.

## 4.5 Exploration of GPT-Based Annotation

Labeling has been considered a costly and complicated task since it requires the hiring of individuals. However, labeling schemes can now be accomplished using sophisticated language models after recent advancements in their competence and affordability, as described in the literature review. In this section, we will describe our development of a GPT-driven labeling strategy, which will then be applied to get our dataset labeled with contextual information for usage in quantitative trials. The results of these experiments will decide the potential of the labeling scheme as a strategy within the final system architecture.

Although having obtained a preprocessed dataset that included labels and relevant contextual information, the validity of the labeling had to be considered. While the method of labeling in Founta et al. was somewhat questionable and not up-to-date, there was a more significant concern for our research. Specifically, the annotators did not have access to contextual information during the annotation process. Therefore, we had to relabel the dataset by inferring the context and incorporating it into the labeling process to obtain a reliable ground truth for our analysis.

Because of the subtle nature of the indicators associated with the labels, we estimated that each sample took at least 40 seconds to label by a human moderator using a multi-layered labeling scheme as outlined by Assimakopoulos et al [5]. This would entail a lower bound of 90 hours for the entire dataset for one person without taking any breaks. We would also risk incorporating our internal biases into the dataset. Moreover, a crucial aspect of our research involves the real-world applicability of a hate speech detection system, which highlights the importance of selecting a labeling scheme that accurately reflects real-world use cases. Therefore, the labeling scheme has to be time-efficient, as well as scalable. Due to these limitations, we decided to outsource the labeling task, as manually evaluating each sample is both too time-consuming and vulnerable to subjective influences. As stated earlier, we recognized this as an opportunity to explore a GPT-driven labeling scheme, a strategy we outline in subsection 3.5.4 of our literature review.

### 4.5.1 Establishing an Unbiased Hate Speech Definition

As we can define the criteria for hate speech ourselves, we wanted to employ a debiasing methodology to mitigate biases when labeling and remove ambiguity from the hate speech definition. Founta et al. developed their definition by consulting contemporary related works and dictionaries. We compared this definition with that of the UN[2] and some others to get a good overlook. The definition of hate speech used by the UN [86] encompasses a broader range of characteristics (e.g., nationality, color, and descent) when compared to the definition used by Founta et al. Hence we added this to our definition. However, the definition of UN also includes the ambiguous term "or other" as an identity factor. Based on our analysis of the literature, it is evident that establishing clear boundaries is crucial in achieving consistent labeling. Hence, "or other" is removed. Consequently, we ended up with the following definition of hate speech which is pretty similar to that of Founta et al. [23], albeit with minor alterations in the attributes included:

*Hate Speech* Language used to express hatred towards a targeted individual or group, or is intended to be derogatory, to humiliate, or to insult the members of the group, on the basis of attributes such as race, religion, ethnicity, sexual orientation, disability, nationality, descent, color or gender.

### 4.5.2 Data Preparation for Outsourcing

Before sending the dataset for third-party labeling, we performed data cleaning and reduction operations to reduce costs, mitigate noise, address ethical considerations, and reduce the risk of privacy violations. Although our data is entirely collected from a public space, we made an effort to protect personal information as users have not given explicit consent for their data to be used for this task. Therefore, we took steps to not outsource any information that does not hold predictive power and that could link a tweet to a specific individual.

#### Selection of Essential Features

The cost of labeling is based on the number of tokens in each sample given as input to the language model, so each sample row is condensed to contain only the critical information required to allow for accurate labeling by an advanced annotator. The child tweet text and parent tweet text provided the actual content of the tweets being evaluated, which is the primary factor in determining hate speech. The child user description and parent user description gave insight into the individuals who wrote the tweets. These descriptions may reveal certain characteristics of the user targeted by hate speech, thereby potentially playing a role in determining the appropriate label. Hence, the sample row is condensed to these four textual features.

[2]Hate speech definition of UN: "any kind of communication in speech, writing or behavior, that attacks or uses pejorative or discriminatory language with reference to a person or a group on the basis of who they are, in other words, based on their religion, ethnicity, nationality, race, color, descent, gender or other identity factors" [86].

| **Labeling Input** | **Example Value** |
|---|---|
| Child tweet | [MENTION] That way they won't be lying under oath when they swear they aren't doing anything illegal. (Someone else is doing it for them). |
| Child user description | ADHD! Family Man, Politics, Conservative, Christian, CANADA #YQR * strongly opposed to the globalist agenda of Trudeau. #TrudeauMustGo! |
| Parent tweet | According 2 this article, they feel Intel agencies r using contractors to illegally spy on Americans using the CIA tools exposed by Wikileak [URL] |
| Parent user description | Free spirit and cool cat at heart. The truth will always prevail, count on it. Accuracy is my motive, if you ever see something wrong, just tell me. [MENTION] |

**Table 4.3:** Data foundation of an example row from the Founta et al. dataset to illustrate what is used for labeling instances with GPT. The specialized tokens are denoted in square brackets.

#### Tokenization of Non-Informative Elements

In the context of the classification task, we regarded mentions of other users in the raw text itself as not informative as the communicative relationship in the conversation is implicitly provided. Similarly, URLs are not deemed significant because they do not contribute directly to the semantic content and meaning of the text, which is the focus of the classification task, while also being a rare occurrence. These non-informative elements are thus replaced with custom tokens. The process is carried out by using the already stored character positions of mentions and URLs in the raw features. The resulting format of each sample outsourced can be seen in Table 4.3.

### 4.5.3 Outlining Trade-Off Determinants for Cost Efficiency

In addition to an adequate hate speech definition and a cost-effective input, there exist numerous configurations for the labeling schemes using language models, each offering a unique trade-off between cost-efficiency, time investment, and performance. We have identified four key factors that influence such trade-offs:

1. Model complexity
2. Instruction method
3. Uncertainty quantile threshold in active labeling
4. Number of in-context examples

Empirical studies from the literature review which could be applied to this strategy were not found. We, therefore, set out to perform empirical trials with different configurations of factors ourselves to find the optimal balance between performance and cost.

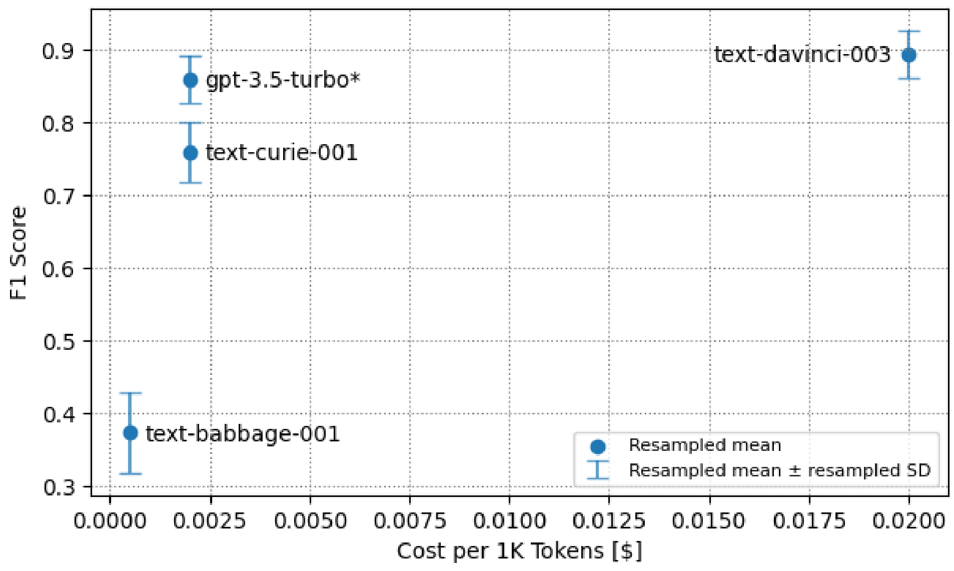
For our trials, we were faced with a dilemma as we tried to balance two opposing considerations; on one hand, we would need to reduce the number of trials to account for budgetary constraints, while on the other hand, we would need to increase the number of trials to test more levels for each factor. A "level" in this context refers to the specific value or setting that a factor can have in an experiment. To address both considerations, we employed a fractional design of experiments (DOE) methodology where we used statistical analysis to approximate optimal performance-cost trade-offs for each factor while reducing the number of trials needed [56]. By analyzing the variance in cost-effectiveness resulting from varying factor levels, we can identify the most influential factors. We went with a three-level factorial design as the interactions between the factors are arguably dependent on each other, and to account for possible non-linear relationships. We now delve into the rationale behind choosing the selected levels for each factor.

### Model Complexity

We decided to use an instruction-based model with a few-shot learning ability so that the model can assign labels based on the definition and some in-context examples that we provide it with. Such models have been shown to provide context-aware and accurate predictions that are more consistent than their non-instruction-based counterparts, which are designed to learn relationships in the data without explicit guidance on how to process instructions [54].

We opted for OpenAI's instruction-based contextualized language models as they are both performative and competitive in price, while easily accessible with an API. As of March 2023, there were five models with varying complexities and prices. Lacking literature applicable to our constraints, we performed small-scale empirical trials in order to find a subset of models with promising cost-effectiveness. In order to establish the ground truth, we labeled 300 samples ourselves using our established definition with the multi-layered labeling scheme by Assimakopoulos et al. [5], as depicted in Figure 4.5. To ensure the positive class was adequately represented, we performed oversampling. Each trial was conducted with only the labeled 300 samples, so to account for non-representative performances, we performed resampling[3] on the predicted values of each model.

[3]Resampling is a statistical method for generating new samples by repeatedly drawing observations from a dataset, usually done to estimate the performance of sample statistics.



* Performance adjusted for non-classifiable predictions

**Figure 4.4:** Estimated performance of OpenAI's instruction-based contextualized language models given their token cost.

From Figure 4.4, we can see that the Davinci model outperforms all the other models with statistical significance, but at a much higher API call cost per one thousand tokens; it is roughly ten times more expensive than the cost-optimized version of GPT-3.5, *gpt-3.5-turbo*, while providing an estimated four percentage points higher $F_1$ score. The GPT-3.5 model does not return the log probabilities of each token, and being able to select the most uncertain samples using human experts is an important property of the active labeling approach. We thus ultimately opted to only proceed with the Davinci model for the experiment, as we were convinced that the other models would not provide us with adequate performance even if we optimized the other factors. Also, not including the cost enabled us to reduce the number of trials, and thus the cost of our experiment, by 67%.

### Instruction Method

As with humans human annotators, it is important to provide concise and precise instructions to the model on how it should classify the instances. In order to achieve consistent labeling, the definition of hate speech must be presented in a manner that reduces the possibility of internal biases. Consequently, we made use of the multi-layered labeling scheme methodology presented by Assimakopoulos et al. [5], which outlines how to reduce biases when annotating. Building upon the previously defined hate speech definition, we developed a series of progressively detailed yes-no inquiries that together reflect the definition. By employing this multi-layer annotation scheme in the subsequent active labeling stage, we also minimized the potential for subjective interpretation by human annotators. This scheme can be seen in Figure 4.5.

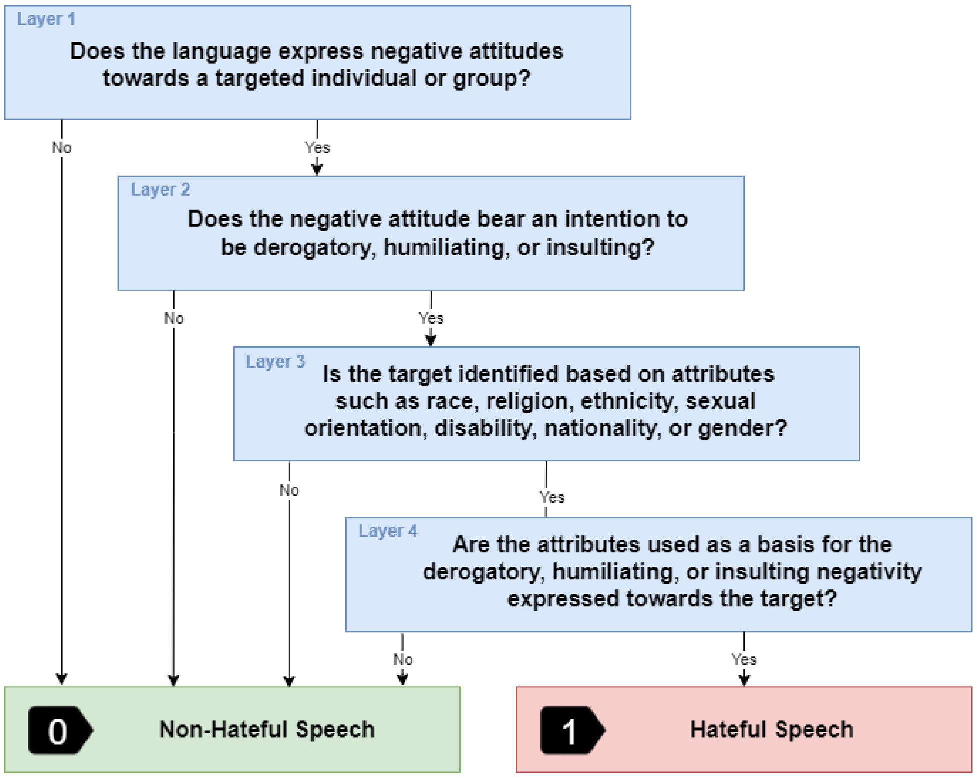

Layer 1
Does the language express negative attitudes towards a targeted individual or group?
No
Yes
Layer 2
Does the negative attitude bear an intention to be derogatory, humiliating, or insulting?
No
Yes
Layer 3
Is the target identified based on attributes such as race, religion, ethnicity, sexual orientation, disability, nationality, or gender?
No
Yes
Layer 4
Are the attributes used as a basis for the derogatory, humiliating, or insulting negativity expressed towards the target?
No
Yes
0
Non-Hateful Speech
1
Hateful Speech

**Figure 4.5:** Condition-based multi-layer labeling scheme designed to classify hate speech in an unbiased matter.

There is however a trade-off between the quality of instruction and the cost. The number of tokens in the prompt given to the language model is indicative of the ultimate cost of labeling. Although we shortened the sample row by employing feature selection, we also had to take into account the length of the instruction in the prompt. We will therefore want to test the cost-efficiency of using a less biased approach by including the instruction method to the GPT model as a factor in the Design of Experiments.

The multi-layered scheme is essentially a series of conditional statements. To make this readable and reliably interpreted by a language model, the scheme was formatted through a series of short-circuited conditionals, where only the ensuing conditional should be evaluated if the previous conditional is satisfied. Using this scheme to mitigate bias will result in an increase in token count from 132 to 247 [77], or about \$0.0023 for the most expensive GPT-based model [78]. There is some redundancy in the branching logic, but we deemed it necessary from our experience through trial and error to ensure output consistency for the language models. Through iterative experimentation, we successfully developed two formulations of the multi-layered scheme, with one of them offering a higher level of detail. The levels for instruction methods can be seen in Table A.2 in the Appendix.

**Level 1** Definition-based

**Level 2** Condition-based with low specificity

**Level 3** Condition-based with high specificity

### Uncertainty Quantile Threshold in Active Labeling

In addition to providing the generated tokens, the OpenAI API also returns the unnormalized log probabilities of the most likely tokens at a given index. These log probabilities can then be used to quantify the uncertainty during classification as long as the model is instructed to generate the classification in a standardized form. This is especially trivial for our case as we instruct the model to output a single token — either a 0 or a 1 — which signifies the predicted class. We used entropy as our uncertainty measure as it is a simple and well-established way of capturing the confidence of classification. The log probabilities, $z_i$, for each class in the index set $I = \{0, 1\}$ at the first token in the generated text were extracted before we applied a Softmax function over the set $J$ to obtain the normalized probabilities. Then, the entropy, $H(I)$, was trivially calculated using the Shannon entropy formula over the index set $I$, as seen in Formula 4.1. We chose base-2 for computing the information content as it aligns with the binary nature of the classification task while also being a widely accepted convention in information theory.

$$\begin{aligned} p_i &= \frac{e^{z_i}}{\sum_{j \in J} e^{z_j}}, \quad \text{for } i \in I \\ H(I) &= -\sum_{i \in I} p_i \log_2 p_i \end{aligned} \tag{4.1}$$

The labeling performance could then be improved by incorporating the uncertainties of each classification in an active labeling scheme, ranking the labels by their uncertainty, and replacing the ones that were the most uncertain with labels assigned by an expert (i.e., a human), as demonstrated by Wang et al. [95].

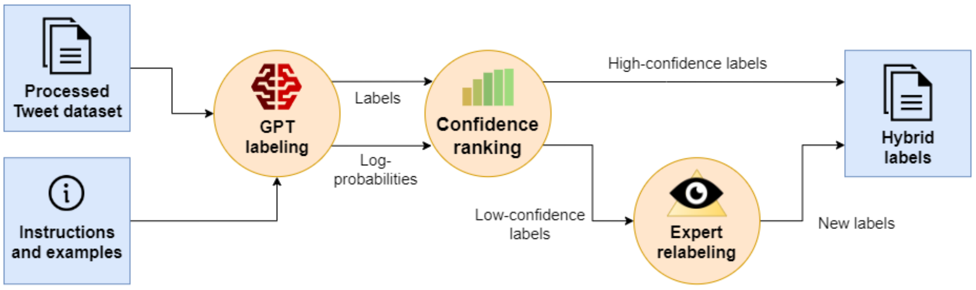


**Figure 4.6:** Active labeling workflow used to relabel the dataset with GPT-3 model. The expert relabels the instances in which the GPT model is the most uncertain.

Our active labeling scheme, which employs a strategic approach to selecting data instances for human annotation based on uncertainty levels, is depicted in Figure 4.6. Given the constraints of a limited budget, it was necessary to carefully select the levels at which the uncertainty quantile threshold for human annotation would be set. These levels represent the percentage of the most uncertain data instances that would be selected for relabeling, with the goal of improving the model's performance, and are as follows:

**Level 1** 0%

**Level 2** 10%

**Level 3** 20%

### Number of In-Context Examples

The introduction of specific classification examples to the language model can be conceptualized as a form of implicit fine-tuning. This process effectively adapts the model to perform more specialized tasks, augmenting its general pre-trained knowledge with specific application-oriented insights. This approach serves as a strategy to increase the model's ability to handle nuanced or ambiguous instances, specifically in the context of categorizing subjective instances of hate speech. As negative samples are more prevalent than positive samples, it was important for us to maintain this same distribution when selecting these samples. This reduces the risk that the model does not become biased towards negative or positive examples when applied to the larger population. Consequently, we chose the following levels for the number of in-context examples factor:

**Level 1** 0-shot

**Level 2** 2-shot, with one positive and one negative classification

**Level 3** 4-shot, with one positive and three negative classifications

### 4.5.4 Estimating the Optimal Combination of Factors

When we had established the levels for each of the four factors, we could proceed with the design of experiments approach. It is important to note that while the factors may improve performance, they will inevitably increase costs. With a limited budget, we wanted to identify the most influential factors in an objective way without performing too many experimental trials.

#### Objective Utility Criterion

We define an objective utility criterion to establish a concrete trade-off relationship between performance and cost such that factor combinations can be objectively valued. Equation 4.2 shows how the cost of a combination is calculated on a per-tweet basis. The cost is determined by scaling the cost of inference per token, denoted as $C_{GPT}$, by the average token count in the combination, $\bar{N_T}$. To this scaled cost, the cost of oracle processing, $CO$, is added, which is scaled by the uncertainty quantile threshold, $T_{UQ}$. The cost of inference is fixed for all combinations as every combination uses the same Davinci model, so $C_{GPT}$ is given a value of \$0.02 in line with the Davinci API pricing [78]. The active labeling cost of an oracle $C_O$ was fixed per sample to \$0.14. We used the AWS Pricing Calculator[4] for estimating the costs of the Sagemaker Ground Truth service in order to retrieve an estimate for the cost of an expert labeler per sample.

$$\bar{C}(C_{GPT}, \bar{N_T}, C_O, T_{UQ}) = C_{GPT} \cdot \bar{N_T} + C_O \cdot T_{UQ} \tag{4.2}$$

We chose to use a parameterized exponential utility function as defined in equation 4.3 such that cost will matter more as the performance grows, as we do not value cost when the performance is low. The parameters $\beta$ and $\gamma$ determine the desired curvature and relative importance between the cost and the performance. We chose a value of 0.25 for $\beta$ and a value of 2 for $\gamma$ as these parameters provided a curve that aligned with our expected predictions. The performance is quantified using the weighted $F_1$ score as defined in equation 2.6 to account for the inherent imbalance between hate and non-hate instances in our dataset.

$$U(F_1, \bar{C}; \beta, \gamma) = F_1 \cdot e^{-\beta \cdot \bar{C} \cdot F_1^{\gamma}} \tag{4.3}$$

It is important to note that the specific configurations of the cost function $\bar{C}$ and the utility function $U$ are generalizable; they can be modified according to the distinct needs of the entity assessing the combinations, demonstrating the method's adaptability to various contexts.

[4]https://calculator.aws/#/addService/SageMakerGroundTruth

### Fractional Factorial Design

It would be beneficial to explore the whole solution space in order to optimize the cost-performance trade-off of a system that is influenced by a range of factors. However, with one level for factor 1, three levels for factor 2, three levels for factor 3, and three levels for factor 4, the total number of possible combinations is 27. Given our financial constraints, performing all 27 combinations was not practical. Therefore, we implemented an orthogonal fractional design, allowing us to estimate the main effects of each factor and some of their interaction effects independently, while minimizing the impact of confounding factors and reducing the number of runs to 9. An overview of trial runs with the performance, cost, and utility scores can be seen in Table 4.4. We conducted tests on 300 samples which we labeled ourselves using the multi-layered labeling scheme presented in Figure 4.5. Fellow students were involved in the active labeling process to differentiate between the ground truth labels established by us and the expert labels utilized in the active labeling phase.

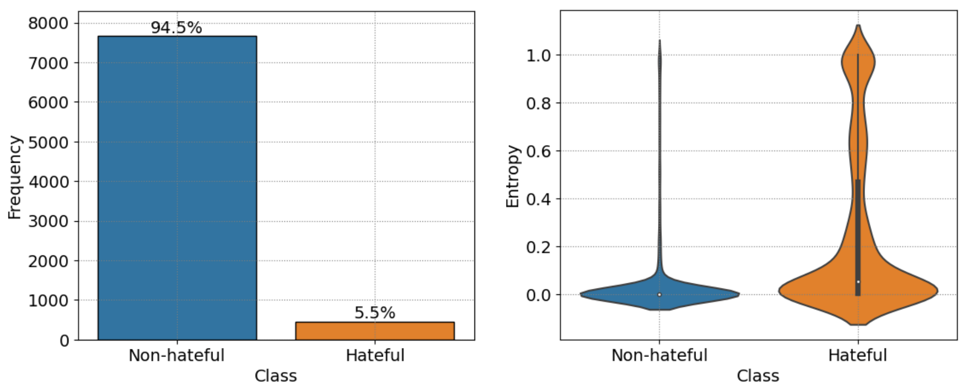
| **Run** | **Instruction** | **Unc. Threshold** | **No. Examples** | $\mathbf{F_1}$ | $\bar{\mathbf{C}}$ | **Utility** |
|---|---|---|---|---|---|---|
| 1 | Definition | 0% | 4-shot | 0.52 | 0.018 | 0.47 |
| 2 | Definition | 10% | 2-shot | 0.67 | 0.026 | 0.63 |
| 3 | Definition | 20% | 0-shot | 0.86 | 0.033 | 0.76 |
| 4 | Condition | 0% | 2-shot | 0.56 | 0.014 | 0.55 |
| 5 | Condition | 10% | 2-shot | 0.73 | 0.028 | 0.68 |
| 6 | Condition | 20% | 2-shot | 0.90 | 0.042 | 0.76 |
| 7 | Condition, specific | 0% | 0-shot | 0.59 | 0.009 | 0.58 |
| 8 | Condition, specific | 10% | 2-shot | 0.77 | 0.029 | 0.71 |
| 9 | Condition, specific | 20% | 4-shot | 0.96 | 0.049 | 0.76 |

**Table 4.4:** Response matrix for fractional factorial design experiments.

To estimate the unique contribution of each factor on the utility, we ran an OLS[5] regression as it is simple while providing easily interpretable coefficients. While the assumption of linearity might not be entirely satisfied, the method can still prove valuable to estimate the overall directional effect of the factors on an output variable, which is the utility metric $U$ in our case. Using this approach, we concluded on an optimal combination when using condition-based instructions with low specificity (Instruction level 2 in the Appendix A.2), 20% uncertainty quantile threshold for active labeling, and 0 in-context examples. We performed a new small-scale test with this new combination from which we obtained a satisfactory $F_1$ score of 0.94. We thus relabeled the entire dataset using this configuration of factors, achieving the second subgoal. The resulting class distribution and uncertainty distribution in the relabeled dataset can be seen in Figure 4.7. Here, we can observe a slight decrease in

[5]Ordinary least squares (OLS) is a statistical method for estimating parameters in a linear regression model.

class imbalance compared to what was mentioned in section 4.4.2, but it remains noticeable, hence requiring consideration in subsequent operations.



**(a)** Bar plot illustrating the class distribution with associated proportions.

**(b)** Violin plot illustrating the uncertainty distribution given the class.

**Figure 4.7:** Comparison of class distribution and the corresponding uncertainty of dataset relabeled with the optimal configuration of factors.

## 4.6 Developing Hate Speech Classifiers for the Evaluation of Quantitative Strategies

With the successful creation of a suitable dataset, we were now ready to advance toward the third subgoal of developing classifiers that could be used to evaluate both the integration of contextual information and the proposed sampling strategy.

To evaluate the integration of contextual information, it was crucial to experiment with various combinations of contextual features, as outlined in section 3.4 of our literature review. Our dataset comprises both textual and numerical contextual information. The textual context encompasses various elements, including the raw data of the tweet, the raw data of the parent tweet, and the two users' descriptions. The numerical context involves different ids (e.g., tweet, conversation, author) and counts (e.g., number of retweets, number of replies, number of likes). Based on our literature review, we discovered that relying solely on numerical features produced unsatisfactory results 3.4. Therefore, we made the decision to create two context classifiers: one that solely considers textual features and another that incorporates both textual and numerical features. Additionally, it was necessary to develop a performative baseline classifier that doesn't utilize contextual features when making predictions to accurately measure the performance gain that could be achieved by adding contextual information.

Most of the contextual features are categorical or numerical features. These don't have the same inherent ordering or dependencies in natural language, so we cannot use a language classifier for these either. Instead, a stacked[6] model architecture can be used in order to handle both textual and numerical data in an effective way.

Apart from being essential for quantitative experiments aimed at analyzing contextual integration, the embeddings of the natural language model can also be leveraged to implement the proposed sampling strategy. These embeddings serve as a metric to gauge the uniqueness of an input. Consequently, we embarked on the development of lightweight natural language classifiers for the following four key reasons:

1. We need a performative baseline classifier that doesn't utilize contextual features when making predictions, in order to accurately measure the performance gains of adding contextual information.
2. The natural language model can be employed independently to classify based solely on textual context.
3. The output obtained from the language classifier can serve as a basis for generating meta-features from textual inputs. These meta-features can then be utilized in conjunction with other numerical features within a stacked model.
4. The embeddings of the hidden layers of the model provide a measure for the uniqueness or diversity of a tweet which will be used in order to test data sampling approaches.

### 4.6.1 Model Architectures and Domain Adaptation Approach

As outlined in the literature review, deep transformer-based neural network models with bidirectional processing are the most effective for our downstream[7] detection task because of their ability to capture nuanced linguistic patterns and cultural references that are often present in hateful language [42]. As referenced earlier, the research done by Gröndahl et al. argues that "the model architecture is less important than the type of data and labeling criteria being used when optimizing the performance of hate speech detection" [25]. Although this is a significant concern when trying to achieve the best possible hate speech detection performance, the choice of model is still critical for ensuring the feasibility and practicality of the actual deployment on a social media platform.

[6]Stacking is a machine learning technique where outputs of base models serve as inputs to a second-level meta-model.

[7]A downstream task refers to a task that is built upon the outputs or representations generated by a pre-trained model.

### Choice of Base Model to Process Textual Features

From the comparison of pre-trained transformer-based models we conducted in our literature review, we gathered that size, performance, and inference time were important factors to consider from the perspective of a social media company. Given that our GPU had a memory capacity of 10 GB, we had to limit our search for a suitable model with this constraint in mind. From this comparison, we gathered that an XLNet model, with its superior performance on benchmarking tasks, or ELECTRA, with its state-of-the-art cost-effective training procedure and faster inference times, was the most appropriate. However, using an already fine-tuned model would prove advantageous to allow the model to grasp the overall characteristics of the task while mitigating overfitting concerns regarding the limited size of the dataset. Such a model would have to be specifically fine-tuned on a hate speech dataset, preferably implicit hate speech. We were not able to identify publicly available XLNet or ELECTRA models already fine-tuned for hate speech classification.

Instead, we found a fitting BERT model accessible through the Transformers library[8]. The researchers behind the ToxiGen dataset, mentioned in subsection 3.5.3, released a fine-tuned version of the HateBERT[9] model, which we used as our foundation for subsequent modifications. We could then perform sequential fine-tuning operations to create classifiers that are calibrated with our definition of hate speech by using the dataset we created. In this manner, we ended up with models that were both initially trained and then sequentially fine-tuned on the task of hate speech detection.

### Choice of Meta-Model to Process Numerical Features

Our approach involved a stacked model architecture, composed of a base model to process textual data, and a second-tier meta-model designed to handle non-textual features. This meta-model incorporated the numerical output of the language model, in addition to other numerical contextual features, as part of its input. We wanted to be able to investigate the usage of a range of numerical features, so a boosting[10] model would be beneficial to handle the high-dimensionality and relatively small dataset. However, it's essential to bear in mind that, despite their strengths, boosting models can risk overfitting when handling high-dimensional data. While neural networks rely on interconnected nodes directly, as described in the background chapter, boosting algorithms combine the predictions of multiple less complex models to create a more

[8]https://huggingface.co/tomh/toxigen_hatebert

[9]HateBERT is a pre-trained model that had been re-trained for abusive language detection in English while being the same size as the original BERT model [10].

[10]Boosting is an ensemble technique used for improving the performance of weak learners, typically decision trees or simple models, by combining them in a sequential manner.

powerful predictive model. Among boosting algorithms, XGBoost[11] was specifically selected for its capacity to manage overfitting through regularization[12] and for its efficient handling of missing data. It is available through the Python library with the same name [87]. The L1 regularization technique it implements practically removes the need for feature selection as it can shrink the coefficients of less important features to zero.

#### Final Classifiers

A total of three classifiers were developed with varying degrees of context awareness. An overview of these three classifiers can be seen in Figure 4.8. We have coined the name *NaiveBERT* for the baseline model, which employs the HateBERT model sequentially fine-tuned on both the ToxiGen dataset and our own dataset. We have designated the name *ContextBERT* for the model that exclusively incorporates textual features into the NaiveBERT baseline model. Finally, we named the model that combines the output of the ContextBERT model with an XGBoost model, while also incorporating numerical features, *ContextBoost*.

### 4.6.2 Design Choices for Sequential Fine-Tuning of BERT models

The training process is a critical component of developing effective models, and its design choices collectively shape the model's performance and generalization capabilities. We, therefore, opted to use the Trainer class in the Transformers library for managing our training process, as it not only streamlines complexities and provides extensive customization options, but also adheres to deep learning best practices and facilitates reproducibility. The training object was configured to use the ROC AUC statistic on the validation set to select the most promising model at each stage, as it provides a comprehensive understanding of the model's performance across various classification thresholds and is less sensitive to the class imbalance of the dataset evident in Figure 4.7a. To account for overfitting, we set up our training object with an early stopping callback and checkpointed the model at each epoch based on the validation set performance. AdamW was used as the optimizer as it incorporates weight decay during training, which can further reduce the risk of overfitting and improve the model's generalization ability, while also being the most conventional optimizer for NLP-related tasks.

[11]XGBoost, short for eXtreme Gradient Boosting, is a machine learning algorithm that excels in leveraging gradient boosting and decision tree ensembles, offering high predictive accuracy, robustness against overfitting, and the ability to handle large-scale datasets with high dimensionality and complex relationships.

[12]Regularization is a machine learning technique applied to models in order to prevent overfitting by adding a penalty term to the loss function. This effectively encourages model simplicity and improves the generalization ability.

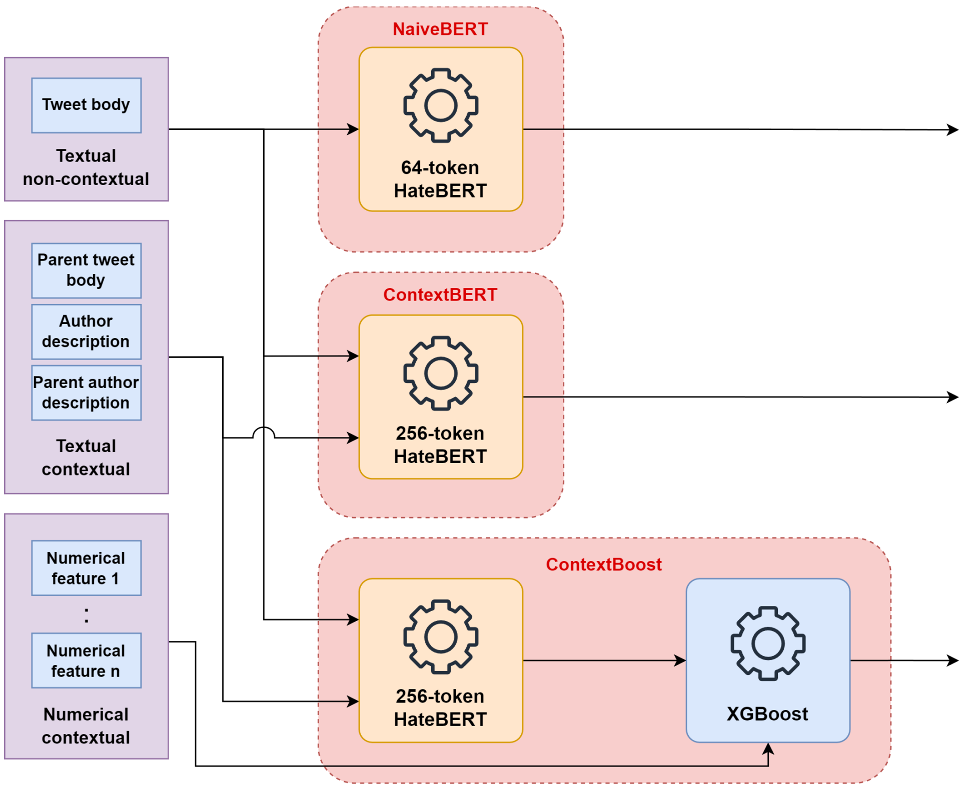


**Figure 4.8:** High-level architectures of hate speech classifiers

Standard loss functions treating all classes equally in imbalanced datasets can bias the model towards the majority class, which may negatively impact the performance of the model on the minority class [12]. This effect can be the most profound in cases where the majority class contains a lot of samples that the model can classify with high certainty, which can result in poor performance on the low-certainty samples of the minority class. To account for this, we utilized the Focal loss function as denoted in equation 4.4 introduced by Lin et al. [38], which has shown large improvements in accuracy for imbalanced datasets without compromising training time. This function adds modulating terms to the cross-entropy loss defined in equation 2.2 which down-weights the contributions of easily classified instances during training.

$$L_F(p_i, y_i; \gamma) = -(1 - p_i)^\gamma \cdot y_i \cdot \log(p_i) - p_i^\gamma \cdot (1 - y_i) \cdot \log(1 - p_i) \tag{4.4}$$

The $\gamma$ parameter was assigned its value through hyperparameter optimization, which we will come back to in subsection 4.6.5.

### 4.6.3 Feature Engineering for Meta Learner

We engineered features from the augmented raw dataset from section 4.4 which we thought would provide predictive power to the classifier in the cases where ContextBERT was uncertain. After applying data cleaning techniques to enhance the data's suitability for analysis, we produced meta-features from the output of the ContextBERT model for the ContextBoost model. In such a sense, the BERT model serves as a complex feature extractor from the four textual features, namely the child tweet text, the parent tweet text, the child author description, and the parent author description. The output of the BERT classifier architecture is a set of logits[13] for each class. Applying the Softmax function to these logits enables us to convert them into class probabilities. Following this, we calculated the entropy of these probability distributions to measure the uncertainty associated with the model's predictions. We hypothesize that incorporating this measure could be advantageous, as it could encourage the model to rely more on other features when uncertainty is present. We further quantified the sentiment of the textual features through the negative polarity scores of the *SentimentIntensityAnalyzer* class in the NLTK library.

We had a lot of numerical features that we needed to transform into more applicable features. While the XGBoost model can handle wide data, we have 16 engagement-type raw features as seen in Table A.1 where each of them might not provide us with a lot of predictive power. We, therefore, used PCA[14] in order to reduce the dimensionality of these features into two representative features. We also engineered a set of features like the time between when the parent and child tweet was posted, whether or not the author uses the default profile (anonymous people might be more hateful), or the ratio of followers between the two authors (trolls have usually not many followers). The resulting features can be seen in Figure 4.5.

### 4.6.4 Preparation of Textual Input Features

Aligning the data representation with that used during pre-training is vital, as the classifiers rely on the knowledge acquired during this process to perform effectively. We utilized the *bert-base-uncased* tokenizer from the Transformers library to make the input sequence formatting align with that of the underlying HateBERT model, thereby ensuring consistency and compatibility.

Multiple input features are fed to the model by adding special separator tokens between the features and concatenating them into a single string. This lets the model

[13]A logit refers to the unnormalized log probabilities for each class predicted by a model.

[14]Principal Component Analysis (PCA) is a statistical method that reduces the complexity in high-dimensional data by transforming it into a smaller number of orthogonal features, known as principal components, while attempting to preserve the variability in the data.

[15]In the context of XGBoost, the F-score represents feature importance, a concept distinct from the usage of F-scores in traditional statistical analysis.

| **Numerical contextual feature** | **Data type** | **F-score**[15] |
|---|---|---|
| Positive class probability of BERT prediction | float64 | 49 |
| Entropy of BERT prediction | float64 | 11 |
| Feature 1 from PCA analysis | float64 | 14 |
| Feature 2 from PCA analysis | float64 | 4 |
| Negative polarity score of tweet | float64 | 1 |
| Negative polarity score of parent tweet | float64 | 18 |
| Negative polarity score of author description | float64 | 3 |
| Negative polarity score of parent author description | float64 | 1 |
| Ratio of author's followers and follows | float64 | 11 |
| Account age difference of authors | int64 | 11 |
| Time between author account and tweet creation | int64 | 0 |
| Author uses default profile image | boolean | 1 |
| Conversation is self reply | boolean | 1 |

**Table 4.5:** Engineered features for meta learner. The respective F-scores were computed using the 'weight' criterion in the *plot_importance* function from the XGBoost library, indicating the frequency of each feature's appearance across all decision trees.

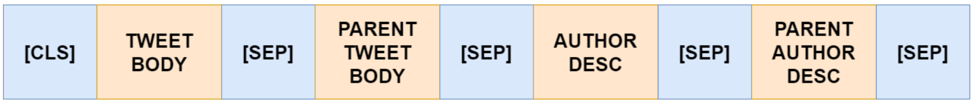


**Figure 4.9:** Sequence input format for the ContextBERT model with the specialized tokens represented with square brackets.

correctly interpret the features as separate and can infer a relationship between them. The format of the input sequences for the BERT model can be seen in Figure 4.9.

The computational complexity of the input size associated with the self-attention mechanism for BERT models mentioned in subsection 2.5.2 is quadratic, so we limited the input size as much as possible without letting it impact the performance. To find out the optimal maximum input length for each classifier, we followed an exploratory approach where we tokenized the inputs and plotted their distribution. The self-attention mechanism operates on smaller chunks of the input sequence, so it's generally recommended to set the length to a multiple of $2^n$. With this methodology, we set the max token length to 64 for NaiveBERT and 256 for ContextBERT.

### 4.6.5 Optimization of Hyperparameters

We used the Optuna library [79] to perform hyperparameter tuning with the Tree-structured Parzen Estimator (TPE), a Bayesian optimization technique, as the search algorithm for the BERT models. The decision to use TPE was made because of its memory-effectiveness, as we are constrained by the GPU memory capacity, and its effectiveness in handling imbalanced datasets and high-dimensional search spaces. The ranges set and the optimal values for the hyperparameters after 50 trials can be seen in Table A.3 in the Appendix. We optimized for the often preferred ROC AUC as it is a metric that is robust against class imbalance and because it provides a complete evaluation of performance across different probability thresholds.

The XGBoost model is part of a different framework and has different hyperparameters, so we instead used the Grid Search Cross Validation class in the Sklearn library [60] with the XGBoost model as the input estimator. These can be seen in Table A.4 in the Appendix.

### 4.6.6 Assessing Impact on Classification Cost

While prioritizing performance by optimizing for ROC AUC, we also incorporated a way to evaluate the classification cost metric to have a metric that more properly illustrated the impact of the performance difference. The classification cost was computed based on the proportion of binary classification outcomes, as defined in subsection 2.4.3, and the associated cost of each outcome. The average cost per classification decision is calculated using the formula provided in equation 4.5. This metric is a weighted average of the costs for each type of classification outcome normalized by the size of the data to be comparable across different dataset sizes.

$$C_C = \frac{TP \cdot C_{TP} + TN \cdot C_{TN} + FP \cdot C_{FP} + FN \cdot C_{FN}}{TP + TN + FP + FN} \tag{4.5}$$

#### Establishing Relative Costs

Without a concrete system to estimate the absolute costs, we assigned costs to each of the four classification outcomes on a relative basis. This way, we could compute the relative savings in the classification costs between any two schemes. We grounded the decision in our literature review to quantify the cost of each classification outcome given its consequences on the properties defined in subsection 4.2 and a flat cost to account for the system load of processing.

- Cost of false negative ($C_{FN}$): Incoming legislation like the Digital Services Act imposes penalties and legal consequences for platforms that fail to detect instances of hate speech. This leads to both direct financial repercussions and indirect consequences, such as damage to reputation. The failure to detect hate speech can ultimately lead to users abandoning the platform, potentially triggering a complete boycott of the platform. Therefore, assigning the highest cost to false negatives reflects the potential financial and reputational damage incurred when hate speech goes undetected. Accordingly, false negatives were assigned a relative cost of 100.

- Cost of false positive ($C_{FP}$): When non-hateful content is mistakenly flagged as hate speech, it creates a burden on human evaluators who need to review and rectify these false positives. This incurs additional costs due to the increased workload imposed on human evaluators, leading to some increased operational expenses. False positives can also result in delayed action against genuine instances of hate speech since human evaluators have to review and label more instances. The longer it takes to identify and address hate speech, the higher the potential harm caused. Moreover, if a user is mistakenly banned, it can result in detrimental consequences for the platform's reputation as well as potential financial implications due to users abandoning the platform. However, the likelihood of experiencing these repercussions is considered lower compared to the occurrence of false negatives. As a result, false positives were assigned a relative cost of 20.
- Cost of true positive ($C_{TP}$): When hate speech is accurately detected, it is sent to human evaluators for further review and action. While this step is necessary, it still incurs a cost due to the involvement of human resources. Assigning a lower cost to true positives reflects the inherent need for human involvement but acknowledges that it is a more desirable outcome compared to false classifications. True positives were therefore assigned a relative cost of 5.
- Cost of true negative ($C_{TN}$): Instances of non-hate speech that are correctly classified as such incur minimal costs. The lower cost assigned to true negatives recognizes that accurately identifying non-hate speech is essential but does not impose a significant burden on resources. Consequently, a relative cost of 1 was assigned to true negatives.

We have chosen these values ourselves, although ideally, multiple relevant actors would have been a part of the decision process. Our chosen values are affected by internal biases, but will however suffice to illustrate how such an evaluation should be conducted. In this manner, false negatives lead to a significant escalation in costs when compared to other outcomes.

### Post-Processing Threshold Optimization

To better adapt our model's predictions to the real-world costs of the established classification errors, we've adopted a post-processing technique known as decision threshold tuning. Instead of using the default threshold of 0.5, which treats false positives and false negatives equally costly, we choose our threshold to minimize our classification cost $C_C$. This optimization process involves iteratively evaluating the cost metric for different thresholds, and then selecting the threshold that results in the lowest cost.

We recognize that a more fitting alternative would be to integrate the cost metric directly into the training process, through a method known as cost-sensitive learning. While this approach could yield further improvements, we decided not to pursue this approach due to the abstract nature of the relative cost values and the complexity of such a scheme. However, we view cost-sensitive learning as a promising area for future work.

## 4.7 Quantitative Analysis of Cross-Validation Trials

In addressing the fourth subgoal, which involved conducting quantitative experiments for strategies where the literature was insufficient, we sought to employ a rigorous cross-validation methodology. K-fold cross-validation is an evaluation technique where the original dataset is divided into k subsets, called *folds*, of equal size. The model is subsequently trained k times, each iteration altering the combination of folds that compose the training and test sets. We performed k-fold cross-validation with k=10 for the trials where statistical significance was a primary concern, demanding a more rigid trial approach. With 10-fold cross-validation, we split the data into 10 smaller chunks referred to as folds. To account for the class imbalance, we utilized a stratified approach to assert the same class imbalance in each split. With 10 performance measures for each classifier, we could then use statistical methods to assert the significance of the potential gains. The cross-validation schemes employed for each trial are visualized in subsections 4.7.1 and 4.7.2.

When comparing the performance of machine learning models, using traditional t-tests is often not appropriate as they impose strict statistical assumptions, such that the data is independent and identically distributed approximately according to a normal distribution. While performance metrics such as the $F_1$ score or ROC AUC are indeed constrained within the range of 0 to 1, their distributions are often skewed due to the properties of the underlying data, rather than these constraints. Therefore, they typically do not satisfy the assumption of normality. Moreover, the assumption of independence can be compromised when performance measures are computed for various models using the same data folds, as this induces a dependence between the evaluations. Therefore, we used the Wilcoxon signed-rank test[16] instead.

[16]The Wilcoxon signed-rank test is a non-parametric statistical hypothesis test commonly used to assess whether there is a significant difference between the median of a paired sample and a hypothesized value, or if the difference between two paired samples is significant.

$$\begin{aligned} H_0 &: M_D = 0 \\ H_1 &: M_D \neq 0 \end{aligned} \tag{4.6}$$

While we anticipate a positive direction of difference, it's possible that the performance could decrease if the added features only introduce noise without contributing predictive power. Therefore, given the uncertainty about the direction of the expected effect, we used a conventional two-sided hypothesis test. This approach is used when the prior hypothesis doesn't strongly suggest a specific direction. The hypotheses can be seen in equation 4.6 where $M_D$ is the median difference between pairs of observations derived from performance metrics such as the $F_1$ score or the ROC AUC.

### 4.7.1 Trial for Evaluating Significance of Contextual Strategy

To compare NaiveBERT with ContextBERT, we trained each model with the same 9 folds and tested on the remaining fold a number of 10 times as outlined in Figure 4.10. We then got a distribution of performances that could assess the significance using the hypothesis test as outlined.

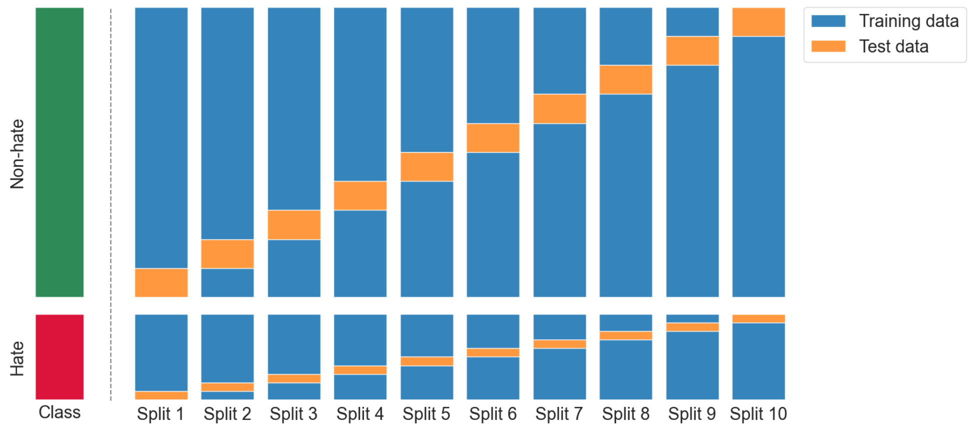


**Figure 4.10:** Stratified 10-fold cross-validation scheme used for testing the performance difference between NaiveBERT and ContextBERT. Class proportions are not to scale.

To test the performance gain from ContextBERT to ContextBoost, we needed the output of ContextBERT for each of the data points as a basis for computing the meta-features. However, this introduced a potential problem with data leakage, where ContextBERT performed inference on samples it had inadvertently been trained on. We avoided this by storing the model's predictions on the test set for each split. These stored predictions were then used as part of the input for ContextBoost during its own cross-validation process. However, this scheme introduced a potential bias. While mitigated to a certain extent by providing diverse training samples across folds in the cross-validation process, the predictions from ContextBERT were inevitably influenced by the specific subset of data used for training in each fold. Consequently, these predictions could reflect the biases inherent to the particular training data used in each fold. However, we found this trade-off necessary to avoid compromising the data integrity in our experiments.

### 4.7.2 Trial for Evaluating Significance of Sampling Strategy

Upon delving into sequential fine-tuning, we found that the existing literature fell short in terms of providing a comprehensive formulation and evaluation of sampling methods for extracting suitable samples for fine-tuning the model post-deployment for the hate speech management domain. As such, we found it necessary to develop a cost-efficient and scalable sampling method for sequential fine-tuning of the detection model. We wanted to test if using the uniqueness and classification complexity of the instances to sample yields better results than sampling randomly. To compute proxies for these properties, we needed a model which had been fine-tuned on instances of the dataset already. To avoid data leakage, we used one of the folds in each split to train the model initially, and then we could use the resulting model to perform inference on 8 of the folds we've designated as the sample space. Each of the sampling methods could then freely use the same sample space, and was set to draw 20% of the data given some approach from this space for further fine-tuning. Afterward, the sequentially fine-tuned model would be tested on the test data similarly to the context trials. This scheme is visualized in Figure 4.11.

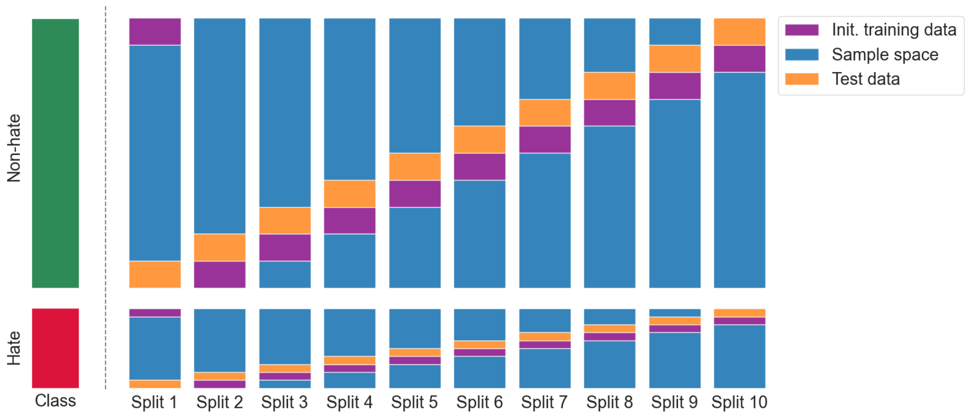


**Figure 4.11:** Stratified 10-fold cross-validation scheme used for testing the performance difference between random sampling and the uniqueness and uncertainty sampling heuristic. Class proportions are not to scale.

## 4.8 Unifying Strategies in a Joint System Architecture

Once the strategies to optimize the identified system properties had been established after verification through both qualitative and quantitative evaluation, we faced a new task. The next step, corresponding to the fifth subgoal, was to combine these strategies effectively in a way that still allowed each system property to be optimized. This necessitated taking into account the interdependencies between the different strategies.

We employed the previously mentioned machine learning lifecycle definition by Ashmore et al. [4] in order to categorize strategies. This involves determining the specific stage of the lifecycle to which each strategy corresponds. It is important to note that the formulated strategies do not exclusively belong to a single stage but were assigned to the most suitable stage. By categorizing the strategies based on the stage in the machine learning lifecycle where they are implemented, we were able to define system components. In this process, we also made reference to our literature review to identify the essential considerations required for the successful implementation of each strategy within the overall architecture. This approach allowed us to leverage existing knowledge and best practices, further enhancing the effectiveness and robustness of the implemented strategies in the system architecture.

Ideally, we would have also been able to perform simulations to evaluate the entire final system architecture. Unfortunately, with limited time, we only performed theoretical analysis to evaluate the architecture developed. This theoretical analysis is based on our initial literature review, as well as the quantitative experiments conducted.

## 4.9 Limitations of Methodology

Although this thesis has identified several system properties for implementing a hate speech management system on social media platforms, it is important to note that there are other properties that have not been considered in this research. As social media platforms and hate speech evolve, new properties may arise, and each platform may prioritize different properties based on their specific needs and objectives. For instance, some platforms may prioritize scalability and might choose to have fewer human moderators, while others may prioritize accuracy and might choose to have more human moderators. This calls for an iterative requirements engineering process rather than the linear process we presented in our methodology.

Some limitations should be acknowledged regarding the methods that we have used to create our dataset. First, it was limited to the English language. The pre-trained language models have shown lower performance for other languages than

English, and hate speech in one language might not be the same as hate speech in another language. Additionally, there are cultural norms and rules attached to each language. These factors could mean that the findings may not prove generalizable to other languages. Secondly, the dataset used in this study was limited in size and diversity. It is likely that some types of hate speech were underrepresented or not included in the dataset. Furthermore, hate speech continues to evolve and adapt to the measures put in place by social media platforms. A new form of hate speech may require additional contextual information to classify accurately. It is also worth acknowledging that we employed a dataset that we created ourselves. This dataset was developed based on our own definition of hate speech, utilizing GPT-labeling with in-context examples that we formulated and active labeling performed by ourselves as annotators. Hence, it is important to acknowledge that this process carries the risk of introducing our implicit biases into the dataset and consequently influencing the outcomes of the quantitative experiments. Ideally, the definition of hate speech, the GPT in-context examples, and the active labeling should have been collaboratively crafted and executed by a group of individuals.

Another limitation is regarding the processing. As social media users become more diverse, platforms may need to consider how their models perform across different languages and cultural contexts. Additionally, our research only takes into account textual posts and their corresponding contextual information. However, a comprehensive hate speech management system could take into account multimodal communication; social media posts can also consist of images and videos.

Our results show that the transformer-based language models that exist today can be used to ensure performance, scalability, and cost-efficiency. However, it is important to note that solely relying on the size, inference time and performance of models may not necessarily yield the optimal strategy. As highlighted in our literature review, research also emphasizes the significance of effectively training the model. While XLNet and BERT may be the most cost-effective in terms of size and performance, ELECTRA employs a more efficient training architecture. In all of our presented strategies, we have made the assumption that the first round of training of the model is already accounted for, which may not always be the case for a social media platform.

# Chapter 5 Results and Discussion

In this chapter, we will outline and discuss the strategies that we have identified, grounded both in experimental findings and existing literature. We will then propose the detection architecture of a hate speech management system that integrates these strategies. Finally, we will consider the practical implications that arise from our findings, the necessary preconditions for integrating the proposed architecture, as well as the limitations of the results.

## 5.1 Specialized Strategies for Hate Speech Management

We will present the strategies in a structured manner, outlining each strategy before evaluating its impact on relevant system properties. Figure 5.1 depicts the mappings between the proposed strategies and the system properties they optimize. The evaluations are based on theoretical analysis from our literature review and the results from the quantitative methods described in the previous chapter. If quantitative experiments were conducted to validate a strategy, we will present and discuss the results.

### 5.1.1 Integration of Contextual Information in Detection Model

In the rigorous context experiment that we carried out, we found that the performance of our baseline classifier as seen in Table 5.1 aligns well with what we have seen in the existing literature. Furthermore, there is a minimal distinction in the performance of the other developed classifiers. This is natural, considering there are often only incremental advancements when approaching the optimal value. In situations where the model does not account for noise, inherent flaws, or complexities in the data, this is to be expected. In relation to the Digital Services Act, lowering the FNR could lead to increased compliance with the regulatory restrictions related to removing hate speech. Hence, even small improvements hold significance, reducing costs for social media platforms. This is the case when including only textual contextual features in the learning process as we now will present.

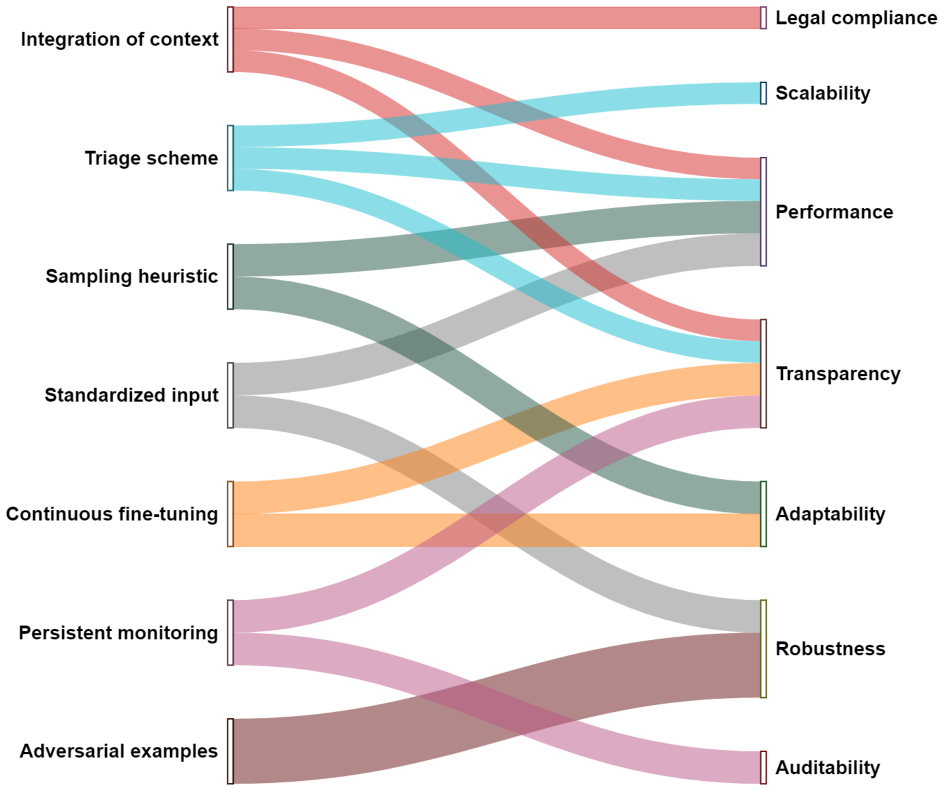


**Figure 5.1:** Mapping between the proposed strategies and the system properties they mainly affect.

The significance of the performance difference between NaiveBERT and ContextBERT, as well as between ContextBERT and ContextBoost, can be found in Table 5.2. The comparison between NaiveBERT and ContextBERT reveals improvements in the precision and the ROC AUC. However, the impact on the recall and the $F_1$ score remains inconclusive. Additionally, incorporating ContextBERT leads to a reduction in cost. When moving from ContextBERT to ContextBoost, we see a gain in the recall but also in the cost metric and reductions in the precision and the $F_1$ score. Due to the lack of significant evidence and the additional complexity introduced by ContextBoost with the two-step architecture, our strategy will not include the integration of numerical contextual features, but only the textual, contextual features. We leave finding a use for additional numerical contextual features for future work.

We can visualize the performance gain of ContextBERT over NaiveBERT with the ROC plot. From Figure 5.2, ContextBERT has significantly higher discrimination

| **Split** | **Precision** | **Recall** | $\mathbf{F_1}$ | **ROC AUC** | $\mathbf{C_C}$ |
|---|---|---|---|---|---|
| 1 | 0.921 | 0.930 | 0.924 | 0.919 | 5.420 |
| 2 | 0.928 | 0.921 | 0.924 | 0.912 | 5.491 |
| 3 | 0.920 | 0.916 | 0.918 | 0.923 | 5.584 |
| 4 | 0.931 | 0.931 | 0.931 | 0.913 | 5.003 |
| 5 | 0.924 | 0.932 | 0.927 | 0.883 | 5.825 |
| 6 | 0.914 | 0.917 | 0.916 | 0.883 | 6.247 |
| 7 | 0.914 | 0.910 | 0.912 | 0.888 | 5.820 |
| 8 | 0.926 | 0.928 | 0.927 | 0.925 | 4.904 |
| 9 | 0.923 | 0.931 | 0.926 | 0.910 | 5.539 |
| 10 | 0.921 | 0.911 | 0.916 | 0.915 | 5.538 |
| **Mean ± SD** | $0.922 \pm 0.005$ | $0.923 \pm 0.008$ | $0.922 \pm 0.006$ | $0.907 \pm 0.015$ | $5.537 \pm 0.390$ |

**(a)** NaiveBERT

| **Split** | **Precision** | **Recall** | $\mathbf{F_1}$ | **ROC AUC** | $\mathbf{C_C}$ |
|---|---|---|---|---|---|
| 1 | 0.926 | 0.926 | 0.926 | 0.926 | 4.998 |
| 2 | 0.933 | 0.922 | 0.927 | 0.924 | 4.906 |
| 3 | 0.927 | 0.921 | 0.924 | 0.924 | 4.882 |
| 4 | 0.936 | 0.927 | 0.931 | 0.937 | 4.604 |
| 5 | 0.924 | 0.906 | 0.914 | 0.905 | 5.332 |
| 6 | 0.920 | 0.907 | 0.913 | 0.907 | 5.707 |
| 7 | 0.931 | 0.927 | 0.929 | 0.913 | 5.444 |
| 8 | 0.937 | 0.919 | 0.927 | 0.924 | 4.834 |
| 9 | 0.929 | 0.915 | 0.921 | 0.918 | 4.553 |
| 10 | 0.927 | 0.916 | 0.921 | 0.918 | 5.045 |
| **Mean ± SD** | $0.929 \pm 0.006$ | $0.918 \pm 0.008$ | $0.923 \pm 0.007$ | $0.921 \pm 0.010$ | $5.031 \pm 0.366$ |

**(b)** ContextBERT

| **Split** | **Precision** | **Recall** | $\mathbf{F_1}$ | **ROC AUC** | $\mathbf{C_C}$ |
|---|---|---|---|---|---|
| 1 | 0.920 | 0.926 | 0.923 | 0.930 | 4.881 |
| 2 | 0.906 | 0.924 | 0.896 | 0.916 | 4.483 |
| 3 | 0.914 | 0.927 | 0.905 | 0.914 | 5.022 |
| 4 | 0.916 | 0.930 | 0.914 | 0.927 | 4.558 |
| 5 | 0.914 | 0.926 | 0.917 | 0.891 | 5.098 |
| 6 | 0.899 | 0.914 | 0.904 | 0.907 | 5.449 |
| 7 | 0.928 | 0.922 | 0.886 | 0.907 | 4.952 |
| 8 | 0.918 | 0.930 | 0.909 | 0.924 | 4.670 |
| 9 | 0.922 | 0.931 | 0.910 | 0.929 | 4.975 |
| 10 | 0.916 | 0.925 | 0.919 | 0.921 | 4.881 |
| **Mean ± SD** | $0.915 \pm 0.008$ | $0.925 \pm 0.005$ | $0.908 \pm 0.011$ | $0.916 \pm 0.012$ | $4.897 \pm 0.280$ |

**(c)** ContextBoost

**Table 5.1:** Performance results of stratified 10-fold cross-validation trials for the three classifiers. The bottom line for each model displays the mean and standard deviation of the metrics.

power; the TPR is higher for every FPR value. Having an increasing TPR for every FPR value indicates that ContextBERT is able to correctly identify more instances of hate speech while maintaining a low FPR. In other words, ContextBERT effectively distinguishes between hate speech and non-hate speech, correctly classifying hate speech instances as positive while minimizing the misclassification of non-hate speech instances as positive.

| | **Precision** | **Recall** | **$F_1$ Score** | **ROC AUC** | **$C_C$** |
|---|---|---|---|---|---|
| **ContextBERT - NaiveBERT** | | | | | |
| Median of paired differences | 0.005 | -0.005 | 0.001 | 0.012 | -0.493 |
| W-statistic | 0 | 14 | 26 | 0 | 0 |
| P-value | <0.001* | 0.193 | 0.922 | <0.001* | <0.001* |
| **ContextBoost - ContextBERT** | | | | | |
| Median of paired differences | -0.015 | 0.006 | -0.007 | -0.007 | -0.164 |
| W-statistic | 0 | 3 | 5 | 10 | 0 |
| P-value | 0.002* | 0.021* | 0.020* | 0.084 | 0.105 |

* Statistically significant given $\alpha = 0.05$

**Table 5.2:** Wilcoxon signed-rank tests for performance differences between the three classifiers. Each comparison measures the performance of the first model relative to the second. For the precision, recall, $F_1$ Score, and ROC AUC, positive median differences indicate higher scores for the second model, while negative differences favor the first model. For the $C_C$ metric, which we aim to minimize, the opposite is true: negative differences indicate better performance for the second model.

While we have seen significant performance gains by including certain contextual features, it is hard to quantify the impact of this on an operational ground by reviewing the classification metrics alone. Table 5.2 demonstrates that there is significance in the reduction of the classification cost $C_C$ given the relative costs and threshold optimization post-processing as described in subsection 4.6.6. Correspondingly, Table 5.1 outlines a 9.14% reduction of the mean classification cost (from 5.537 to 5.031), which is a substantial amount. However, there is a trade-off between performance and the inference cost. Following the quadrupling of the input size, we saw the mean of the *eval_samples_per_second* metric, which represents the number of samples the model can perform inference on per second, in the Trainer class decrease from 868 to 271. While these numbers should be interpreted cautiously due to their dependency on hardware conditions and may lack rigorous generalizability, they serve to illustrate the potential cost-performance trade-off that needs to be considered. This 69% decrease could have serious financial consequences on an already strained infrastructure of detection models. Figure 5.3 plots the classification cost over the classification threshold for NaiveBERT and ContextBERT.

Using the optimized thresholds based on lowering the classification cost, we plotted the resulting classification outcomes in confusion matrices to contextualize the impact of the performance gain from NaiveBERT to ContextBERT and from using a standardized classification threshold to a cost-optimized one. Figure 5.4 illustrates that the overall classification patterns changes for both models when adopting a cost-optimized threshold. Both models experience reductions in the false negative rate as expected since we have designated this as the most costly

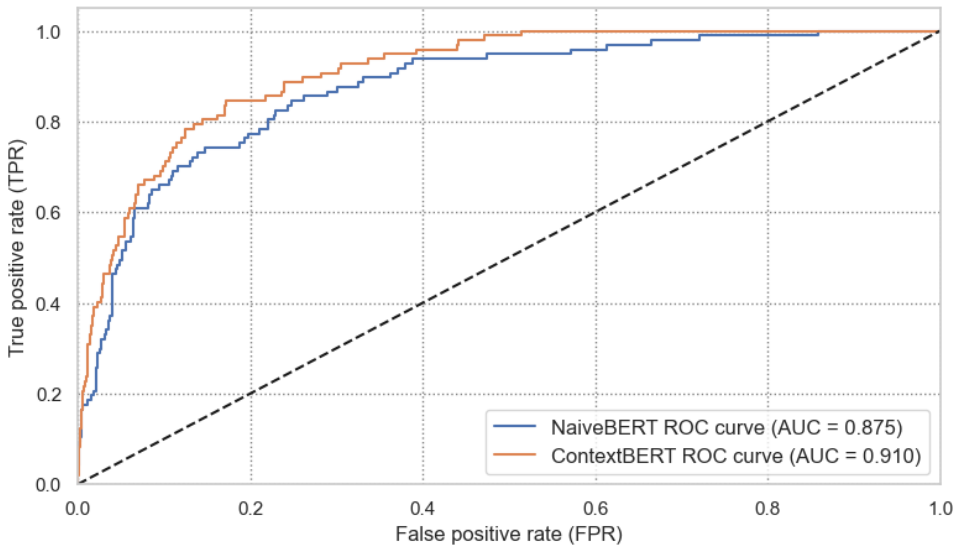


**Figure 5.2:** ROC curves for NaiveBERT and ContextBERT for the last cross-validation split, displaying the relationship between the TPR and FPR across classification thresholds.

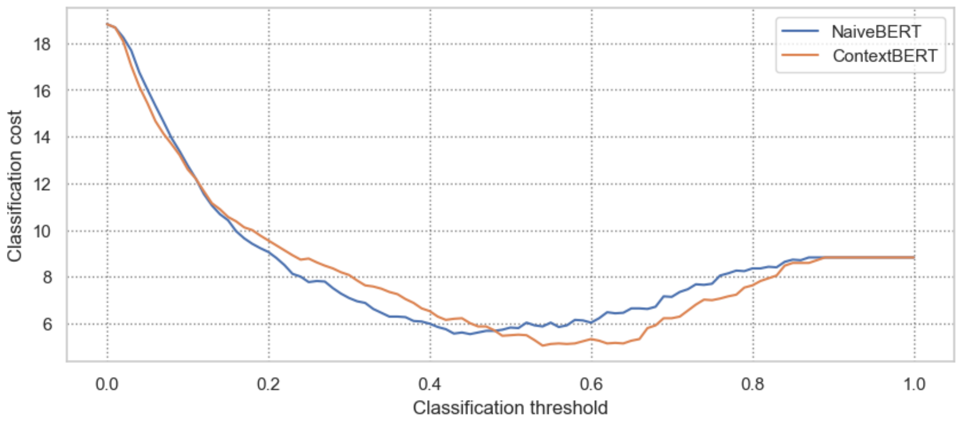
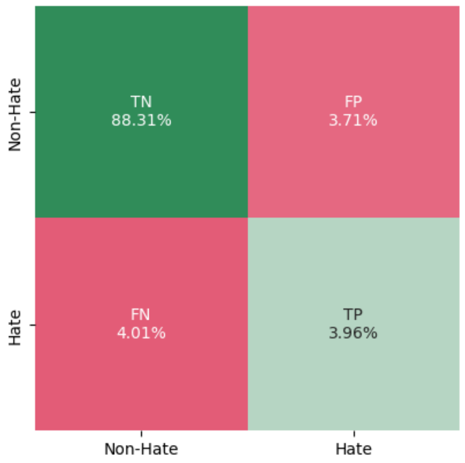
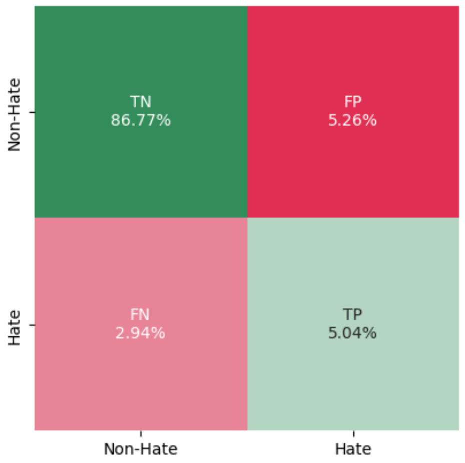


**Figure 5.3:** Classification costs given classification threshold for NaiveBERT and ContextBERT for the last cross-validation split.

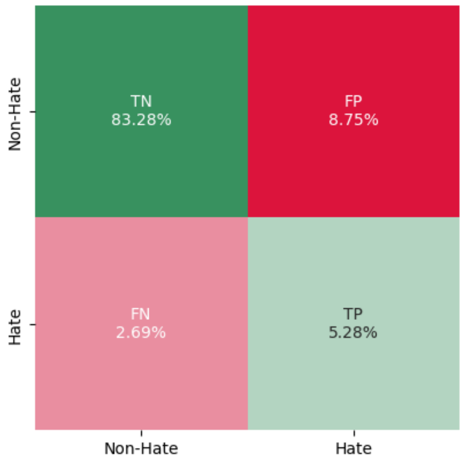
outcome. Nevertheless, it is apparent that this post-processing optimization comes at a price; there has been a notable increase in the misclassification rate. This needs to be countered by further review and verification, which is the current industry practice as outlined in subsection 2.3.1, as such dramatic false positive rates would lead to unjustified actions, jeopardizing legitimate users' experience, and negatively impacting the credibility of the platform.



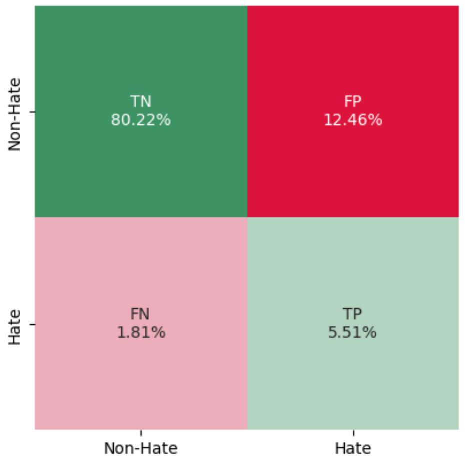
**(a)** NaiveBERT with standard classification threshold



**(b)** ContextBERT with standard classification threshold



**(c)** NaiveBERT with cost-optimized classification threshold



**(d)** ContextBERT with cost-optimized classification threshold

**Figure 5.4:** Confusion matrices of aggregated classification outcomes with various post-processing techniques for trials of contextual information.

In addition to the performance property, there is also a concern about transparency. Incorporating contextual information will lead to an increase in the number of features, which in turn would have a negative impact on the transparency of the detection system. Transparency will be affected if the system fails to provide clear explanations or insights into how the contextual information is used and weighted in the machine learning model. This can be achieved by conducting an analysis to determine the importance and weighting of the features used in the model. This analysis should involve techniques such as feature importance scores and sensitivity analysis that provide insights into how each feature contributes to the model's predictions [72]. This could in turn lead to higher interpretability and transparency.

### 5.1.2 Standardizing Input Format for Detection Model

Our literature review revealed adversarial attacks as a significant problem for hate speech detection. Often, adversarial attacks in the context of hate speech detection aim to maintain the post's readability while letting the hateful speech of the post go undetected by performing small alternations to the input text, as can be seen in Table 3.4. One cost-effective approach to increase the model's robustness to such attacks is to preprocess in order to standardize the input text. Standardizing the input through preprocessing involves making necessary modifications to the text, ensuring uniformity. The applied modifications to the text will effectively undo some of the alterations introduced by the adversarial attack. This enables us to counter certain types of adversarial attacks and restore the integrity of the data. To standardize the input, we implement the TSAR framework by Moh et al. [47]. The framework consists of three preprocessing steps to account for three different types of adversarial attacks that Moh et al. developed specifically for hate speech detection. It's important to note that individually, the attacks listed below will likely not bypass the detection of hate speech when employing a state-of-the-art transformer-based model. However, when combined, these attacks may potentially evade detection by the model. These preprocessing steps are as follows:

1. Firstly, to account for the lexical attacks that exploit the removal of whitespaces to evade the detection model, a mitigation approach involves segmenting composite words into their smallest root forms. For a transformer-based model such as BERT, removing whitespaces would alter the tokenization process, leading to incorrect interpretation. Combined with the rest of the listed attacks, the text might become unintelligible. By reverting this attack, we can better preserve the original intent of the text [47].

2. The second essential preprocessing step involves considering intentional typographical errors. To undermine the model's recognition of hate speech, an adversary may rearrange the letters within a word. To mitigate this attack, a dictionary lookup technique matches the unknown word with words containing the same number of characters and total ASCII value. This dictionary may include both widely recognized hateful words and negatively charged terms. By employing this approach, the unknown word is substituted with the word

that is deemed most probable to be the original word prior to rearranging its letters.

3. Finally, to counteract the attack of injecting a word with an overwhelmingly positive sentiment into the hateful post, it is necessary to analyze the post's grammatical sentence structure. Words that violate grammar rules and lack negative sentiment are identified and removed as a precautionary measure. This can be done through part-of-speech tagging and sentiment analysis [47].

Implementing the preprocessing steps of TSAR significantly decreases the likelihood of these specific adversarial attacks by standardizing the input format. Employing preprocessing to enhance robustness against adversarial attacks offers the advantage of reducing input size and thus lowering the cost of inference. However, it is important to note that this approach introduces some additional latency due to the inclusion of extra preprocessing steps.

While improving robustness against adversarial attacks, these measures may also decrease the model's accuracy on non-adversarial examples. This is because some preprocessing defense techniques may also alter the clean examples and introduce noise that affects the model's accuracy [47]. For instance, there are cases where the composite word carries a distinct meaning that is not simply a combination of the separated root words. In such situations, applying this approach can lead to unintended repercussions. Hence, the extent to which these techniques are implemented must be appropriately adjusted.

### 5.1.3 Detection Triage Scheme

To have a high classification performance across all levels of complexity of a social media post, one needs an annotation scheme that can consistently classify instances with high complexity. Traditionally, human annotators have been utilized to ensure the correctness of the detection outputted from the machine learning model. However, such annotators have much higher costs and are less scalable. Therefore, there are real incentives for reducing cost while introducing flexibility and scalability to the detection module. We have analyzed annotation with BERT, GPT, and human moderation in order to develop an optimal annotation strategy. It's important to note that BERT was employed as a demonstration to showcase the concept of a lightweight language model. Therefore, our strategy encompasses a broader scope and is not limited to the use of BERT alone. Similarly, we have employed a specific GPT model to showcase the potential impact of utilizing such a model.

[1]Kernel Density Estimation is a non-parametric method for estimating a probability density function by using kernels as weights.

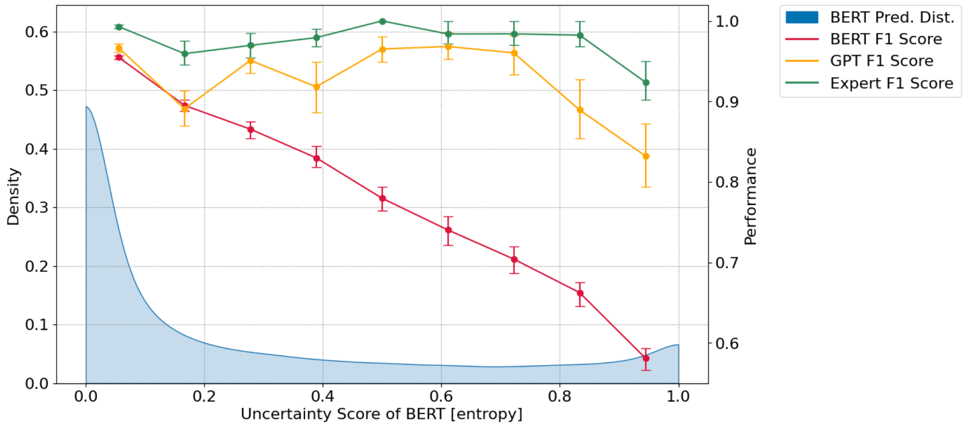


**Figure 5.5:** The $F_1$ score of BERT, GPT, and experts (i.e., human moderators) given BERT's uncertainty score. The performance of BERT is estimated via cross-validation on the entire dataset, while the GPT and expert performances stem from the optimal configuration tests during relabeling. Each scheme's performance reflects resampled means across nine bins, with a 95% confidence interval per bin. The prediction distribution is computed using a KDE[1]-weighted approach with a bandwidth parameter of 0.05.

By exploiting the log-probability output of the BERT model, which acts as our default detection model, we are capable of using the entropy as a reliable measure of the model's expected performance on a given instance. It demonstrates a predictable correlation with BERT's performance. In Figure 5.5, we can see the performances of annotation with BERT, GPT, and human moderators given the entropy of the BERT model. Intuitively, we can observe that the performance of BERT rapidly degrades as the entropy of its predictions increase. While this is a cheap and scalable way of detecting hate speech, it is seemingly not able to capture all the nuances in more complex instances.

GPT models have the ability to process large volumes of text quickly and efficiently, allowing for the automated analysis of a vast amount of data. By employing GPT, both the time and cost associated with annotation will be significantly reduced. We observe that the performance remains relatively stable until around the 70th percentile and that it can process more intricate pieces of hate speech. However, the transparency in the annotation process will be reduced when using a GPT-based model. GPT models are considered black-box models, meaning their decision-making process is not easily interpretable or explainable. To mitigate the transparency challenges with GPT-based models, it is possible to instruct the GPT model to append a reasoning for the classification, hence incorporating an important XAI principle similar to the technique used by Hendricks et al. [29] as explained in the literature review. To implement this approach, the prompt given to the GPT model should be modified to include specific instructions that guide the model to generate a justification for the predicted label, ensuring transparency and explainability of the model's decision-making process.

The human experts consistently outperform other entities irrespective of the level of uncertainty, but experts are expensive, and it's hard to scale day-to-day. Human moderators have limitations in terms of their capacity to review content manually. They may be restricted by time, availability, and the sheer amount of data to analyze. Scaling up a human moderation team to handle increasing volumes of content can be challenging and costly. Consequently, relying solely on human moderators for addressing the issue of hate speech annotation is not only expensive but also not scalable in the event of a surge in its intensity.

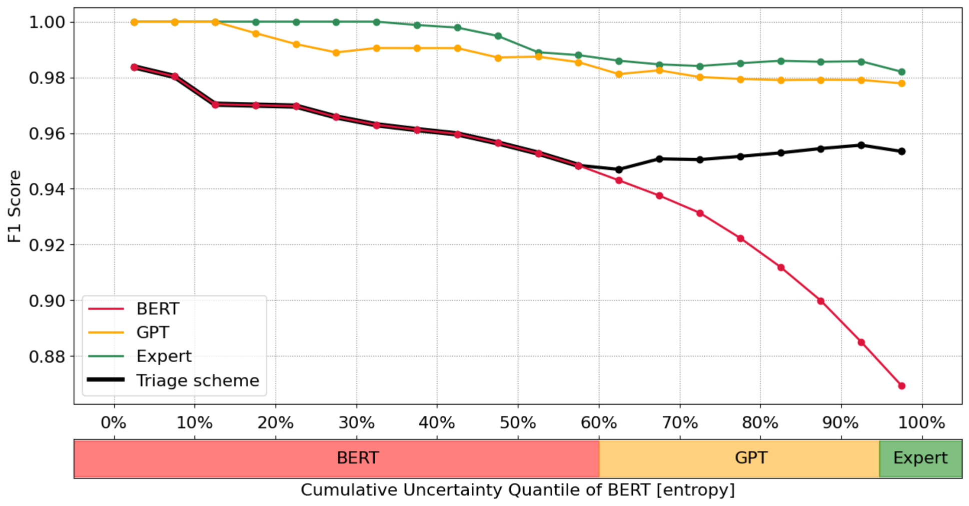


**Figure 5.6:** Cumulative performance of an example 60-35-5 resource split for a data annotation triage system. The cumulative performances of the individual labeling schemes are also shown. BERT is utilized for the least uncertain 60% samples of the data, GPT covers the 60th to 95th percentile, and experts are employed for the remaining top 5% quantile. The contribution of the annotation method for each quantile within the triage system is illustrated in the lower section.

These findings pave the way for the design of a triage system that uses a lightweight language model as the default model for detection and facilitates further processing for the more challenging cases. In the cases when the uncertainty reaches a certain level that has empirically shown suboptimal performance, we can pass on the annotation job to a more complex (and expensive) annotator. The choice between the GPT model and human annotators would be made contingent upon the initial uncertainty level as identified by the lightweight language model and the optimal trade-off between performance and cost as identified by the company.

The split between resources is customizable and would depend on aspects such as the cost of the resources, the load on each of the resources, and the set trade-off between cost and performance. A data annotation triage scheme is shown in Figure 5.6, where we can see that the cumulative performance would be comparable to solely using GPT or experts for annotation, but with added automation and flexibility. In this manner, the scalability of the annotation is substantially increased.

It is important to point out that the performance of our BERT model should be regarded as a lower limit because our model only has been trained on a small dataset. In a real setting, we would expect a lightweight language model such as BERT to have significantly more comparable results with larger models, and that our results using the specific BERT model should be merely treated as a proof of concept. Consequently, such a triage system will be included in our detection architecture.

### 5.1.4 Continuous Sequential Fine-Tuning

From our literature review, we have gathered that continuous fine-tuning can ensure consistent performance over time [32]. In the specific use case of hate speech detection, where the rate of the linguistic shift occurring in online social media is high [30], continuous fine-tuning is essential. However, instructing the model to be fine-tuned is not the largest challenge; it is the supporting architecture that involves precise validation and labeling that needs to be defined.

The process of continuous sequential fine-tuning commences with a pre-trained model with the appropriate classification head, as described in subsection 2.5.3, designed specifically for detecting hate speech. However, since the model's knowledge is based on a specific training dataset, it may struggle to accurately identify newly emerging forms of hate speech or account for shifts in language usage. The process of continuous sequential fine-tuning involves routinely adjusting the model with recently labeled data which accurately represents the changing patterns of hate speech. One of the key advantages of such a strategy is its ability to maintain the existing knowledge of the model while incorporating new information. Rather than discarding the previously learned patterns, continuous sequential fine-tuning leverages them as a foundation, allowing the model to build upon its existing understanding incrementally. Therefore, this strategy enables the model to adapt and maintain its performance over time as the distribution of upstream data changes.

In regards to transparency, the use of fine-tuning can make the model more complex and harder to interpret. This can make it more difficult to understand how the system functions, and hence it will be harder to identify any potential sources of bias or error. Versioning and documentation of each fine-tuning cycle will make the fine-tuning process auditable, as well as allow for a quick and easy rollback to a previous version in the event of errors or issues arising in the latest version. This will help administrators understand why the system may behave unexpectedly and how to interpret the results, subsequently increasing transparency.

Ensuring the system's adaptability by continuously updating the model with regular fine-tuning would likely increase costs. The cost of the additional human resources needed for human evaluation, the cost of collecting and labeling tweets, and the cost of storing datasets may add up. However, the benefits of an adaptable model may outweigh the added costs in the long run, as it can improve the model's accuracy and effectiveness in identifying novel hate speech while being robust against adversarial attacks.

| **Parameter** | **Coef** | **Std. Err.** | **t** | **P>\|t\|** |
|---|---|---|---|---|
| Intercept | 0.6590 | 0.005 | 123.986 | 0.000 |
| Instruction method | 0.0367 | 0.011 | 3.259 | 0.047* |
| Uncertainty threshold | 0.1050 | 0.011 | 9.310 | 0.003* |
| In-context examples | -0.0089 | 0.004 | -2.222 | 0.113 |
| Instruction method : uncertainty threshold | -0.0089 | 0.004 | -2.222 | 0.113 |
| Instruction method : in-context examples | 0.0034 | 0.014 | 0.248 | 0.820 |
| Uncertainty threshold : in-context examples | -0.0176 | 0.014 | -1.275 | 0.292 |

| | | | |
|---|---|---|---|
| **Dep. Variable:** | Utility | **R-squared:** | 0.990 |
| **Model:** | OLS | **Adj. R-squared:** | 0.973 |
| **Method:** | Least Squares | **F-statistic:** | 58.57 |
| **No. Observations:** | 9 | **Prob (F-statistic):** | 0.00344 |

* Statistically significant given $\alpha = 0.05$

**Table 5.3:** Regression model summary of fractional factorial design trials. The interaction terms between two parameters are denoted with a colon (:) in between.

Continuous fine-tuning inherently necessitates continuous labeling since it is a supervised learning task. Depending on the frequency of fine-tuning, this can incur a significant cost. To account for this, we want to use the knowledge we gained about GPT-based annotation in section 4.5. This involves using a GPT-driven labeling scheme similar to that outlined in the triage scheme in subsection 5.1.3, but without a lightweight language model such as BERT as the default model as we need to avoid feedback loops. The significance of the factors for the GPT-driven labeling scheme on the utility metric as defined in equation 4.3 can be seen in Table 5.3. We can see that both the instruction method and the uncertainty quantile threshold for active labeling have a significant influence on the utility. To gain a deeper understanding of the significance, we plotted the main effects of each factor in Figure 5.7. Here, it is

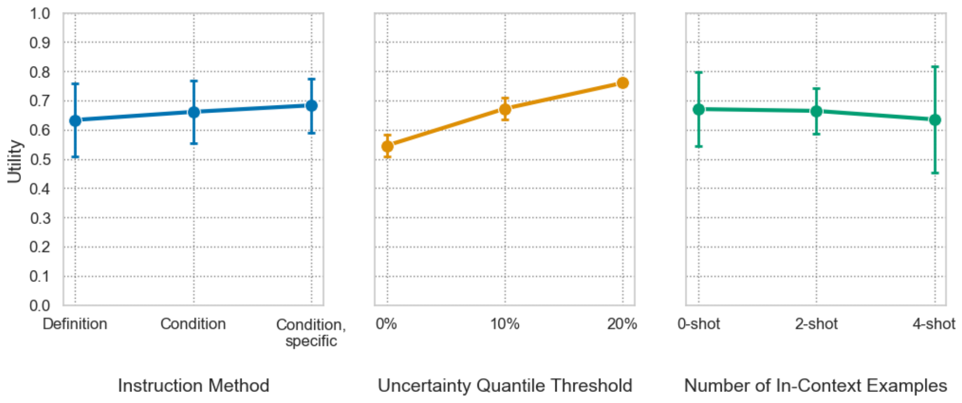


**Figure 5.7:** Main effect of factors on utility with standard deviation of fractional factorial design trials.

evident that the effect of the number of in-context examples remains inconclusive. However, the active labeling scheme demonstrates a notably significant impact. This suggests that employing GPT with active labeling for labeling during continuous fine-tuning emerges as a cost-effective and competitive alternative to relying solely on human annotators.

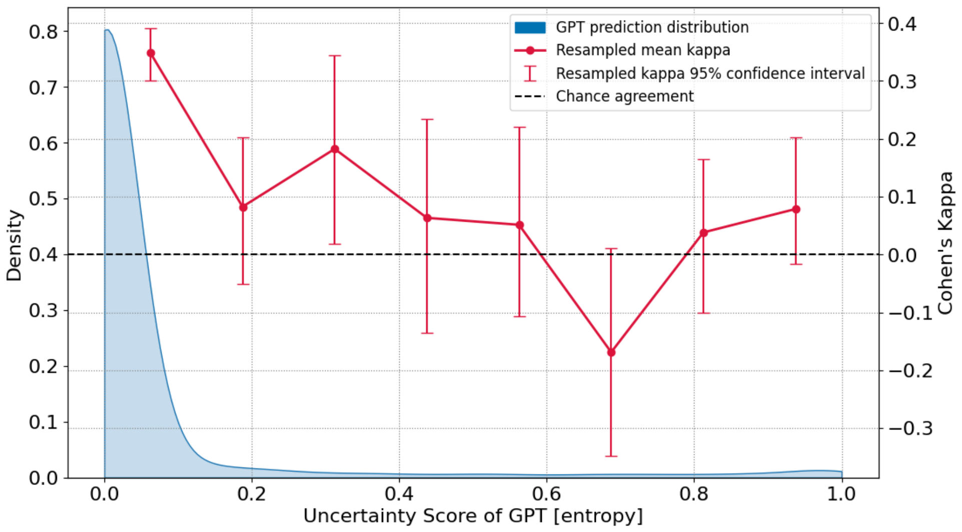


**Figure 5.8:** Cohen's kappa between the original Founta et al. labels, where the "abusive" class was assigned non-hateful, and the GPT scheme labels of our dataset. Resampling was utilized to compute each value and its corresponding confidence interval of the kappa values. The prediction distribution given the uncertainty score is computed using KDE.

To ground the performance of the labeling scheme, we compare it with the original labels of the Founta et al. dataset using Cohen's kappa[2] metric. In data annotation, Cohen's Kappa is frequently employed to evaluate the agreement between different annotators categorizing or labeling data. It's preferred over conventional performance metrics like the ROC AUC, as both sets of labels serve as the ground truth. In Figure 5.8, we plotted the agreement of the annotators given the uncertainty of the GPT model. Identifying a trendline for the GPT model presents more of a challenge than in the BERT model. This is largely due to the large confidence intervals associated with the GPT model, which arise from its more sporadic entropy distribution, with values typically further removed from both 0 and 1.

Furthermore, we notice a negative value indicating a level of disagreement that is lower than what would be expected by chance. This occurrence is highly improbable and is likely attributed to the aforementioned noise. Nevertheless, we can still observe that the agreement is likely highest when the uncertainty of the GPT model is low. Considering the significance of the previously discussed active labeling scheme tests, it becomes evident that both the GPT and BERT models possess the capability to distinguish instances based on their complexity, a measure determined by the degree of difficulty the model encounters when trying to predict the correct label. This demonstrates the concept that the uncertainty of the GPT model, in a similar fashion to the uncertainty of the BERT model, can be effectively utilized as a basis for determining whether further review by a human is warranted.

It is important to note that implementing such a strategy carries the risk of introducing positive feedback loops. The human evaluators may have their own biases or misunderstandings of what constitutes hate speech, which could result in incorrect labels. This could then feed back into the system and potentially reinforce the biases or errors in the model, resulting in even less accurate predictions. We can reduce the probability of such occurrences by using multiple evaluators for each label in combination with the above-mentioned GPT-based active labeling scheme.

### 5.1.5 Enhanced Sampling Heuristic of Uniqueness and Uncertainty

In implementing a fine-tuning strategy, the existing literature strongly emphasized the importance of developing a sampling strategy that takes both the uniqueness and uncertainty into account [103]. A quantifiable uniqueness metric is the cosine similarity of the input data compared to earlier processed data. By leveraging the embeddings from a lightweight language model like the base layers of BERT, which

[2]Cohen's Kappa is a statistical coefficient that measures the level of inter-rater reliability or the degree of agreement between two raters, while taking into account the probability of agreement occurring by chance.

is a high-dimensional vector representation of the input text, we can estimate the uniqueness of the input data in relation to what it has already been trained on. This is done by computing the high-dimensional centroid[3] for each class (hateful or non-hateful) for the small-scale set of samples the model had already been trained on. However, the output size of BERT base layer embeddings is 768, which could pose computational and practical challenges due to the curse of dimensionality[4]. To mitigate these issues and increase computational efficiency, we apply Principal Component Analysis (PCA) to reduce the dimensionality of the embeddings. With this technique, we reduce the dimensionality to 50, determined by selecting the point using the elbow heuristic[5], where adding further dimensions resulted in a significantly smaller increase in the explained variance.

With both the cluster and the embeddings of each instance reduced, we compute the cosine similarity to the nearest cluster of each embedding, normalized to [0,1] instead of [-1, 1]. The classification complexity was represented using the uncertainty of the model, through computing the entropy as we have done earlier. The combined metric is then the weighted sum of the two metrics. This workflow can be seen in Figure 5.9.

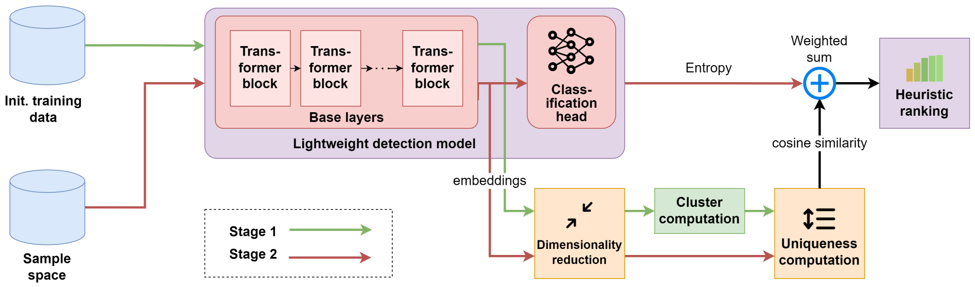


**Figure 5.9:** Sampling heuristic workflow for computing the combined uniqueness and uncertainty metric.

The evaluations of the cross-validation sampling trials using this sampling scheme are presented in Table 5.4. In this case, we can observe reduced generalizability across all metrics when comparing the model's performance on the test set to the context trials presented in Table 5.2. This outcome is anticipated since the model was trained on a significantly smaller dataset for each split. Instead, we are interested

[3]A clustering centroid is a central point computed by taking the mean in each dimension within a group of data points.

[4]The curse of dimensionality is when distances between data points lose their informativeness or discriminatory power in high-dimensional data due to diminishing differences.

[5]In PCA, the elbow heuristic identifies the optimal number of components by analyzing the plot of cumulative explained variance. It pinpoints the point where the variance explained starts diminishing.

in the performance gain by moving from random sampling to our heuristic sampling method. We see improvements in all metrics except for the precision.

| **Split** | **Precision** | **Recall** | **$F_1$** | **ROC AUC** | **$C_C$** |
|---|---|---|---|---|---|
| 1 | 0.894 | 0.863 | 0.877 | 0.781 | 7.271 |
| 2 | 0.897 | 0.838 | 0.862 | 0.766 | 6.896 |
| 3 | 0.902 | 0.864 | 0.880 | 0.844 | 6.170 |
| 4 | 0.884 | 0.815 | 0.848 | 0.814 | 6.974 |
| 5 | 0.893 | 0.852 | 0.869 | 0.744 | 7.302 |
| 6 | 0.890 | 0.828 | 0.854 | 0.725 | 7.584 |
| 7 | 0.899 | 0.800 | 0.838 | 0.795 | 7.180 |
| 8 | 0.906 | 0.816 | 0.850 | 0.825 | 6.899 |
| 9 | 0.912 | 0.837 | 0.865 | 0.827 | 6.162 |
| 10 | 0.911 | 0.820 | 0.853 | 0.835 | 6.242 |
| **Mean ± SD** | 0.899 ± 0.009 | 0.833 ± 0.022 | 0.860 ± 0.013 | 0.795 ± 0.040 | 6.869 ± 0.510 |

**(a)** Random Sampling

| **Split** | **Precision** | **Recall** | **$F_1$** | **ROC AUC** | **$C_C$** |
|---|---|---|---|---|---|
| 1 | 0.893 | 0.867 | 0.879 | 0.796 | 7.131 |
| 2 | 0.891 | 0.848 | 0.867 | 0.772 | 6.709 |
| 3 | 0.906 | 0.855 | 0.875 | 0.866 | 6.311 |
| 4 | 0.884 | 0.862 | 0.872 | 0.819 | 6.857 |
| 5 | 0.888 | 0.869 | 0.878 | 0.752 | 7.162 |
| 6 | 0.881 | 0.881 | 0.881 | 0.736 | 7.560 |
| 7 | 0.887 | 0.874 | 0.880 | 0.806 | 6.852 |
| 8 | 0.895 | 0.838 | 0.862 | 0.832 | 6.612 |
| 9 | 0.909 | 0.877 | 0.890 | 0.834 | 6.031 |
| 10 | 0.916 | 0.895 | 0.904 | 0.852 | 5.890 |
| **Mean ± SD** | 0.895 ± 0.011 | 0.867 ± 0.016 | 0.879 ± 0.012 | 0.806 ± 0.043 | 6.712 ± 0.521 |

**(b)** Uniqueness and uncertainty heuristic sampling

**Table 5.4:** Cross-validation results of random sampling and sampling heuristic of uniqueness and uncertainty. The NaiveBERT model was used for both cases.

The significance of the findings is presented in Table 5.5. We see significance in all the metrics, which shows improvement moving from random sampling to the sampling heuristic, and non-significance in the decrease in the precision. Sampling with uniqueness and uncertainty leads to statistically significant improvements in the recall, $F_1$ score, ROC AUC, and our classification cost. Hence, we regard such a sampling strategy as viable in regard to performance. Using such a sampling scheme for their lightweight detection model, a social media company might enhance the chances of refining the model with hate speech data while making the sequential fine-tuning process more cost-effective by minimizing the possibility of repeating the same refinements unnecessarily.

However, we would need to make a small trade-off; a degree of randomness must be maintained to ensure a distribution representative of actual data [99] [62]. We did not include randomness in the trials because it might dilute the observed effects of the sampling strategy. In some cases, the model may exhibit confidence in instances it perceives as having low complexity. If we solely rely on this sampling scheme, the model would never be trained on such samples. This would be particularly problematic for the model's adaptability, as it would struggle to handle new linguistic

| | **Precision** | **Recall** | **$F_1$ Score** | **ROC AUC** | **$C_C$** |
|---|---|---|---|---|---|
| **Heuristic - Random** | | | | | |
| Median of Paired Differences | -0.004 | 0.031 | 0.018 | 0.010 | -0.141 |
| W-statistic | 10 | 3 | 3 | 0 | 3 |
| P-value | 0.084 | 0.010* | 0.010* | 0.002* | 0.010* |

* Statistically significant given $\alpha = 0.05$

**Table 5.5:** Hypothesis test of the performance of the sampling strategy. The comparison measures the performance of the sampling heuristic relative to random sampling. For the precision, recall, $F_1$ Score, and ROC AUC, positive median differences indicate higher scores for the sampling heuristic, while negative differences favor random sampling. For the $C_C$ metric, which we aim to minimize, the opposite is true: negative differences indicate better performance for random sampling.

expressions that emerge over time. The sampling method could then miss out on instances crucial for updating the model, as it may not comprehend the nuances of previously unseen hate speech. Therefore, it is crucial that this heuristic is not treated as an absolute. Instead of relying solely on our heuristic for sampling, our approach aims to increase the likelihood of sampling based on the heuristic while still maintaining a certain level of randomness.

### 5.1.6 Advancing Robustness During Fine-Tuning

Including adversarial examples in the fine-tuning can increase the robustness against adversarial attacks. This involves continuous research into the latest adversarial attacks and techniques. Once identified, we can generate samples of those attacks and add them to the fine-tuning batches. During training, the model will learn to recognize and defend against those specific attacks. To ensure that the model is continuously updated to defend against the latest adversarial attacks, we need to regularly review and add new samples of attacks as they arise to the fine-tuning batch. This will involve ongoing monitoring of the model's performance and identifying any weaknesses or vulnerabilities that need to be addressed.

However, including adversarial examples during model fine-tuning may also increase the cost, as this requires continuous research into the latest adversarial attacks and techniques. Another aspect to consider is the potential impact on model performance and generalization when including adversarial examples in the fine-tuning process. Continuous monitoring and evaluation can help identify any degradation in performance on non-adversarial inputs and guide adjustments in the training methodology to ensure a balance between robustness and generalization.

### 5.1.7 Persistent Monitoring With Explainable AI

After deploying the model an important strategy is regularly scheduled monitoring. The monitoring should include regularly scheduled data analysis. This can be achieved through the use of XAI techniques, such as the feature importance techniques described earlier 3.6.3, which provide insights into how the model works and why it makes certain predictions. These techniques can help identify areas that need improvement. For example, if certain metadata features consistently lead to misclassifications, the model can be fine-tuned to better handle those features. It can also help identify potential sources of bias in the model's decision-making process, allowing for targeted interventions to address those biases. In analyzing the data it is important to involve diverse stakeholders, including individuals from underrepresented or marginalized communities who may be disproportionately affected by hate speech. This can help ensure that the system takes into account a range of perspectives and is developed in a way that is fair and equitable.

Using XAI for monitoring improves auditability and transparency. By regularly analyzing data, organizations can create documentation that tracks changes in upstream and downstream data, as indicated in the literature review 3.6.4. This ensures a clear record of modifications and enables easy traceability of those changes. This is especially important for evolving systems as it enables organizations to maintain a comprehensive understanding of how the system has evolved over time. By leveraging XAI for monitoring, organizations can gain valuable insights into the decision-making processes of their models and ensure accountability in their operations.

## 5.2 Implementations of Strategies for Unified System Architecture

With our domain-specific strategies defined, we will now unify these into one system architecture using the methodology outlined in section 4.8. Our system architecture builds upon the initial depiction of a hate speech management system presented in Figure 2.3 from the background chapter. Based upon this initial illustration, we define the detection approach, the preprocessing and post-processing parts, and extend the support architecture encompassing monitoring and the essential modules necessary for continuously fine-tuning the lightweight detection model.

Figure 5.10 illustrates the stages of the machine learning lifecycle in conjunction with the proposed strategies. This categorization enables us to define the system components of the detection architecture more effectively. The strategies of the data management stage constitute the preprocessing component which receives an input data stream and thereby is the first system component in the detecting architecture

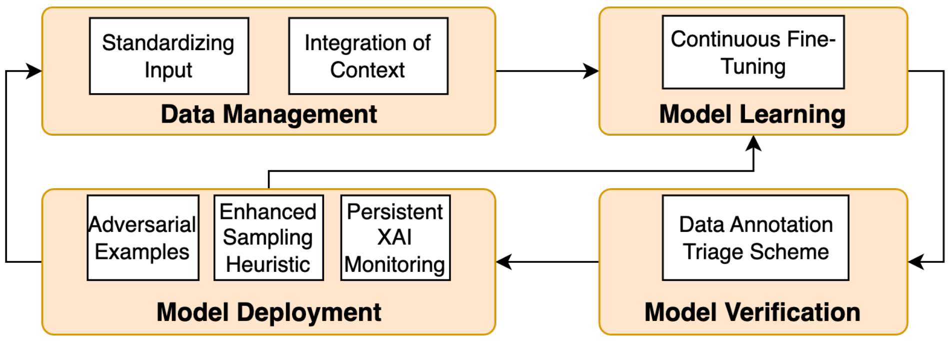


**Figure 5.10:** Proposed strategies integrated into the machine learning lifecycle defined by Ashmore et al. [4]., thereby illustrating the system components of the final architecture.

of a hate speech management system. Similarly, the model learning and deployment stages form the support architecture designed to accommodate the evolving language model. Finally, the model verification stage generates the labeled data stream that feeds into subsequent stages of the hate speech management system, such as an action center, and therefore is the last system component of the architecture. With this categorization of the established strategies, we can observe that the architecture forms an iterative process, in which the system continuously adapts.

Following the establishment of the system components, we utilized relevant findings from our literature review to increase the validity and reliability of the components when combining them into one architecture. This led us to identify various considerations and adaptions that were necessary in order to unify the strategies into one architecture. This section will first outline these necessary considerations and adaptions before presenting the final detection architecture. While we have incorporated relevant literature to support our findings, it is essential to validate and enhance the proposed detection architecture through quantitative experimentation, including validation and reliability testing. We acknowledge that this aspect requires further attention and investigation in future work.

## 5.2.1 Preprocessing

In the first step of our detection architecture, we preprocess the incoming data stream. In our case of using Twitter as the data source, this consists of Tweets but could be adapted to any type of data stream objects. Then, we need to extract the necessary features that show predictive power. These features should constantly be adapted to the changing nature of online communication in an ongoing feature engineering process. From what we have tested and observed in subsection 5.1.1 regarding the contextual information strategy, contextual features should be extracted here. With the features extracted, we need to defend against specific adversarial attacks using the TSAR framework as outlined in the strategy in section 5.1.2 by standardizing the features that can be influenced by users such as the tweet body. With these features standardized, we need to anonymize in accordance with GDPR as outlined in the literature review 3.6.1. This does not necessarily imply the complete removal of sensitive information; rather, it necessitates transparent treatment of the data with the explicit consent of the users involved. Lastly, we carry out essential data transformations, such as tokenization, which are imperative for the data to be processed by the lightweight detection model. These transformations pave the way for subsequent feature extraction and processing within the detection model.

### 5.2.2 Implementation of Triage Scheme

As stated in subsection 5.1.3, the triage scheme strategy combines the use of a lightweight language model, an instruction-based GPT model, and human moderators on the basis of the uncertainty level of the lightweight language model. We use this to assess whether to employ GPT or human moderators for further review, strategically utilizing a GPT model when suitable, and relying on human moderators when required.

To implement the triage scheme, we utilize a prioritizer to assign priorities to the instances based on some prioritization parameters (such as the cost of each predictive outcome, e.g., classification costs), the predicted class, and the prediction uncertainty of the lightweight language model. Then, a routing module will decide what happens next to the sample based on the system state and the sample's priority. The system state variable is determined by continuously monitoring factors such as the intensity of incoming data and the availability of human moderators, which can be quantified through a queue occupancy metric. This allows for a dynamic distribution of the output samples generated by the lightweight language model. Consequently, this approach facilitates the redirection of samples with high prediction uncertainties to human moderators. If the queue of samples awaiting human moderation surpasses a predefined threshold or if the prediction uncertainty falls within specific thresholds, samples may be directed toward GPT-based labeling. Likewise, samples with prediction uncertainties below a certain threshold can be directly forwarded to the subsequent stage of the hate speech management system, such as an action center. Additionally, by incorporating prioritization parameters and the predicted class as parameters in the routing of the outputted samples from the language model, the routing of the samples become even more customizable. For instance, a system could opt to exclusively route positive samples to human moderation or prioritize samples with high classification costs. By incorporating a routing strategy that considers all these parameters, the system achieves enhanced scalability while still preserving adequate performance and transparency through human moderation.

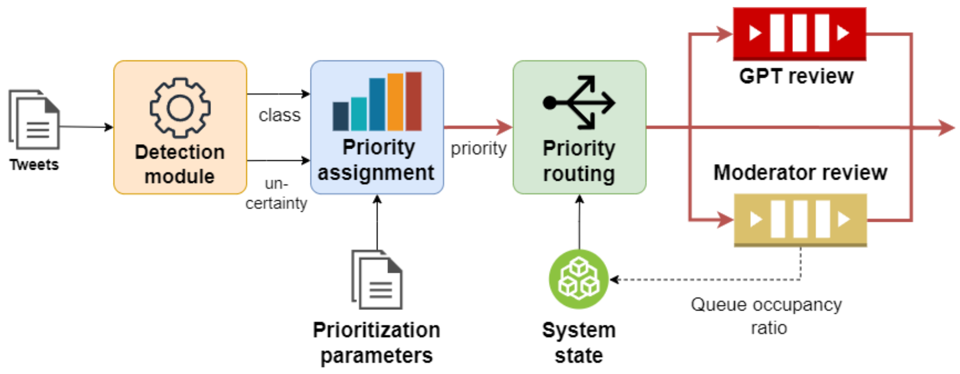


**Figure 5.11:** Implementation of adaptive triage scheme

Figure 5.11 illustrates the potential appearance of the described scheme. The lightweight language model generates the predicted class and its corresponding uncertainty. This information, along with a set of prioritization parameters, is utilized to assign a priority to the sample. The assigned priority then determines whether the sample should undergo GPT review, human moderator review, or no review at all. Additionally, the system state, including factors like the queue size for samples awaiting human moderator review, may influence the routing process.

### 5.2.3 Monitoring-Governed Fine-Tuning Triggers

The sampling rate of the sampling strategy should be determined by considering the available resources and the rate at which posts are generated. To achieve the most cost-effective scheduling of fine-tuning, the trigger for fine-tuning should be both performance-based and based on changes in the distribution. When the performance of the model falls below a designated threshold, it will trigger the fine-tuning pipeline automatically. One way to detect whether the model has become outdated is by monitoring the distribution of the data in production through upstream data analysis. Hence, there is an inter-dependency between the sampling strategy and the monitoring strategy using XAI. This analysis involves examining the characteristics of the input data, such as the frequency of different types of hate speech and language patterns. Then using anomaly detection techniques or statistical methods, such as tracking the mean or variance of specific features, we can detect substantial changes in the data distribution. Both triggers require the establishment of thresholds, which must be determined by considering the resources at hand.

However, it is also important to ensure that the interval between each fine-tuning session is long enough to allow for a large enough batch to culminate each time, hence a minimum interval time should be upheld in addition to the triggers. Subsequently, it is necessary to take into account the available computational resources for fine-tuning as well as the time required to accumulate a sufficiently large batch of data. Similarly, to account for the possibility of one of the triggers for fine-tuning being met after a significant duration, it is crucial to implement measures that prevent the fine-tuning batch from growing excessively. One approach is to set a maximum size for the batch, whereupon reaching this threshold, the oldest samples are replaced with newer ones. By employing this approach, the batch will consistently remain up-to-date, even if the triggers for fine-tuning are not met for an extended period.

### 5.2.4 Real-Time Stream Sampling

We apply the sampling strategy outlined in subsection 5.1.5 through a real-time data sampling module. This module effectively captures and processes every instance flowing through the data stream in the detection model. The sampling process leverages the detection model's uncertainty and the uniqueness calculated from the base layer embeddings of each tweet. To mitigate biases introduced by the sampling heuristic for a proper evaluation of the performance, we use a separate sampling stream that samples the tweets completely at random. The resulting two sample streams could then be labeled using the active labeling scheme with GPT and human experts as outlined in the sequential fine-tuning strategy in subsection 5.1.5. This functionality is illustrated in Figure 5.12.

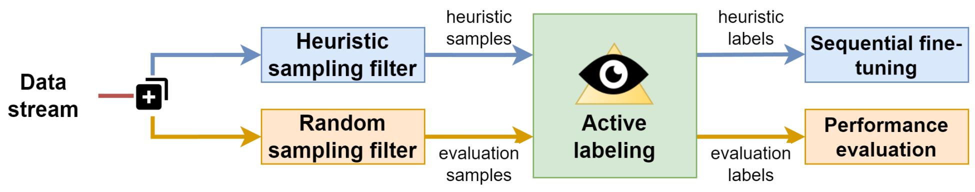


**Figure 5.12:** Real-time sampling streams for sequential fine-tuning and performance evaluation.

### 5.2.5 Validation of Fine-Tuned Models

After each fine-tuning of a model, the model needs to be evaluated in terms of properties such as performance and transparency before replacing the previous model. Hence, the batch of samples obtained after the proposed sampling strategy will have to be split into a training set and a test set. The test set will be used for evaluating the performance of the fine-tuned model. If the performance shows a statistically significant improvement compared to the previous model, deploy the newly fine-tuned version. If it does not meet the criteria, continue using the previous version of the model.

It is important to acknowledge that even if the fine-tuned model demonstrates improved performance metrics compared to the previous model on the current test set, it does not automatically imply that the fine-tuned model would have exhibited better performance metrics when tested on previous test sets. This difference in performance is what separates the two properties, performance, and adaptability. To ensure that the continuously evolving model does not deviate significantly from the initial model (i.e., the model before any fine-tuning), it is advantageous to conduct regular performance testing using the designated baseline dataset mentioned earlier. This approach helps prevent the system from overfitting on a specific type of speech, as well as ensures that the system is still able to detect previously hate speech it has been previously trained on.

The newly fine-tuned model should also be evaluated in terms of transparency. This entails an examination of the batch it has been fine-tuned and the subsequent documentation of relevant information regarding the data of the batch. To ensure an accurate representation of the data that the model has been trained and sequentially fine-tuned on, the designated baseline dataset should consist of an extract from the original training set that is regularly updated with additional samples, in order to maintain a fair representation of the real-world distribution as suggested by Sato et al. [66]. To avoid excessive growth, a maximum size for the benchmark dataset should be defined. When this maximum size is reached, the addition of new samples should replace the oldest additional samples in the dataset. The extract of the original training set should constitute the largest portion and should remain unchanged. In this manner, both the performance testing and the comparison with incoming tweets to retrieve unique tweets will be kept up-to-date.

An illustration of the detection architecture can be seen in Figure 5.13. Here, the data flow is represented by red arrows in which incoming data is inputted into the preprocessing module while the final labels are produced from the triage scheme.

## 5.3 Implications for Industry

The practical implications of this thesis are beneficial for social media companies that seek to implement an effective hate speech management system. By identifying system properties and proposing strategies to optimize them, this thesis offers research and insights into the field of hate speech detection, serving as a guiding resource for companies in the development and deployment of a hate speech management system. Moreover, the strategies and system architecture proposed in this thesis provide a generalizable framework that can assist social media companies in developing and implementing an effective hate speech management system. The presented architecture serves as a solid foundation for the detection architecture of a hate

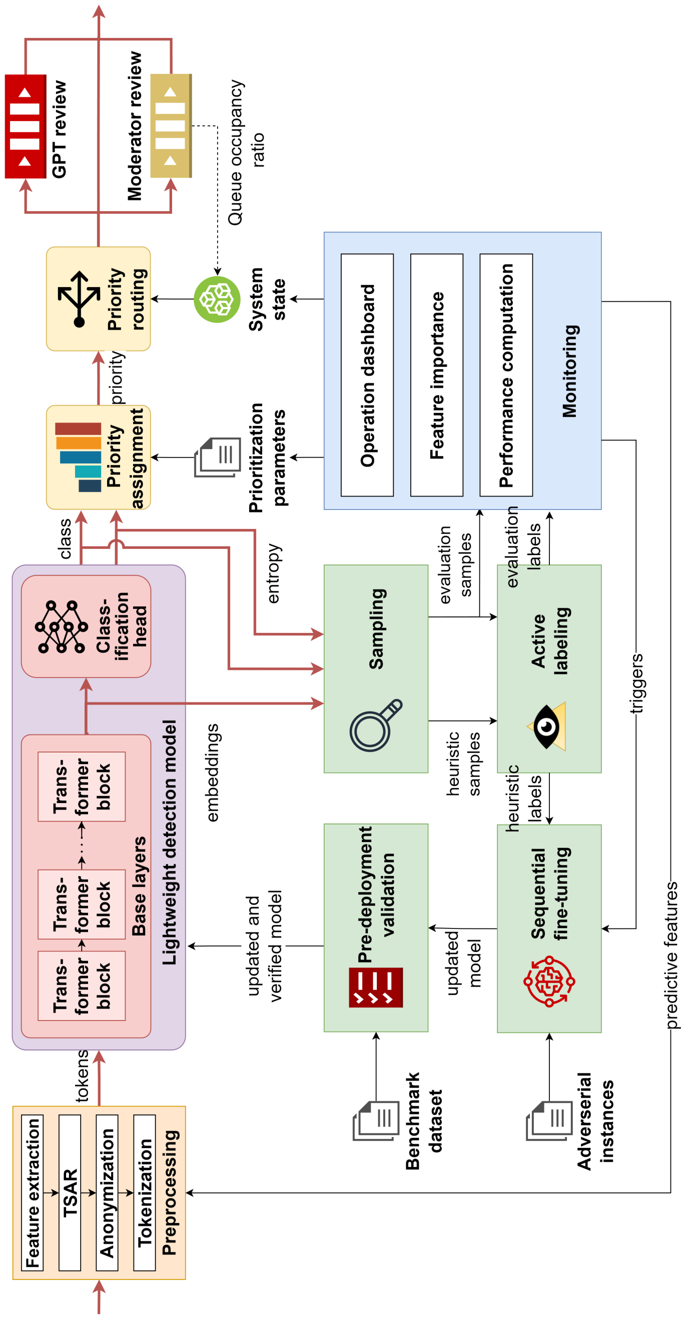


**Figure 5.13:** Proposed detection architecture for hate speech management system, implementing the established strategies. The on-line flow of data is displayed in red.

speech management system, allowing for flexible integration of strategies that can be easily added or removed as needed.

After conducting the literature review, it is evident that the field of hate speech detection is constantly and rapidly evolving. Therefore, it is crucial for a social media platform to keep up with the latest advances in the field. It is worth noting that while the presented strategies are specific, they can be customized and are merely suggestions as to what a social media company should focus on. In this manner, the strategies are intended to serve as exploratory suggestions of how a social media company can approach the problem of hate speech on their platform, rather than strict rules. Hence, they should be regarded as flexible guidelines that can be adjusted and tailored to meet the unique challenges and requirements of each platform.

Before deploying the proposed architecture, thorough testing and simulation of the proposed strategies and the architecture as a whole must be conducted. As our proposed architecture is generalizable, we have not defined any specific thresholds or other values necessary to implement the detection architecture. This involves various tasks, such as developing the prioritizer within the triage scheme to allocate priorities according to a specific routing strategy, establishing an optimal sampling rate for the sampling strategy, and defining thresholds for performance-based and distribution-based triggers, among others. Before implementing the architecture, system requirements must be therefore be specified in order to define necessary thresholds and values. Once these thresholds and values have been established, the detection architecture can be integrated into the overarching hate speech management system. Within this system, further considerations need to be made, including determining the approach for data extraction and transfer of this data to the detection architecture, as well as defining the appropriate actions to be taken based on the labeled output.

## 5.4 Limitations of Results

While the results were promising, they came with a set of limitations inherent to the exploratory nature of our research design. First of all, we did not implement and test all of the proposed strategies in the final system architecture as this was not the scope of our work. Despite the system architecture's basis in previous research and available documentation, it is possible that actual implementation may reveal additional issues or interdependencies that have not been considered. Therefore, it is crucial to note that the strategies presented are only suggestions and serve as an example of how a hate speech management system should consider factors beyond mere hate speech detection. Future research will need to address the actual implementation of the proposed system architecture.

While our quantitative experiments yielded significant results, it is important to question the validity of these findings given the nature of our dataset. First, our dataset is relatively small in size, which may limit the generalizability of our conclusions. Furthermore, the ratio between hateful speech and non-hateful speech in our dataset may not accurately reflect the distribution observed on popular social media platforms. If the prevalence of hate speech on social media platforms is considerably lower than what is represented in our dataset, our strategies could have a lesser impact.

We did not find any significant predictive power in the numerical features when used as the only input features, and not surprisingly, neither when added to the BERT output. This might be a problem regarding the quality of the numerical features that we had access to. As the performance with the language model alone was pretty high, there is not a lot of room for improvement. For numerical features to impact predictive power, we would have had to find features that would provide guidance in the cases where the model is unsure from the textual input alone. A model with user profiling might have had other results as one might find interesting correlations with earlier behavior of the same user.

While we had chosen BERT based on the relevant literature, it is important to note that the results obtained from the various quantitative experiments may vary when using a different model. To ensure consistency, the same experiments should be performed using the different language models presented in the comparison of models from our literature review in subsection 3.2.2. Generally, more quantitative experiments are needed to ensure the validity of our results.

When implementing the described system there is also the decision of actions to take when hate speech is detected, which we have only briefly touched upon. Freedom of speech is a fundamental right, and social media platforms should ensure that their policies and hate speech management systems do not unduly restrict free expression. Therefore, it is essential to strike a balance between protecting users from hate speech and ensuring that the policies and systems do not infringe upon free speech. Our results did not account for this as of now.

# Chapter 6
# Conclusion

This chapter will serve as the concluding section of the thesis, offering an overview of the key findings. It will also align the research questions with the corresponding answers derived from the results, and discuss the significance of our results for society. Lastly, potential avenues for future work will be discussed.

## 6.1 Summary of Thesis

The primary aim of this thesis was to investigate a practical machine learning approach for managing hate speech on social media platforms, specifically focusing on the impact of integrating contextual information and optimizing system properties. This essentially entailed the development of a detection architecture for a hate speech management system. To fulfill this aim, we posed the following research questions:

*What are the strategies that optimize system properties of a hate speech management system, with particular attention to the incorporation of contextual information as a potential strategy?*

*What is a system architecture that integrates strategies optimizing system properties, while ensuring that all properties are still optimized to the same degree?*

By addressing these research questions, this thesis aimed to analyze strategies that optimized system properties of the detection architecture of a hate speech management system. We then proposed a detection architecture that unified the developed strategies into one architecture. The ultimate goal was to contribute to the ongoing efforts to create a safer and more inclusive online environment, where users can express themselves without fear of harassment or discrimination.

### Addressing the 1st Research Question

In addressing the first research question, we first identified relevant system properties and then formulated strategies that optimized them. In doing so we began by identifying multiple relevant strategies from our initial literature review. Subsequently, we thoroughly evaluated these strategies through qualitative and quantitative methods to ascertain their viability for integration into the final system architecture. By employing this methodology, a set of strategies was generated, which were deemed viable through a combination of relevant literature and quantitative experiments.

### Addressing the 2nd Research Question

To address the second research question, we employed the strategies formulated and categorized them according to the machine learning lifecycle definition by Ashmore et al. [4]. Additionally, we consulted relevant literature to ensure the compatibility of these strategies within the same system, while still optimizing the desired system properties. This involved addressing specific edge cases that required careful consideration. Furthermore, we developed an illustration of the system architecture incorporating the formulated strategies.

## 6.2 Main Contributions

This thesis contributes to the applied machine learning field for hate speech management on social media platforms by focusing on the integration of contextual information and the optimization of system properties. The key contributions are:

- **Identification and formulation of strategies:** This research introduces unique strategies derived from a comprehensive literature review. These strategies tackle critical system properties for effective hate speech detection and management, such as the standardization of input, adversarial examples, and persistent monitoring using XAI techniques.
- **A novel approach to data labeling:** The developed strategy for cost-effectively labeling hate speech data using a GPT model and active labeling provides a practical solution that maintains high accuracy while reducing costs.
- **Validation of contextual information importance:** The work validates the critical role of contextual information in hate speech detection, demonstrating improved performance of a BERT model when textual contextual information is incorporated.
- **Innovation in fine-tuning process:** The thesis introduces a novel sampling strategy for fine-tuning a BERT model, which prioritizes unique samples

and instances with high uncertainty. This approach significantly outperforms traditional random sampling methods.

- **Triage scheme proposal:** A dynamic triage scheme is proposed that routes classifications made by the lightweight language model to a GPT model, human moderation, or immediate action. This scheme optimizes resource allocation and improves the system's overall efficiency.
- **Integrated detection architecture:** The proposed strategies are integrated into a system architecture following a machine learning deployment workflow definition. The architecture considers implementation aspects to ensure that the strategies coexist effectively, providing a comprehensive blueprint for social media companies to adapt and employ.

Through these contributions, the thesis provides a concrete and generalizable framework for managing hate speech on social media platforms, advancing the field and offering practical tools for companies and researchers alike. More details about these contributions can be found in the respective chapters.

## 6.3 Significance for Society

Proposing research that aids in developing a hate speech management system for social media platforms holds significance for society. Implementing such a system promotes a safer and more inclusive digital environment. By actively detecting and mitigating instances of hate speech, the system contributes to creating a more welcoming space for diverse perspectives and encourages respectful dialogue.

Furthermore, the management of hate speech helps protect vulnerable communities from harm. Online platforms have become breeding grounds for harassment and targeted abuse, which can have severe psychological, emotional, and even physical consequences. By swiftly identifying and addressing hate speech, the system reduces the risk of harm and supports the well-being of individuals who may otherwise be subject to discrimination, threats, or harassment.

## 6.4 Future Work

Although the proposed system architecture draws upon literature findings and results from quantitative experiments of specific strategies within the architecture, it is important to note that the architecture as a whole has not undergone quantitative experimentation. Therefore, future work should focus on implementing and evaluating the proposed system architecture to validate its functionality as intended. In doing this, a simulation environment is needed in which posts can be generated continuously and fed to the system. The simulation time should be long enough to ensure a realistic depiction of how the incoming distribution changes over time. In this manner, we can verify the adaptable nature of the system. Furthermore, it is crucial to employ an appropriate number of human moderators for both the GPT labeling of the fine-tuning batches and the triage scheme.

Given the limitations of our dataset in terms of size and content, it is recommended that future work replicate the experiments using a larger dataset. This would provide a more robust and comprehensive analysis of the proposed methods. There are several approaches to creating a comprehensive dataset. It is crucial to collect data over an adequate period of time and ensure a reliable and unbiased labeling process. In this manner, exploring the potential of GPT labeling can be a valuable avenue for further investigation. Additionally, the reliability of machine-generated datasets could also be further researched to uncover their full potential and implications.

Another intriguing aspect to explore is the fine-tuning of both XLNet and ELECTRA models for hate speech detection. Although these models were identified as suitable candidates for a hate speech management system, we were unable to obtain a pre-trained model specifically fine-tuned for hate speech. It would be valuable to investigate whether the results differ when utilizing another model for this purpose. The model's pre-training objectives and architectures might vary, leading to differences in how they learn and represent language, potentially influencing their effectiveness in hate speech detection.

Lastly, our strategies have based a wide range of decision-making on aspects such as entropy, as a proxy for uncertainty, and cosine similarity to centroids, as a measure of uniqueness. While these metrics have proven to work for our experiments, future research should consider exploring these further. Instead of manually computing the entropy from the output post-analysis, machine learning architectures that inherently incorporate these should be explored. Similarly, a more comprehensive exploration of uniqueness is warranted to fully understand the benefits and applications.

## 6.5 Closing Remarks

In conclusion, our Master's thesis presents strategies for the detection architecture of a hate speech management system, aimed at addressing the pervasive issue of online hate speech on social media platforms. Through extensive research, analysis, and the formulation of innovative strategies, we have provided a solid foundation for a proactive and effective approach to combating hate speech on social media platforms.

While our work lays the groundwork for a realistic hate speech management system, it is important to recognize that the fight against hate speech is an ongoing and ever-evolving battle. As technology advances and societal dynamics shift, it will be crucial to adapt and improve our methods and strategies continually. Successfully implementing the proposed system architecture requires collaboration and cooperation among various stakeholders, including social media platforms, policymakers, researchers, and users.

As we conclude this thesis, we hope that our research inspires further exploration, innovation, and practical applications in the field of hate speech management. Through continuous efforts, we can work towards a future where social media platforms not only amplify voices but also actively promote respect, empathy, and understanding, fostering a society that embraces diversity and rejects hate in all its forms.

# References

[1] Sweta Agrawal and Amit Awekar. «Deep Learning for Detecting Cyberbullying Across Multiple Social Media Platforms». In: *CoRR* abs/1801.06482 (2018). URL: http://arxiv.org/abs/1801.06482.

[2] Aymé Arango, Jorge Pérez, and Barbara Poblete. «Hate Speech Detection is Not as Easy as You May Think: A Closer Look at Model Validation». In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR'19. Paris, France: Association for Computing Machinery, 2019, pp. 45–54. URL: https://doi.org/10.1145/3331184.3331262.

[3] Sylvain Arlot and Alain Celisse. «A survey of cross-validation procedures for model selection». In: *Statistics Surveys* 4.none (2010), pp. 40–79. URL: https://doi.org/10.1214/09-SS054.

[4] Rob Ashmore, Radu Calinescu, and Colin Paterson. «Assuring the machine learning lifecycle: Desiderata, methods, and challenges». In: *ACM Computing Surveys (CSUR)* 54.5 (2021), pp. 1–39.

[5] Stavros Assimakopoulos, Rebecca Vella Muskat, et al. «Annotating for Hate Speech: The MaNeCo Corpus and Some Input from Critical Discourse Analysis». English. In: *Proceedings of the Twelfth Language Resources and Evaluation Conference.* Marseille, France: European Language Resources Association, May 2020, pp. 5088–5097. URL: https://aclanthology.org/2020.lrec-1.626.

[6] Md Rabiul Awal, Rui Cao, et al. *On Analyzing Annotation Consistency in Online Abusive Behavior Datasets.* 2020.

[7] Pinkesh Badjatiya, Shashank Gupta, et al. «Deep Learning for Hate Speech Detection in Tweets». In: (June 2017).

[8] Alexander Brown. «What is Hate Speech? Part 2: Family Resemblances». In: *Law and Philosophy* 36.5 (Oct. 2017), pp. 561–613. URL: https://doi.org/10.1007/s10982-017-9300-x.

[9] Tom B. Brown, Benjamin Mann, et al. *Language Models are Few-Shot Learners.* 2020. URL: https://arxiv.org/abs/2005.14165.

[10] Tommaso Caselli, Valerio Basile, et al. *HateBERT: Retraining BERT for Abusive Language Detection in English.* 2021.

[11] Kevin Clark, Minh-Thang Luong, et al. *ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators.* 2020.

[12] Yin Cui, Menglin Jia, et al. «Class-Balanced Loss Based on Effective Number of Samples». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[13] Maral Dadvar and Kai Eckert. *Cyberbullying Detection in Social Networks Using Deep Learning Based Models; A Reproducibility Study*. 2018. URL: https://arxiv.org/abs/1812.08046.

[14] Thomas Davidson, Dana Warmsley, et al. «Automated Hate Speech Detection and the Problem of Offensive Language». In: *Proceedings of the International AAAI Conference on Web and Social Media* 11 (May 2017), pp. 512–515. URL: https://ojs.aaai.org/index.php/ICWSM/article/view/14955.

[15] Jacob Devlin, Ming-Wei Chang, et al. «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding». In: 2018. URL: https://arxiv.org/abs/1810.04805.

[16] Bosheng Ding, Chengwei Qin, et al. *Is GPT-3 a Good Data Annotator?* 2022. URL: https://arxiv.org/abs/2212.10450.

[17] Derek Doran, Sarah Schulz, and Tarek R. Besold. *What Does Explainable AI Really Mean? A New Conceptualization of Perspectives*. 2017.

[18] Mehdi Elahi, Dietmar Jannach, et al. «Towards responsible media recommendation». In: *AI and Ethics* 2.1 (Feb. 2022), pp. 103–114. URL: https://doi.org/10.1007/s43681-021-00107-7.

[19] European Commission. *Digital Services Act: Commission welcomes political agreement on rules ensuring a safe and accountable online environment*. URL: https://ec.europa.eu/commission/presscorner/detail/en/ip_22_2545.

[20] European Commission. *Europe fit for the Digital Age: new online rules for platforms*. URL: https://commission.europa.eu/strategy-and-policy/priorities-2019-2024/europe-fit-digital-age/digital-services-act-ensuring-safe-and-accountable-online-environment/europe-fit-digital-age-new-online-rules-platforms_en#tailored-asymmetric-obligations.

[21] European Parliament and of the Council. *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*. [Online; accessed March 27, 2023]. 2016. URL: https://gdpr-info.eu/.

[22] Paula Fortuna, Juan Soler, and Leo Wanner. «Toxic, Hateful, Offensive or Abusive? What Are We Really Classifying? An Empirical Analysis of Hate Speech Datasets». English. In: *Proceedings of the Twelfth Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, May 2020, pp. 6786–6794. URL: https://aclanthology.org/2020.lrec-1.838.

[23] Antigoni-Maria Founta, Constantinos Djouvas, et al. *Large Scale Crowdsourcing and Characterization of Twitter Abusive Behavior*. 2018.

[24] Edita Grolman, Hodaya Binyamini, et al. «HateVersarial: Adversarial Attack Against Hate Speech Detection Algorithms on Twitter». In: *Proceedings of the 30th ACM Conference on User Modeling, Adaptation and Personalization.* UMAP '22. Barcelona, Spain: Association for Computing Machinery, 2022, pp. 143–152. URL: https://doi.org/10.1145/3503252.3531309.

[25] Tommi Gröndahl, Luca Pajola, et al. *All You Need is "Love": Evading Hate-speech Detection.* 2018. URL: https://arxiv.org/abs/1808.09115.

[26] Karl Hansson, Siril Yella, et al. «Machine learning algorithms in heavy process manufacturing». In: *American Journal of Intelligent Systems* 6.1 (2016), pp. 1–13.

[27] Thomas Hartvigsen, Saadia Gabriel, et al. «ToxiGen: A Large-Scale Machine-Generated Dataset for Adversarial and Implicit Hate Speech Detection». In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 3309–3326. URL: https://aclanthology.org/2022.acl-long.234.

[28] *Hateful Conduct.* Website. Twitter. URL: https://help.twitter.com/en/rules-and-policies/hateful-conduct-policy (last visited: June 5, 2023).

[29] Lisa Anne Hendricks, Zeynep Akata, et al. *Generating Visual Explanations.* 2016.

[30] Bernie Hogan and Anabel Quan-Haase. «Persistence and Change in Social Media». In: *Bulletin of Science, Technology & Society* 30.5 (2010), pp. 309–315. URL: https://doi.org/10.1177/0270467610380012.

[31] György Kovács, Pedro Alonso, and Rajkumar Saini. «Challenges of Hate Speech Detection in Social Media». In: *SN Computer Science* 2.2 (Feb. 2021), p. 95. URL: https://doi.org/10.1007/s42979-021-00457-3.

[32] Christoph Käding, Erik Rodner, et al. «Fine-tuning deep neural networks in continuous learning scenarios». In: *Computer Vision–ACCV 2016 Workshops: ACCV 2016 International Workshops, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part III 13.* Springer. 2017, pp. 588–605.

[33] L1ght. *Toxicity during coronavirus: Report by L1ght.* https://l1ght.com/Toxicity_during_coronavirus_Report-L1ght.pdf. Accessed: 05 18, 2023. 2020.

[34] A. van Lamsweerde. «Goal-oriented requirements engineering: a guided tour». In: *Proceedings Fifth IEEE International Symposium on Requirements Engineering.* 2001, pp. 249–262.

[35] Zhenzhong Lan, Mingda Chen, et al. *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations.* 2019. URL: https://arxiv.org/abs/1909.11942.

[36] Zachary Laub. «Hate Speech on Social Media: Global Comparisons». In: (June 2019). Updated on June 7, 2019 3:51 pm (EST). URL: https://www.cfr.org/backgrounder/hate-speech-social-media-global-comparisons (last visited: June 5, 2023).

[37] David Leslie. «Understanding artificial intelligence ethics and safety». In: *arXiv preprint arXiv:1906.05684* (2019).

[38] Tsung-Yi Lin, Priya Goyal, et al. «Focal Loss for Dense Object Detection». In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV).* Oct. 2017.

[39] Yinhan Liu, Myle Ott, et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach.* 2019. URL: https://arxiv.org/abs/1907.11692.

[40] Edward Loper and Steven Bird. *NLTK: The Natural Language Toolkit.* 2002. URL: https://arxiv.org/abs/cs/0205028.

[41] *Lov om likestilling og forbud mot diskriminering (likestillings- og diskrimineringsloven).* 2017. URL: https://lovdata.no/pro/NL/lov/2017-06-16-51.

[42] Jitendra Singh Malik, Guansong Pang, and Anton van den Hengel. *Deep Learning for Hate Speech Detection: A Comparative Study.* 2022.

[43] Ilia Markov and Walter Daelemans. «The Role of Context in Detecting the Target of Hate Speech». In: *Proceedings of the Third Workshop on Threat, Aggression and Cyberbullying (TRAC 2022).* Gyeongju, Republic of Korea: Association for Computational Linguistics, Oct. 2022, pp. 37–42. URL: https://aclanthology.org/2022.trac-1.5.

[44] Stefano Menini, Alessio Palmero Aprosio, and Sara Tonelli. *Abuse is Contextual, What about NLP? The Role of Context in Abusive Language Annotation and Detection.* 2021. URL: https://arxiv.org/abs/2103.14916.

[45] *Meta Transparency Rweport.* https://transparency.fb.com/data/community-standards-enforcement/. Accessed: May 28, 2023.

[46] Selina Meyer, David Elsweiler, et al. «Do We Still Need Human Assessors? Prompt-Based GPT-3 User Simulation in Conversational AI». In: CUI '22. Glasgow, United Kingdom: Association for Computing Machinery, 2022. URL: https://doi.org/10.1145/3543829.3544529.

[47] Melody Moh, Teng-Sheng Moh, and Brian Khieu. «No "Love" Lost: Defending Hate Speech Detection Models Against Adversaries». In: *2020 14th International Conference on Ubiquitous Information Management and Communication (IMCOM).* 2020, pp. 1–6.

[48] Swapnanil Mukherjee and Sujit Das. «Application of Transformer-based Language Models to Detect Hate Speech in Social Media». In: *Journal of Computational and Cognitive Engineering* (Dec. 2021). URL: https://ojs.bonviewpress.com/index.php/JCCE/article/view/105.

[49] Raymond T Mutanga, Nalindren Naicker, and Oludayo O Olugbara. «Hate Speech Detection in Twitter using Transformer Methods». In: *International Journal of Advanced Computer Science and Applications* 11.9 (2020). URL: http://dx.doi.org/10.14569/IJACSA.2020.0110972.

[50] Brigitte L. Nacos, Robert Y. Shapiro, and Yaeli Bloch-Elkon. «Donald Trump: Aggressive Rhetoric and Political Violence». In: *Perspectives on Terrorism* 14.5 (2020), pp. 2–25. URL: https://www.jstor.org/stable/26940036 (last visited: May 15, 2023).

[51] Bashar Nuseibeh and Steve Easterbrook. «Requirements Engineering: A Roadmap». In: *Proceedings of the Conference on The Future of Software Engineering.* ICSE '00. Limerick, Ireland: Association for Computing Machinery, 2000, pp. 35–46. URL: https://doi.org/10.1145/336512.336523.

[52] Rajvardhan Oak. «Poster: Adversarial Examples for Hate Speech Classifiers». In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. CCS '19. London, United Kingdom: Association for Computing Machinery, 2019, pp. 2621–2623. URL: https://doi.org/10.1145/3319535.3363271.

[53] Marwan Omar and David Mohaisen. «Making Adversarially-Trained Language Models Forget with Model Retraining: A Case Study on Hate Speech Detection». In: *Companion Proceedings of the Web Conference 2022*. WWW '22. Virtual Event, Lyon, France: Association for Computing Machinery, 2022, pp. 887–893. URL: https://doi.org/10.1145/3487553.3524667.

[54] Long Ouyang, Jeff Wu, et al. *Training language models to follow instructions with human feedback*. 2022.

[55] Andrei Paleyes, Raoul-Gabriel Urma, and Neil D. Lawrence. «Challenges in Deploying Machine Learning: A Survey of Case Studies». In: *ACM Comput. Surv.* 55.6 (Dec. 2022). URL: https://doi.org/10.1145/3533378.

[56] Gyung-Jin Park. «Design of experiments». In: *Analytic methods for design practice* (2007), pp. 309–391.

[57] Sara Parker and Derek Ruths. «Is hate speech detection the solution the world wants?» In: *Proceedings of the National Academy of Sciences* 120.10 (2023), e2209384120. URL: https://www.pnas.org/doi/abs/10.1073/pnas.2209384120.

[58] Amandalynne Paullada, Inioluwa Deborah Raji, et al. «Data and its (dis)contents: A survey of dataset development and use in machine learning research». In: *Patterns* 2.11 (2021), p. 100336. URL: https://www.sciencedirect.com/science/article/pii/S2666389921001847.

[59] John Pavlopoulos, Jeffrey Sorensen, et al. «Toxicity Detection: Does Context Really Matter?» In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 4296–4305. URL: https://aclanthology.org/2020.acl-main.396.

[60] Fabian Pedregosa, Gaël Varoquaux, et al. «Scikit-learn: Machine Learning in Python». In: (2012). URL: https://arxiv.org/abs/1201.0490.

[61] Fabio Poletto, Valerio Basile, et al. «Resources and benchmark corpora for hate speech detection: a systematic review». In: *Language Resources and Evaluation* 55.2 (June 2021), pp. 477–523. URL: https://doi.org/10.1007/s10579-020-09502-8.

[62] Md Mustafizur Rahman, Dinesh Balakrishnan, et al. *An Information Retrieval Approach to Building Datasets for Hate Speech Detection*. 2021. URL: https://arxiv.org/abs/2106.09775.

[63] Ashley Reichelmann, James Hawdon, et al. «Hate Knows No Boundaries: Online Hate in Six Nations». In: *Deviant Behavior* 42.9 (2021), pp. 1100–1111. URL: https://doi.org/10.1080/01639625.2020.1722337.

[64] Victor Sanh, Lysandre Debut, et al. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2019. URL: https://arxiv.org/abs/1910.01108.

[65] Maarten Sap, Swabha Swayamdipta, et al. *Annotators with Attitudes: How Annotator Beliefs And Identities Bias Toxic Language Detection.* 2021. URL: https://arxiv.org/abs/2111.07997.

[66] Danilo Sato, Arif Wider, and Christoph Windheuser. «Continuous delivery for machine learning». In: *Martin Fowler* 9 (2019).

[67] David Sculley, Gary Holt, et al. «Hidden technical debt in machine learning systems». In: *Advances in neural information processing systems* 28 (2015).

[68] Qadeem Khan Shams-ul-Arif and SAK Gahyyur. «Requirements engineering processes, tools/technologies, & methodologies». In: *International Journal of reviews in computing* 2.6 (2009), pp. 41–56.

[69] Or Sharir, Barak Peleg, and Yoav Shoham. «The cost of training nlp models: A concise overview». In: *arXiv preprint arXiv:2004.08900* (2020).

[70] Mayuri S Shelke, Prashant R Deshmukh, and Vijaya K Shandilya. «A review on imbalanced data handling using undersampling and oversampling technique». In: *Int. J. Recent Trends Eng. Res* 3.4 (2017), pp. 444–449.

[71] Emma Strubell, Ananya Ganesh, and Andrew McCallum. «Energy and policy considerations for modern deep learning research». In: *Proceedings of the AAAI Conference on Artificial Intelligence.* Vol. 34. 09. 2020, pp. 13693–13696.

[72] Mirka Saarela and Susanne Jauhiainen. «Comparison of feature importance measures as explanations for classification models». In: *SN Applied Sciences* 3.2 (Feb. 2021), p. 272. URL: https://doi.org/10.1007/s42452-021-04148-9.

[73] Thomas L Saaty. *What is the analytic hierarchy process?* Springer, 1988.

[74] Zeerak Talat, Aurélie Névéol, et al. «You reap what you sow: On the Challenges of Bias Evaluation Under Multilingual Settings». In: *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models.* virtual+Dublin: Association for Computational Linguistics, May 2022, pp. 26–41. URL: https://aclanthology.org/2022.bigscience-1.3.

[75] The ADL website. URL: https://www.adl.org/resources/report/online-hate-and-harassment-american-experience-2022 (last visited: May 9, 2023).

[76] The New York Times website. URL: https://www.nytimes.com/2020/06/29/business/dealbook/facebook-boycott-ads.html (last visited: May 9, 2023).

[77] The OpenAI website. URL: https://platform.openai.com/tokenizer (last visited: Mar. 27, 2023).

[78] The OpenAI website. URL: https://openai.com/pricing/ (last visited: Mar. 13, 2023).

[79] *The Optuna Python library.* https://optuna.org/. Accessed: May 29, 2023.

[80] *The PyTorch Python library.* https://pytorch.org/. Accessed: May 29, 2023.

[81] *The Scikit-learn Python library.* https://scikit-learn.org/. Accessed: May 29, 2023.

[82] *The Transformers Python library.* https://huggingface.co/docs/transformers/index/. Accessed: May 29, 2023.

[83] The Twitter website. URL: https://help.twitter.com/en/rules-and-policies/enforcement-options (last visited: Feb. 8, 2023).

[84] The Twitter website. URL: https://help.twitter.com/en/rules-and-policies/enforcement-philosophy (last visited: Mar. 10, 2023).

[85] The Twitter website. URL: https://blog.twitter.com/engineering/en_us/a/2013/new-tweets-per-second-record-and-how (last visited: Mar. 29, 2022).

[86] The UN website. URL: https://www.un.org/en/genocideprevention/hate-speech-strategy.shtml (last visited: Mar. 21, 2023).

[87] *The XGBoost Python library.* https://xgboost.ai/. Accessed: May 29, 2023.

[88] Robin Thompson. «Radicalization and the Use of Social Media». In: *Journal of Strategic Security* 4.4 (2011), pp. 167–190. URL: http://www.jstor.org/stable/26463917 (last visited: June 5, 2023).

[89] Shoujie Tong, Qingxiu Dong, et al. *Robust Fine-tuning via Perturbation and Interpolation from In-batch Instances.* 2022.

[90] *Twitter GDPR.* https://gdpr.twitter.com/. Accessed: May 18, 2023.

[91] Stefanie Ullmann and Marcus Tomalin. «Quarantining online hate speech: technical and ethical perspectives». In: *Ethics and Information Technology* 22.1 (Mar. 2020), pp. 69–80. URL: https://doi.org/10.1007/s10676-019-09516-z.

[92] Elise Fehn Unsvåg. *Investigating the Effects of User Features in Hate Speech Detection on Twitter.* URL: http://hdl.handle.net/11250/2560363.

[93] Ashish Vaswani, Noam Shazeer, et al. *Attention Is All You Need.* 2017. URL: https://arxiv.org/abs/1706.03762.

[94] Alex Wang, Amanpreet Singh, et al. *GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding.* 2019.

[95] Shuohang Wang, Yang Liu, et al. «Want To Reduce Labeling Cost? GPT-3 Can Help». In: *Findings of the Association for Computational Linguistics: EMNLP 2021.* Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 4195–4205. URL: https://aclanthology.org/2021.findings-emnlp.354.

[96] Zeerak Waseem, Thomas Davidson, et al. «Understanding Abuse: A Typology of Abusive Language Detection Subtasks». In: *Proceedings of the First Workshop on Abusive Language Online.* Vancouver, BC, Canada: Association for Computational Linguistics, Aug. 2017, pp. 78–84. URL: https://aclanthology.org/W17-3012.

[97] Zeerak Waseem and Dirk Hovy. «Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter». In: *Proceedings of the NAACL Student Research Workshop.* San Diego, California: Association for Computational Linguistics, June 2016, pp. 88–93. URL: https://aclanthology.org/N16-2013.

[98] Vitor Werner de Vargas, Jorge Arthur Schneider Aranda, et al. «Imbalanced data preprocessing techniques for machine learning: a systematic mapping study». In: *Knowledge and Information Systems* 65.1 (Jan. 2023), pp. 31–57. URL: https://doi.org/10.1007/s10115-022-01772-8.

[99] Michael Wiegand, Josef Ruppenhofer, and Thomas Kleinbauer. «Detection of Abusive Language: the Problem of Biased Datasets». In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 602–608. URL: https://aclanthology.org/N19-1060.

[100] Matthew L Williams, Pete Burnap, et al. «Corrigendum to: Hate in the Machine: Anti-Black and Anti-Muslim Social Media Posts as Predictors of Offline Racially and Religiously Aggravated Crime». In: *The British Journal of Criminology* 60.1 (Sept. 2019), pp. 242–242. URL: https://doi.org/10.1093/bjc/azz064.

[101] Tomer Wullach, Amir Adler, and Einat Minkov. *Fight Fire with Fire: Fine-tuning Hate Detectors using Large Samples of Generated Hate Speech*. 2021. URL: https://arxiv.org/abs/2109.00591.

[102] Zhilin Yang, Zihang Dai, et al. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. 2020.

[103] Ziqi Zhang and Lei Luo. *Hate Speech Detection: A Solved Problem? The Challenging Case of Long Tail on Twitter*. URL: https://arxiv.org/abs/1803.03662.

[104] Didar Zowghi and Chad Coulin. «Requirements Elicitation: A Survey of Technique, Approaches and Tools». In: Jan. 2005, pp. 19–46.

[105] Andreas Aarrestad and Santhosh Shanmugam. *Analyzing the Performance Gain of Hate Speech Detection by Providing Machine Learning Models with Contextual Resources*. Project report in TTM4502. Department of Information Security, Communication Technology, NTNU – Norwegian University of Science, and Technology, Dec. 2022.

# Appendix A
# Supplementary Figures and Tables

| Feature | Data Type | Non-Null Count | Unique Values |
|---|---|---|---|
| tweet id | String | 8103 | 8103 |
| conversation id | String | 8103 | 7954 |
| child author id | String | 8103 | 7955 |
| parent author id | String | 8103 | 7238 |
| child tweet place id | String | 324 | 255 |
| parent tweet place id | String | 293 | 228 |
| tweet raw | String | 8103 | 8103 |
| tweet type | int64 | 8103 | 2 |
| tweet retweet count | int64 | 8103 | 48 |
| tweet reply count | int64 | 8103 | 18 |
| tweet like count | int64 | 8103 | 72 |
| tweet quote count | int64 | 8103 | 12 |
| tweet mentions | object | 5822 | 5519 |
| tweet urls | object | 2257 | 2252 |
| tweet hashtags | object | 1022 | 1015 |
| tweet created at | Timestamp | 8103 | 8066 |
| author name | String | 8103 | 7795 |
| author username | String | 8103 | 7955 |
| author description | String | 8103 | 7054 |
| author profile image url | String | 8103 | 7786 |
| author followers count | int64 | 8103 | 3742 |
| author following count | int64 | 8103 | 3107 |
| author tweet count | int64 | 8103 | 7428 |
| author listed count | int64 | 8103 | 595 |
| author created at | Timestamp | 8103 | 7954 |
| parent tweet raw | String | 8103 | 7961 |
| parent tweet language | String | 8103 | 40 |
| parent tweet type | int64 | 8103 | 3 |
| parent tweet retweet count | int64 | 8103 | 1046 |
| parent tweet reply count | int64 | 8103 | 650 |
| parent tweet like count | int64 | 8103 | 1326 |
| parent tweet quote count | int64 | 8103 | 528 |
| parent tweet mentions | object | 3423 | 3831 |
| parent tweet urls | object | 3188 | 3437 |
| parent tweet hashtags | object | 1511 | 1456 |
| parent tweet created at | Timestamp | 8103 | 7924 |
| parent author name | String | 8103 | 7119 |
| parent author username | String | 8103 | 7238 |
| parent author description | String | 8103 | 6731 |
| parent author profile image url | String | 8103 | 7186 |
| parent author followers count | int64 | 8103 | 5936 |
| parent author following count | int64 | 8103 | 3270 |
| parent author tweet count | int64 | 8103 | 7079 |
| parent author listed count | int64 | 8103 | 2042 |
| parent author created at | Timestamp | 8103 | 7238 |
| label | int64 | 8103 | 3 |
| **Total** | | | **8103** |

**Table A.1:** Raw feature summary of developed dataset

**Instruction Level 1: Definition-Based**

Assign a class to a given child tweet, using the provided classification examples and the contextual information that includes the user's description, the parent tweet and the parent tweet user's description. The class is determined using the following definition of speech:
*"Language used to express hatred towards a targeted individual or group, or is intended to be derogatory, to humiliate, or to insult the members of the group, on the basis of attributes such as race, religion, ethnicity, sexual orientation, disability, nationality, descent, colour or gender."*
Assign the speech to Class 1 if it fits with this definition, and to Class 0 if it does not.

**Instruction Level 2: Condition-Based**

Assign a class to a given child tweet, using the provided classification examples and the contextual information that includes the user's description, the parent tweet and the parent tweet user's description. The class is determined using the following evaluation process:

1. Does the language express negative attitudes towards a targeted individual or group?
   - If "no", assign the speech to Class 0 and stop evaluating.
   - If "yes", proceed to the next question.
2. Is the negative attitude intended to be either derogatory, humiliating, or insulting?
   - If "no", assign the speech to Class 0 and stop evaluating.
   - If "yes", proceed to the next question.
3. Is the target identified based on attributes such as race, religion, ethnicity, sexual orientation, disability, nationality, or gender?
   - If "no", assign the speech to Class 0 and stop evaluating.
   - If "yes", proceed to the next question.
4. Are the attributes used as a basis for the derogatory, humiliating, or insulting negativity expressed towards the target?
   - If "no", assign the speech to Class 0 and stop evaluating.
   - If "yes", assign the speech to Class 1.

**Instruction Level 3: Condition-Based with High Specificity**

Assign a class to a given child tweet, using the provided classification examples and the contextual information that includes the user's description, the parent tweet and the parent tweet user's description. The class is determined using the following evaluation process:

1. Does the language express negative attitudes towards a targeted individual or group? Consider context and tone; avoid hasty judgments based on isolated words.
   - If "no", assign the speech to Class 0 and stop evaluating.
   - If "yes", proceed to the next question.
2. Is the negative attitude intended to be derogatory, humiliating, or insulting? Assess the intention by examining the overall message and tone.
   - If "no", assign the speech to Class 0 and stop evaluating.
   - If "yes", proceed to the next question.
3. Is the target identified based on attributes such as race, religion, ethnicity, sexual orientation, disability, nationality, or gender? Be cautious of implicit or coded language targeting these attributes.
   - If "no", assign the speech to Class 0 and stop evaluating.
   - If "yes", proceed to the next question.
4. Are the attributes used as a basis for the derogatory, humiliating, or insulting negativity expressed towards the target? Evaluate whether the negativity directly targets the attribute, rather than a general or unrelated critique.
   - If "no", assign the speech to Class 0 and stop evaluating.
   - If "yes", assign the speech to Class 1.

**Table A.2:** Instruction levels

| **Hyperparameter** | **Range** | **NaiveBERT** | **ContextBERT** |
|---|---|---|---|
| Learning rate | $5 \cdot 10^{-6}$ to $5 \cdot 10^{-5}$ | $3.74 \cdot 10^{-5}$ | $3.70 \cdot 10^{-5}$ |
| Batch size | 8, 16, 32 | 32 | 32 |
| Number of epochs | 2 to 5 | 2 | 2 |
| Warmup steps | 0 to 500 | 220 | 263 |
| Weight decay | 0.0 to 0.01 | 0.00082 | 0.0070 |
| Dropout rate | 0.0 to 0.5 | 0.19 | 0.29 |
| Focal parameter $\gamma$ | 1, 2, 3, 4 | 2 | 2 |

**Table A.3:** Hyperparameters for NaiveBERT and ContextBERT

| **Hyperparameter** | **Range** | **ContextBoost** |
|---|---|---|
| Max depth | 3, 5, 7, 10 | 3 |
| Learning rate | 0.001, 0.01, 0.1, 0.2 | 0.01 |
| Subsample fraction | 0.5, 0.7, 1.0 | 0.5 |
| Feature subsampling fraction | 0.4, 0.6, 0.8, 1.0 | 0.8 |
| Number of estimators | 100, 200, 500 | 200 |
| L2 regularization parameter $\lambda$ | 0.01, 0.1, 1 | 1 |
| Focal parameter $\gamma$ | 1, 2, 3, 4 | 2 |

**Table A.4:** Hyperparameters for ContextBoost

NTNU

Norwegian University of
Science and Technology