Innovative Approaches for Real-Time Toxicity Detection in Social Media Using Deep Reinforcement Learning

Arezo Bodaghi

A Thesis

In the Department

of

Concordia Institute for Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of

Doctor of Philosophy (Information and Systems Engineering) at

Concordia University

Montréal, Québec, Canada

September 2024

© Arezo Bodaghi, 2024

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: **Arezo Bodaghi**

Entitled: **Innovative Approaches for Real-Time Toxicity Detection in Social Media Using Deep Reinforcement Learning**

and submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY (Information and Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. Emre Erkmen	Chair
Dr. Omair Shafiq	External Examiner
Dr. Hovhannes Harutyunyan	Arm's Length Examiner
Dr. Farnoosh Naderkhani	Examiner
Dr. Jeremy Clark	Examiner
Dr. Ketra A. Schmitt	Thesis Supervisor
Dr. Benjamin C. M. Fung	Thesis Supervisor

Approved by _____
Dr. Farnoosh Naderkhani, Graduate Program Director

10/10/2024 _____
Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering Computer Science

ABSTRACT

Innovative Approaches for Real-Time Toxicity Detection in Social Media Using Deep Reinforcement Learning

Arezo Bodaghi, Ph.D.

Concordia University, 2024

Toxic comments on social media discourage user engagement and have serious consequences for mental health and social well-being. Such negativity heightens feelings of anxiety, depression, and social isolation among users, ultimately diminishing their experience on these platforms. For businesses, these toxic interactions are detrimental as they lead to reduced user engagement, subsequently affecting advertising revenue and market share. Creating a safe and inclusive online environment is essential for business success and social responsibility. This requires real-time detection of toxic behavior through automated methods. However, many existing toxicity detectors focus mainly on accuracy, often neglecting important factors including throughput, computational costs, and the impact of false positives and negatives on user engagement. Additionally, these methods are evaluated in controlled experimental settings (offline tests), which do not reflect the complexities of large-scale social media environments. This limitation hinders their practical applicability in real-world scenarios. This thesis addresses these limitations by introducing a Profit-driven Simulation (PDS) framework for evaluating the real-time performance of deep learning classifiers in complex social media settings. The PDS framework integrates performance, computational efficiency, and user engagement, revealing that optimal classifier selection depends on the toxicity level of the environment. High-throughput classifiers are most effective in low- and

high-toxicity scenarios, while classifiers offering moderate accuracy and throughput excel in medium-toxicity contexts. Additionally, the thesis tackles the challenge of imbalanced datasets by introducing a novel method for augmenting toxic text data. By applying Reinforcement Learning with Human Feedback (RLHF) and Proximal Policy Optimization (PPO), this method fine-tunes Large Language Models (LLMs) to generate diverse, semantically consistent toxic data. This approach enhances classifier robustness, particularly in detecting minority class instances. The thesis also proposes a Proximal Policy Optimization-based Cascaded Inference System (PPO-CIS), which dynamically assigns classifiers based on performance and computational costs. This system improves efficiency by using high-throughput classifiers for initial filtering and more accurate classifiers for final decisions, reducing the workload on human moderators. Extensive evaluations on datasets such as Kaggle-Jigsaw and ToxiGen demonstrate significant improvements in processing time, detection accuracy, and overall user satisfaction, contributing to the development of scalable, cost-effective toxicity detection systems for social media platforms.

Preface

This thesis has been prepared in a manuscript-based format under the co-direction of Dr. Ketra A. Schmitt from the Centre for Engineering in Society, Concordia University, and Dr. Benjamin C. M. Fung, Canada Research Chair in Data Mining for Cybersecurity and Professor at the School of Information Studies, McGill University. This research was generously supported by Concordia University through Graduate Fellowships, International Tuition Award of Excellence, the Mitacs Business Strategy Internship (BSI), and the Mitacs Accelerate Fellowship in collaboration with Scrawlr. Additionally, McGill University provided support via a Research Assistantship in the School of Information Studies using NSERC Discovery Grants and the Canada Research Chairs Program.

All the articles presented in this thesis were co-authored and reviewed prior to submission for publication by Dr. Ketra A. Schmitt and Dr. Benjamin C. M. Fung. The author of this thesis acted as the principal researcher and was responsible for the development of models, programming of solution algorithms, analysis and validation of results, and writing the initial drafts of the articles.

The first article [1] entitled "Technological Solutions to Online Toxicity: Potential and Pitfalls ", co-authored by Dr. Benjamin C. M. Fung, and Dr. Ketra Schmitt was published in *IEEE Technology and Society Magazine (**TSM**)*.

The second article [2] entitled "A Profit-driven Simulation (PDS) Framework for Comparison of Deep Learning Models for Real-time Toxicity Detection in Social

Media", co-authored by Dr. Benjamin C. M. Fung, Dr. Jonathan Shahen, and Dr. Ketra Schmitt is submitted to *ACM Transactions on Knowledge Discovery from Data (**TKDD**)*.

The third article [3] entitled "AugmenToxic: Leveraging Reinforcement Learning to Optimize LLM Instruction Fine-Tuning for Data Augmentation to Enhance Toxicity Detection", co-authored by Dr. Benjamin C. M. Fung, and Dr. Ketra Schmitt is in press by *ACM Transactions on the Web (**TWEB**), Special Issue on Advances in Social Media Technologies and Analysis*.

The forth article [4] entitled "Real-Time Adaptive Toxicity Detection with Cascaded Classifiers Optimized by Proximal Policy Optimization", co-authored by Dr. Benjamin C. M. Fung, and Dr. Ketra Schmitt is submitted to *IEEE Transactions on Computational Social Systems (**TCSS**), Special Issue on Trends in Social Multimedia Computing: Models, Methodologies, and Applications* .

This thesis is dedicated to

My beloved parents, Alireza and Akram,

for their endless love and unlimited support.

My loving husband, Mohammad,

for his boundless support, inspiration, and invaluable assistance.

My dearest family,

Abbas, Davoud, Hossein, Hasan, Saeedeh, and Kaynoosh,

for their unwavering support and encouragement.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor, Dr. Ketra A. Schmitt. Her vast knowledge and invaluable advice have been pivotal throughout my PhD journey. Her unwavering support, even during the most challenging times, has been akin to that of a sister. I remain forever indebted to her boundless enthusiasm, encouragement, and patience.

I extend my heartfelt thanks to my co-advisor, Prof. Benjamin C. M. Fung, for his invaluable guidance, tireless efforts, and incisive comments throughout my thesis. It has been an honor working with him. His vast knowledge, inspiration, and exemplary humility have been truly motivating.

I am profoundly grateful to Dr. Jonathan Shahen, Lead Developer at Scrawlr, for imparting numerous technical skills and providing unwavering support. My gratitude also extends to Scrawlr, a company offering a platform for unrestricted, global interaction with internet content and users. As a Mitacs intern at Scrawlr, I had the opportunity to work on a project that formed the basis of my thesis. Scrawlr's ongoing support was crucial in enabling the successful implementation of my PhD research.

To the members of the Systems Risk Laboratory and Data Mining and Security Lab (DMaS), I extend my heartfelt gratitude. I would also like to extend my heartfelt thanks to my friends who have not only been wonderful colleagues but have also become lifelong friends throughout this journey.

I express my deep appreciation to the Concordia Institute for Information Systems Engineering (CIISE) for providing an enriching and friendly work environment.

Last but not least, my special thanks go to my husband, Mohammad, my parents, and my brothers. Without Mohammad's unwavering support and my family's constant encouragement, completing this journey would not have been possible. Along this path, we have shared moments of joyous laughter and heartfelt tears, and I am immensely grateful for having them all by my side through it all.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement and Motivation

The rapid development of communication technology and the internet has transformed social media platforms such as Facebook, X (formerly known as Twitter), Sina Weibo, Instagram, and Reddit into dynamic virtual communities, fostering accessibility and user interaction. However, this evolution has also brought significant challenges, particularly regarding antisocial behavior, including toxic comments, hate speech, cyberbullying, and other forms of online harassment [5]–[10]. These toxic behaviors can deter participants from engaging in meaningful conversations, leading to self-censorship and disengagement from social media platforms [11]. Major online platforms have acknowledged the need to improve their services to protect users, especially women, from harassment and abuse [12].

Detecting toxic behavior in a timely and automated manner is crucial for ensuring a safe and inclusive online environment that fosters user satisfaction and engagement [13]. The overwhelming flow of user-generated content (UGC) presents significant challenges in real-time dissemination, computing, processing, and analysis. For instance, platforms such as Twitter (now X) handle millions of daily active users

and thousands of tweets per second, making content moderation a monumental task [14]–[16]. Various Deep Learning (DL) and Machine Learning (ML) techniques have been developed to counteract the spread of harmful content, yet implementing these methods in real-time for every comment remains resource-intensive and challenging [17]–[19].

Social media content moderation currently relies on a collaborative approach between human moderators and automated tools. Platforms utilize ML and DL algorithms to analyze UGC, generate predictive scores, and classify content into different categories [20]–[22]. Despite advancements, the dynamic nature of language, regional variations, and the contextual subtleties of human communication pose significant challenges, necessitating ongoing research and development to enhance the accuracy and effectiveness of these technologies [23]–[25].

Moreover, while DL and transformer-based models have demonstrated effectiveness in identifying toxic content, they can also exacerbate data bias issues. These models rely heavily on large datasets that may not be inclusive of all user groups, potentially marginalizing certain voices [25]. Another challenge is the issue of imbalanced datasets. Training datasets are often skewed towards nontoxic content, which can result in models that are overly optimistic in their detection rates [26]. This problem is further compounded by the use of self-reported data or crowdsourcing for annotation, which may not accurately capture the diversity of toxic language used online [27].

Additionally, the inherent complexities of language present further challenges. Sarcasm, irony, and other forms of figurative language are difficult for algorithms to detect [28]. Users may also intentionally obfuscate their language or use coded expressions to evade detection, complicating the task of identifying toxic content [29], [30]. ML algorithms typically excel at identifying previously reviewed toxic content but struggle with new and emerging forms of harmful content [31].

Another significant limitation is the potential unsuitability of these models for real-time applications. The evaluation of ML systems often relies on metrics such as accuracy or area under the receiver operating characteristic curve (AUROC), which are useful for classification tasks but do not consider the model's efficiency in real-time scenarios [19]. Throughput and latency are critical factors, especially for applications where timely detection and response to toxic content are essential. A study by Hosseinmardi et al. [32] found that while DL algorithms achieved high accuracy in detecting hate speech, they also exhibited high latency and low throughput, making them unsuitable for real-time applications. This underscores the importance of evaluating not only the accuracy but also the efficiency and scalability of ML algorithms for detecting toxic language in real-world settings.

Addressing these challenges requires innovative approaches that not only improve detection accuracy but also enhance the efficiency and scalability of content moderation systems.

1.2 Contributions

In this thesis, we focused on three main challenges in toxicity detection on social media platforms and proposed innovative solutions to address these issues effectively.

Firstly, the overwhelming volume and diversity of user-generated content (UGC) present a significant challenge for real-time detection and moderation of toxic behavior. To address this, we developed a novel **Profit-Driven Simulation (PDS)** Framework. This framework evaluates the effectiveness and efficiency of various toxicity detection algorithms by considering different levels of toxicity in social media environments. It identifies the most profitable detector that accurately detects toxic content at the lowest cost and within a reasonable time frame. The PDS framework also integrates user satisfaction as a critical factor in selecting the optimal detector, emphasizing the

importance of minimizing false positives and false negatives to prevent user churn and enhance engagement.

Secondly, the issue of imbalanced datasets significantly hampers the performance of toxicity detection models. To mitigate this, we introduced **AugmenToxic**, a technique for sentence-level text data augmentation (TDA) specifically targeting toxic language. This approach leverages reinforcement learning from human feedback (RLHF) using the Proximal Policy Optimization (PPO) algorithm, applied to an instruction fine-tuned large language model (LLM). By optimizing the model to paraphrase text while maintaining semantic coherence and maximizing toxicity, AugmenToxic significantly expands the available toxic text data. This method enhances classifier accuracy and performance by generating a more balanced and diverse dataset, surpassing traditional data augmentation techniques, and ensuring better representation of toxic content in training data.

Lastly, ensuring both accuracy and efficiency in real-time toxicity detection remains a crucial challenge. To overcome this, we proposed the **Proximal Policy Optimization-based Cascaded Inference System (PPO-CIS)**, an adaptive multi-stage inference system for toxicity detection that employs a dynamic cascade of classifiers. This system uses high-throughput classifiers for initial filtering and highly accurate classifiers for final detection, optimizing both speed and accuracy. It dynamically selects the most appropriate classifiers from the first and second stages to ensure optimal performance. By integrating deep reinforcement learning (DRL) with Proximal Policy Optimization (PPO), PPO-CIS identifies the most cost-effective set of cascaded classifiers, balancing classification accuracy, processing time, and computational cost. This scalable and efficient content moderation system reduces the burden on human moderators and enhances user satisfaction by promptly and accurately identifying toxic content in real-time.

These contributions address critical challenges in toxicity detection on social media platforms, providing robust solutions that improve the efficiency, accuracy, and scalability of content moderation systems.

This thesis aims to address the significant challenges in toxicity detection on social media platforms by proposing and developing innovative solutions. The primary objectives are as follows:

1.2.1 Profit-Driven Simulation (PDS) Framework

- Propose a novel PDS framework to evaluate the effectiveness and efficiency of toxicity detection algorithms across different toxicity levels in social media environments.
- Identify the most profitable detector that can accurately detect toxic content at the lowest cost, within a reasonable time, and with the highest level of user engagement under various circumstances.
- Design a profit model tied to the simulation approach to identify the most lucrative toxicity classifier for different environments on social media platforms.
- Consider user satisfaction as a critical factor in selecting the optimal detector, highlighting the significance of false positive and false negative predictions, which can lead to user churn.

1.2.2 Text Data Augmentation (TDA) for Toxic Language

- Develop a novel method for enhancing toxic text data through instruction fine-tuning on the pretrained FLAN-T5 model, crafted for paraphrasing with semantic equivalence using the PAWS dataset.

- Apply Proximal Policy Optimization (PPO) to further fine-tune the instruction-tuned FLAN-T5, incorporating a reward model within the PPO framework to ensure generated responses maintain the specified level of toxicity.

- Utilize the Google Perspective API to score toxicity and assign rewards accordingly, while implementing KL-Divergence as a penalty in the reward function to ensure generated text maintains human-like responses.

- Expand the imbalanced Jigsaw dataset into a balanced dataset comprising over 278,000 samples, significantly improving the balance of toxic and nontoxic samples.

- Outperform other data augmentation techniques such as zero-shot learning, back-translation, and instruction-tuned LLMs by leveraging RLHF optimization.

1.2.3 Proximal Policy Optimization-based Cascaded Inference System

- Develop an adaptive multi-stage inference system for toxicity detection, optimized using deep reinforcement learning (DRL) through Proximal Policy Optimization (PPO), balancing accuracy and processing efficiency.

- Implement a novel cascaded classification methodology where high-throughput classifiers perform initial filtering and highly accurate classifiers finalize the detection, thus optimizing both speed and accuracy.

- Design an optimized reward function within the DRL framework that reduces processing time and enhances classification accuracy by minimizing false positives and false negatives.

- Conduct extensive evaluations on multiple datasets, demonstrating significant improvements in processing time and detection accuracy with the proposed PPO-CIS framework.

- Provide a scalable, cost-effective solution for real-time content moderation, reducing the burden on human moderators and enhancing user satisfaction by promptly and accurately identifying toxic content.

By addressing these objectives, this thesis aims to provide robust solutions that enhance the efficiency, accuracy, and scalability of toxicity detection and content moderation systems on social media platforms.

1.3 Organization of the Thesis

This manuscript is organized into six chapters, each focusing on different aspects of toxicity detection on social media platforms and the innovative solutions developed to address these challenges.

Chapter 2 covers the essential preliminaries for this thesis, including deep reinforcement learning, proximal policy optimization, Actor-Critic with Experience Replay (ACER), text generation, instruction fine-tuning of large language models, and parameter-efficient fine-tuning.

Chapter 3 explores the current literature in toxicity detection, discussing various approaches and techniques used in the field. It identifies the limitations and problems in existing methods, highlighting the need for improved detection frameworks to enhance accuracy and efficiency.

Chapter 4 details the first proposed technique, focusing on the PDS framework. This chapter describes the methodology for evaluating the effectiveness and efficiency

of toxicity detection algorithms, explaining the profit-driven approach and user satisfaction considerations. It includes an experimental setup and results demonstrating the capabilities of the framework.

Chapter 5 addresses the issue of imbalanced datasets. This chapter introduces AugmenToxic, a technique for sentence-level text data augmentation specifically targeting toxic language. It describes the development and implementation of AugmenToxic, leveraging reinforcement learning from human feedback (RLHF) with Proximal Policy Optimization (PPO). The chapter provides a detailed methodology for paraphrasing toxic text while maintaining semantic coherence, along with experimental results showcasing improvements in dataset balance and classifier performance.

Chapter 6 delves into the challenge of real-time toxicity detection and the proposed PPO-CIS. It presents the development of an adaptive multi-stage inference system that employs a dynamic cascade of classifiers. The chapter explains the methodology for using high-throughput classifiers for initial filtering and highly accurate classifiers for final detection, optimizing both speed and accuracy. It highlights the integration of deep reinforcement learning (DRL) with PPO for optimal classifier selection, supported by experimental setup and results demonstrating the efficiency and accuracy of PPO-CIS.

Chapter 7 summarizes the key findings and contributions of the thesis, discussing the impact of the proposed solutions on toxicity detection. It identifies potential limitations and areas for improvement, and suggests future research directions and advancements in the field.

Chapter 2

Preliminaries

2.1 Text Generation

Imagine a language model, denoted as M, which responds to a consistent prompt P by generating a response y. The process involves the model sampling from its distribution $M(P)$ through decoding, represented as $y \sim M(P)$. In typical text generation scenarios, M calculates the probability distribution for the next token tk based on the prior context $C_{<tk}$, expressed as $p_\omega(tk|C_{<tk})$ [33]. The model learns parameters ω in training by maximizing the likelihood of observed data. This learned probability distribution ($p_\omega(tk|C_{<tk})$) is crucial in guiding the model as it decodes the next token, shaping the coherence of the generated text. By leveraging its learned parameters ω, the model captures relationships between tokens, enabling the creation of coherent and contextually relevant sequences. Various decoding algorithms, including greedy decoding, beam search, temperature sampling, and top-p sampling [34], play a pivotal role in how the model selects and arranges tokens, contributing to the overall coherence and relevance of the generated output.

In the context of text paraphrasing, for two sentences l and l' that serve as paraphrases of each other, we express this relationship as $l \equiv l'$, indicating their

equivalent meanings (Semantic Meaning(l) = Semantic Meaning(l') where $l \in L$ and $l' \in L'$). Consequently, when using a prompt to generate a paraphrase for l, we denote the paraphrased version as l', where $M(l|P) = l'$. Please note that a prompt is composed of an instruction I and an input $x \in X$ so that $P = I(x)$. Therefore, $(M(P) = y) \rightarrow (I(x)) = y)$. In the case of paraphrasing, where the instruction is a request for paraphrasing and the input is $l \in L$, then $M(I(l)) = l'$ where $l \equiv l'$.

Response generation employs various methodologies, including zero-shot learning, one-shot learning, few-shot learning, and fine-tuning language models on datasets with instructional annotations [35], [36]. Zero-shot learning enables a model to generate responses for categories or prompts it hasn't been explicitly trained on by leveraging its understanding of underlying concepts or patterns from the training data [37]. Conversely, few-shot learning involves training the model with minimal examples of a specific category or prompt, yet it still learns to generalize and produce responses for similar, previously unseen categories or prompts, showcasing its adaptability and generalization prowess [38]. Research has shown that instruction tuning substantially enhances zero-shot performance on unseen tasks [39]. In the following section, we will delve deeper into the details of Instruction Fine-tuning.

2.2 Instruction Fine-tuning (ITune)

Generative LLMs are initially pre-trained on an extremely large and diverse public dataset, and their weights can be fine-tuned for each task of interest using a much smaller task-specific dataset. Instruction fine-tuning (ITune) is a process in which a pre-trained model, represented by parameters ω, undergoes refinement based on a specialized instruction dataset D_{I}. This dataset includes input-output pairs that serve as explicit instructions or prompts for the model. Therefore, $D_{\text{I}} = \{(I_i, x_i, y_i)\}_{i=1}^{n}$ represents an instruction dataset with each sample consisting of an instruction (I_i),

an input sequence ($x_i \in X$), and its corresponding response ($y_i \in Y$), with n total samples in the dataset.

The objective is to adapt a M to better understand and generate responses aligned with the provided instructions. The updated parameters after fine-tuning are denoted as ω', and the fine-tuning process can be expressed as $\omega' = \text{ITune}(\omega, D_{\text{I}})$. This notation captures the transformation of the model's parameters to enhance its performance in generating contextually relevant outputs in response to specific instructions provided in the training dataset.

2.3 Parameter-efficient Fine-tuning (PEFT)

In the conventional fine-tuning process, model weights are usually copied from a pre-trained language model and adapted for a specific downstream task, requiring the generation of new weights for each task. However, full fine-tuning of parameters becomes impractical due to the rapidly growing size of models, making it infeasible to fine-tune the entire model and store separate copies of parameters for numerous downstream tasks [40]. Parameter-efficient techniques have been introduced to address concerns related to storage and computational costs associated with full fine-tuning [41]–[43]. As a noteworthy contribution to parameter-efficient fine-tuning (PEFT) techniques, one approach is Low-Rank Adaptation (LoRA) [44]. In the LoRA methodology, the pre-trained model weights remain frozen, and trainable rank decomposition matrices are introduced into each layer of the Transformer architecture. For each M, the hyperparameters r (representing the rank of the update matrices) and α_{LoRA} (a scaling factor crucial for stabilizing training) are fine-tuned [45]. This innovative technique effectively reduces the number of trainable parameters for downstream tasks, lowering GPU memory requirements and demonstrating a commitment to parameter efficiency in the adaptation process.

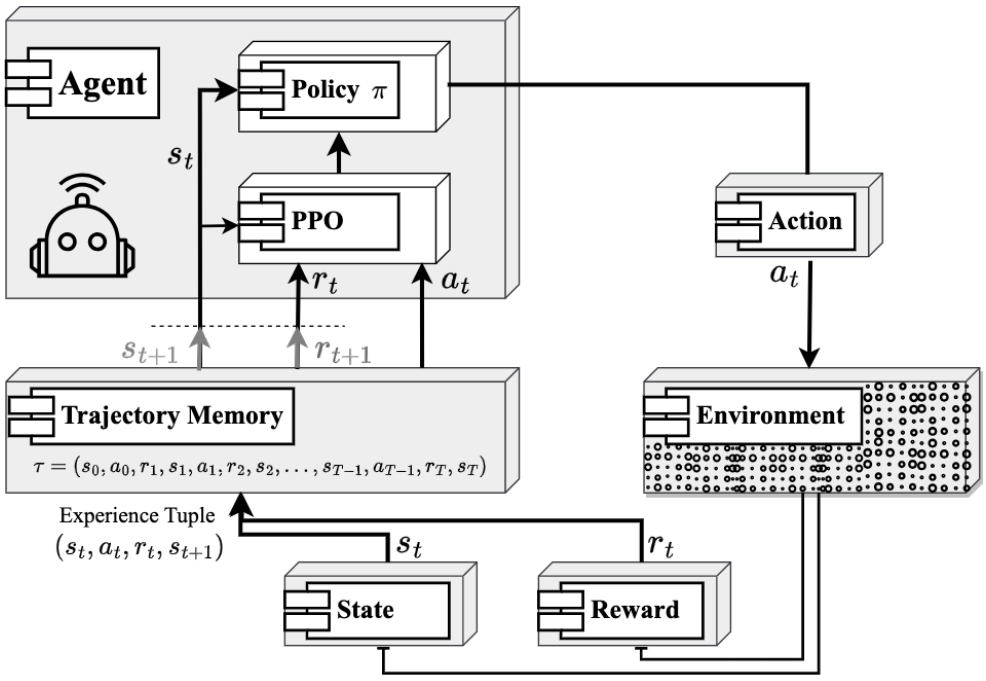


Figure 1: PPO Model Architecture Overview

2.4 Reinforcement Learning (RL)

Reinforcement Learning (RL) is a machine learning paradigm where an agent learns to make sequential decisions by interacting with an environment. The agent takes actions ($a \in A$), receives rewards ($r \in R$) or penalties, and adjusts its strategy to maximize cumulative rewards over time. RL involves sequences of states ($s \in S$), actions ($a \in A$), and rewards ($r \in R$), known as trajectories (τ). The main objective in RL is for the agent to learn a policy (π_θ), parameterized by θ, that maps states to actions to maximize expected cumulative rewards.

Policies in RL can be deterministic or stochastic [46]. A deterministic policy ($\pi : S \to A$) maps each state directly to a specific action, while a stochastic policy ($\pi(a|s) : S \to P(A)$) maps each state to a probability distribution over possible actions, indicating the likelihood of selecting each action given a particular state. Deterministic policies offer clear action choices but may be limited in adversarial environments and

computationally expensive in large action spaces [47].

During interactions, the agent selects actions a_t at time step t, transitioning from state s_t to s_{t+1} and receiving reward r_t. The objective is to maximize the total future rewards R_T:

$$R_T = \sum_{t=1}^{T} r_t$$

where T is the final time step. The policy $\pi(a, s)$ dictates the probability of selecting action a in state s, and the Q-function $Q(s, a)$ estimates the expected future reward for taking action a in state s under the current policy π. RL algorithms are evaluated based on their capacity to develop effective strategies across diverse environments, enabling agents to explore actions, exploit learned knowledge, and maximize long-term rewards in dynamic and complex environments.

Reinforcement Learning from Human Feedback (RLHF)

Reinforcement Learning from Human Feedback (RLHF) is a technique used to train AI systems to align with human goals, and it has become the primary method for fine-tuning state-of-the-art large language models (LLMs) [48], [49]. RLHF customizes pre-trained large language models (LLMs) by defining a reward model and fine-tuning the LLM using reinforcement learning. This process incorporates human feedback to capture desired sentiments in the model's responses, with the reward model mapping the model's output to a scalar reward representing human preferences [50]. By leveraging the extensive knowledge and capabilities of LLMs, RLHF encourages desired responses and behaviors, leading to safer, higher-performing, and more controllable AI systems [51].

2.5 Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) represents an advanced approach that combines the principles of RL with the power of Deep Learning (DL) [52], [53]. Unlike traditional RL, which often faces challenges in handling high-dimensional and complex data spaces, DRL leverages Deep Neural Networks (DNNs) to effectively manage and process such data [54], [55]. In contrast, DRL utilizes DNNs to approximate the Q-function $Q(s, a; \theta)$, where θ represents the parameters of the neural network [56]. This approach is highly effective for handling large or continuous state-action spaces [57]. Consequently, while the Q-function in traditional RL is simpler and less scalable, the Q-function in DRL provides superior representational power and generalization capabilities, enabling it to perform well in complex environments [46].

2.5.1 Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO), introduced by Schulman et al. [58], stands out as a robust and widely used deep reinforcement learning algorithm. PPO is known for its stability and efficiency in optimizing policies against specified reward functions, consistently achieving state-of-the-art performance across a wide range of challenging tasks. The core idea behind PPO involves iteratively updating the policy π_θ to improve its performance while ensuring stability during training. It aims to balance exploration and exploitation effectively, where exploration refers to the agent's ability to try out different actions to learn about the environment, and exploitation involves utilizing the knowledge gained to select actions that maximize expected rewards. Achieving a balance between exploration and exploitation presents a challenge [59].

Policy Gradient Improvement and Surrogate Objective Function

PPO is designed to improve the stability and reliability of policy gradient methods [60], which can suffer from high variance and instability as updates to the policy can sometimes lead to large, detrimental changes. PPO addresses these issues by ensuring that policy updates are more controlled and constrained. It achieves this through the use of a surrogate objective function that limits the magnitude of policy changes. The key idea is to optimize a clipped objective function, which penalizes changes to the policy that move too far from the current policy [61].

The objective function in PPO can be expressed as:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \right) \right] \tag{1}$$

where:

- $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio,
- π_θ is the new policy with parameters θ,
- $\pi_{\theta_{\text{old}}}$ is the old policy with parameters θ_{old},
- $\hat{A}_t$ is the advantage estimate at time step t,
- ϵ is a hyperparameter that controls the clipping range.

The clipping mechanism in PPO ensures that the ratio $r_t(\theta)$ stays within the range $[1-\epsilon, 1+\epsilon]$, thus preventing large deviations from the old policy. This constraint helps to maintain the stability of updates and improves overall training performance.

PPO can also incorporate the Kullback–Leibler (KL) divergence [62] as an additional penalty term or as a monitoring tool to ensure the policy does not deviate too

much from the previous policy. The KL divergence between the old policy $\pi_{\theta_{\text{old}}}$ and the new policy π_θ is given by:

$$D_{\text{KL}}(\pi_{\theta_{\text{old}}} \| \pi_\theta) = \mathbb{E}_{s \sim \rho_{\pi_{\text{old}}}} \left[\sum_a \pi_{\theta_{\text{old}}}(a|s) \log \frac{\pi_{\theta_{\text{old}}}(a|s)}{\pi_\theta(a|s)} \right]$$

In some variants of PPO, an adaptive KL penalty is added to the objective function to penalize large deviations:

$$L^{\text{KL}}(\theta) = L^{\text{CLIP}}(\theta) - \beta \mathbb{E}_{s \sim \rho_{\pi_{\text{old}}}} [D_{\text{KL}}(\pi_{\theta_{\text{old}}} \| \pi_\theta)]$$

where β is a coefficient that determines the strength of the penalty. The KL divergence serves as a broader measure of change and can adaptively adjust the penalty based on how much the policies diverge, providing another layer of control to ensure stability.

In summary, PPO enhances traditional policy gradient methods by incorporating a clipped surrogate objective function to ensure controlled and stable policy updates. This approach mitigates the risk of large, destabilizing changes to the policy, resulting in more reliable and efficient learning in reinforcement learning applications. PPO is widely used in various RL tasks due to its simplicity, robustness, and effectiveness, striking a good balance between performance and computational efficiency, making it a preferred choice for many complex and high-dimensional environments.

2.5.2 Actor-Critic with Experience Replay (ACER)

Actor-critic with experience replay (ACER) [47] is a type of actor-critic [63] method, which means it comprises two main components: the actor and the critic. The actor is responsible for selecting actions based on a policy $\pi(s; \theta_\pi)$, where θ_π are the parameters of the policy network. The critic, on the other hand, evaluates the actions taken by

the actor by estimating the value function $Q(s, a; \theta_Q)$, where θ_Q are the parameters of the value network. This dual structure allows ACER to simultaneously learn a policy for selecting actions and a value function for assessing the quality of those actions. One of the key innovations in ACER is the use of experience replay. In traditional actor-critic methods, updates are made using the most recent experiences, which can lead to inefficient learning and instability. ACER addresses this issue by storing past experiences in a replay buffer and using these experiences to perform off-policy updates. This approach not only improves sample efficiency by reusing past experiences but also stabilizes training by breaking the correlation between consecutive updates. Another critical aspect of ACER is the use of a trust region policy optimization (TRPO) [64] approach to ensure that the updates to the policy do not deviate too much from the current policy, preventing large and potentially destabilizing changes. Additionally, ACER employs importance sampling techniques to correct for any bias introduced by using off-policy data from the replay buffer. In summary, ACER enhances traditional actor-critic methods by integrating experience replay for better sample efficiency, trust region optimization for stable policy updates, and importance sampling for unbiased learning. These features enable ACER to effectively learn in complex and high-dimensional environments, making it a powerful algorithm in the field of deep reinforcement learning.

Chapter 3

Literature Review

This chapter is an extended version of the article entitled "Technological Solutions to Online Toxicity: Potential and Pitfalls", which was published in the *IEEE Technology and Society Magazine*, vol. 42, no. 4, pp. 57-65, Dec. 2023, doi: 10.1109/MTS.2023.3340235.

Social media platforms present a perplexing duality, acting at once as sites to build community and a sense of belonging, while also giving rise to misinformation, facilitating and intensifying disinformation campaigns and perpetuating existing patterns of discrimination from the physical world. The first step platforms take in mitigating the harmful side of social media involves identifying and managing toxic content. Users produce an enormous volume of posts that must be evaluated very quickly. This is an application context that requires machine learning (ML) tools, but as we detail in this chapter, ML approaches rely on human annotators, analysts, and moderators. Our review of existing methods and potential improvements indicates that neither humans nor ML can be removed from this process in the near future. However, we see room for improvement in the working conditions of these human workers.

Dealing with this problem is challenging due to the impracticality of manually removing toxic content, given the volume, velocity, and variety of online material. Consequently, platforms have increasingly adopted moderation systems incorporating machine learning (ML) models. However, these models have limitations, including biases and discriminatory outcomes. As a result, some platforms have opted to engage human moderators in assessing content flagged as potentially toxic by ML models and making the final decisions. However, these systems still undeniably have limitations that require deep investigation. Therefore, this chapter aims to comprehensively review moderation systems for automatic toxicity detection on social media, emphasizing the need to understand their constraints.

3.1 Online Toxicity: A Comprehensive Review

Over the past decade, there has been tremendous growth in social networking services, resulting in the creation of millions of data by platform users. As an example, Facebook

has nearly 2.85 billion users, while YouTube has 2 billion, and Twitter and Reddit have 350 million and 430 million users, respectively [16]. On Twitter, for instance, there are 192 million daily active users with an average of 6,000 tweets per second, 350,000 tweets per minute, and 500 million tweets sent each day, while the number rises to more than 140,000 tweets per second during certain events (natural disasters, breaking news, etc.) [14]. There are around 100 queries per second on Facebook, and 100 billion rows are processed per second, with average response times of less than 1 second [65].

Microblogging platforms are networks of real-time information, and their true value lies in their ability to absorb timely and relevant information that is significant to users [66]. Real-time promising an immediate experience, users also anticipate that their generated content will be immediately visible to followers, ideally within seconds, and any prolonged delay in this process could potentially lead to user churn [67]. The overwhelming flow of content generated by users, who are imagined as potential super processors [68], creates a significant challenge in real-time dissemination, computing, processing, and analysis of data. It is difficult to effectively process big data as a result of its variety, velocity, volume, and value [15]. Therefore, the aspect of real-time processing plays a pivotal role in addressing online toxicity, and the developed models should be able to handle a substantial volume of data within a second. Unfortunately, this aspect has been largely overlooked in the literature and we were unable to locate any related work that considered these features for the detection of toxicity on social networks.

3.1.1 Toxicity in Online Content

Although a significant volume of content is shared every second, the majority of it is relatively harmless, nontoxic, and knowledge-based. This leaves only a small fraction

that is toxic, often representing attacks on other users, individuals, or minority groups [69], [70]. This difference in rates of toxic and nontoxic content complicates the task of flagging toxic content and increases the risk of false positives.

A dataset provided by [71] consists of 80K tweets, each annotated with five judgments. The dataset is categorized into four groups: Abusive (11%), Hateful (7.5%), Spam (22.5%), and Normal (59%). It reveals that over half of the tweets do not contain any toxic language and are classified as normal. Furthermore, the analysis of 293 million English tweets using Google's Perspective API models by Qayyum et al. [72] demonstrated that 80 percent had toxicity scores below 0.40. In particular, users exhibiting high toxicity are often reported for misconduct violations and subsequently banned [73]. However, the enforcement of rules can sometimes be uneven.

A study on 1.18M Twitter conversations, encompassing 58.5M tweets and 4.4M users, found that users with moderate activity levels displayed a higher proportion of toxic tweets than both low- and high-activity users [74]. The study conducted in this paper determined that users posting between 10 and 1000 tweets were responsible for 22% to 23% of toxic tweets. In contrast, highly active users with a substantial number of posts (2,000-10,000) exhibited a lower percentage of toxic tweets (2%-19%) than low-activity users. Furthermore, more than 50% of the users did not post any toxic tweets. It should be noted that tweets from influential users often attract more replies from users with fewer followers, increasing the likelihood of influential users being targeted with toxic comments.

In another study by Radfar et al. [75], 178K tweets were collected from users with different follower-friend relationships. Radfar et al. found that almost 9.6% shared tweets between users who are mutual friends were deleted due to the high ratio of toxicity. In contrast, Twitter removed almost 40% of tweets shared between users who are not mutually connected or there is only a one-way connection due to toxicity. They

later selected 6.7K tweets and annotated them by human experts. Tweets between users who do not have a connection were found to be nearly three times more likely to be toxic than tweets between mutual friends. In addition, they also found that users follow great users to read their posts while targeting them with a toxic message.

Thus, the variety in social media content and users emphasizes the limits of relying on a single classifier to handle all platform situations effectively. Consequently, the dynamic nature of social media platforms demands different techniques for different cases. However, identifying the detector most suitable for various scenarios remains a significant challenge.

3.1.2 The Social Impact of Toxicity

Social Media platforms play a vital role in our daily lives by enabling users to stay in touch, express their views in real time, and providing instant access to information. However, the smooth sharing of content, and the cover of anonymity on microblogging platforms along with the lack of normative cues in online interactions also contribute to the widespread dissemination of antisocial and toxic behaviours [76].

The pervasive use of toxic language on social media has serious social implications. Individuals may hesitate to express their opinions or participate in discussions due to the fear of being targeted with harmful content [77]. In severe cases, this phenomenon can lead to mental health issues and social isolation, particularly impacting teenagers [78], [79]. Toxicity sometimes cooccurs with hate speech and violent threats and can cross over into the physical realm through doxing, swatting, or stalking [80], [81].

Manual removal of toxic content is infeasible due to the sheer volume and multilingual diversity of online content [19]. Human removal would require enormous staffing levels and expense, prompting the adoption of ML techniques as a viable solution [21]. Although ML models exhibit effectiveness in large-scale real-time content

classification, their performance is not without limitations, particularly in the context of toxic language [82].

Toxicity detection models may confidently make incorrect predictions based on spurious lexical features [83], [84]. In addition, subtle wording similarities can lead to inaccurate outcomes, highlighting a significant concern: bias in modeling, training, and usage [85], [86]. This bias may result in discrimination against specific social subgroups, including black users, women, and LGBTQI+ communities, in automated decision-making systems [87].

Despite these technical limitations, the urgent need to address antisocial online behavior has led platforms to frequently adopt moderation systems that integrate ML-based models. These systems are used to identify potentially harmful content, and some platforms also involve human moderators to review flagged content and make the final decision. It is worth noting that human judgments are utilized for annotating training datasets to develop ML models. As a result, both ML techniques and human individuals (moderators and annotators) remain crucial components for effective moderation systems to control online toxicity.

This collaboration between humans and ML has constraints that necessitate a deep dive into potential causes of misidentification and poorly managed outcomes. Our aim in this chapter is to comprehensively review ML techniques designed for automatic toxicity detection, emphasizing the need to examine and understand their limitations in the dynamic and varied landscape of social media content. We categorize these methods into social, policy and technical approaches, with a particular focus on technical solutions. We explore the potential of strictly technical approaches to address these risks. We also explore the limitations inherent in a purely "technological fix".

3.2 Existing Tools for Detecting Online Toxicity

3.2.1 Machine Learning and Deep Learning-based Strategies

Recent years have witnessed a substantial surge in research dedicated to exploring both the social and computational dimensions of toxic content detection [82], [88]. Existing research has tackled diverse aspects of online toxicity across various platforms. This includes endeavors such as the detection and categorization of toxic content [89], [90], assessment of the impact of online toxicity on communities [91], characterization of different types of toxicity [75], and identification of toxic users [92].

Machine learning (ML) approaches, both classical and deep learning (DL) algorithms, have been extensively implemented to identify toxicity in online conversations [93]. Numerous studies have applied various classical ML techniques to tackle online toxicity. For example, Logistic Regression (LR), Decision Trees (DT), Random Forest (RF), and Support Vector Machine (SVM) have been used effectively [92], [94]–[97]. Furthermore, combining Latent Semantic Analysis (LSA) with SVM and using Latent Dirichlet Allocation (LDA) [98] have been explored. Common feature extraction methods include bag-of-words (BOW), term-frequency-inverse document frequency (TF-IDF), and word embeddings [99]. The literature offers numerous studies comparing different techniques, considering various feature extraction and preprocessing methods [100], [101].

Data-driven and ML approaches for detecting, categorizing, and measuring toxic content are gaining popularity due to their scalability, robustness, and high performance [102]. However, traditional ML methods often face challenges in detecting toxic comments due to the high variability and complexity of language, including slang, sarcasm, and cultural nuances [103]. The accuracy of these algorithms largely depends on feature extraction from data [103].

Since 2006, research has shifted towards DL-based models derived from Neural Networks (NN), which are rapidly gaining traction [104]. Various Deep Neural Networks (DNN)-based models have been used extensively for detecting toxic content, such as Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), Long Short-term Memory (LSTM), Bidirectional LSTM (BiLSTM), Gated Recurrent Units (GRUs), and Bidirectional GRU (BiGRU) [105]–[108]. CNNs, specifically, have been widely used to classify toxic content due to their effectiveness in performing convolutions [7], [93], [109].

With the introduction of pre-trained word embeddings, recent trends in text classification tasks have switched to using language models trained on large unlabeled corpora. The work by Collobert and Weston [110] established the utility of pre-trained word embeddings. Mikolov et al. [111] created word2vec, followed by Global Vectors (GloVe) [112]. FastText embeddings, developed by Facebook AI Research, is another technique trained on Common Crawl and Wikipedia for 157 languages [113], [114]. These embeddings can be used as input features to DNN classifiers. For instance, Mohammed et al. [115] compared CNN, RNN, LSTM, bi-LSTM, GRU, and bi-GRU models using standard embedding layers and pre-trained embedding corpora such as GloVe, word2vec, and fastText.

Bidirectional Encoder Representations from Transformers (BERT) is another successful pre-trained language model introduced by Devlin et al. in 2018 [116]. It uses transformer-based architecture to comprehend contextual word relationships, generating contextualized word embeddings. Several BERT variants have been developed to improve its efficiency and effectiveness. These include BERT-base [116], ALBERT [117], DistilBERT [118], and RoBERTa [119]. Studies have demonstrated the superior performance of fine-tuning BERT for toxic language classification tasks [120]. Zhao et al. [121] evaluated different pre-trained language models, including BERT, RoBERTa,

and XLM, using various architectures such as bi-LSTM + BERT/RoBERTa/XLM and CNN + BERT/RoBERTa/XLM.

Deep learning techniques, such as neural networks and transformers, have shown greater success in this area by capturing nuanced language patterns and context, leading to improved detection accuracy [93]. DL models autonomously extract complex features from raw data without the need for manually crafted features, showcasing their ability to learn representations directly from input datasets [122]. Various studies have explored different DL models and their variations for toxicity detection, comparing their performance across different approaches [122], [123]. Some research has focused on developing hybrid DL models that blend multiple methods to improve detection accuracy and efficiency, yielding promising results [11], [124]. However, hybrid models can sometimes underperform with small datasets, making generalization challenging [125].

According to Jahan and Oussalah [126], prominent deep neural network architectures for this task include CNN [55], LSTM networks [127], Bi-LSTM architectures [128], GRUs [129], and BERT [116]. CNNs, with character embedding models, have shown excellent performance with an accuracy of 94%. Character-level embedding has improved performance over word-level embedding for CNNs [90].

3.2.2 Ensemble Learning-based Strategies

Ensemble learning is a method used to enhance the performance of classifiers by combining multiple models to create a stronger overall model. The three main techniques of ensemble learning are bagging, boosting, and stacking.

Bagging: short for bootstrap aggregation, is a technique introduced by Leo Breiman in 1996 [130]. Bagging involves drawing multiple bootstrap samples from the original dataset and applying a prediction method to each sample. The predictions

are then combined through averaging for regression or simple voting for classification, which reduces variance and improves the stability and accuracy of the final prediction.

Boosing: introduced by Robert Schapire in 1990 [131], enhances a weak learning algorithm by iteratively refining and combining multiple models for improved performance.

Stacking: By integrating predictions from multiple models into a single meta-learner, the approach demonstrates strong prediction capability, improving model generalizability and accuracy.

Numerous ensemble learning techniques have been proposed for detecting toxic content [90], [132]. For example, one approach involves combining various Recurrent Neural Network (RNN) models to identify hate speech in short texts, incorporating both textual content and user-related information [133]. Another method combines three base classifiers trained using word2vec for hate speech detection [134]. A meta-classifier (logistic regression) then combines the outputs and features from these base classifiers to generate the final output. The study demonstrates that this approach outperforms single classifiers, majority voting, and standard stacking in distinguishing hateful and non-hateful tweets across four datasets. Alternatively, a multi-view stacked classifier is proposed, leveraging different feature spaces to enhance model robustness against unintended gender bias [135]. Furthermore, other works have explored ensemble learning techniques such as parallelizing bagging, A-stacking, and random sub-space to accelerate and improve hate speech detection on social media [136]. This approach enhances efficiency and accuracy, particularly when handling large-scale, high-dimensional data, addressing the challenge of processing vast volumes of data with high accuracy and fault tolerance.

In addition, various innovative approaches have been proposed to address the issue of toxicity using ensemble learning. For example, DeL-haTE is a novel framework that

focuses on tackling class imbalance and model variability in hate speech detection [137]. This framework utilizes an ensemble of deep learning models with customizable CNN and GRU layers. The CNN layer extracts high-level features from the word embedding matrix, which are then processed by the GRU layer to identify important features from word sequences.

Another approach is the Stacked Weighted Ensemble (SWE), which combines multiple standalone classifiers to detect toxic language [138]. In many studies, ensemble learning techniques, especially those involving deep learning models, outperform single models. This highlights the importance and benefits of ensemble learning over individual models. However, these proposed techniques are not yet widely applicable in real-world scenarios. Additionally, employing the entire ensemble for every sample, although efficient, frequently results in high costs and redundancy [139]. Moreover, the most current techniques focus on achieving high accuracy in detecting toxicity but often neglect the need for throughput when processing large volumes of data. Therefore, these models must be tested in real-time, large-scale applications.

Social media platforms such as Facebook and Twitter currently lack real-time hate speech detection systems, often taking corrective measures only after harmful content has been posted online [103]. Given the large volume of data shared on these platforms every second, it is essential for models to process vast amounts of data efficiently and effectively. Balancing accuracy and speed is a significant challenge, making it crucial to choose the most suitable classifier for each specific use case. Moreover, CMSs often incorporate human oversight with ML/DL models, so optimizing collaboration between the two is essential for achieving the best performance [19]. Additionally, models that perform well on their own may not necessarily achieve the best results when working alongside a human moderator [140]. Unfortunately, there is a lack of research addressing this challenge in toxicity detection, highlighting the need for greater attention to this

issue. Google's Jigsaw team explored how human moderators and machine learning models can work together in content moderation systems to enhance their overall performance. They presented an approach that integrates model uncertainty into the collaboration and developed metrics to measure the system's effectiveness [19]. Although human moderators may be necessary to remove toxic content, automated tools should still be enhanced to handle the large scale of user-generated content in real-time. Improving these tools can reduce errors and misclassifications, ultimately lightening the workload for human moderators and enhancing online safety. To achieve this, we are exploring research beyond toxic language detection to discover advanced methods for processing samples effectively and efficiently.

The literature suggests that employing a tree of classifiers is recognized as an effective approach for achieving high-throughput and accurate data processing. A "One-class-at-a-time" approach [141], a multistage cascading classification technique was proposed for triaging psychiatric patients using ML on textual patient records. This method classifies one class at a time and uses the most accurate classifier at each stage. It outperformed traditional multiclass classifiers, achieving an overall high accuracy rates for individual classes. This approach helps reduce expert effort and serves as decision support for triaging psychiatric patients based on the severity of their condition. However, this approach is not high-throughput enough to accurately process large volumes of data in seconds.

Employing multiple learning models for a single task is a common practice seen across different fields, enhancing classification accuracy but also presenting challenges including increased time consumption, costs, and scalability issues [142]. Different strategies have been proposed to address this, such as an AdaBoost-based algorithm and a cascading approach for rapid and accurate visual object detection, aiming to prioritize computational resources on relevant areas while ignoring backgrounds [143].

Employing multiple learning models for a single task is a common practice seen across different fields, enhancing classification accuracy but also presenting challenges such as increased time consumption, costs, and scalability issues [142].

Different strategies have been proposed to address this, such as using an AdaBoost-based algorithm and a cascading approach for rapid and accurate visual object detection, aiming to prioritize computational resources on relevant areas while ignoring backgrounds [143].

3.2.3 Reinforcement Learning-based Strategies

Reinforcement Learning (RL) is a computational approach designed to understand and automate goal-oriented learning and decision-making processes [144]–[146]. It involves an agent that interacts with a dynamic environment, taking actions and receiving feedback in the form of rewards or penalties [147]. The agent's objective is to learn optimal behavior over time through trial-and-error interactions, aiming to maximize cumulative rewards and achieve specified objectives [148], [149]. This iterative process entails exploring extensive solution spaces and developing efficient strategies across diverse domains, demonstrating RL's capability to tackle complex decision-making challenges effectively [150].

However, accurately estimating expected rewards for every possible state-action combination becomes impractical in scenarios with extensive state and action spaces due to computational complexity [46]. Consequently, approximation-based techniques, such as employing neural networks, are often utilized [63]. These techniques enable RL to scale effectively and perform well in complex environments by approximating the value function [151].

Coupled with deep learning [109], [152], RL has significantly advanced, enabling it to tackle previously intractable decision-making problems with high-dimensional state

and action spaces [148]. Deep reinforcement learning (DRL) algorithms have been successfully applied to a wide range of problems. In robotics, DRL enables the learning of control policies directly from camera inputs in real-world environments, surpassing traditional controllers that were either hand-engineered or relied on low-dimensional features of the robot's state [153], [154]. In the realm of complex games, DRL has achieved groundbreaking milestones, such as the success of AlphaGo in mastering the game of Go, highlighting its potential in strategy formulation and decision making [151]. In adversarial learning, DRL plays a crucial role in developing robust models that can withstand adversarial attacks, thereby enhancing cybersecurity measures [155]. Additionally, DRL has shown promising results in malware detection, providing adaptive techniques to identify and mitigate evolving threats [156]. Natural Language Processing (NLP) has leveraged the advantages of DRL across various applications, including dialogue system [157], [158], machine translation [159], and text generation [160].

Recently, DRL has been applied to optimize fine-tuned Large Language Models (LLMs) [161] for generating toxic or nontoxic textual content [162], [163].

While research on using RL for online toxicity detection is still emerging, several notable contributions have been made. The first significant work in this area, introduced in 2019, proposed a self-learning model using Deep Q-Learning Networks to detect toxicity in online conversations [164]. Although the results did not surpass the baseline neural network model, the technique achieved competitive results compared to other baseline machine learning models, whose F1-scores ranged from 50-60%.

Another important development is Q-Bully, designed to detect cyberbullying on various social media and online gaming platforms using RL combined with NLP techniques [165]. This study incorporated RL by feeding messages and posts from bullies

and victims to an RL agent for classification. Additionally, ConBERT-RL, a policy-driven DRL-based approach, was proposed to detect homophobia and transphobia in low-resource languages [166]. This framework leverages a concatenated representation that combines BERT's contextualized outputs with a pre-trained classifier model within a reinforcement learning setting. Specifically, it uses the REINFORCE algorithm to capture problem-specific features and understand nuances in transliterated Tamil words into English, significantly improving classification accuracy for offensive comments. This demonstrates ConBERT-RL's robustness and effectiveness in capturing language-specific features.

3.2.4 Optimizing Ensemble Models with Reinforcement Learning

DRL emerges as a promising approach for economically selecting classifiers, as demonstrated by innovative solutions such as Security Policy Implementation using Reinforcement Learning (SPIREL) [142] and Cost Effective Transfer of Reinforcement learning policies (CETRA) [139]. These approaches leverage DRL for dynamically allocating detectors in a sequential manner for malware detection.

SPIREL is a DRL-based framework, specifically Actor–Critic, where the reward function assigns values based on both correct and incorrect classifications, as well as the runtime needed for analyzing each sample. This method ensures a balanced trade-off between accuracy and computational efficiency, making it suitable for real-time applications.

Building upon SPIREL, CETRA empowers organizations to establish desired benchmarks for key performance indicators, such as memory usage, running time, or Area Under the ROC Curve (AUC). CETRA dynamically adjusts its reward function to achieve these objectives, providing a flexible and adaptive approach to maintaining

optimal performance across different operational contexts.

By incorporating these advanced DRL techniques, both SPIREL and CETRA demonstrate how reinforcement learning can effectively manage and optimize classifier selection processes, offering significant improvements in resource allocation and overall system efficiency.

3.2.5 Evaluation Metrics

Various metrics have been used to assess algorithm performance and compare different techniques developed to detect toxic language. A systematic review of ML techniques for toxic comment classification [93] indicates that F1-score, accuracy, and area under the ROC curve (AUC ROC) are the most widely used evaluation metrics. Additionally, other metrics including Log Loss, Hamming Loss, Mean Precision, Mean Recall, Specificity, and Mean Error Rates are utilized. These metrics primarily focus on gauging the model's ability to accurately differentiate between toxic and nontoxic content. Although crucial for identifying the most accurate model, this assessment is insufficient, especially in the context of real-time information networks such as social media platforms.

3.3 Existing Tools for Addressing Class Imbalance in Toxicity Detection

In the literature, various solutions have been proposed to the class imbalance problem. These techniques can be categorized into Data Level, Algorithmic Level, Ensemble Learning, and Data Augmentation, each of which is briefly discussed in this section. Furthermore, we examine methodologies employed in toxicity detection.

3.3.1 Data-level Approaches

In managing the class imbalance at the data level, the primary goal is to adjust the distribution of classes by strategically resampling the data space. This involves increasing instances of the underrepresented class through techniques including oversampling and reducing instances of the overrepresented class through methods such as undersampling. In many cases, a combination of both oversampling and undersampling techniques may be employed to achieve optimal results [167]–[170]. Each of these resampling techniques, along with their variations and applications, will be explored individually and in detail in the following subsections.

3.3.1.1 Undersampling

In the undersampling method, the primary focus is on the majority class within the dataset, from which instances are extracted either randomly or through specific techniques to achieve class balance [171], [172]. Undersampling, while incurring the main drawback of information loss through the deletion of examples from the training data, nonetheless, offers the benefit of reducing the time required to train models by diminishing the size of the training dataset [172]. The most straightforward method of undersampling involves randomly choosing a portion of samples from the majority class [173]. The random undersampling (RUS) strategy poses a significant risk of eliminating potentially valuable data from the majority class [174], [175]. In response to this limitation, specific methodologies, such as informative undersampling techniques, selectively eliminate insignificant patterns from the majority class, thereby aiming to maintain performance levels and overcome this drawback [176]. One suggested informative undersampling method, the Condensed Nearest Neighbour (CNN) discussed in [177], serves as a data reduction approach to create a representative subset of the original training set, proficient in accurately classifying all instances [176].

Similarly, the Edited Nearest Neighbors (ENN) uses a K-nearest neighbors (K-NN) approach to identify atypical examples within their neighborhood and subsequently removes them [178]. Furthermore, one-sided selection [179] serves as an alternative approach utilizing Tomek links [180] to detect and eliminate such atypical instances. These approaches are not very useful for text data; they are more applicable to numerical data. In the case of toxic language detection, this technique proves ineffective because we still need a sufficient number of samples from the nontoxic class to successfully train classifiers for distinguishing toxic from nontoxic content. Therefore, our proposed method is specifically developed for textual data and can effortlessly handle unstructured sentences while increasing toxicity scores. There is no need to remove samples from the nontoxic class.

An alternative approach proposed to tackle the RUS limitation involves replacing the strategy with a clustering technique as discussed by [174]. Employing cluster-based techniques aims to group similar objects, or data samples, into the same clusters, with objects in distinct clusters differing in their feature representations. In 2017, imbalanced-learn, an open-source Python toolbox, aimed to address imbalanced dataset challenges in ML and pattern recognition by incorporating state-of-the-art techniques grouped into four categories: (i) under-sampling, (ii) over-sampling, (iii) combined over- and under-sampling, and (iv) ensemble learning methods [181]. Imbalanced-learn [1] offers tools such as ClusterCentroids and RandomUnderSampler. ClusterCentroids reduces the majority class by substituting a cluster with the centroid from a KMeans algorithm. RandomUnderSampler swiftly balances data by selecting a subset randomly. Mediratta and Oswal [182] employed RandomUnderSampler to tackle the imbalance issue in a toxic content classification model. In their comparison of various machine learning models (Support Vector Machine (SVM), Naïve Bayes, Gated Recurrent Unit

[1] https://imbalanced-learn.org/stable/

(GRU), Long Short-term Memory (LSTM), they observed that SVM and Naïve Bayes achieved high accuracy even without addressing the imbalance, effectively learning from imbalanced datasets. The most promising outcomes occurred with GRU when handling imbalance with a random sampler and employing GloVe (Global Vectors for Word Representation) word embedding. Rupapara et al., [101] introduced an ensemble technique for toxic comment detection, comparing its performance with other ML classifiers on both imbalanced and balanced datasets. They employed various resampling methods, including RUS and oversampling, highlighting that machine learning models achieved superior performance when using oversampling for dataset balance.

3.3.1.2 Oversampling

Undersampling techniques work well for datasets with a lower class imbalance ratio, while oversampling methods effectively manage high-class imbalance [183]. Yet, oversampling tends to expand the training set size by replicating patterns, leading to extended learning times and potential overfitting [170], [174], [184]. Similar to random undersampling, oversampling can occur randomly but involves replicating instances from the minority class to achieve dataset balance [185].

An alternative oversampling approach, referred to as informative oversampling, focuses on amplifying the smaller class. In contrast to generating new samples, this method selectively chooses samples from the minority class for resampling instead of employing a random approach [170].

Another technique for oversampling is synthetic oversampling, where artificial samples are generated for the minority class [186]. These additional samples supplement vital information to the minority class, preventing misclassification of its instances. The Synthetic Minority Oversampling Technique (SMOTE), introduced by [167],

operates as an oversampling approach that aims to generate additional instances for the minority class by interpolating between various neighboring instances within that class [185].

Continuing the exploration of oversampling techniques for the minority class, additional methods include the Modified Synthetic Minority Oversampling Technique (MSMOTE) [187], and the Selective Preprocessing of Imbalanced Data (SPIDER) [188]. MSMOTE modifies SMOTE by categorizing minority class instances into safe, border, and latent noise groups, adjusting the neighbor selection strategy accordingly, and SPIDER combines local minority class oversampling with identifying noisy majority class instances, implementing different preprocessing methods to enhance the minority class and eliminate remaining noisy majority class examples [185].

In the case of toxic language, where the number of samples in the nontoxic class significantly surpasses the number of samples in the toxic class, oversampling can be a beneficial strategy. However, it must be performed with caution. The nature of language, particularly sentences, implies that the combination of various words together can be quite intricate. Some words may not be toxic when considered individually, but they can become toxic when used in combination.

In our proposed method, we leverage pretrained large language models that have been trained on extensive datasets and exposed to numerous sample sentences. This approach helps us in generating new samples in the minority class, thereby increasing the diversity of samples.

3.3.1.3 Hybrid Sampling

Both undersampling and oversampling techniques present challenges by respectively risking the removal of vital majority class examples, potentially causing underfitting, and inducing overfitting through an increased number of specific but potentially

misleading minority class samples affecting the model's decision boundaries [167], [189]. A sought-after approach involves integrating the benefits of both techniques to manage imbalanced medical diagnostic data.

3.3.2 Algorithmic-level Approaches

This approach involves developing new algorithms or adapting existing ones to be more responsive to class imbalance issues [190], [191]. In this approach, the focus is on addressing the minority class, preventing the learner from exhibiting bias toward the majority class to mitigate the overall cost associated with misclassification [192], [193]. Usually, these methods include the utilization of cost-sensitive and ensemble approaches [194]–[196]. Resampling techniques and algorithmic methods alone may prove insufficient in addressing class imbalance challenges in high-dimensional scenarios [197], [198].

3.3.2.1 Cost-sensitive Learning

The cost-sensitive learning framework integrates strategies at both the data and algorithmic levels, considering the increased costs associated with misclassifying samples from the positive class compared to the negative ones [185], [190], [199]. This approach has been applied to address imbalanced labels in toxic content detection by incorporating it into machine learning and deep learning models to enhance overall performance. However, it is crucial to note that accurate estimation of misclassification costs is necessary, and it can be challenging to achieve in practical applications [200].

3.3.3 Ensemble Learning Approaches

The limitation of traditional approaches (sampling, algorithm level, and cost-sensitive) lies in the requirement to define misclassification costs, often unavailable in datasets,

leading to the introduction of ensemble-based methods that combine ensemble learning algorithms with data-level and cost-sensitive techniques to address class imbalance, although the challenge of defining costs persists [185]. Boosting, bagging, and stacking stand out as the most frequently employed techniques within this category [201]. According to a survey on the application of ensemble learning methods for class imbalance problems, Random Forest (RF) and XGBoost have emerged as the most commonly utilized methods in the literature, with both demonstrating reliable performance [202]. Another review paper focusing on ensemble learning and data augmentation models for class imbalance issues demonstrated that various combinations of ensemble learning and oversampling techniques, including SMOTE-LightGBM and random oversampling-LightGBM (ROS-LightGBM), are effective approaches for addressing this challenge [203].

Addressing data imbalance is a crucial aspect in ML, and various techniques have been proposed to tackle this issue across different domains. However, when it comes to textual data, especially in the context of toxic language detection, existing methods face challenges. Traditional data imbalance techniques, such as oversampling or undersampling, may not be directly applicable to textual data due to its unique characteristics. Moreover, the nature of toxic language data often involves intricate linguistic nuances, making it difficult to apply standard resampling methods effectively. In light of these challenges, there is a growing need for specialized techniques in data augmentation tailored for textual data, which can help alleviate data imbalances and enhance the performance of toxic language detection models. Hence, we further explore text data augmentation techniques, particularly those designed for balancing datasets to improve online toxicity detection.

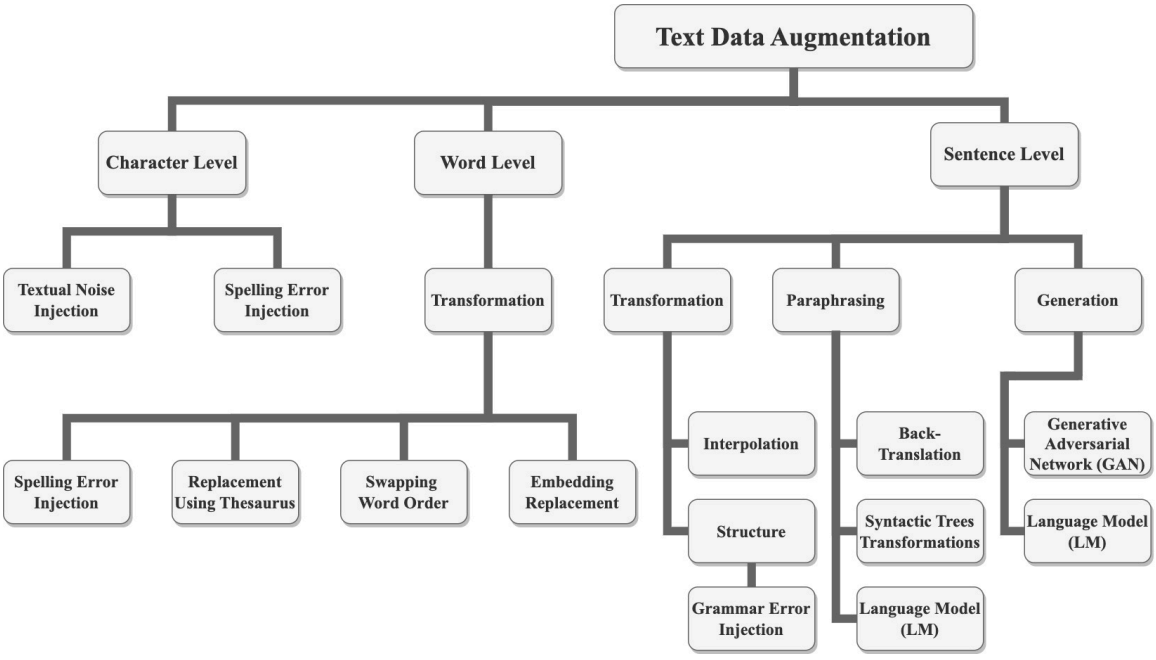


Figure 2: Taxonomy of Text Data Augmentation Techniques

3.4 Text Data Augmentation

Text data augmentation (TDA) involves generating additional training data from existing data, thereby expanding the dataset available for training classifiers or classification models [204]. Unlike image data augmentation, where simple transformations such as rotation and translation easily preserve the original label, these methods for TDA present a greater challenge in maintaining the original label after perturbations [205]. In recent years, researchers have proposed both unsupervised and supervised TDA methods, generating synthetic data through advanced techniques, where unsupervised methods do not rely on labeled data and supervised methods utilize labeled data for augmentation [204]. The taxonomy of TDA techniques is illustrated in Figure 2, accompanied by a summary of citations included in Table 1. Comprehensive reviews of TDA methodologies can be found in survey papers, such as [205]–[207]. These surveys provide an in-depth synthesis of the field. TDA encompasses diverse strategies

applied at different levels, including character, word, sentence, and document levels. At the character level, techniques involve Textual Noise Injection and Spelling Error Injection. Moving to the word level, augmentation includes the introduction of spelling errors, random deletion, replacement using the thesaurus, swapping word order, and embedding replacement. In the realm of toxic language, a specific instance involved replacing words with their synonyms [208] using word embeddings such as Word2Vec [209], GloVe [112], Fattest [114]. Word replacement can also be performed using features obtained from ConceptNet relations and their descriptions extracted from Wikidata [210].

Expanding to larger units, such as phrases or sentences, TDA incorporates transformation, paraphrasing, and sentence generation. These techniques can be further segmented into specialized methods. For instance, back-translation involves translating a sentence from one language to another and then translating it back into the original language. The back-translation method was employed for data augmentation in the context of hate speech and cyberbullying, involving the initial translation of English text to German and then translating it back to English [220]. Syntactic tree transformations [221] are additional techniques applied at this level. Moreover, the generation of new sentences is achieved through advanced approaches, including Generative Adversarial Networks (GANs) and generative language models. These models contribute to the creation of diverse and contextually relevant text during the augmentation process. As an illustration, generative language models such as Generative Pretrained Transformer 2 (GPT-2) [222] were employed to generate extra-textual samples for the minority class by fine-tuning on existing minority class samples [223], [224]. ToxiGen, a large-scale dataset of toxic and benign statements about minority groups, generated using a demonstration-based prompting framework and an adversarial classifier-in-the-loop decoding method [163]. They demonstrate that

Table 1: Summary of Text Data Augmentation Techniques with Citations

TDA Level	Method	Reference
Character Level	Textual Noise Injection	[207]
	Spelling Error Injection	[207]
Word Level	Spelling Error Injection	[211]
	Replacement Using Thesaurus	[212]
	Swapping Word Order	[213]
	Embedding Replacement	[214]
Sentence Level	Interpolation	[214]
	Structure	[215]
	Grammar Error Injection	[216]
	Back-Translation	[205]
	Syntactic Trees Transformations	[207]
	Generation- Language Model (LM)	[217]
	Generative Adversarial Network (GAN)	[218]
	Paraphrasing - Language Model (LM)	[219]

ToxiGen improves the performance of toxicity classifiers on human-written data and can also help fight machine-generated toxicity. Leveraging the extensive ToxiGen dataset, ConPrompt introduces an innovative pre-training strategy tailored for implicit hate speech detection. Through contrastive learning and prompt-based positive sampling, ConPrompt, embodied by ToxiGen-ConPrompt, emerges as a leading solution. Experimental findings undeniably demonstrate ToxiGen-ConPrompt's superiority over established models including HateBERT and BERT, showcasing its exceptional generalization and bias mitigation [225].

In certain studies, researchers assigned the minority class label to each newly generated sample [226]. Alternatively, in other studies, the classifiers were fine-tuned for toxicity detection, and only the samples identified as toxic by the classifier were retained after analysis [227]. To ensure that augmentation samples capture target class features, using off-the-shelf language models is limited due to their undirected and random generation, as noted by Liu et al. [228]. In response, they presented Data Boost, a text data augmentation framework guided by reinforcement learning and based on an off-the-shelf language model (GPT-2). The approach involves computing a Salience Score for each word and selecting the top-N highest-scoring words as the salient lexicon for the target class label. Lee et al. [229] explore alignment algorithms, particularly Direct Preference Optimization (DPO), and their role in reducing toxicity in pre-trained language models such as GPT-2-medium. Their research delves into toxicity representation and elicitation in these models, showcasing how DPO can curb toxic outputs while preserving learned abilities. Furthermore, they present a method to revert models to toxic behavior, underscoring the significance of understanding alignment algorithms in natural language processing.

Comprehensive utilization of these techniques not only aids in overcoming overfitting but also enriches the input feature range, enhancing the overall robustness of

classification models [230], [231]. However, current methodologies for data augmentation exhibit significant limitations and remain imperfect. For example, labeling all generated samples with the minority class label [226] is flawed because it does not guarantee the preservation of the intended target class label or ensure that the generated samples adhere to the characteristics of their supposed label. Furthermore, analyzing the toxicity of generated samples using a classifier trained on the same data used for fine-tuning generative language models may introduce bias, given its lack of data agnosticism and potential inefficacy on diverse datasets [232]. In addition, there is a common deficiency in instruction fine-tuning of large language models (LLMs) for specific tasks. In the context of conditional generation, the proposed reward function, which employs Salience Gain [228], tends to prioritize token similarity over toxicity level and may exhibit limited improvement in tasks involving challenging class modeling. Moreover, it faces challenges in extracting explicit lexical features for metaphor, sarcasm, and formality. These techniques may struggle to effectively implement data augmentation, particularly on a large scale, especially for unstructured sample sentences from social media. Additionally, generated samples may deviate from the intended scope of the work, possibly leading to hallucinatory outcomes [233]–[235]. To address these limitations, we propose a sentence-level data augmentation technique based on paraphrasing. In this approach, we employ fine-tuning a pretrained Large language model (LLM) through instruction specifically for the task of text paraphrasing. We optimize the model by using reinforcement learning to generate toxic samples. Additionally, data-agnostic models are used to assess the toxicity of generated samples, enabling the assignment of toxic rewards accordingly. This approach is suitable for generating samples at a large scale and has demonstrated superior performance compared to other data augmentation techniques, such as back-translation.

3.5 Challenges in Online Toxicity Detection

Little scholarship exists exploring the limitations of innovative techniques for addressing antisocial behaviors. Some existing studies have examined natural language processing (NLP) techniques for automatic hate speech detection while also recognizing and addressing their limitations [236]. Others have explored the reliability of pretrained large language models (LLMs), assessing their effectiveness in decision-making tasks that involve aspects of uncertainty, robust generalization, and adaptation [237]. Furthermore, a comprehensive analysis of the risks associated with LLMs has been conducted to better guide responsible innovation [238]. This analysis draws from various fields, including computer science, linguistics, and social sciences. The examination and consideration of uncertainty in ML techniques, including the realm of online toxicity detection, have prompted the proposal and application of methods designed to address this issue [239], [240].

Despite these efforts, there is uncertainty about the efficacy of ML techniques in detecting and managing toxic speech. Notably, the presence of biases in both data and algorithms poses a significant issue, potentially resulting in increased discrimination [241]. Bias in datasets causes posts from minoritized groups to be over-flagged (false positives (FPs) identifying toxicity when none existed) [242]. Furthermore, research has concentrated predominantly on textual content, and the conclusions drawn about uncertainty in toxicity detection using ML techniques for textual posts may not necessarily apply to multimodal content. Consequently, a comprehensive understanding of uncertainty in identifying toxic multimodal content is necessary.

Deploying ML techniques at the platform level poses broader challenges. Some studies indicate that considering the prediction uncertainty of neural networks facilitates detecting complex text inputs. This includes short or lengthy texts with

less informative tokens and potentially incorrect predictions, which requires manual verification. Google's Jigsaw team [19] presented Collaborative Toxicity Moderation in the Wild (CoToMoD), a benchmark to assess the effectiveness of systems involving both ML models and human moderators. They introduced principled metrics such as collaborative accuracy of the oracle model (OC-ACC), collaborative area of the oracle model under the receiver operating characteristic curve (ROC) (OC-AUC) and review efficiency, which evaluate the performance of the system in utilizing human attention and decisions, going beyond traditional predictive performance or uncertainty calibration measures.

Obstacles to effective ML detection occur at various stages of content moderation. We start by examining challenges related to obtaining labeled datasets, which can introduce uncertainty and impact prediction accuracy. Following that, we delve into the modeling approach, covering both training and testing phases.

3.5.1 Challenges in Data Preparation

ML techniques, particularly supervised learning, are valuable tools for detecting online toxicity [125]. The initial and crucial step in these techniques involves data collection, which, in the context of online toxicity, entails gathering data from various platforms. Social media platforms have a global reach, empowering people worldwide to create profiles. However, the pattern of use differs among regions, with certain platforms being more popular in one region than another, and others being outright banned by federal governments. This regional variability in use patterns adds a layer of complexity to analysis but does not significantly impact the diverse user base across social media platforms. The diversity results in vast amounts of unstructured and multilingual content being shared every second on social media platforms, rendering its management exceptionally complex. Once a dataset is collected and cleaned, it needs to be reviewed

and labeled by annotators to be used for training classifiers. This task itself has many challenges. Many datasets are publicly available for toxicity detection, including the Jigsaw Toxicity Dataset [243], which involves Wikipedia comments labeled by human annotators, and ToxiGen [163], comprising a large-scale machine-generated dataset that addresses adversarial and implicit hate toward minority groups. However, precise guidelines on how these datasets were annotated are not available. In some cases, tools such as Google's Perspective API [2], HATECHECK [3] or pretrained models including HateBERT [244] are used to annotate the dataset. However, they may not perform perfectly, indicating that the labeled dataset may not be perfectly accurate and can potentially introduce bias [29]. Therefore, for each specific task, a rubric is required for annotation. A rubric is a set of guidelines on how specific words should be interpreted in different contexts. However, this guide document is not enough to ensure unbiased annotation of potentially toxic posts. Annotators need an ethical framework to direct the task. Two additional challenges relate to the annotators themselves. Human annotators are sometimes exposed to violent and disturbing content, which often has a hidden human cost. These annotators are often paid far less than the prevailing minimum wage and below a living wage [245], [246].

These poor working conditions compound the second challenge human annotators face, which is having sufficient identity knowledge of the target of antisocial behavior to accurately identify toxicity. For example, when detecting toxic comments against LGBTQIA2+ people, the annotators should ideally include people from that group to review the provided dataset and label each sample according to the rubric. However, detailed demographic information on the annotators is rarely available. Moreover, another significant challenge arises from the ambiguity that exists between various forms of toxic language, including hate speech and offensive language. In addition,

[2]https://perspectiveapi.com/
[3]https://hatecheck.ai/

annotating multilingual datasets are also challenging. Many ML algorithms are trained primarily on English-language data, leading to potential shortcomings in their performance when applied to other languages. Furthermore, these models often struggle to accurately identify subtle content in statements that carry multiple meanings. An illustrative example is the challenge posed by dog-whistle phrases in various languages. A dog whistle refers to a pejorative term deliberately crafted to be discernible only by individuals actively engaged in discrimination against a particular group while remaining undetected by the general population—those who neither experience discrimination in this manner nor partake in discriminatory behavior [247].

Deep learning (DL) and transformer-based models have demonstrated effective results in identifying toxic content [122]. However, they may also worsen the problem of data bias since these models rely heavily on large amounts of training data that may not be inclusive of all user groups [25]. Although LLMs have recently shown impressive capabilities in a wide range of applications and tasks, including natural language understanding, generation, and translation, it is important to note that they can also inherit biases from the training data, as these data can mirror the societal biases present during their collection. One potential remedy to inherited data bias is to create models with logic awareness. Adam et al. [248] conducted a study to determine if language models that incorporate logic-awareness could successfully mitigate the presence of harmful biases. Their findings revealed that when a language model lacks explicit logic acquisition, it often displays a significant degree of biased reasoning, while, integrating logic learning into the language model can decrease.

Rule enforcement is another approach to reducing antisocial behavior. Platforms work to keep users safe by enforcing rules such as removing toxic posts or suspending accounts from users who frequently engage in hateful conduct. However, it is not clear if suspending users who behave badly reduces the overall prevalence of toxic

speech online. Suspending an account may encourage bad actors to migrate to other platforms with fewer rules. Ali et al. [249] analyzed posting behavior on Twitter (now called X) and Reddit and compared the same individuals' content on Gab, a site known for promoting hateful conduct and not enforcing behavioral rules. The results of this comparison revealed that users exhibited increased toxicity when they experienced suspension on one platform and were compelled to migrate to another. Additionally, their level of activity rises, leading to a higher frequency of posts.

In terms of the process of managing online toxicity, removing toxic content, and suspending users are critical moments because these actions have important consequences. Content moderation involves both people and ML models working together. For instance, the model can pick outposts that probably break the rules, and then human moderators can take a closer look at them. Accurate classification is crucial in this situation. If toxic content is mistakenly identified as nontoxic (FN), users will see harmful content (even if they have settings to prevent this). Moreover, having a high number of FPs or FNs can lead users to disengage from discussions and become less active on these platforms. In addition, over-flagging content as toxic can also drive users away, as it is likely that many of their posts will be marked as toxic,discouraging them from posting or commenting due to the high chance of their content getting blocked. Jhaver et al. [250] examined how Reddit users responded to the platform's moderation process. Their findings showed that 18% of the participants agreed that their posts were correctly removed, 37% were uncertain about the reasons behind their post removal, and 29% expressed frustration regarding the removal of their posts.

3.5.2 Challenges in Model Construction

After preparing the dataset, the subsequent step involves training and testing the model, where the importance of a well-prepared dataset cannot be overstated. Quality and quantity both play vital roles in this phase. To ensure the reliability of results, it is imperative to maintain a near balance in the number of samples across different classes, creating a balanced dataset. Datasets with imbalanced class distributions pose a frequent challenge across various classification tasks [102]. This issue is particularly challenging in the domain of toxic language detection because toxic language is typically less frequent when compared to nontoxic language [26]. As an example, consider a dataset of 100,000 tweets gathered from Twitter (now called X), where the proportion of toxic tweets was relatively low, accounting for approximately 5% (around 5,000 instances), while the majority of the data consisted of nontoxic content [71]. Consequently, this data imbalance can result in models excelling at recognizing nontoxic language but struggling with identifying toxic content. This issue can be compounded by the use of self-reported data or crowd-sourcing for annotation, which may not accurately reflect the diversity of toxic language used online [27]. Moreover, proposed techniques may be limited by the very nature of language itself. Sarcasm, irony, and other forms of figurative language can be difficult for algorithms to detect [28]. Various solutions have been proposed to address class imbalance issues at both the data and algorithmic levels [184]. At the data level, these solutions involve different types of resampling, such as random oversampling, random undersampling, directed oversampling, directed undersampling, oversampling with informed generation, and their combinations [173], [251], [252]. At the algorithmic level, solutions include adjusting class costs, adjusting the probabilistic estimate at the tree leaf, adjusting the decision threshold, and recognition-based learning [253], [254]. While these techniques have shown promising results in improving the performance

of models on imbalanced datasets, they also have limitations. Oversampling and undersampling can lead to overfitting and underfitting, respectively, and may not work well when the dataset is extremely imbalanced [255]. Cost-sensitive learning requires accurate estimation of the misclassification costs, which may be difficult in practice [200]. These oversampling techniques are particularly effective in image data but may not perform optimally with textual content. Therefore, data augmentation methods, such as synonym replacement, back-translation, and text generation, have been introduced, primarily tailored to address these limitations in textual data. While employing back-translation techniques can enhance accuracy by balancing the dataset, it is crucial to acknowledge that this approach may still lead to a significant number of false detections, which can be prohibitively costly in real-world applications [220]. The occurrence of misclassifications highlights an additional challenge, which is closely tied to the existing evaluation metrics. The most frequently used evaluation metrics encompass accuracy, precision, recall, and the area under the ROC curve (AUCROC). These metrics effectively measure the model's performance on training and testing data but may not accurately reflect how the model will perform in real-world applications. In the work done by [29], they provided evidence that an adversary can make subtle changes to a highly toxic phrase, causing the system to assign a significantly lower toxicity score. Their experiment involved applying this method to the sample phrases provided on the Perspective website, consistently reducing the toxicity scores to the level of nontoxic phrases. This finding underscores the detrimental effect of adversarial examples on the usability of toxic content detection systems.

Another challenge to address is the requirement for high throughput, particularly in the context of toxicity detection due to the enormous data flow received every second. While throughput is a critical aspect of ML techniques, in general, it holds even greater significance in the realm of toxicity detection. In a study by [32], it was found that

although DL algorithms achieved high accuracy in detecting hate speech, they also had high latency and low throughput, making them unsuitable for real-time applications. This implies that the models we develop should not only be highly accurate, but also exceptionally fast, as users expect their posts and comments to appear for their followers within seconds. The most proposed techniques in the literature are tested in laboratory conditions on relatively small datasets and timescales, where processing speed is not an urgent consideration. To implement these models at a platform level, their performance must be characterized at large scales and in real-time. At present no/only one/few published articles engage with platform-level performance, and only one considers processing speed. Ensuring that toxicity detection techniques perform adequately at scale requires constructing measures of uncertainty along with the development of reliable performance evaluation metrics that fully consider uncertainty continues to be an ongoing concern.

3.6 Humans and Machine Learning: A Team Approach to Detection

Our exploration of the methods available to limit toxic speech online revealed that both ML techniques and human intervention are necessary through the process of data collection, annotation, analysis, and action.

While data collection processes are easily automated, human annotators are still vital in enabling any ML technique to operate on data. While automated toxicity detectors exist, our analysis and others have shown that these have very low reliability. A human annotator must interact with content to understand its meaning within a context and community. But while these human annotators are needed to enable this process, their jobs are not well remunerated, nor are they pleasant for the person who

must sift through toxic content. Once trained, several approaches can do a relatively effective job of identifying toxic posts.However, certain techniques are necessary to limit data bias that can inadvertently over-flag posts from marginalized communities. Human intervention is again needed to verify automated detection approaches and to make more permanent decisions such as content removal or user suspension (or to verify automated decisions on these topics).

Given the sheer volume of posts that are made within and across platforms, the consequences of FPs and FNs from automated detection methods seem unacceptably high. But so do the social and ethical implications of hiring human annotators.

What other solutions exist beyond these two poles? Individual users can take action to protect themselves from toxic speech. This can include settings to prevent inappropriate or offensive images from showing, as well as muting or blocking problematic posters. Groups of people also create block lists to create safe online experiences for communities with common values or identities. Some platforms also allow different levels of sharing for identified groups.

We then have three possible approaches: 1) individual protective measures, 2) human-based measures by the platform, and 3) automated approaches. All these approaches involve some level of tradeoff—time and effort for individual protective measures, harm and low pay for human annotators, and risks of bias and FPs and FNs for automated approaches. What can be done to mitigate these risks?

Improve Working Conditions for Annotators and Human Moderators

Gig-based approaches, such as those monetized through the Amazon Mechanical Turk program, have found a way to provide a technological version of piecework, a classic way to underpay workers by paying them a low rate for a unit of work, rather than

paying a fair wage per hour. Because online work can cross international boundaries, they can find workers willing to take poor wages to annotate toxic posts. But contract work plays other critical roles for social media platforms. Contract workers are the first line in content moderation and face serious mental health consequences due to the violent and disturbing content that they remove [256]. Workers are forced to sign nondisclosure agreements (NDAs) which prevent them from sharing the horrifying nature of their working conditions. Again, the global reach of the internet allows for platforms to hire contracting companies to pay workers in the global south less and provide them with fewer protections [257].

Develop Social Media Platform Cultures that Prioritize Care and Respect

Platforms, like other communities, have cultures. We recognize certain platforms for very high levels of toxicity (Gab, 4Chan, 8Chan, and TruthSocial), but online spaces also exist that prize kindness and good conduct. Reddit is an example of collectively managed communities that manage to enforce their own norms of behavior. One insight into these communities is that they may be connected through common interests, and, therefore, have stronger potential bonds and common values. For example, Wattpad [258], a networking site where authors can share work in progress, or Reddit subcommunities like r/fountainpens which bring together people with niche interests. At the same time, community-based platforms can clearly have other cultures and prioritize other types of behavior. However, these examples may provide some insights into how platforms can become kinder places.

Improve Algorithmic Approaches

LLMs have improved at an astonishing pace. While detection methods are improving, they are still beset by a number of challenges including misidentification, and difficulty in differentiating sentiment and toxicity in specific contexts. The ways to address these shortcomings are not immediately clear.

Hire Adequate Staff at The Platform Level and Treat Them Decently

Recent cuts across a swathe of social media companies have given rise to concerns that platforms will lack the resources to combat serious threats such as disinformation, false information, and online toxicity [259]. X, in particular, has seen a rapid rise in disinformation and toxicity [260], [261]. The cultures of the social media platform organizations themselves have influences on their employees. Just as poor working conditions and pay for gig workers who annotate posts for ML influence the quality of annotation and raise ethical concerns, the toxicity of the corporate culture at X under the new leadership influences the performance of workers employed by the platform. Just like anyone else, the human annotators, programmers, and staff tasked with removal and suspension decisions need safe and stable work environments.

3.7 Human Role in Machine Learning

A common narrative of artificial intelligence (AI) processes states that the human element can be removed from work, including decision-making as well as boring, repetitive tasks. However, this review of the process of online toxicity detection demonstrates how much this narrative leaves out. Toxicity detection begins with

dataset annotation that is typically carried out by human annotators (a boring, repetitive task). Although pretrained tools are employed in some instances, human analysis and intervention are often necessary due to the imperfect accuracy of these tools, reaffirming the central role of human annotation in the process. The ML tasks only begin once annotation is complete, when the classifier can be trained and tested on a small dataset.

These techniques are then applied to user-generated content to distinguish between toxic and nontoxic content. In some cases, potentially incorrect classifications are forwarded to human moderators for content removal decisions. In other words, a human worker retains decision-making in the moderation process. The classification results are of utmost importance, with both FPs and, in particular, FNs incurring significant costs. Both human annotators and the ML algorithms produce errors. Moreover, results from our lab and others indicate that ML techniques for annotation without a human user are prone to much higher error rates. This means that for the near future, human involvement remains integral to the moderation process to tackle toxicity. However, it is crucial to emphasize the importance of prioritizing human safety within the moderation framework. That means that for the near future, humans remain a critical part of the moderation process used to control online toxicity, but these humans need protection: for all the talk about AI safety, human safety should be a priority.

After a thorough review of the literature, we have identified three main problems that this thesis will address with proposed solutions:

Uncertainty in Practical Application of Toxicity Detection Techniques: While numerous techniques for detecting toxicity exist in the literature, it is often unclear how and when they add value to different tasks or how they perform in complex real-world settings. These algorithms are typically tested in simplified

experimental setups that do not accurately reflect the complexity of large-scale social media environments. This limitation makes it difficult for social media companies to predict the cost and effectiveness of these algorithms when deployed at scale in real-time. Consequently, there is a lack of information that hinders widespread adoption and places additional testing burdens on companies.

Moreover, algorithms have distinct features such as diverse formulations, computational times, and memory footprints. Their performance can vary across social media platforms with different toxicity levels, influenced by the user's network and reach. Algorithms can also exhibit biases, inadvertently flagging content from certain groups more often than others, leading to unfair treatment and reduced user trust.

To address these challenges, Chapter 4 presents a pioneering effort that prioritizes the creation of high-accuracy deep learning models with fast processing capabilities, managing a significant number of data points per second. This chapter introduces an innovative simulation framework to help social media companies strategically choose efficient classifiers across varying toxicity levels. This framework considers operational costs, advertising revenue, user satisfaction, and advertisement visibility. Recognizing the influence of toxicity detectors on user satisfaction and revenue, the study highlights the importance of considering more than just accuracy when selecting a toxicity classifier. User satisfaction and engagement are crucial elements that impact revenue generation. The framework also addresses scalability issues, ensuring solutions are effective and efficient even as the data volume grows.

Imbalanced Datasets in Toxicity Detection: The literature shows that imbalanced datasets can significantly impact the performance of classification tools, particularly in toxicity detection tasks. Addressing this imbalance is crucial for improving classifier performance. Existing methods predominantly utilize supervised approaches, which heavily depend on labeled datasets. This poses a challenge due to

the inherently imbalanced nature of toxic language datasets [262].

Hence, Chapter 5 aims to tackle the imbalance in toxic datasets to enhance classifier performance in toxicity detection by fine-tuning large language models (LLMs) using reinforcement learning from human feedback (RLHF) section 2.4.

Challenges in Current Moderation Systems: Current moderation systems employ a combination of human moderators and automated tools using machine learning algorithms. However, these systems often struggle to balance classification accuracy and speed, leading to frequent misclassifications. This increases the workload for human moderators and results in longer overall process times.

Therefore, Chapter 6 introduces an innovative deep reinforcement learning (DRL)-based framework for dynamically selecting classifiers in toxicity detection. The framework is designed to deliver both high-throughput and precise solutions, enabling rapid and accurate identification of toxic content amid the vast volumes of data exchanged on online platforms every second.

Chapter 4

A Profit-driven Simulation (PDS) Framework for Comparison of Deep Learning Models for Real-time Toxicity Detection in Social Media

This chapter is dedicated to the article entitled *"A Profit-driven Simulation (PDS) Framework for Comparison of Deep Learning Models for Real-time Toxicity Detection in Social Media"*. This article is under review by *ACM Transactions on Knowledge Discovery from Data*. The titles, figures, and mathematical formulations have been revised to keep the coherence through the manuscript.

The growing prevalence of toxic comments in social media has heightened the need for a safe and inclusive online environment, where the automatic and swift detection of such behavior is crucial for enhancing user satisfaction and engagement. Most of the existing toxicity detection techniques in the literature are mainly evaluated by performance metrics such as accuracy and F-score. However, key algorithmic aspects, such as throughput and the potential impact of false positives and false negatives on user engagement, are often overlooked. Moreover, the operational characteristics of these algorithms are typically tested in simple experimental setups, which may not accurately reflect the complexity of large-scale social media environments. These simple setups make it difficult for social media companies to predict the cost and effectiveness of these algorithms when running at scale in real-time environments. To address these drawbacks, this study presents a novel profit-driven simulation (PDS) framework to evaluate the effectiveness and efficiency of toxicity detection algorithms in complex social media environments. This PDS framework enables us to evaluate the real-time performance of toxicity classifiers by incorporating their effectiveness, computational costs, and user engagement level. We conduct comprehensive experiments to verify the effectiveness and efficiency of the proposed PDS framework, employing multiple Deep Learning (DL) algorithms including Convolutional- and Transformer-based. More precisely, two variants are developed for each classifier: one highly accurate but slower, and another with higher throughput but slightly less accurate. Our findings demonstrate that the highest throughput classifier is the most profitable in low- and high-toxic social media environments. On the contrary, the moderately accurate classifier with acceptable throughput shows superior performance in a medium-toxic environment. This finding highlights the significance of the proposed PDS framework and emphasizes that there is no universally suitable classifier for all toxic environments.

4.1 Problem Statement & Contributions

Antisocial behavior on social media platforms is a serious social issue, with unequal impacts across social groups [5], [6]. Toxic comments are a form of this behavior that forces users to abandon conversations and may include insults, threats, hate speech, and cyberbullying [7]–[9]. Fear of online abuse can lead to self-censorship and disengagement from social media platforms [11]. In 2021, major online platforms promised to improve their services to protect users, especially women, following widespread criticism that existing mechanisms for harassment protection are inadequate [12].

Detecting toxic behavior in a timely and automated manner is crucial to ensure a safe and inclusive online environment that fosters user satisfaction and engagement [263]. Over the past decade, there has been tremendous growth in social networking services, resulting in the creation of millions of data by platform users. On Twitter, for instance, there are 192 million daily active users with an average of 6,000 tweets per second, 350,000 tweets per minute, and 500 million tweets sent each day, while the number rises to more than 140,000 tweets per second during certain events (natural disasters, breaking news, etc.) [14]. The overwhelming flow of generated content creates a significant challenge in real-time dissemination, computing, processing, and analysis of the data. It is difficult to effectively process big data as a result of its variety, velocity, volume, and value [15]. Various Deep Learning (DL) techniques have been developed to counteract the spread of harmful content on social media platforms, enabling them to successfully identify toxic content. However, implementing these methods in real-time for every comment can be resource-intensive [17], [18]. The speed of a detection model is critical for social media applications since users expect real-time visibility of their content [16], [67]. Toxic content must be flagged and hidden promptly to prevent its spread. This time-sensitive process poses a challenge

for existing methods, which may not be optimal when delays occur. To measure the effectiveness of a detection model, its accuracy is typically the main focus rather than its real-time prediction speed, cost, or memory usage, although the ultimate objective is to identify and address toxic content before it reaches users [264], [265]. It is obvious that accurately flagging toxic content is crucial to prevent user churn, as wrongly labeling nontoxic content as toxic or vice versa can lead to user dissatisfaction. This highlights the importance of high-performing and accurate detection models [266].

Although a large volume of content is shared per second, most content is relatively harmless, nontoxic, and knowledge-based, leaving only a small fraction offensive, and represent attacks on other users, individuals, or minority groups [69], [70]. This differential in rates of toxic and nontoxic content complicates the task of flagging toxic content and increases the risk of false positives and false negatives. Moreover, the most common method of identifying toxicity focuses on specific content terms within exchanges, but this approach may not be sufficient to determine if the content is truly toxic [267]. An example of this is how teenagers in North America use language among themselves that may be considered toxic by adults [18]. Furthermore, when addressing toxic content and toxic users on social media, it is crucial to acknowledge the significant variation in responsiveness based on users' follower and following counts. Some users consistently post content and receive numerous comments per second, while others rarely post or comment [268]. Hence, the diversity observed in social media content and users underscores the limitation of relying exclusively on a single classifier, particularly when its evaluation and selection are based solely on accuracy, to effectively address all cases on a platform. Consequently, the large-scale and dynamic nature of social media platforms necessitates techniques capable of both effective and efficient analysis of high-throughput streams of unstructured content in real-time [269]. However, identifying the most suitable detector for different scenarios poses a

significant challenge.

This chapter proposes a novel approach to determining the most suitable model for detecting toxic content, offering the following contributions:

- A novel profit-driven simulation (PDS) framework is proposed to evaluate the effectiveness and efficiency of toxicity detection algorithms across different toxicity levels in social media environments.
- The proposed PDS framework can identify the most profitable detector that can accurately detect toxic content, at the lowest cost, within a reasonable time, and the highest level of user engagement under different possible circumstances.
- A profit model tied to this simulation approach is designed to identify the most lucrative toxicity classifier for various environments on social media platforms.
- This framework considers user satisfaction as a critical factor when selecting the optimal detector for different environments under realistic scenarios. It highlights the significance of false positive and false negative predictions, which lead to user churn.

To the best of my knowledge, our approach is the first to reveal that evaluation metrics such as accuracy and F score are inadequate to effectively and efficiently identify toxic content on social media. Additional considerations encompass computational cost, real-time data processing at scale, and the influence of detector performance on user engagement must also be taken into account. Additionally, this chapter categorizes toxicity into three distinct levels based on social media characteristics, and the PDS framework can adeptly choose the most suitable detector for each of these levels.

A brief preview of what will be covered in the following sections can be found below: Section 4.2 presents a comprehensive overview of our methodology, while section 4.3

delves into the specifics of the experimental setup and explains how various toxicity classifiers were fine-tuned and chosen to function as a detector in the PDS framework. The results of the PDS framework are presented in section 4.4. This chapter concludes with a summary of our contributions in section 4.5.

4.2 Methodology

This section delves into the intricacies of our POS framework, providing insights into how it was developed. Initially, we highlight common features shared by most social media platforms in Equation 4.2.3.2. Following that, we detail the simulation process in subsection 4.2.3, explaining how we simulated various social media components and assessed the impact of different classifiers on users and overall profit.

To illustrate the structure and features of the PDS framework, we begin by offering an overview of the input graph used in the simulation, along with how user behavior is defined and controlled throughout the simulation (explained in subsubsection 4.2.3.1). Subsequently, in subsubsection 4.2.3.2, we provide a comprehensive explanation of how the simulation operates, covering the calculation of costs, revenue, and profit.

Finally, in subsubsection 4.2.3.3, we subject the simulation to null cases to validate its accuracy before employing it for experimentation.

4.2.1 Generic Social Media Model (GSMM)

We will briefly cover some of the common features shared by most social media platforms (see Figure 3). The primary feature is the **user** who can create and share information on the platform. After creating an account, users can set up a profile, and they are then considered part of the platform's user base. Users can choose to leave the platform at any time and return later. In addition, they have the option to

follow or unfollow other users, with each user having a list of followers and followings. The list of followers shows users who follow the user, while the following list displays users that the user follows.

A **post**, also known as user-generated content, can be in the form of text (unstructured), visual, or audio. Users can view posts generated by other users in their following list and interact with them by reacting or replying. Social media platforms offer various ways for users to react, such as commenting to express opinions, offer praise, disagree, ask questions, and participate in online conversations about social content. Credibility is another significant feature of social media, where users can indicate their agreement or disagreement with a post using features such as "Likes" on Facebook or "Retweets" on Twitter, which also facilitate the dissemination of information.

On a page called the **front page**, users can find a list of the most recent posts from their following list, which are available for a specified period. Meanwhile, the **trending page** showcases the most popular posts determined by a generic algorithm that takes into account factors such as the time since the post was made and the number of comments or likes it has received.

The maintenance of social media platforms greatly relies on **advertisement** revenue as their primary funding source [270]. For example, Google discloses an annual advertising revenue of $50.57 billion, which represents 91% of their yearly earnings. Furthermore, according to a report by Statista [1], advertising constituted 98.3% of Facebook's revenue and 84% of Twitter's revenue in the first quarter of 2021.

Taking these general features into account, we simulate the social media environment. Further details on the development and simulation process are outlined in subsection 4.2.3. It is important to note that, while we utilize these generic features

[1] https://www.statista.com/statistics/279456/share-of-revenue-of-facebook-by-source/

in our study, they may vary for different platforms.

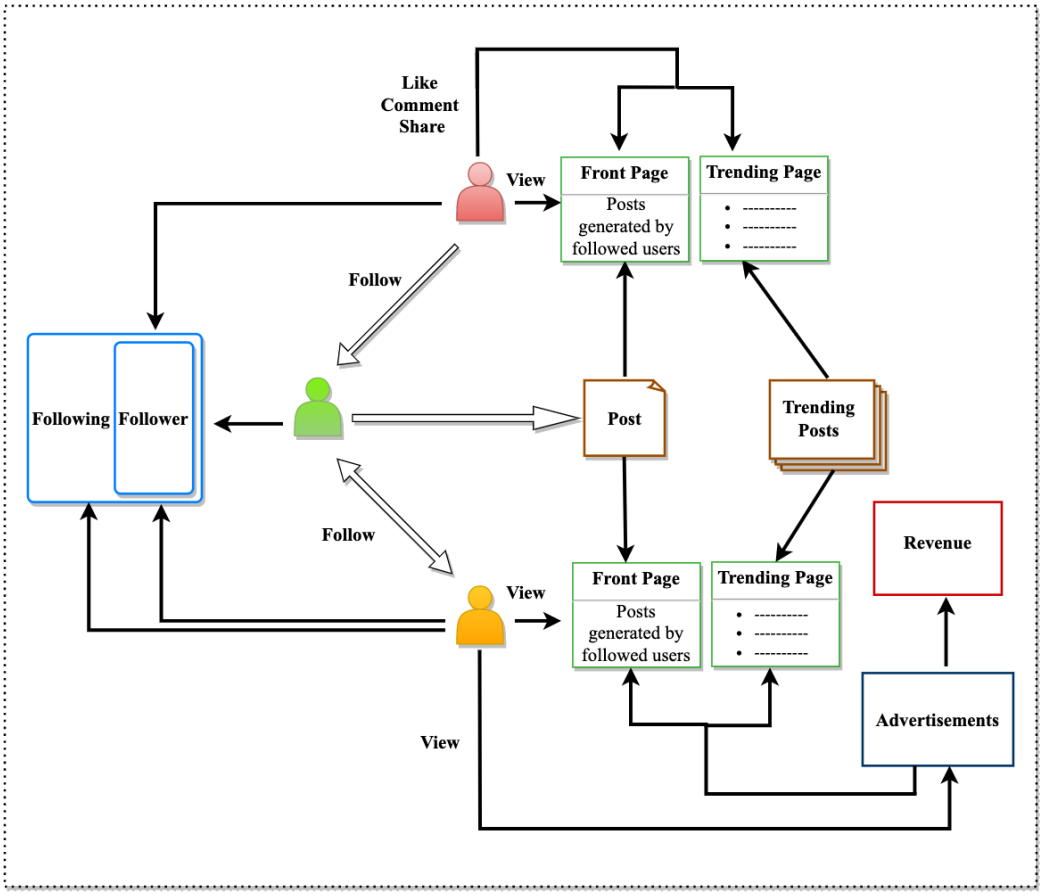


Figure 3: The architecture of social media

4.2.2 Environments

As discussed in subsection 3.1.1, it is evident that social media is not uniformly toxic. Numerous factors play a direct role in influencing users' encounters with or the creation of toxic content.

Initially, the likelihood of users engaging or receiving toxic content hinges on their network affiliation and follower/following ratio. Within the realm of social media, symmetric connections, where users mutually follow each other and often share mutual friends, are commonplace, termed as friendship networks. In this context, the

likelihood of encountering toxic content is notably low, although not completely absent. Essentially, toxic interactions are less probable between individuals who share direct connections such as friendships, family ties, work associations, and more. However, such instances can still arise due to differing viewpoints, competition, and aspirations for status.

Another essential aspect to consider is the reach rate, which measures the number of users who come across content from an account. Some users receive many comments, reactions, or reposts, which brings their posts to the attention of a wide audience [268]. For instance, this is often the case with celebrities, influencers, and public figures, commonly referred to as "high-reach" users in this chapter, who have substantial reaching rates from both followers and non-followers. Whether these users encounter hurtful comments depends on their expertise and their audience. Influencers, activists, and politicians, for example, are more likely to face such negativity. Also, users who post a lot, especially those who get into arguments, are more likely to be attacked or criticized. As the reach rates increase, these users become more vulnerable to encountering toxic content, even if they themselves generate less toxic content. It is exceptionally uncommon for high-reach users to respond negatively to comments from their followers

Moreover, for high-reach users, the balance between the number of their followers and the limited number of accounts they follow is another crucial aspect in addition to the reaching rate. These users often have a significantly larger number of followers compared to the few accounts they follow. This discrepancy increases the chances of encountering toxic comments. In a broader context, interactions that involve users who lack any previous connection, indicating that they do not follow each other, tend to be more toxic. The concept of reaching rates is also applicable to the content they share. High-reach users' posts, along with public content and trending pages, typically

reach a larger audience compared to personal posts from users with fewer followers or private accounts. As a result, both creators of widely seen posts and users who frequently comment face an increased risk of encountering toxic content. In summary, environments characterized by frequent highly visible posts often exhibit a higher frequency of toxic occurrences.

Hence, the varied nature of toxicity within social media underscores the necessity for an appropriate toxic content detector that effectively aligns with this diversity. A single detector cannot reliably perform optimally in different environments. Additionally, the implementation of classifiers customized for each specific environment has significant potential for profitability for social media companies. Subsequent paragraphs will delve into this hypothesis with more comprehensive details.

Consequently, we define three levels of social media toxicity: **low**, **medium**, and **high**, determined by factors such as the user network and the reach rate. Within these toxicity tiers, we will evaluate the effectiveness of the detectors in diverse settings.

4.2.3 Social Media Profit Simulation

4.2.3.1 Input Graph and User Behavior

Our consideration involves a group of users connected to each other, with their own list of followers and followings. Users can create posts visible to their followers, and interact with posts by reading, liking, and commenting. We focus only on users who do not want to be exposed to toxic content, although some users may view social media as a platform for self-expression and do not mind encountering toxic content.

User engagement on online platforms can affect the likelihood of encountering harmful content, and personal attacks may lead to decreased activity or platform abandonment [16]. Inaccurate toxicity detection systems can result in users receiving multiple toxic messages (false negatives) and leaving the platform to avoid further

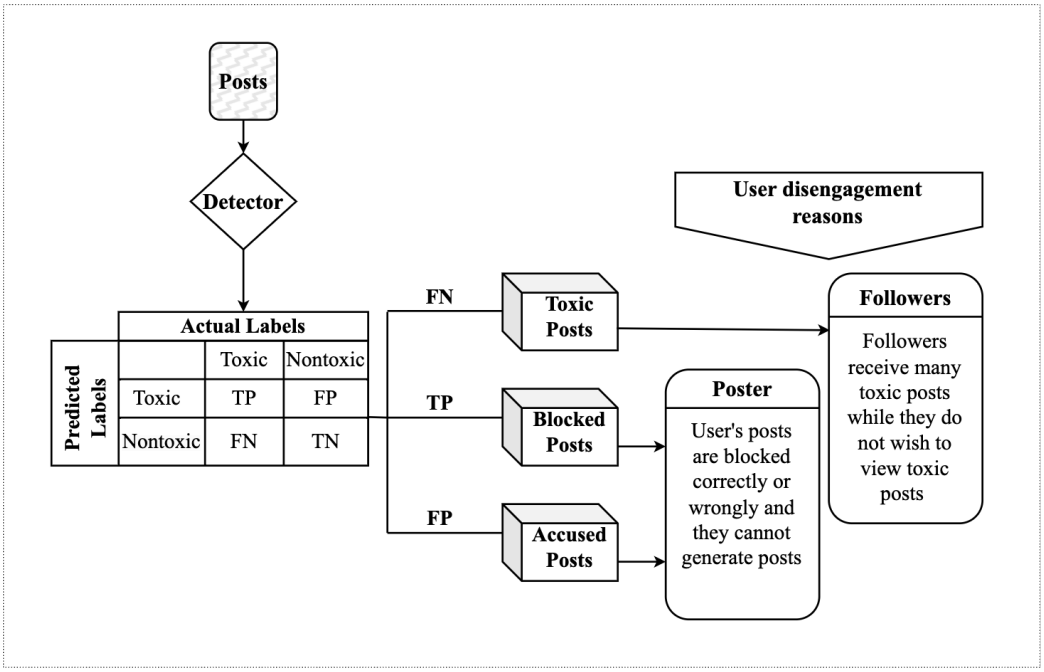


Figure 4: Different user disengagement reasons

exposure.

Disengagement from a platform can occur due to various reasons beyond user sensitivity to harmful content. For instance, frequent post-flagging as toxic can lead to disengagement, particularly when false positives are produced by the classifier. While accurate blocking (true positive) may not significantly affect user engagement, the effect of false positives can be significant. Classifier performance typically has a significant impact on user engagement, with a well-functioning classifier potentially increasing engagement. Simulation frameworks can be used to compare the impact of different toxicity classifiers under various conditions and evaluation metrics. The PDS framework assesses user behavior according to three factors that can cause dissatisfaction with their social media experience (see Figure 4). It is important to note that there is a pre-established limit for the number of posts received for different reasons, and exceeding this limit may lead to user disengagement and platform abandonment. The reasons are as follows:

Toxic posts: The simulation tracks the number of visible toxic posts (false negative) received by users. If a user is exposed to too many toxic posts, they may become disinterested in the platform and leave, resulting in zero revenue for the company. User engagement is measured by the amount of time they spend reading content each day, and a decrease in reading time may be a result of the platform's toxic content.

Blocked posts: The simulation records the number of blocked posts (true positive) for each user due to the toxicity detector's evaluation. If the detector is too stringent in its assessment, it may block several posts that are not highly toxic, or if it is biased in its classification of toxic content.

Accused Posts: The simulation records and keeps track of the number of false positives for each user, where nontoxic posts are wrongly detected as toxic.

4.2.3.2 How the POS Framework works

The simulation system incorporates multiple inputs, such as the user graph, pre-trained classifiers, and previously calculated predictions. As mentioned earlier, the user graph comprises users, their followers, followings, and different sensitivity levels.

In order to simulate the real-time system used by social media, the PDS framework involves the concurrent operation of three distinct threads, as illustrated in Figure 5. Further elaboration on each of these threads is provided below.

Reader Thread: The simulation involves the replication of user activities such as reading, commenting, and liking posts. To accomplish this, we have defined three distinct states namely **Waiting**, **Read Session**, and **Read Post**. Initially, all users are in the waiting state. They are subsequently assigned to their unique reading session using a designated function. Each reading session contains all the new posts

available on the front page and the trending page, which are specific to each user. The front page displays all posts created by the user's followings, while the trending page features popular posts whose creators are not on the reader's followings list. Notably, the trending page is dynamic and changes at the end of the simulated day. In order to guarantee that all posts are available and not concealed, snapshots of the front and trending pages are captured when a user enters the reading session. These snapshots are later inspected to identify any posts that have been removed. Once a toxicity detector is implemented to distinguish between toxic and nontoxic posts, some posts may be hidden from the user based on the detector's classification.

We made the assumption that users would first read all unread posts from the people they follow before moving on to unread posts from the trending page, if time permits. During the Reading session, the time spent by users is recorded. Reading sessions are scheduled for a certain amount of time, and some users may read in many small sessions throughout the day, while others may read in fewer or larger chunks. If a user reads a post, they are moved to the next state, Reading post, and then return to the reading session to read the next post. The time spent in the Reading post state is also recorded. Users are permitted to follow or unfollow the creators of posts they are reading. If a reader decides to follow the creator of an additional post while reading it, their list of followings will expand during the simulation. As a result, the number of posts that they can read will increase, potentially leading to a decrease in the number of readers leaving their reading session early. Users can also like and/or comment on each post during the reading session. A user's engagement increases when they like or comment on a post, while reading toxic content decreases their engagement. When the allotted time for the reading session is over, the user returns to the waiting state until they decide to check for updates and view generated content again.

For the creation of the trending page, we take into account the number of likes

(α) and the number of comments (β) received by each public post, along with the time it takes (γ) to accumulate those likes and comments. This information is used to compute the post's score using the formula:

$$\textbf{Score} = \alpha w_1 + \beta w_2 - \gamma w_3 \tag{2}$$

where $w_1, w_2, w_3 > 0$ and $w_1 + w_2 + w_3 = 1$.

By ranking posts (created within a simulated time) in descending order based on their scores, we select the top 100 posts as trending posts for updating the trending page. The trending page is refreshed every simulated day and prior to users entering their reading session, the system checks if an update to the trending page is needed.

Poster Thread: It simulates how users create new content (toxic and nontoxic). It only includes the Waiting state, where all users are present and can decide whether or not to post. A time will be assigned for each user. Based on their unique probability of generating toxic posts provided by the user graph, a Normal distribution (Gaussian distribution) determines whether the user creates a toxic/nontoxic post. If the user decides to generate a post, then a post will be created. Moreover, the system keeps track of the time per user for generating a post, indicating when each post is generated. This means that the time interval between each post generated by a user can be determined. It should be noted that prior to receiving newly generated posts from the Poster Thread, certain posts have already been forwarded to the Reader Thread, providing users with some content to peruse during the simulation.

Detection Thread: which simulates detecting and hiding toxic posts in real-time. This thread includes two states, Waiting and Processing. A detector is first in the waiting state until it receives posts as input for classifying. Note that, all three threads

happen simultaneously, then the number of available posts for processing is different every time. Additionally, we considered different batch sizes for detectors to take input for classification, which resulted in different processing times.

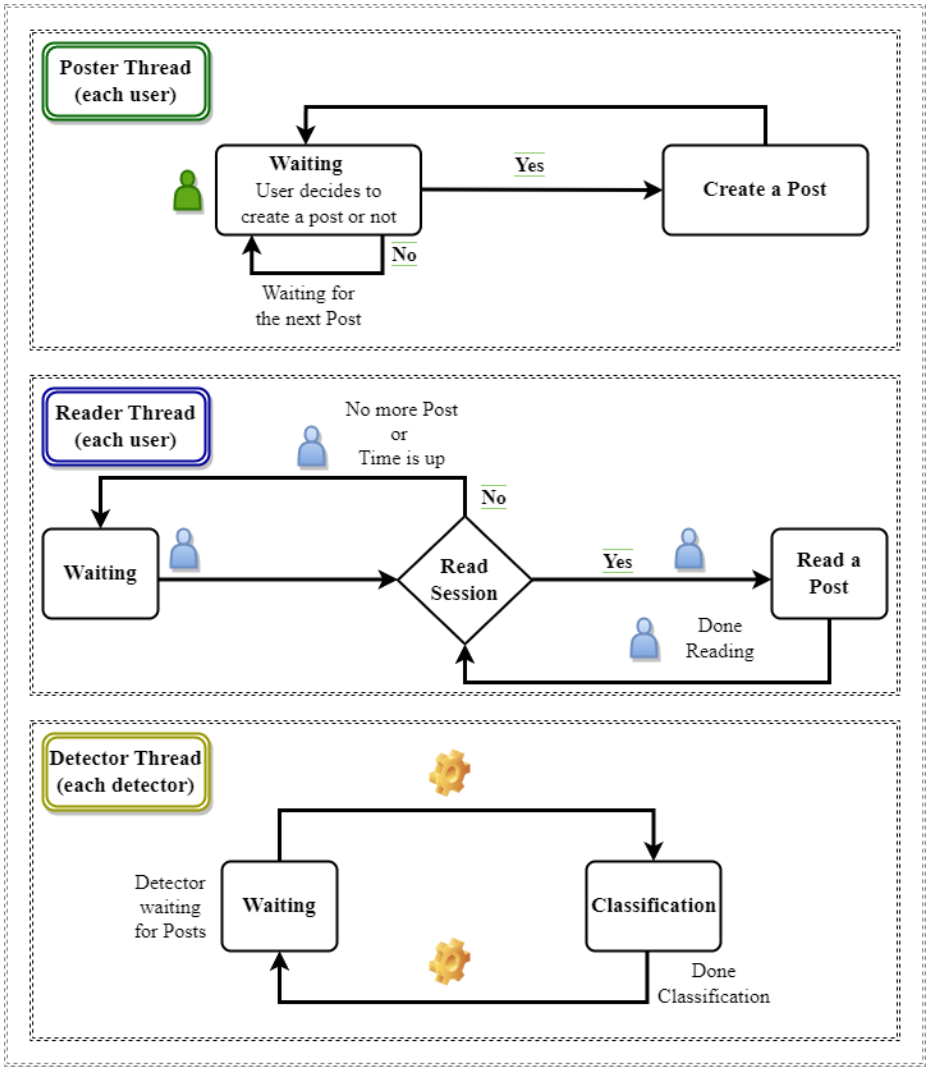


Figure 5: Architecture for the POS Framework

In terms of time, the PDS framework runs in a squeezed time, where we can simulate a long-running simulation within a short span of time, while still operating in a real-time system. In addition, precalculation is done within the simulation to speed up the system so that we can perform long-running simulations.

Social media toxicity is influenced by the user's network and reaching rate. Toxic

content is less likely in friendship networks but still possible. Users with high-reaching rates are more likely to receive toxic comments, especially on high-reaching posts. Therefore, we have categorized toxicity into three levels: low, medium, and high, and we will evaluate the detectors' efficacy in different environments across these three levels of toxicity.

Finally, the outputs of the simulation include **profit**, **revenue** from user engagement, **costs** of the machine for using classifiers, the number of users who did not leave the platform, the number of uncreated posts (because the user left the platform), the **confusion matrix** (error matrix) of the detection results, the total number of posts created by users, and the total number of blocked posts because of toxicity. Note that a confusion matrix is used to summarize the performance of classifiers. This method is useful for measuring Precision, Recall, Accuracy, F1-score, and AUC-ROC curves which will be explained later in Section 4.3.2.

Cost: We precompute the actual time that a detector takes to do the batch computation and then using a linear transform between the two estimates the amount of computation time will be calculated. To calculate the cost of a classification task using instances, we first determine the total cost per second for an instance used to run the batch classification. Then, the total cost for each classifier is calculated by multiplying the cost per second of the instance used for batch classification and the total detection time. In this context, $\mathbf{W}_i$ represents the cost incurred by instance **i** when processing and classifying samples within a given time frame. Similarly, $\mathbf{T}_{x,i}$ denotes the total time taken by classifier **x** running on instance **i** to complete the classification of **S** samples. Therefore, the total cost for classifying n samples using instance (i) per classifier (x) is given by the following formula:

$$\mathbf{T}_x(\mathbf{t},\ \mathbf{s}) = \sum_{s \in S} t_s \tag{3}$$

- $\mathbf{T}_{x,i}$**(t, s)** is the total time for processing S samples by classifier x.
- $\mathbf{t}_s$: time required for classifying sample s.

therefore the total cost is equal to:

$$\mathbf{C(x,\ i)} = T \times W \tag{4}$$

Revenue: As mentioned earlier in , advertising is the primary source of revenue for social media companies. Consider that every post includes an advertisement, which generates revenue per view when users view it. By acknowledging that every post incorporates an advertisement, revenue is generated in accordance with the views it accumulates from users. In other words, each view encompasses both content and ads, and a higher number of views results in more revenue. This is due to the platform receiving payments for showcasing advertisements to its users, effectively leading to increased revenue.

Assume A_v represents the revenue earned per advertisement's view.

Let u $\in U_m$ is a user on platform **m**, and **N(u, m)** shows the total number of views by user **u** on the platform **m**. Then

$$\mathbf{V} = \sum_{u \in U} N(u, m) \tag{5}$$

V represents the total number of views on platform **m** that have been carried out by **U** users. Thus, the total revenue generated from viewing posts is given by:

$$\mathbf{R} = A \times V \tag{6}$$

Profit: Profit can be calculated by subtracting the total cost from the revenue generated.

$$\mathbf{P}_{x,i} = R - C \tag{7}$$

$\mathbf{P}_{x,i}$ represents the profit generated by classifier x, while the **R** and **C** determine the revenue and cost associated with the application of classifier **x** using instance **i**, respectively.

4.2.3.3 Null Case Verification

The main goal of this study is to identify efficient techniques for addressing online toxicity and increasing user engagement. Thus, using the PDS framework we will test different scenarios to find the best solution per case. To ensure that the PDS framework meets expectations, we first tested several null cases. All null classifiers have zero computation costs, and all machines are the same type. The key parameters for simulating null cases are presented in Table 2.

Perfect Detector (PD): Always returns the correct detection for Toxic/nontoxic.

Always Toxic Detector (AT): Returns the toxic for all detections.

Always nontoxic Detector (AN): Returns the nontoxic for all detections.

Random Detector (RD): Equally random chance of returning toxic/nontoxic for all detections.

The Perfect Detector (PD) is the most ideal detector. It is unlikely that a user will leave this platform because of toxic or accused posts. Nevertheless, blocking posts may cause some users to leave the platform. It is important to note that revenue is derived from views of posts, but not from hidden posts. Therefore, a perfect detector does not generate more revenue since toxic posts will be blocked and no one will see them.

The Always Toxic Detector (AT) is the worst detector. Due to the fact that no post will be visible to users, it will generate zero profit in any environment. Users will

eventually leave the platform.

Always nontoxic Detector (AN) does not hide any toxic posts, then users who are sensitive to toxic posts will leave the platform. It generates the most revenue because all posts even toxic ones will be shown to users. The revenue decreases over time due to users leaving the platform because of toxic posts. The worst case happens in a highly-toxic environment, and the highest revenue will be generated in a slightly-toxic environment.

Random Detector (RD) fails to function accurately. The posts are incorrectly classified as toxic or nontoxic in this case. It is possible that some users will leave the platform because of posts that are toxic but the classifier failed to detect them; or because of posts that are falsely detected and blocked as toxic; or due to blocked posts. The revenue generated by RD varies depending on the environment.

The simulation was run with a shrink factor of 100 $\times$, enabling it to model a week of activity over the course of 6048 seconds, while incorporating 10,000 users during the process. The use of a shrink factor in the simulation is a common technique used to speed up the execution time of simulations. By reducing the scale of the simulation, it is possible to run it more quickly while still maintaining the overall accuracy of the results. In this case, the shrink factor of 100 allowed the PDS framework to execute much more quickly than it would have if it had run at full scale, while still representing a full week of activity. The four detectors were evaluated across environments with varying levels of toxicity: from low toxicity, indicating approximately 20%, to medium toxicity encompassing around 50%, and highly toxic environments comprising over 80%. In this case, zero cost is assumed, resulting in the profit being equal to the revenue. As discussed, revenue is also generated based on the number of views by users. To determine the revenue earned per view of an advertisement, **Google**

Table 2: PDS Framework Parameters for Null Cases Verification

Parameter	Value
Shrink Factor	100 ×
Simulation Duration	604800 sec
Users	10,000
Ad_revenue	$0.0086
Cost	0
Low Toxic ratio	∼20%
Medium Toxic ratio	∼50%
High Toxic ratio	∼80%
Machine Type	Paperspace P4000

AdSense [2] was utilized. Selecting the **North America** region and the **People and Society** content category would yield a revenue of $0.0086 per view, according to Google AdSense. This projection suggests that 50,000 monthly views would generate an annual revenue of $5,202. So all three threads (Poster, Reader, Detector) will begin simultaneously. This results in some posts that are not analyzed by the detector and are displayed to users as is. Table 3 shows the profit generated by each detector in different environments.

The simulation results validate our null hypotheses across multiple environments and confirm the reliability of the simulation. Apparently, PD offers the best performance and profitability regardless of the environment. AT, on the other hand, made no profit because all posts were treated as toxic content and were hidden from users, so they couldn't be seen or read by other users. In all environments, AN outperforms RD.

Figure 6 provides an overview of user fall over the simulation time. It is evident that across all environments, the PD classifier demonstrates the lowest user churn rate when compared to other classifiers. Specifically, PD exhibits the least user attrition in the low-toxic environment, while experiencing the higher user churn in medium- and

[2] https://adsense.google.com/

Table 3: Profit Variation Across Different Environments in Null Cases

Environment	Detector	Profit ($)
Low Toxic	PD	381,990
	RD	40,313
	AT	0
	AN	83,853
Medium Toxic	PD	214,315
	RD	13,568
	AT	0
	AN	23,044
High Toxic	PD	146,164
	RD	10,209
	AT	0
	AN	15,094

high-toxic environments. Moreover, AT and PD experience a sudden rise in user fall at the beginning of the simulation, which then gradually increases. Similarly, RD and AN show a notable number of users leaving, with a major churn of more than 7500 users taking place within the initial 2000 seconds, followed by a gradual increase.

According to Figure 7 which shows the number of users who left the platform per second for various reasons, when PD is applied in all environments, users leave the platform only because they get many posts blocked because of toxicity, not because they receive toxic content or their posts are incorrectly blocked. AN, however, causes many users to leave even in low-toxic environments due to no control over shared content and toxicity. When AN is used, fewer users leave the low toxic environment compared to medium/high toxic environments.

It was evident that AT blocked all posts, so users eventually left the platform after their posts were blocked incorrectly or correctly and there were no posts for followers to read. Additionally, more users left because of wrongly blocked posts in higher and medium-toxic environments than in a low-toxic environment. As toxicity increases (medium and high-toxic), the number of users who leave due to blocked posts by AT

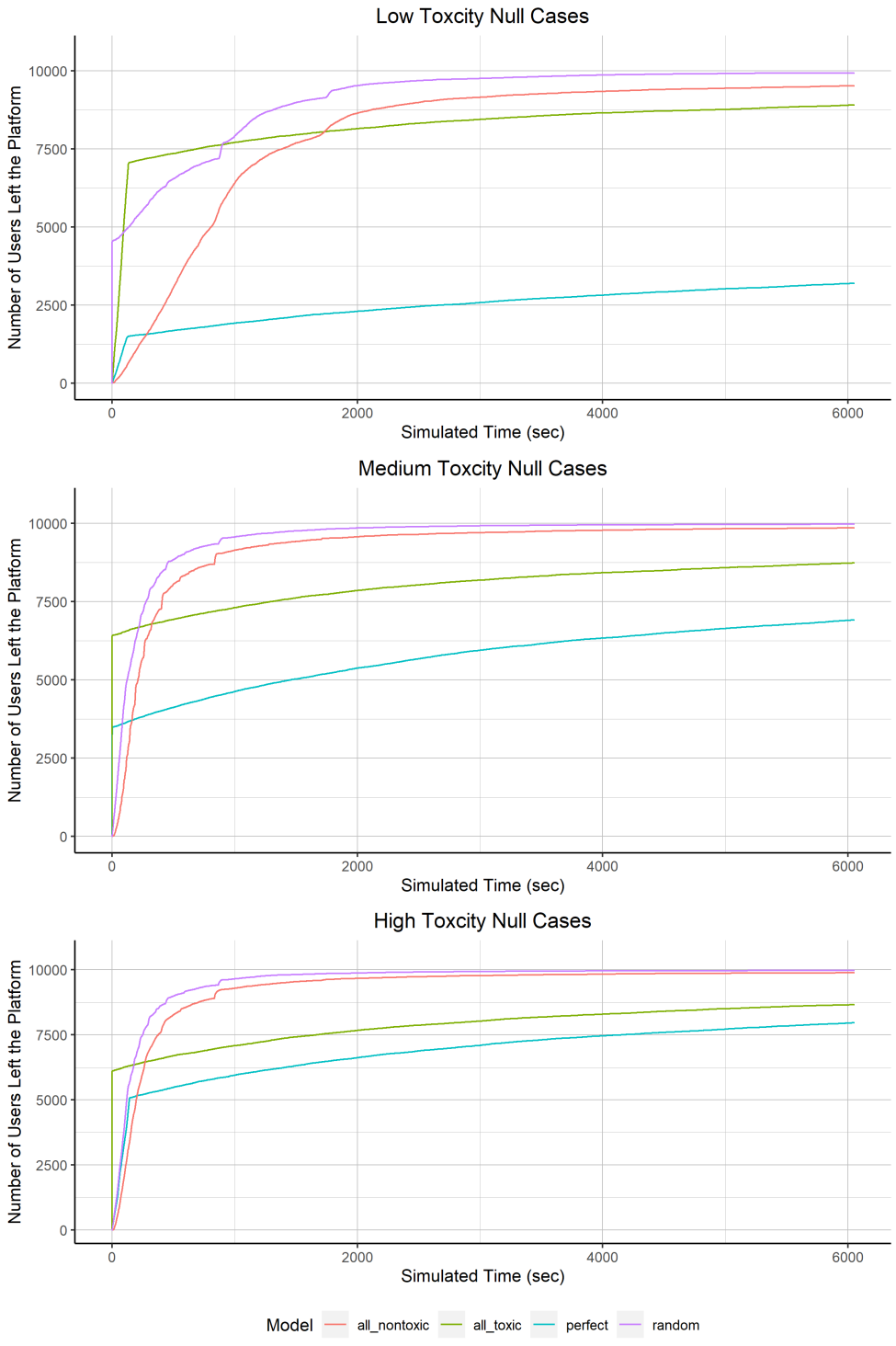


Figure 6: User churn over the simulation period for Null Cases

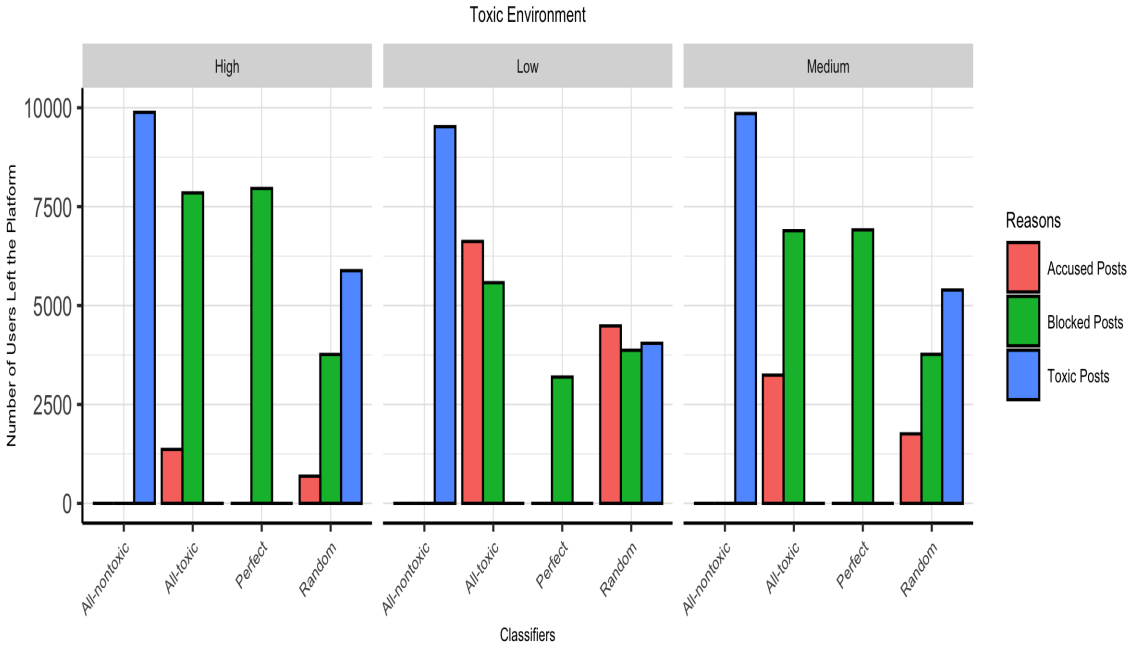


Figure 7: The number of users left for different reasons based on the classifier used

increases, while the number of users who leave due to wrongly blocked posts decreases. RD results in users leaving for all three reasons in all environments. In addition, a higher level of toxicity leads to a larger number of left users who don't wish to see toxic content.

Users were tracked throughout the simulation and both their departure and reasons were recorded. PD clearly outperforms other approaches, and the line chart shows that it gradually increases over simulation time, even if it starts out sharply. Initially, AT loses the most users, but their numbers gradually increase over time. In both RD and AN, the maximum number of left users has finally been reached.

In general, the simulation outcomes illustrate that diverse classifiers' effectiveness and efficiency can be evaluated in a range of settings, and user engagement can be tracked throughout the simulation process.

4.3 Experimental Setup

This section describes our experiment setup, including the toxic dataset for toxicity detection, the graph dataset for our PDS framework, and the classifiers used in the detection thread. We also explain the improvements made to the classifiers for higher throughput and provide details on the computing resources and tools utilized in our experiments.

4.3.1 Data Sets

4.3.1.1 Toxic text data set

We utilized a dataset from the Google Jigsaw team [271] to train our detectors. This dataset comprises 159,571 Wikipedia comments that have been human-rated for toxicity across six categories: toxic, severe toxic, insult, obscene, threat, and identity hate. Additionally, the dataset contains comments labeled as nontoxic. For classifier development, the dataset was randomly split into three sets: a training set (70%), a validation set (20%), and a test set (10%).

4.3.1.2 Graph data set

To better reflect the typical behavior of social media users, a dataset consisting of 10,000 users was generated, with consideration given to various follower/following ratios. The follower/following ratio, which is calculated by dividing the number of users following an individual (followers) by the number of users they follow (followings), can provide insight into the type of user on social media platforms [272]. Normal users typically have an equal ratio of followers to followings, while celebrities or influencers have a higher number of followers and a lower number of followings, resulting in a high follower/following ratio. New or less active users tend to have a lower number of

followers and a higher number of followings. Fake accounts often have few followers, while normal users tend to have identifiable followers such as family and friends. In addition, profiles can be public or private, allowing users to choose who sees their content. Popular content can be found on the trending page.

To ensure user engagement and retention, each user in the experiment was assigned maximum sensitivity levels based on various factors. For instance, a toxic content threshold was assigned to each user using a normal distribution (Gaussian distribution) to represent their sensitivity to toxic content. Additionally, random numbers were generated for each user's threshold to account for scenarios such as posts being blocked correctly or incorrectly. Each user also had a unique probability for liking, commenting, following, unfollowing, and creating new posts.

4.3.2 Detectors

To detect toxic text, DL models are considered as viable options. However, their effectiveness, efficiency, and ability to handle large volumes of data must be balanced for use in social media platforms. To this end, we developed multiple DL models to be used in the "Detection Thread" of the PDS framework. We can finally identify the most profitable detector for real-time toxicity detection across social media environments.

We use Accuracy, AUC-ROC, F1-score, and Throughput as metrics to evaluate model performance. Using multiple metrics, we can evaluate various aspects of the models' strengths and weaknesses and choose the best one based on our evaluation. This ensures that our selected models are not only accurate but also efficient in terms of processing time.

Accuracy: It measures the proportion of correctly classified samples and is a crucial indicator of a model's performance, calculated from the confusion matrix.

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

AUC-ROC: It measures the model's ability to rank positive and negative samples and is helpful in imbalanced datasets with fewer positive samples [273]. The formula for AUC can be expressed as the area under the ROC curve (receiver operating characteristic curve):

$$\textbf{AUC} = \int_0^1 TPR(FPR^{-1}(t))dt \tag{9}$$

where TPR is the true positive rate (sensitivity) and FPR is the false positive rate (1-specificity), both of which range between 0 and 1. FPR^{-1}(t) represents the threshold value that corresponds to a given false positive rate t [273].

F1-score: It assesses test accuracy using precision and recall. Precision is the ratio of true positive results to all positive results (including misidentified ones), while recall is the ratio of true positive results to all actual positive samples.

$$\textbf{F1-score} = 2 * (precision * recall)/(precision + recall) \tag{10}$$

Throughput: It is the number of samples processed per unit of time, and is important for applications that require fast predictions. We calculated the throughput per classifier in terms of how many samples can be processed and classified in a second, using only the test set data.

$$\textbf{Throughput} = \frac{N}{t} \tag{11}$$

where N is the total number of samples processed and t is the time taken to process the samples.

This study involves developing eight candidate models for social media toxicity

detection, categorized into two sets: CNN- and Transformer-based. The objective is to select the most accurate and the fastest (higher throughput) model variants in each category. For CNN-based, both CNN and CNN-fastText model variants are considered, with modifications made to improve throughput. For Transformer-based, variants of BERT-based and RoBERTa-based models are tested, including distilled and BERT miniatures to improve performance. The chosen models will be assessed within a PDS framework to determine their efficiency and effectiveness in identifying toxic contents on social media. The goal is to identify the most profitable model for this task. Details on developing high-accuracy models can be found in sections 4.3.2.1.1 and 4.3.2.1.2, while sections 4.3.2.2.1 and 4.3.2.2.2 provide information on enhancing throughput.

4.3.2.1 Detectors selected for best accuracy

4.3.2.1.1 ***CNN-based***

The input to the CNN model undergoes data cleaning and preprocessing, which involves removing URLs, punctuations, digits, stop-words, and normalizing cases, acronyms, and abbreviations. Tokenization is performed, and the text comments are truncated or padded to a fixed length before being converted to vector words. The model comprises four 1D convolutional layers with multiple window sizes and output channels. A pooling layer is utilized to reduce the dimensionality of the convolutions, followed by a global max pooling to further reduce the dimension. The hidden layers of the model employ ReLU activation function, while the output layer uses Sigmoid activation function. During training, binary cross entropy is used as the loss function to measure the difference between predicted and actual labels. Additionally, Adam optimizer is employed to update the model's parameters during training, which helps

Table 4: Classification Experiment Results and Parameters for CNN-based Models with the Best Accuracy

Model	Epoch	Batch size	Learning rate	Dropout rate	Filters	Accuracy	AUC	F1-score	Throughput (s)
CNN	30	128	2.00E-05	0.1	128	0.92	0.70	0.72	231.9
CNN fastText Crawl	3	256	0.002	0.4	300	0.92	0.77	0.75	384.9

to speed up convergence and improve accuracy.

We incorporated pre-trained fastText word embeddings into our CNN model to accelerate the training process (CNN-fastText). Our experiments employed the "crawl-300d-2M" embeddings, which consist of 2 million word vectors trained on Common Crawl using a continuous bag of words (CBOW) with position weights, character n-grams of length 5, and a window of size 5 and 10 negatives. The CNN model convolves over the embedded vectors, which are then passed through a max pooling layer to reduce the dimensionality. The final classification is performed using fully connected layers with a Sigmoid activation function. The use of pre-trained word embeddings helped to significantly reduce the training time of our CNN model. The hyperparameter tuning for these models was performed using a random search.

The experimental results for the most accurate CNN and CNN-fastText models are shown in Table 4.

According to the results, the CNN model with 30 epochs, 128 batches, 2.00E-05 learning rate, and 0.1 dropout rate achieved the highest accuracy (92%), and F1-score (72.33%) followed closely by the CNN model with fastText embedding, which had a higher AUC, F1-score and throughput. Please note that, while these models are accurate, their throughputs are currently inadequate for effective application in social media scenarios. Hence, we are actively exploring ways to enhance their throughput without compromising their accuracy.

4.3.2.1.2 ***Transformer-based***

We utilized the Huggingface transformer library, which is compatible with Tensorflow 2.4.1. To detect toxicity, we tested various versions of BERT such as BERT-base, AlBERT, Distilbert, RoBERTa-base, RoBERTa-large, and DistilRoBERTa. To prepare text data for BERT-based models, we used BERT tokenizer, which transforms raw text into BERT-compliant tokens. A pre-trained BERT model has a built-in tokenizer and can handle up to 512 tokens, which is the maximum limit in BERT. During the training phase, we limited the token length to 200, 256, and 400 tokens to address computational constraints or minimize the risk of overfitting on lengthy sequences.

The BERT architecture utilizes the Self Attention mechanism of the Transformer model to acquire context by examining word embeddings in other sentences. We trained our models using different epochs and batch sizes, employed the binary cross-entropy loss function, and Adam adaptive optimizer with varying learning rates for better GPU utilization.

To identify the most accurate BERT and RoBERTa architectures, we conducted experiments and tested various variants by adjusting random hyperparameters, such as dropout probabilities and ReLU activation. We randomly explored a different set of hyper-parameters for each model to identify the one that produced the most accurate results. We tested the Accuracy, AUC, F1-score, and Throughput values for each model, and their respective optimal hyper-parameters are reported in Table 5 for both BERT and RoBERTa models.

The results demonstrate that all BERT and RoBERTa-based models exhibit nearly identical accuracy and F1-score in detecting online toxicity, with variations in throughput. BERT-base has the highest AUC (79%) and F1-score (78%), but the

Table 5: Classification experiment results and parameters for BERT-based models with the best accuracy

Model	Epoch	Batch size	Learning rate	Dropout rate	Accuracy	AUC	F1-score	Throughput (s)
BERT	3	8	2.00E-05	0.1	0.92	0.79	0.78	2.5
Distilbert	1	2	2.00E-05	0.2	0.92	0.71	0.75	6.4
ALBERT	1	16	2.00E-05	0.3	0.92	0.70	0.73	4.4
RoBERTa	4	16	2.00E-05	0.1	0.92	0.78	0.77	5.4
DistilRoBERTa	3	16	2.00E-05	0.2	0.92	0.77	0.77	9.1

lowest throughput (25). Distilbert has a lower AUC and F1-score compared to BERT-base but a higher throughput (64). RoBERTa-base and DistilRoBERTa are very close in terms of accuracy, AUC, and F-score, but they have different throughputs. However, in the context of social media, where thousands of posts and comments are received every second, faster predictions are necessary. Despite their acceptable accuracy, AUC, and F1-score, transformers have low throughput and require improvements for practical use.

In the upcoming sections, we will detail the modifications applied to the CNN, CNN-fastText, BERT-based, and RoBERTa-based models with the aim of improving their throughput and profitability.

4.3.2.2 Modified detectors selected for best profit

Although the developed models are accurate in classification tasks, they are not optimal for high-throughput toxicity classification. To address this challenge, a variety of approaches have been suggested and put into practice to find the right balance between throughput and accuracy. However, it's important to note that the classifier with the highest throughput may not necessarily be the optimal choice for toxicity detection and application on platforms. There are numerous other factors that can not be adequately evaluated using metrics such as accuracy or throughput. Therefore, at the end of the assessment, models exhibiting both high throughput and high accuracy

will be selected for testing within the PDS framework. Ultimately, the most profitable model, which is best suited for each environment, will be chosen.

4.3.2.2.1 ***CNN-based***

(1) Tuning Parameters: A potential avenue for enhancing throughput involves adjusting parameters including learning rates and filters. We experimented with various parameter sets for CNN-based models to evaluate the impact on throughput.

(2) Shallower Network Topology: To potentially increase the throughput of a CNN, we experimented with reducing the depth of the network. While deeper networks can improve accuracy, they can also be computationally expensive and slow down the process. A shallower network with fewer convolutional 1D layers can be a good compromise between accuracy and throughput. For instance, we tested a CNN with four convolution layers and found that while it performed accurately enough, its throughput needed to be improved. To achieve this, we experimented with reducing the number of convolution layers to three and two, while still maintaining good accuracy.

(3) Smallest pre-trained embedding: Using larger pre-trained models can improve accuracy, but we also tested how smaller pre-trained fastText embedding would affect CNN performance. We compared two fastText pre-trained embeddings: wiki-news-300d-1M (16B tokens) and crawl-300d-2M (600B tokens) [3]. The wiki-news-300d-1M, which is trained on Wikipedia 2017, UMBC web-based corpus, and statmt.org news dataset has 1 million word vectors compared to crawl-300d-2M.

It is worth noting that while tuning model parameters and adjusting network topology can improve throughput, hardware and software optimization, such as using specialized hardware and efficient libraries, can also impact it significantly. However,

[3] https://fasttext.cc/index.html

Table 6: Throughput Improvement for CNN Models

Throughput (s)	**Accuracy**	**AUC**	**F1-score**	**Hidden layers**	**Filter**	**Batch Size**	**Learning Rate**
169.4	91.14%	64.50%	61.41%	4	128	512	0.01
231.9	92.06%	70.00%	72.33%	4	128	128	2.00E-05
3359	90.91%	75.25%	70.59%	4	400	128	5.00E-05
484.5	91.82%	69.98%	70.76%	4	128	256	0.01
608.5	90.98%	73.34%	69.54%	3	300	256	0.00.2
788.7	90.41%	74.12%	68.73%	3	128	128	0.01
8,65.1	90.27%	74.00%	68.03%	3	128	64	0.01
1,077.7	91.40%	73.91%	71.72%	2	128	64	0.01
1,135.6	91.26%	69.80%	71.37%	2	200	64	0.02
1,193.4	91.35%	71.38%	68.61%	2	300	64	0.0002
1,255.0	91.47%	72.12%	70.98%	2	300	64	0.0002

these factors were not considered in this study and will be explored in future work.

After hyperparameter tuning and adjusting the network topology, the CNN model achieved the highest throughput of 1,255 per second with 2 convolution layers, batch size of 128, learning rate of 0.01, maximum features of 50000, and filter size of 64. However, it is important to note that only parameter tuning could not significantly improve throughput; a shallower network topology had a more substantial impact. Combining both a shallower network and tuned parameters led to improved throughput. Similar techniques were applied to CNN-fastText, considering different word embeddings. Results revealed that the CNN model without pre-trained embeddings performed best, while among the fastText pre-trained embeddings, the Crawl news model achieved higher throughput than the wiki-news model. The CNN model using Crawl News achieved a throughput of 936.4 per second with 2 convolution 1D layers, GlobalMaxPooling1D, batch size of 128, learning rate of 0.01, maximum features of 40000, and filter size of 32. In contrast, the CNN model using wiki-news reached a throughput of 686.3 per second with 4 convolution layers. Additional details are available in Tables Table 6 and Table 7 for CNN and CNN-fastText models, respectively.

Table 7: Throughput Improvement for CNN-fastText Models

Throughput (s)	**Accuracy**	**AUC**	**F1-score**	**Hidden layers**	**Sources**	**Filter**	**Batch Size**	**Learning Rate**
186.0	91.74%	76.83%	73.81%	4	crawl	300	512	0.002
202.8	90.60%	77.25%	74.05%	4	wiki	300	256	0.0002
241.6	91.32%	79.25%	74.26%	4	wiki	300	128	0.002
384.9	92.14%	77.65%	75.25%	4	crawl	300	256	0.002
408.5	91.08%	80.30%	74.03%	4	crawl	300	64	5.00E-5
549.8	91.20%	80.70%	74.96%	3	crawl	300	64	5.00E-5
564.2	91.28%	80.65%	74.52%	3	crawl	300	64	5.00E-5
686.3	90.88%	84.40%	74.71%	2	wiki	128	128	0.001
687.7	91.92%	71.43%	73.28%	4	crawl	128	256	0.01
786.7	91.76%	82.28%	76.39%	2	crawl	64	128	0.001
843.2	91.39%	80.91%	75.04%	2	crawl	64	128	0.0001
936.4	91.74%	73.80%	73.62%	2	crawl	32	128	0.01

4.3.2.2.2 ***Transformer-based***

Transformers are known for their high computational requirements due to a large number of parameters and multiple layers of transformers. This limitation often makes them unsuitable for real-time applications that require high throughput. To address this issue, we explored various BERT models, including BERT-base, DistilBERT, ALBERT, RoBERTa, and DistilRoBERTa, to identify the most efficient models that offer high throughput. In addition, we tested all 24 BERT miniatures and found that BERT-Tiny, with two transformer layers and a hidden embedding size of 128, was the most suitable model for our memory and throughput requirements. Table 8 illustrates detailed information about the BERT miniatures and their characteristics. Through multiple experiments involving adjustments to parameters, we observed that BERT-Tiny performed optimally with epoch=1, batch-size=16, learning-rate=2.00E-05, and dropout-rate=0.3, yielding a throughput of 152.7 and an accuracy of 91%, along with an AUC and F1-score of 68.79% and 70.36%, respectively. Among RoBERTa-based variants, DistilRoBERTa achieved the highest throughput at 11.5 per second, accompanied by accuracy, AUC, and F1-score of 89.76%, 60.03%, and 67%, respectively.

A comparison of the throughput among different versions of BERT and RoBERTa is separately illustrated in Figure 8. The results point out that utilizing smaller variants of BERT and RoBERTa embeddings led to higher throughput.

Table 8: Characteristics of 24 Smaller BERT Models (English only, Uncased, Trained with WordPiece masking)

	H=128	**H=256**	**H=512**	**H=768**
L=2	2/128 (BERT-Tiny)	2/256	2/512	2/768
L=4	4/128	4/256 (BERT-Mini)	4/512 (BERT-Small)	4/768
L=6	6/128	6/256	6/512	6/768
L=8	8/128	8/256	8/512	8/768
L=10	10/128	10/256	10/512	10/768
L=12	12/128	12/256	12/512	12/768 (BERT-Base)

As you observe, classifiers exhibit significant differences in terms of evaluation metrics. The results are derived from testing algorithms in simple experimental setups, which may not accurately reflect the complexity of large-scale social media environments. Consequently, we have selected the best models based on their accuracy, F1-score, and throughput to compare and evaluate them using the PDS framework, while other aspects such as cost, revenue, and user satisfaction will also be evaluated this time. Table 25 presents the list of chosen classifiers. Notably, BERT-base boasts the highest accuracy (92.96%) and F1-score (78.26%) among all the models selected for PDS framework, while CNN exhibits the highest throughput (1,255). To differentiate between the models selected for PDS framework, we utilized the suffixes v1 and v2, where v1 indicates the version with higher accuracy, and v2 denotes the version with superior throughput.

4.3.3 Environments

The evaluation of classifiers spanned environments with diverse toxicity levels, ranging from low toxicity (about 20%), to medium toxicity (around 50%), and highly toxic environments (over 80%)

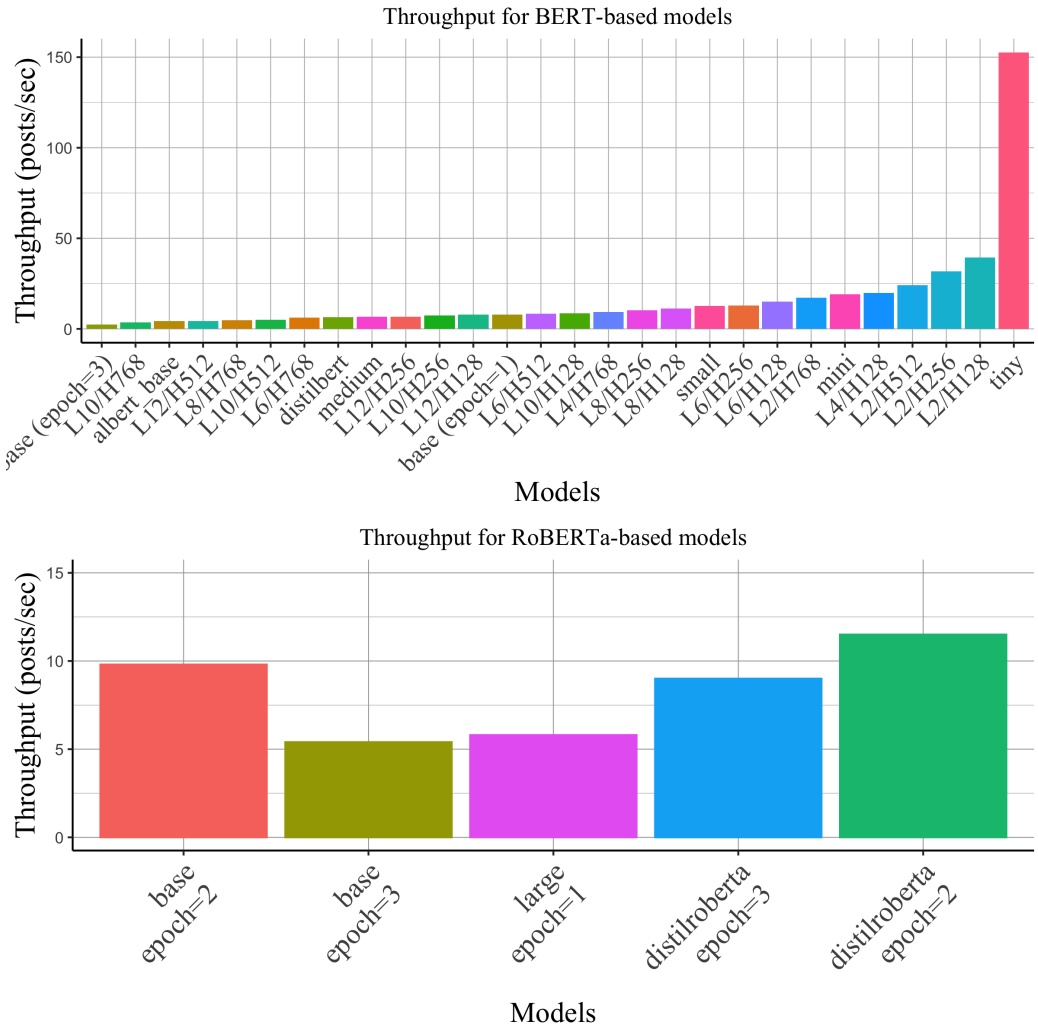


Figure 8: Throughput for BERT-based and RoBERTa-based Models per second (L is the Number of layers and H is Hidden size).

Table 9: All Models selected for experimentation

	Model	Accuracy	AUC	F1-score	Throughput (s)
Accuracy	CNN_{v_1}	92.06%	70.06%	72.33%	231.9
	$CNN\text{-}fastText_{v_1}$	92.14%	77.65%	75.25%	384.9
	$BERT\text{-}base_{v_1}$	92.96%	78.50%	78.26%	2.5
	$RoBERTa\text{-}base_{v_1}$	92.88%	78.63%	77.44%	5.4
Throughput	CNN_{v_2}	91.47%	72.12%	70.98%	1,255
	$CNN\text{-}fastText_{v_2}$	91.74%	77.65%	75.25%	936.4
	$BERT\text{-}Tiny_{v_2}$	91.50%	68.79%	70.36%	152.7
	$DistilRoBERTa\text{-}base_{v_2}$	89.76%	60.03%	60.67%	11.5

4.3.4 Computer Power

We used the cloud computing instance "Paperspace P4000" with NVIDIA GPUs [4] for our study. P4000 has 1 GPU with 8 GB memory, NVIDIA QUADRO P4000 type, 30 GB RAM, 8 vCPUs, Intel Xeon processor, and 2.60 GHz Clock Speed. The cost per second for a classification task on P4000 has been previously calculated as $0.000142. Therefore, according to Equation 4, the total cost of each classifier can be determined as follows: C(x, P4000) = $0.000142 multiplied by Tx, i, where Tx, i indicates the total detection time for classifier x running on instance i. Although we wrote all of the classifiers in Python 3.8, we opted to use Java 18 for the PDS framework as it significantly improved the running time and resulted in faster performance. We also visualized the PDS framework results using "InfluxDB" [5], a high-performance, open-source, distributed, and scalable time-series database designed to handle high volumes of time-stamped data. InfluxDB is widely used for real-time monitoring and provides a comprehensive set of tools for data visualization and analysis, making it easy to create real-time dashboards and reports.

[4] https://www.nvidia.com/en-us/design-visualization/quadro/
[5] https://www.influxdata.com/

4.3.5 Profit

The ultimate metric employed to evaluate the performance of distinct classifiers is profit. The computation of profit within the PDS framework is carried out using Equation 7. The calculation of costs was elucidated in the previous section. In terms of revenue assessment, the same configuration and features used for verifying null cases were adopted. This process is comprehensively explained in subsubsection 4.2.3.3. As a quick reminder, the revenue earned per advertisement view is determined by "Google AdSense" at a rate of $0.0086 per user view for the "North America region". More specifically, within the "People and Society" content category. The total revenue is calculated using Equation 6.

4.4 Experimental Results

Individually, every classifier was employed as the toxicity detector within the simulation's detector thread to exhibit its real-time performance within scaled environments. These environments encompassed scenarios with 10,000 users over the span of one week, incorporating varying degrees of toxicity. The outcomes are condensed in Table 10 outlining the profit achieved by each classifier in distinct environmental conditions.

The results show that in a low-toxic environment, the model with the highest throughput is the most profitable solution. A CNN_{v_2} with 91.47% accuracy, 70.98% F1-score and 1,255 throughput produces $402,046 profit while a CNN_{v_1} with 92.06% accuracy, and 72.33% produces $343,759 profit. Although BERT_{v_1} is the most accurate model, it could not be the most profitable in this situation. Moreover, the profits produced by BERT_{v_1} ($341,727) are very close to those produced by CNN_{v_1} ($343,759), while their throughputs are very different. Even though $\text{DistilRoBERTa}_{v_2}$ has a higher throughput (115) than other RoBERTa variants, it achieves the lowest profit ($88,796)

among all models due to its low accuracy (89.76%).

Under moderately toxic conditions, CNN_{v_1} generated the highest profit of $70,807 outperforming other models. There are other classifiers, such as CNN-fastTextv_1, that exhibit higher accuracy, AUC, F1-score, and throughput when compared to CNNv_1. However, despite their superior performance, they were unable to generate higher profits. It is interesting to note that neither the model with the highest throughput nor the most accurate model made the most profit in this environment. Similar to the low-toxic environment, $\text{DistilRoBERTa}_{v_2}$ makes the lowest profit. The reasoning makes sense, given that when a classifier underperforms in a low-toxic environment, while having neither higher accuracy nor throughput than others, it will likely underperform in a medium or high-toxic environment.

As a result of a highly toxic environment, CNN_{v_2} with the highest throughput ranks first in terms of profit ($26,196). Next, it is followed by the most accurate model (BERT-base_{v_1}) which generates $24,143 profits. With a profit of $22,201, CNN_{v_1} takes third place. In line with our expectations, $\text{DistilRoBERTa}_{v_2}$ has the worst performance with a profit of $15,162.

According to Figure 9, 36.05% of users left low-toxic environments when CNN_{v_2} was applied, while 95.4% left by $\text{DistilRoBERTa}_{v_2}$. The total percentage of users left over the simulation varied between 50% and 80% for other classifiers. There is a sharp increase of 58% in $\text{DistilRoBERTa}_{v_2}$, followed by a gradual increase, while in other classifiers the rate gradually increases from the beginning. All classifiers encountered the maximum number of left users in medium and high-toxic environments. Compared to high-toxic environments, medium-toxic environments have a slower rate of user fall. There are fewer left users in CNN_{v_1} until 22000 seconds, but CNN_{v_2} meets it and shows fewer falls in the rest of the test. CNN_{v_2} keeps increasing very closely to CNN_{v_1} (90.55%) in a highly toxic environment, but CNN_{v_2} finally achieves the lowest number

Table 10: PDS framework results for different classifiers

Model	Profit ($)		
	Low	Medium	High
CNN_{v_1}	343,759	**70,807**	22,201
$CNN\_fastText_{v_1}$	311,540	47,186	19,243
$BERT\_base_{v_1}$	341,727	46,678	24,143
$RoBERTa_{v_1}$	312,819	48,000	18,667
CNN_{v_2}	**402,046**	63,123	**26,196**
$CNN\_fastText_{v_2}$	282,555	39,945	18,371
$BERT\_Tiny_{v_2}$	279,594	37,016	16,783
$DistilRoBERTa_{v_2}$	88,796	23,017	15,162

(99.3%).

In addition, the number of users who left the platform for different reasons is also shown by Figure 10. The results demonstrate that by applying CNNv_2 in low- and high-toxic environments, the main reason that pushes users away from the platform is a large number of blocked posts, while in a medium-toxic environment, the main reason for leaving the platform is a large number of toxic posts. Moreover, applying RoBERTav_2 in all environments resulted in user churn due to a large number of toxic posts.

Overall, the CNN_{v_2} classifier outperformed the others in low and highly toxic environments, but CNN_{v_1} was the most profitable classifier in the medium-toxic environment. The study shows that CNN-based models are the best candidates for detecting toxic language online.

4.5 Summary

In conclusion, this chapter has presented a novel PDS framework that effectively and efficiently determines the most profitable toxicity classifier for various environments on social media platforms characterized by differing levels of toxicity. The proposed

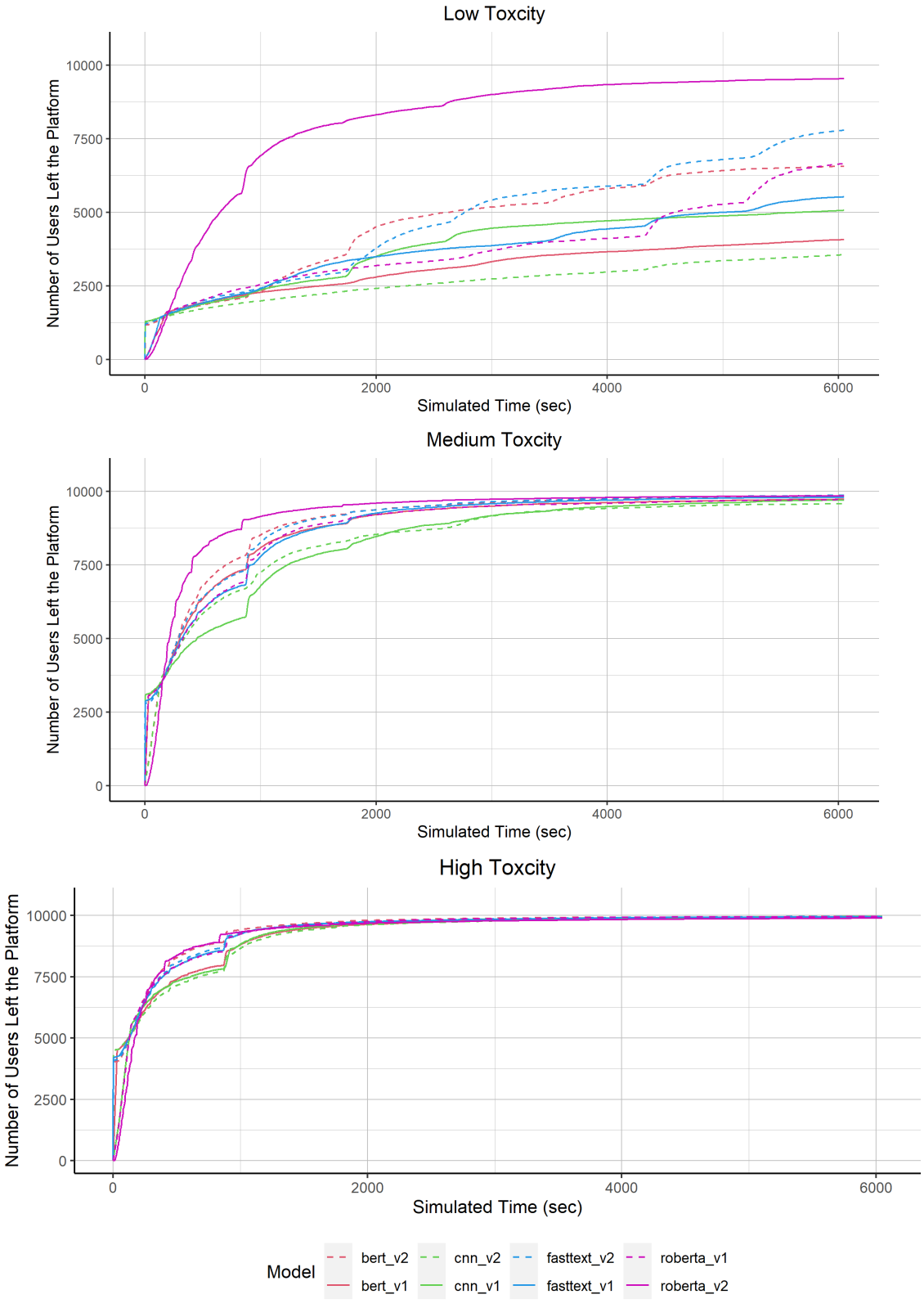


Figure 9: Users churn over the simulation time

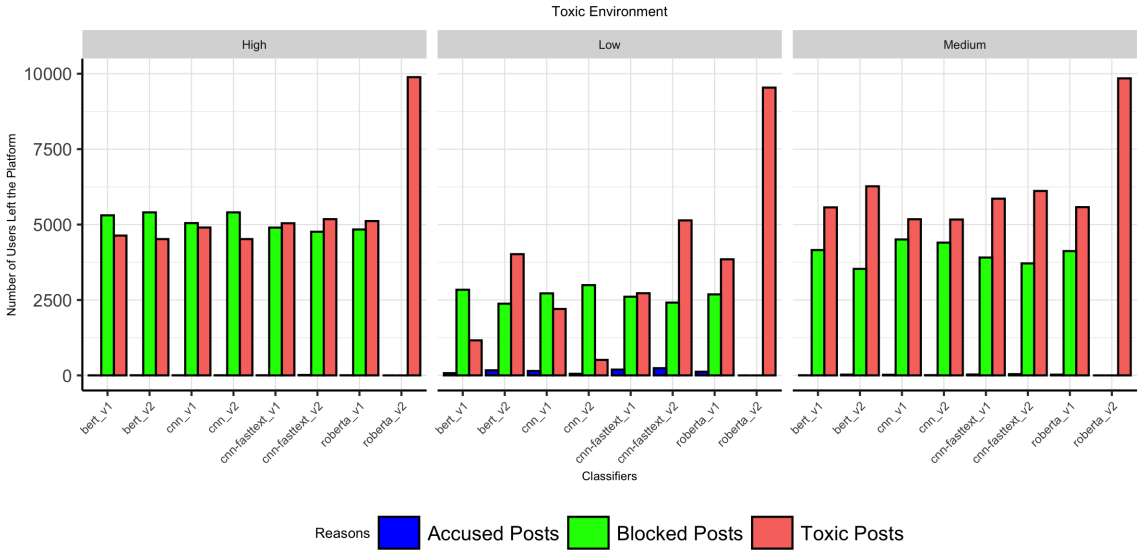


Figure 10: The reasons for leaving a platform in different environments by applying detectors

PDS framework evaluates toxicity classifiers by considering computational cost, real-time data processing at scale, and the impact of false positives, false negatives, true positives, and true negatives on user engagement and ad-generated revenue. Ultimately, this framework selects the optimal classifier for each scenario based on the highest profit achieved. The simulation results demonstrated that some classifiers with higher accuracy, AUC and F scores were not necessarily appropriate for all environments. Therefore, it is necessary to trade off the accuracy and the throughput and choose the model which ultimately increases user satisfaction and engagement and results in more profit for social media companies.

In addition, the PDS framework offers a valuable contribution to the field of social media analysis by providing researchers and practitioners with a tool that can better understand the complex nature of toxic behaviour on platforms and select the most effective and efficient classifiers to address these issues.

In addition, during our experiments, we observed that our deep learning classifiers, particularly transformer-based models, did not have acceptable throughput to process

a large volume of data. As a result, we took further action to improve their throughput while ensuring that these models were well-fitted without suffering from either underfitting or overfitting.

The PDS framework has shown promise in identifying the most suitable toxicity detector for social media platforms, but limitations remain. To ensure the effectiveness of the system, validation with real-world datasets containing actual content and connections is crucial. In addition, developing other classifiers using various datasets can also enhance the accuracy. Finally, integrating the PDS framework into A/B testing in a social media model can optimize the overall user experience and evaluate the system's effectiveness in a real-world scenario.

Acknowledgement We gratefully acknowledge the support of MITACS through the Accelerate and Business Strategy programs. Additionally, we extend our thanks to Kexin Yan, software engineer at Scrawlr, for his contributions to this research.

Chapter 5

AugmenToxic: Leveraging Reinforcement Learning to Optimize LLM Instruction Fine-Tuning for Data Augmentation to Enhance Toxicity Detection

This chapter is dedicated to the article entitled *"AugmenToxic: Leveraging Reinforcement Learning to Optimize LLM Instruction Fine-Tuning for Data Augmentation to Enhance Toxicity Detection"*. This article was submitted to the *ACM Transactions on the Web* and is under review. The titles, figures, and mathematical formulations have been revised to maintain coherence throughout the theis.

Addressing the challenge of toxic language in online discussions is crucial to developing effective toxicity detection models. This pioneering work focuses on addressing imbalanced datasets in toxicity detection by introducing a novel approach to augment toxic language data. We create a balanced dataset by instructing fine-tuning of Large Language Models (LLMs) using Reinforcement Learning with Human Feedback (RLHF). Recognizing the challenges in collecting sufficient toxic samples from social media platforms for building a balanced dataset, our methodology involves sentence-level text data augmentation through paraphrasing existing samples using optimized generative LLMs. Leveraging generative LLM, we utilize the Proximal Policy Optimizer (PPO) as the RL algorithm to fine-tune the model further and align it with human feedback. In other words, we start by fine-tuning a LLM using an instruction dataset, specifically tailored for the task of paraphrasing while maintaining semantic consistency. Next, we apply PPO and a reward function, to further fine-tune (optimize) the instruction-tuned LLM. This RL process guides the model in generating toxic responses. We utilize the Google Perspective API as a toxicity evaluator to assess generated responses and assign rewards/penalties accordingly. This approach guides LLMs through PPO and the reward function, transforming minority class samples into augmented versions. The primary goal of our methodology is to create a balanced and diverse dataset to enhance the accuracy and performance of classifiers in identifying instances from the minority class. Using two publicly available toxic datasets, we compared various techniques with our proposed method for generating toxic samples, demonstrating that our approach outperforms all others in producing a higher number of toxic samples. Starting with an initial 16,225 toxic prompts, our method successfully generated 122,951 toxic samples with a toxicity score exceeding 30%. Subsequently, we developed various classifiers using the generated balanced datasets and applied a cost-sensitive learning approach to the original imbalanced

dataset. The findings highlight the superior performance of classifiers trained on the data generated using our proposed method. These results highlight the importance of employing RL and a data-agnostic model as a reward mechanism for augmenting toxic data, thereby enhancing the robustness of toxicity detection models.

5.1 Problem Statement & Contributions

The rapid development of communication technology and the internet has transformed virtual communities, making social media platforms more accessible and user-friendly, but also presents significant challenges [10]. Toxic language, a prevalent issue in online discussions, is frequently characterized by disrespectful responses that can deter participants from engaging in meaningful conversations [7], [274]. To ensure the safety of online civil discussions and mitigate the potential harm caused by toxic language, the vast amount of user-generated content (UGC) requires the implementation of data-driven techniques, including Machine Learning (ML) algorithms, for the automatic classification of UGC within modern content moderation systems [19], [275], [276]. Developing efficient detection systems for toxic content is heavily dependent on the availability of appropriate training datasets. This is in line with the fundamental principle in data-driven research, which states that the quality of outputs is directly influenced by the quality of inputs [102].

Imbalanced datasets and lack of annotated samples pose significant challenges in various classification tasks, such as toxicity detection [173], [276]. In these datasets, one class is often much more numerous than the others, typically known as the majority class, which can result in the development of biased models that yield unsatisfactory results when dealing with the underrepresented minority class [173], [277]. The problem is particularly acute in toxicity detection, as the frequency of toxic samples is often low compared to nontoxic ones. In other words, it is difficult to collect roughly the same number of samples for both toxic and nontoxic classes. For instance, Madukwe et al. [278] highlighted a pronounced class imbalance problem in hate speech detection, with the hate class constituting less than 12% in multi-class datasets and less than half in binary datasets. Additionally, a systematic review of datasets for automatic

hate speech detection revealed that 41% of the datasets are small (0-5k posts), with 37% containing less than 20% offensive content [82]. This confirms the challenges associated with obtaining extensive labeled data and underscores the potential pitfalls of class imbalance in training datasets. The imbalance in data distribution can lead to overfitting and hinder generalizability, especially for Deep Learning (DL) models. This imbalance may result in models that excel in detecting nontoxic language but perform poorly on toxic content.

To address imbalanced classification challenges, a diverse set of techniques is utilized at both the data and algorithmic levels [184]. At the data level, these techniques encompass various re-sampling approaches, including oversampling of minority classes and undersampling of the majority class [173], [251]. Undersampling techniques consist of methods such as random undersampling [252], Inverse random undersampling [279], and directed undersampling (informed undersampling) [280]. On the contrary, oversampling techniques [281] include random oversampling with replacement, directed oversampling (which entails informed choices for replacing samples, rather than random selection), and oversampling methods with informed generation of new samples. Moreover, an alternative strategy involves a hybrid method [282] that combines both undersampling and oversampling techniques. In addition to data-level strategies, at the algorithmic level, methods such as cost-sensitive learning (which entails adjusting costs associated with different classes), asymmetric classification, dimension reduction, expert systems, and ensemble techniques such as bagging, boosting, and stacking all assume crucial roles [170], [201], [253], [254], [283], [284]. Although these techniques have shown promising results in improving the performance of models on imbalanced datasets, they also have limitations. Oversampling and undersampling can lead to overfitting and underfitting, respectively, and may not work well when the dataset is extremely imbalanced [255], [285]. Cost-sensitive learning requires an accurate

estimation of the misclassification costs, which may be difficult in practice [200]. In the case of ensemble methods, as sampling techniques are used to balance the data in each iteration, they can potentially eliminate valuable information and be prone to overfitting [175]. To mitigate overfitting, an alternative method is automatic data augmentation (DA), involving the creation of synthetic data based on an existing dataset [204]. This can contribute to improving the generalizability of text classification models, making them more adept at performing well with unseen data [286]. The methods for augmenting text data depend on the specific task and the type of text data under consideration [204]. Text data augmentations (TDA) can be classified into several categories, including the injection of textual noise or spelling errors, word replacement using a thesaurus, and the generation of paraphrases through syntactic tree transformations, back-translation, and pre-trained transformer networks [207], [287]. However, these techniques are not yet ideal, and their effectiveness is hindered because the structure and meaning of the text are closely connected, making it challenging to manipulate one aspect without affecting the other satisfactorily [286].

Therefore, to address the challenge of balancing datasets and enhancing the detection of toxic language through a data augmentation approach, there is a pressing need to develop effective techniques for generating toxic text. While some solutions have been proposed to generate toxic text [163], and general text generation techniques have been tested for toxic language detection [223], [224], [226], they all exhibit limitations. For instance, certain methods necessitate precise and well-written prompts to function effectively, potentially performing inadequately when dealing with original, unrefined samples. Others rely heavily on zero-shot learning, which involves fine-tuning models solely on toxic or non-toxic samples. This approach often neglects the significant role that instruction fine-tuned language models play in handling specific tasks. Additionally, another issue arises from the practice of assigning the same label to

generated samples as the prompt sample. This can lead to inaccuracies, especially if the generated content deviates from the intended toxicity, resulting in the misclassification of non-toxic content as toxic, or vice versa.

As we address these challenges, our objective is to introduce a novel technique for sentence-level TDA, specifically targeting toxic language. This approach aims to overcome the limitations observed in previous works by making a substantial contribution to the development of techniques for creating a balanced and diverse toxic dataset, ultimately enhancing classifier accuracy and performance.

We present a TDA framework guided by Reinforcement Learning from Human Feedback (RLHF) [288], specifically employing the Proximal Policy Optimization (PPO) algorithm [58]. This framework operates on an Instruction Fine-Tuned (IFT) Large Language Model (LLM) [161] and focuses on refining the model's ability to paraphrase text. The process begins by fine-tuning the LLM using an instruction dataset derived from the PAWS dataset (Paraphrase Adversaries from Word Scrambling) [289]. PAWS is selected because it is human-labeled, enabling it to accurately discern between paraphrases with equivalent semantic meaning and those with high lexical overlap but are not true paraphrases. This ensures that the paraphrasing task maintains semantic coherence. Throughout our experiment, we exclusively concentrate on paraphrase pairs exhibiting identical semantic meaning. After the initial fine-tuning, we optimize the model further using PPO in conjunction with a reward model. PPO guides the model through the augmentation process by facilitating fine-tuning against this reward model. Operating within a hybrid architecture that seamlessly combines value-based and policy-based methods, Proximal Policy Optimization (PPO) enhances training stability by iteratively updating the model's policy in a controlled and proximal manner. This approach prevents drastic changes that could potentially disrupt the training process, thereby ensuring a more stable and reliable learning

experience.

To encourage paraphrasing while maximizing toxicity, we utilize the Google Perspective API to evaluate the toxicity level of each generated text. This API assigns toxicity scores ranging from 0 to 1, indicating the likelihood of a text being toxic. This incentivizes the model to produce toxic samples as it seeks to maximize rewards. Additionally, to prevent the LLM from generating peculiar and non-human-like responses solely to maximize rewards, we employ Kullback-Leibler (KL) Divergence as a penalty. This divergence is calculated between the active policy, influenced by reinforcement signals for toxicity, and the reference policy derived from the initial instruction-tuned LLM. This measure ensures that when the model is hallucinated, it aligns closer to the reference model, striking a balance between maximizing rewards and maintaining human-like response characteristics. By imposing this penalty, the model is motivated to generate paraphrases that closely align with the reference model, thus achieving a balance between maximizing rewards and preserving human-like response characteristics. We evaluated the proposed method and compared it with other techniques using the Jigsaw toxic dataset [271] and the ToxiGen dataset [163].

This is the first work to employ an optimized instruction-fine-tuned language model (LLM) to paraphrase existing unstructured data, thereby augmenting toxic textual samples in the minority class. Furthermore, our dataset is one of the largest and most balanced available. Additionally, we applied zero-shot learning, and back-translation techniques to benchmark our developed model against other methods, resulting in the creation of the largest balanced dataset for toxicity detection, generated through back-translation from nine different languages into English. In summary, our contributions can be outlined as:

- A novel method for enhancing toxic text data through Instruction Fine-tuning on the pretrained FLAN-T5 model, precisely crafted for paraphrasing with semantic

equivalence using PAWS.

- Applying Proximal Policy Optimization (PPO) to further fine-tune (optimize) the instruction-tuned FLAN-T5; our approach incorporates a reward model within the PPO framework to ensure the generated responses maintain the specified level of toxicity.

- Utilizing the Google Perspective API to score toxicity and assign rewards accordingly, while implementing KL-Divergence as a penalty in the reward function to ensure the generated text maintains human-like responses.

- Expanded the imbalanced Jigsaw dataset, which originally included 143,346 nontoxic samples and 16,225 toxic samples, into a balanced dataset comprising over 278,000 samples.

- Outperforming other data augmentation techniques, such as zero-shot learning, back-translation, and instruction-tuned LLMs, which lack RLHF optimization.

The chapter is organized as follows: section 5.2, covers the proposed method. Section 5.3 details the experimental setup, and the results are presented in section 5.4. Section 5.5 concludes the chapter with remarks and discussions.

5.2 Methodology

In this section, we outline the structure of our proposed approach. Initially, we conduct fine-tuning on the generative LM by utilizing an instruction dataset to paraphrase samples while preserving their semantic meaning (Section 5.2.1). Subsequently, to optimize the process, we employ PPO and reward function to guide the model toward toxic paraphrasing, aiming to generate more toxic responses (Section 5.2.2).

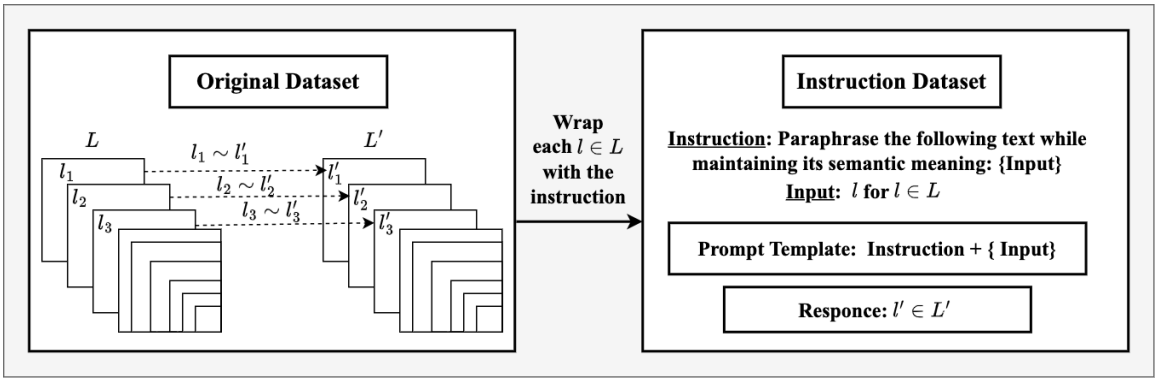


Figure 11: Illustration of the process to convert the dataset into an instruction format.

5.2.1 Supervised Instruction Fine-tuning

Our primary goal is to increase the number of minority class samples through paraphrasing techniques using generative Language Models (LMs). Although models like zero-shot learning may offer simplicity, they are less effective in paraphrasing toxic samples from online conversations. The subtle and context-dependent nature of toxicity in online discussions poses challenges beyond the capabilities of general language understanding methods, including zero-shot, one-shot, or few-shot learning [290], [291]. Limited exposure to samples can hinder the model's paraphrasing accuracy, making it challenging to maintain meaningful output [292]. To address this, instruction fine-tuning emerges as a promising alternative. This approach involves training an LM on a specific task with explicit instructions, enhancing its ability to paraphrase toxic content while preserving semantic meaning. Instruction fine-tuning offers a focused and tailored training process, enabling the model to adapt more effectively to the nuances of paraphrasing toxic language in unstructured online comments, ultimately improving the quality of generated samples.

5.2.1.1 Instruction Dataset

An instruction dataset D_{I} typically refers to a specific dataset designed to provide explicit instructions for training a model on a particular task. It contains examples paired with clear instructions on how the model should interpret or respond to those examples. The purpose of an instruction dataset is to guide the model's learning process and help it acquire specific skills or behaviors.

To construct D_{I}, we utilize a structured format containing examples, making it more intuitive for the generative model M to learn in accordance with our requirements. The dataset includes pairs $s \in S$ and $s' \in S'$ representing paraphrases of each other ($s \sim s'$) while maintaining semantic equivalence.

To convert the dataset into an instruction format, each sample s undergoes a wrapping process with an instruction, as illustrated in Figure 11. The instruction provided is formulated as follows: "Paraphrase the following text while maintaining its semantic meaning: {text}." This instruction serves as a directive for the model, guiding it to generate paraphrases that retain the same semantic meaning as the original text. Additionally, all samples, both input prompt (instruction + s) and output (s'), undergo tokenization using the LM's tokenizer. Padding is also applied to the tokenized sequences, ensuring they have the same maximum length.

5.2.1.2 Instruction Fine-tuning

With the constructed instruction dataset D_{I} (refer to Section 5.2.1.1), the pre-trained model M undergoes fine-tuning in a fully supervised manner. This process entails training M to predict each token in the output sequentially, guided by the instruction dataset [293]. The fine-tuning adapts M's parameters based on the task-specific information embedded in the instruction dataset, thereby enhancing its performance in generating paraphrases while preserving semantic meaning.

Recognizing the advantages of Parameter-Efficient Fine-Tuning (PEFT) techniques, such as reducing computational costs, minimizing memory usage during training, streamlining the storage and deployment of task-specific fine-tuned parameters [294], directed us to utilize PEFT as a substitute for the full fine-tuning process in our instruction fine-tuning approach. Furthermore, its demonstrated superiority over full fine-tuning across a diverse array of tasks [45] further supported our decision.

PEFT methods (refer to Section 2.3) involve freezing the majority of parameters in pre-trained models while still demonstrating comparable capabilities in downstream tasks [295]. Specifically, we considered fine-tuning through Low-Rank Adaptation (LoRA) [44] as an additive fine-tuning scheme, as defined in Section 2.3. The instruction fine-tuning of the LM for the specific task of paraphrasing is illustrated in Figure 12.

While the fine-tuned model M_{LoRA} excels at paraphrasing existing samples in the minority class and generating new samples, it may encounter challenges when faced with unstructured input from online conversations. Particularly, its performance might diminish when dealing with less well-written or unstructured prompts, even after pre-processing. This challenge is exacerbated when the model is tasked with generating multiple responses for each prompt. Moreover, we anticipate scenarios where the model may generate toxic samples, and there is a need for the LM to rephrase the input while retaining or even increasing its toxicity. Therefore, following the fine-tuning process via LoRA (M_{LoRA}), an additional optimization step using Reinforcement Learning from Human Feedback (RLHF) becomes essential. RLHF aims to guide the LM in rephrasing while preserving toxicity or potentially intensifying it, based on human feedback expressed through a reward function. The subsequent section will delve into the specifics of the optimization process using RLHF.

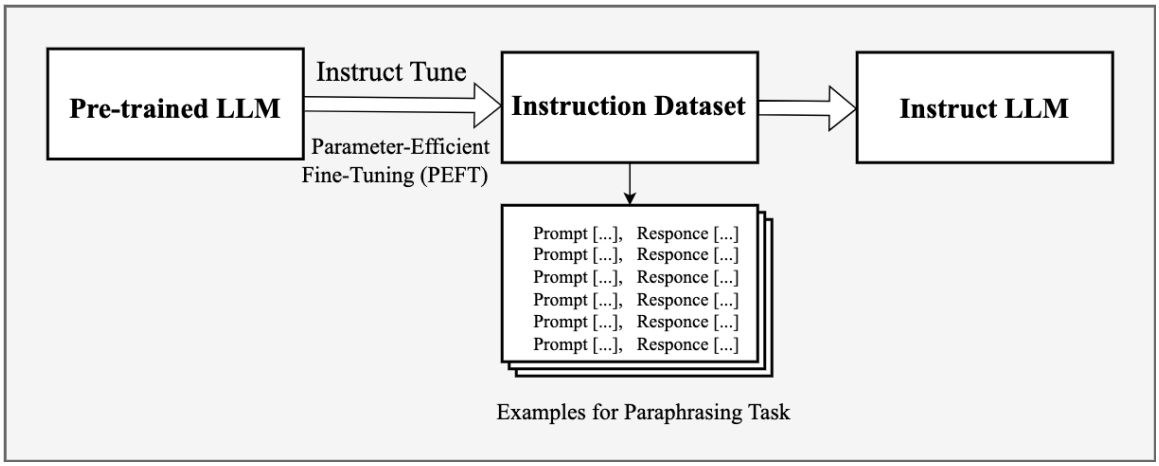


Figure 12: Instruction fine-tuning of pre-trained Large Language Models (LLMs) for paraphrasing

5.2.2 Optimization using Reward Function

In this study, we employ Reinforcement Learning from Human Feedback (RLHF) Section 2.4, specifically utilizing Proximal Policy Optimization (PPO) as described in Section 2.5.1. Our objective is to optimize the fine-tuned model M_{LoRA} to generate toxic responses, acknowledging that not all augmented sentences may exhibit toxicity. The development of efficient optimization algorithms is crucial across various scientific disciplines, as researchers seek faster and stronger algorithms capable of optimizing a wide range of functions [296]–[299]. After the initial fine-tuning with LoRA, where only a portion of parameters was trained, we seek further refinement by updating these trainable parameters to obtain an optimized fine-tuned model. To achieve this, we incorporate a data-agnostic classifier to assess the toxicity of generated responses and assign rewards or penalties accordingly. It is important to note that D_{I} used for instruction fine-tuning is nontoxic, but our aim is to increase toxicity using the reward model, directing M_{LoRA} to generate toxic tokens.

As a reward model, we leverage the Google Perspective API (API) [1], a machine

[1] https://perspectiveapi.com/

learning-based tool designed to detect abusive comments. This API furnishes toxicity scores ranging from 0 to 1, serving as a probability indicator without delineating severity. Higher scores indicate a greater likelihood of resembling patterns observed in toxic comments. We employ PPO to fine-tune M_{LoRA} with respect to the reward model, resulting in $M_{\text{PPO-API}}$. While M_{LoRA} has initially been trained using the instruction dataset, our aim is to optimize its performance leveraging the reward model.

In the proposed PPO framework, M_{LoRA} serves as an active model (M_{act}) during training and as a reference model (M_{ref}) when not trainable. The generative model functions as an agent, selecting tokens during language generation. The agent initializes its policy with M_{LoRA} and, at each time step t, observes the current state s_t (previously generated tokens), takes action a_t according to the policy ϕ, and transitions to the next state. The agent receives a reward r from the reward model, aiming to maximize the expected reward during PPO training. The framework is visually depicted in Figure 13.

A prompt x_t from D_{I} is simultaneously inputted into both the active model M_{act} and the reference model M_{ref}. Active policy $\pi_{\theta_{\text{act}}}$ and reference policy $\pi_{\theta_{\text{ref}}}$ are initialized. The active model M_{act} generates a response (paraphrased input), such that $\pi_{\theta_{\text{act}}}(a_t \mid s_t) \longrightarrow M_{\text{act}}(x_t) = y_{\text{act}_t}$. Similarly, the reference model M_{ref} generates a response $\pi_{\theta_{\text{ref}}}(a_t \mid s_t) \longrightarrow M_{\text{ref}}(x_t) = y_{\text{ref}_t}$). Subsequently, the response generated by the active model y_{act_t} is decoded and passed to the Google Perspective API, which assigns a toxicity score. We have defined a reward of 1 for samples with toxicity exceeding 0.7; otherwise, it receives a penalty of -10. This reward scheme is formalized as follows:

$$r_{\text{toxic}} = \begin{cases} +1 & \text{if API}(y_{\text{act}_t}) \geq 0.7 \\ -10 & \text{if API}(y_{\text{act}_t}) < 0.7 \end{cases}$$

We assigned greater value to punishment compared to reward because if both were given equal values, such as +1 and -1, the agent might learn to generate only nontoxic responses. In such a scenario, the penalty for producing a nontoxic response would be only 1, which would not sufficiently differentiate between the reward and the penalty. Additionally, to discourage M_{act} from producing unnatural responses solely for increased rewards, a reference model with frozen weights serves as a fixed point of reference. The Kullback-Leibler (KL) Divergence is calculated between the two policies $\pi_{\theta_{\text{ref}_t}}$ and $\pi_{\theta_{\text{act}_t}}$ as follows:

$$r_{\text{kl}} = \beta D_{\text{KL}}(\pi_{\theta_{\text{act}_t}} \parallel \pi_{\theta_{\text{ref}_t}}) = -\beta \log \left(\frac{\pi_{\theta_{\text{act}_t}}}{\pi_{\theta_{\text{ref}_t}}} \right)$$

r_{kl} serves as a penalty, ensuring that when M_{act} generates hallucinations, it aligns closer to M_{ref}. This penalty is added to the toxicity reward, constraining the update within a trust region defined by the distance between the two policies. The total reward is computed as:

$$r_t = r_{toxic} + r_{kl} \tag{12}$$

This cumulative reward guides PPO through multiple prompt-response experiments, facilitating ranking averages and employing backpropagation to optimize the response of M_{act}. The proposed framework is illustrated in Figure 13.

5.3 Experimental Setup

5.3.1 Instruction Dataset

To fine-tune the LLM for text paraphrasing, we employed the Paraphrase Adversaries from Word Scrambling (PAWS) dataset [289], introduced by Google AI Language

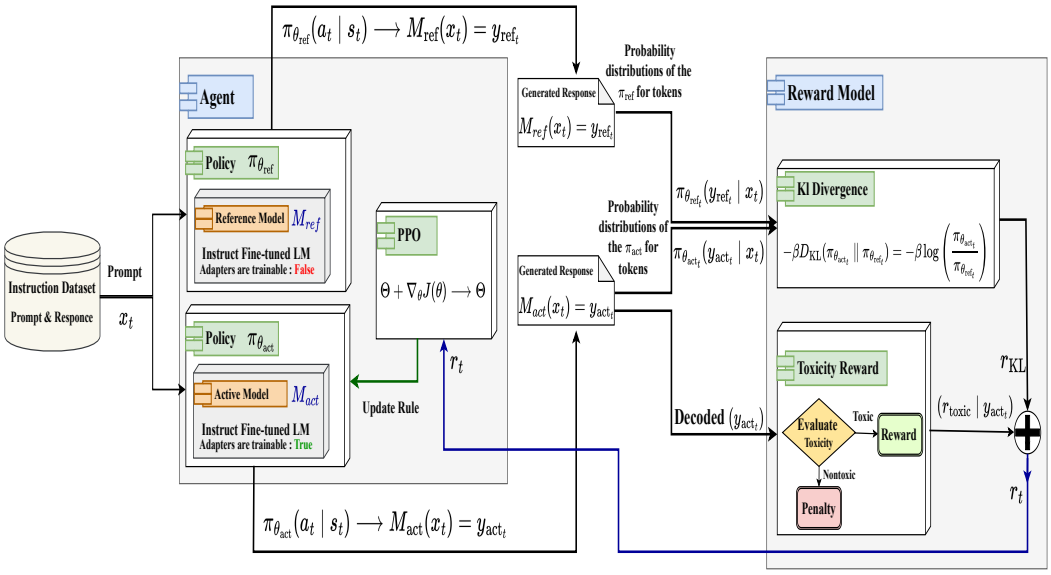


Figure 13: The proposed solution for paraphrasing toxic samples

in 2019. This openly accessible dataset comprises 108,463 thoughtfully crafted pairs, encompassing both paraphrases and non-paraphrases with significant lexical overlap. Specifically, we utilized the PAWS-Wiki Labeled "Final" version, which includes 65,401 pairs generated through both word swapping and back translation methods. All pairs have undergone human assessments for both paraphrasing fidelity and fluency. The dataset is divided into Train, Validation, and Test sets containing 49,401, 8,000, and 8,000 respectively with no overlap of source sentences across sets. Maintaining high quality, the dataset is structured with three columns—sentence 1 (L_1), sentence 2 (L_2), and a label (0 or 1). In this organization, sentence 1 ($l_1 \in L_1$) represents the primary text, and sentence 2 ($l_2 \in L_2$) serves as its counterpart. A label of 1 signifies that sentence 2 is a paraphrase of sentence 1, while a label of 0 indicates a distinct semantic meaning between the two. Combining all sets (train, validation, and test) into one dataset, comprising "65,401" samples, we exclusively consider the "28,904" samples with label (1), denoting paraphrasing, and discarding samples with label (0).

Subsequently, an 80-20 split was performed, allocating “23,123” samples to the training set and “5,781” samples to the test set. Within the training set, a further subdivision was implemented for hyperparameter tuning, with ‘2,312” samples, constituting 10% of the training data, reserved for validation purposes.

5.3.2 Toxic Datasets

5.3.2.1 Jigsaw-dataset

We employed a publicly available dataset provided by Google Jigsaw and Kaggle [271], which comprises 159,571 Wikipedia comments human-rated for toxicity across six categories: toxic (15,294), severe toxic (1,595), insult (7,877), obscene (8,449), threat (478), and identity hate (1,405). The remaining data (143,346), which is not included in any type of toxicity, is considered nontoxic. Notably, this dataset exhibits a significant imbalance, with the majority class consisting of nontoxic samples. Within this dataset, aside from nontoxic samples, others may bear multiple labels. For instance, a sample could be labeled as both insult and obscene without carrying the toxic label, while another might solely be labeled as insult, and yet another could have labels for both insult and toxic. Unfortunately, precise descriptions or definitions for the various types are not published. Consequently, we classify all samples with at least one form of toxicity as toxic and convert the dataset into binary labels (toxic, nontoxic). It is crucial to emphasize that, in this chapter, we use the term “toxic” to encompass any type of toxicity. As a result, the final dataset includes 143,346 nontoxic samples and 16,225 toxic samples. This implies that, for every toxic sample, approximately eight nontoxic examples exist in the dataset. To ensure an adequately balanced dataset, we decided to generate nine samples per toxic sample, resulting in a total of 146,025 samples. Since not all generated samples are necessarily toxic, we opted for generating additional samples to maintain a sufficient number even if nontoxic ones are removed,

thereby achieving an equal balance of toxic and nontoxic samples in the final dataset.f

5.3.2.2 ToxiGen-dataset

ToxiGen is a machine-generated dataset comprising 274,000 statements, encompassing both toxic and benign content associated with 13 distinct minority groups [163]. They gathered human-written sentences showcasing implicit toxicity directed at the 13 minority groups, resulting in 26 sets of prompts. Each set includes two variations (benign and toxic) for every target group. From these sets, we specifically chose 2000 prompts displaying hateful content. To achieve our objective, we aim to rephrase toxic prompts and generate additional toxic samples. Subsequently, to ensure a balanced dataset, we randomly select benign samples from the dataset equal to the number of augmented toxic samples.

5.3.3 Instruction Fine-tuning

In our study, we employed FLAN-T5 [292] for the data augmentation task by paraphrasing existing samples in the minority class. FLAN-T5 is a Fine-tuned Language Network (FLAN) built on the T5 (Text-To-Text Transfer Transformer) architecture [300] and pre-trained on an extensive text corpus. It demonstrates robust generalization across multiple tasks [301]. We specifically employed **flan-t5-base**, which features 250 million parameters, an encoder-decoder architecture, and span corruption [292].

To fine-tune the flan-t5-base model through instruction, we employed the PAWS dataset Section 5.3.1 for the targeted paraphrasing task. Following the methodology explained in Section 5.2.1.1, we first constructed the instruction dataset. The PAWS dataset was adapted into the instruction dataset, where the prompt is generated by wrapping inputs ($\forall s_1 \in S_1$) with paraphrasing instructions, and the response is

provided by S_2. The minimum and maximum lengths of the input data are set to 10 and 300, respectively. As mentioned earlier in Section 2.3, we utilized LoRA to efficiently fine-tune the flan-t5-base model with limited computational resources. The training consisted of 15 epochs with a learning rate of 1×10^{-5}, where Lora-rank (r) was set to 70, α to 70, and dropout to 0.05. The total number of trainable parameters for the original model is 255,319,296. With the use of PEFT, this figure is reduced to 7,741,440, signifying that only 3.03% of the model parameters are now trainable.

To assess the performance of the model for paraphrase generation before and after instruction fine-tuning, there is a lack of consensus on the metrics appropriate for these task-specific models, resulting in variations in measurements across different studies. One commonly utilized metric in the assessment of summarization tasks is **ROUGE** which is Recall-Oriented Understudy for Gisting Evaluation [302]. This metric, which predominantly emphasizes recall, is widely employed and extends its applicability to paraphrase evaluation [303], [304]. ROUGE captures the n-gram overlap between responses generated by LM and those generated by humans. It is diversified into several types, including ROUGE-1, ROUGE-2, ROUGE-N, and ROUGE-L, ROUGE-W, ROUGE-S, each tailored to specific features. As an example, ROUGE-N emphasizes gram count and calculates recall by examining matching unigrams in the context of unigram analysis (ROUGE-1). Conversely, ROUGE-L evaluates the Longest Common Subsequence (LCS), ROUGE-W focuses on Weighted LCS, ROUGE-S delves into skip-bigram co-occurrence statistics, and ROUGE-LSUM shows the Length of LCS normalized by the total words in the reference. Additionally, we incorporated other metrics to assess the quality of generated responses, including **METEOR** (Metric for Evaluation of Translation with Explicit Ordering) [305], and **BERTScore** [306]. METEOR integrates semantic understanding into its translation evaluation process by assessing matches in terms of exactness, stemming, or synonymy

[307]. BERTScore leverages pre-trained contextual embeddings from BERT-based models to compare words between the source and generated texts, employing cosine similarity for matching [308]. We utilized METEOR 1.5 [309] and the 'bert-base-uncased' model for BERTScore.

Table 11 presents the assessment outcomes using $ROUGE$, $METEOR$, and $BERTScore$ metrics, comparing the initial model pre-finetuning using Zero-shot learning ($flant5_{Zero-shot}$) with the instruct-tuned flan-t5-base ($flant5_{LoRA}$).

The findings indicate that the $flant5_{LoRA}$ exhibits absolute percentage enhancements compared to the $flant5_{\text{Zero-sho}}$, with improvements in all metrics. Please note that PAWS does not include any toxic samples, and, so far, flan-t5-base is only instruct-tuned for the paraphrasing task while preserving semantic meaning. It may not perform well in countering toxic content or may unintentionally remove toxic words to avoid generating harmful samples. Therefore, further optimization is needed using a reward function to perform paraphrasing while preserving semantic meaning and addressing or potentially increasing the toxicity level.

In the following section, we explain our experimental setup for optimizing instruction fine-tuning of flan-t5-base using a reward function.

5.3.4 Optimization

While the instruct-tuned flan-t5-base ($flant5_{LoRA}$) has demonstrated improvements over the flan-t5-base using zero-shot learning ($flant5_{\text{initial}}$), further optimization is possible through the integration of a reward function and PPO. Due to the unstructured nature of toxic samples, characterized by online comments deviating from ordinary grammar and vocabulary, the task of paraphrasing becomes notably challenging. To address this challenge, optimization involves leveraging a reward function capable of evaluating text toxicity. By rewarding or penalizing the model accordingly, the

Table 11: Comparing Model Performance in Paraphrasing Tasks Pre and Post Instruct-Finetuning

Model	Metric	Value
$flant5_{\text{Zero-shot}}$	ROUGE-1	0.324
	ROUGE-2	0.289
	ROUGE-L	0.318
	ROUGE-LSUM	0.321
	METEOR	0.317
	BERTScore-Precision	0.332
	BERTScore-Recall	0.323
	BERTScore-F1	0.326
$flant5_{LoRA}$	ROUGE-1	**0.416**
	ROUGE-2	**0.392**
	ROUGE-L	**0.412**
	ROUGE-LSUM	**0.413**
	METEOR	**0.526**
	BERTScore-Precision	**0.547**
	BERTScore-Recall	**0.531**
	BERTScore-F1	**0.539**

objective is to encourage the generation of toxic samples compared to nontoxic ones.

For optimization, we follow the method explained by Section 5.2.2 and use $flant5_{\text{LoRA}}$ as a reference model ($flant5_{\text{RF}}$), where its adaptors are not trainable, and all weights are frozen, and an active model ($flant5_{\text{ACT}}$) with trainable adaptors. We also employ the Google Perspective API [2] as the toxicity detector in the reward model to evaluate the toxicity of generated samples. The Perspective API utilizes machine learning to identify abusive comments, providing toxicity scores between 0 and 1 as a probability indicator, not a severity measure. Higher scores indicate a greater likelihood of resembling patterns in toxic comments, and developers can set thresholds based on these scores without quantifying the degree of toxicity [310].

Additionally, we utilize another toxicity detector to score the toxicity of generated

[2] https://perspectiveapi.com/

paraphrased samples and gain a toxicity reward. The **facebook/roberta-hate-speech**[3], which we refer to as $HateRoBERTa$ in this chapter, is a RoBERTa model fine-tuned on a hate/toxic speech dataset [311], available on Hugging Face[4], specifically designed for detecting hate/toxic speech.

The goal is to compare the performance of $flant5_{\text{LoRA}}$ when optimized by PPO, and the toxicity detector in the reward function is Perspective API ($flant5_{\text{PPO-API}}$), and when the optimization is done using $HateRoBERTa$ as the toxicity detector ($flant5_{\text{PPO-RoBERTa}}$). Perspective API is more data-agnostic than $HateRoBERTa$, and we expect to see superior performance.

The toxicity reward is then added to the penalty. The reference model ($flant5_{\text{RF}}$) also acts as a fixed point of reference, ensuring that when the active model hallucinates, it aligns closer to the reference model, providing positive responses that are not bizarre. KL-Divergence is then calculated and used as a penalty. This penalty is added to the toxicity reward, and the total reward is passed to the value function (PPO) to update the policy accordingly.

To optimize both optimization techniques utilizing various toxicity reward functions, we utilized the trl package [312]. This involved setting generation parameters to Top-k=0.0, Top-p=1.0, output-min-length=50, output-max-length=512, and implementing a maximum of 20 PPO steps.It is important to note that the Top-k sampler restricts sampling to the k most probable tokens, while the Top-p (nucleus) sampler constrains sampling to the smallest set of tokens [34], [313], [314]. Additionally, we conducted iterative tests using a validation set and experimented with random values for hyperparameters to determine the optimal settings.

To assess the quality of paraphrased responses generated by $flant5_{PPO-API}$ and $flant5_{PPO-RoBERTa}$, we utilized ROUGE, METEOR, and BERTScore metrics to

[3] https://huggingface.co/facebook/roberta-hate-speech-dynabench-r4-target
[4] https://huggingface.co/

Table 12: Examining Paraphrasing Model Performance Enhanced by PPO Using Diverse Toxicity Reward Mechanisms

Model	Metric	Value
$flant5_{\text{PPO-API}}$	ROUGE-1	**0.798**
	ROUGE-2	**0.793**
	ROUGE-L	**0.797**
	ROUGE-LSUM	**0.798**
	METEOR	**0.831**
	BERTScore-Precision	**0.924**
	BERTScore-Recall	**0.916**
	BERTScore-F1	**0.920**
$flant5_{\text{PPO-RoBERTa}}$	ROUGE-1	0.788
	ROUGE-2	0.784
	ROUGE-L	0.789
	ROUGE-LSUM	0.789
	METEOR	0.803
	BERTScore-Precision	0.893
	BERTScore-Recall	0.889
	BERTScore-F1	0.892

analyze the impact of PPO optimization on paraphrasing quality. Table 12 presents the results for these two models across various metrics. The findings indicate that their performance is quite comparable, with optimization via PPO leading to noticeable improvements in paraphrasing quality. Furthermore, a comparison between the results presented in Table 12 and those in Table 11, which includes all four models (including $flant5_{\text{Zero-shot}}$, $flant5_{LoRA}$, $flant5_{PPO-API}$, and $flant5_{PPO-RoBERTa}$), underscores the significant enhancement in generated response quality achieved through PPO optimization compared to Zero-shot learning or instruct-finetuning.

Furthermore, to assess the ability of the developed models to generate toxic samples, we employed a test set for sentence paraphrasing tasks. We assessed the toxicity of the test set, calculated the average and standard deviation of toxic scores for paraphrasing samples generated by the reference model ($flant5_{\text{RF}}$), and compared it with the toxicity of samples generated by $flant5_{\text{PPO-API}}$ and $flant5_{\text{PPO-RoBERTa}}$. Table 13 outlines the

Table 13: Percentage Enhancement in Toxicity Scores for Paraphrasing Post-Optimization with PPO

Model	Improvement in Toxicity Score (%)	
	Average	Standard Deviation
$flant5_{\text{PPO-RoBERTa}}$	12.35	27.44
$flant5_{\text{PPO-API}}$	**28.42**	25.18

results, demonstrating the enhancement in the toxicity of generated samples through the optimization task. Specifically, $flant5_{\text{PPO-API}}$ could generate more toxic samples compared to $flant5_{\text{PPO-RoBERTa}}$ on average by 21.68% and 7.57%, respectively.

Please note that the toxicity of generated responces for samples in the test set was evaluated by HateBERT [244]. In other words, for samples generated by $flant5_{\text{PPO-API}}$, and $flant5_{\text{PPO-RoBERTa}}$, $HateBERT$ was employed to score toxicity and facilitate comparison between the reference model and the optimized model.

After constructing the models for sentence-level augmentation through paraphrasing, we utilized the toxic datasets detailed in Section 5.3.2 to evaluate our proposed model.

5.3.4.1 Jigsaw

As previously mentioned, the jigsaw toxic dataset comprises 143,346 nontoxic samples and 16,225 toxic samples. The objective is to generate additional toxic samples to achieve a balanced dataset with an almost equal number of toxic and nontoxic samples. To accomplish this, we aimed to augment toxic samples approximately $9 \times 16,225 = 146,025$, considering that not all augmented samples are necessarily toxic, and some may be removed later.

To meet this objective, we chose to generate nine different paraphrases per toxic sample using all instruct-tuned models $flant5_{LoRA}$, $flant5_{\text{PPO-RoBERTa}}$, $flant5_{\text{PPO-API}}$.

Subsequently, we transformed the toxic dataset into an instruction dataset by encapsulating all 16,255 toxic samples with instructions to form prompts. These prompts were then fed into the models with the following generation parameters: minimum length, top-k, top-p, and the number of returned sequences set to $(5, 0.0, 1.0, 9)$. The results will be presented in the results section Section 5.4.

5.3.4.2 ToxiGen

We followed a similar approach as with the ToxiGen dataset, starting with 2,000 toxic samples. Utilizing all models including $flant5_{LoRA}$, $flant5_{\text{PPO-RoBERTa}}$, and $flant5_{\text{PPO-API}}$, we generated an additional 2,000 samples. In other words, we paraphrased every sample in the ToxiGen dataset, resulting in 2,000 augmented samples. The outcomes are detailed in the following section.

5.3.5 Baselines

We evaluate our proposed method for augmenting toxic samples through paraphrasing by comparing it against four baselines. The first baseline is zero-shot learning technique ($flant5_{\text{Zero-shot}}$) , the second one is the instruct-tuned FLAN-T5 model, represented as $flant5_{\text{LoRA}}$. These baseline allows us to observe how the incorporation of RLHF can enhance model performance, especially in the context of toxic paraphrasing.

The third baseline is the optimized model, $flant5_{PPO-RoBERTa}$, using toxicity detector of HateRoBERTa in the reward model. A comparison between our proposed technique and this baseline enables us to assess the impact of different reward models on performance.

Additionally, we aim to broaden our comparison to include other data augmentation techniques, specifically, back-translation. We have already detailed the setup for the first three baselines. In the following subsection, we will guide you through the setup

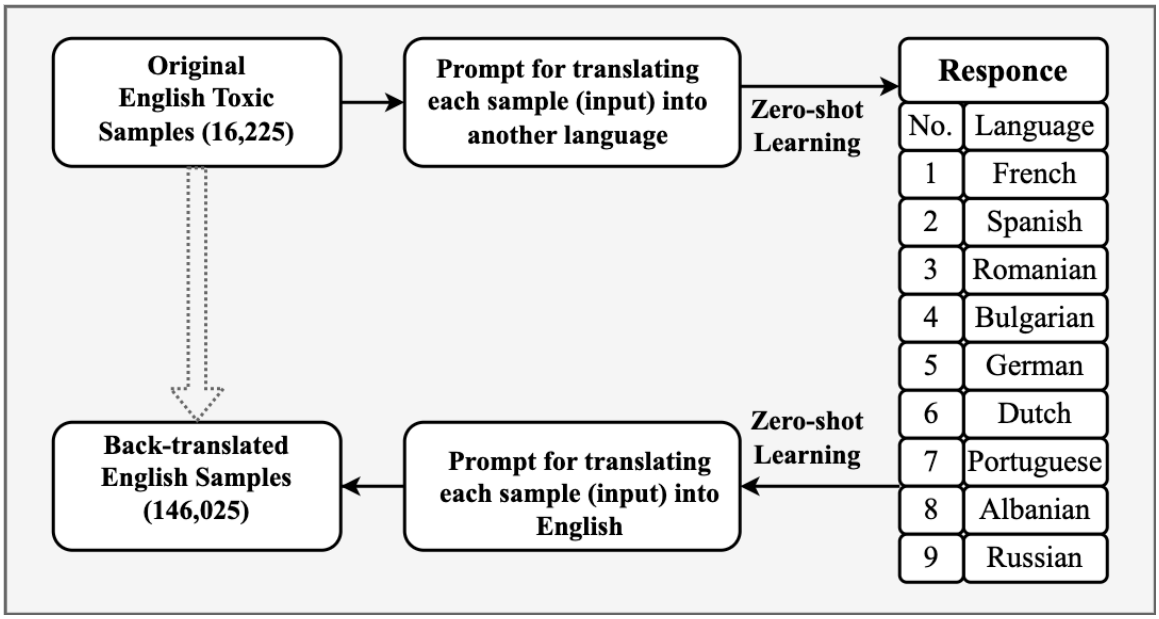


Figure 14: Illustration of the Back-Translation Technique, where English toxic samples are translated into multiple languages and then back into English for data augmentation.

for data augmentation using back-translation.

5.3.5.1 Zero-shot Learning

In our approach to generating toxic text via paraphrasing, we leveraged zero-shot learning, which involves training a model without explicit examples of the task it is meant to perform. We employed "flan-t5-base" or this purpose.

First, we transformed our toxic dataset (Section 5.3.2) into an instruction dataset. This involved encoding each toxic sample along with a prompt requesting its paraphrasing while maintaining the same semantic meaning. This method allowed us to generate toxic text without explicitly providing examples of such text. Instead, the model learned to generate toxic text by understanding the underlying semantics of the provided prompts and applying paraphrasing techniques accordingly.

5.3.5.2 Back Translation

To evaluate our proposed approach against existing text augmentation techniques, we utilized back-translation to generate additional toxic samples. For Jigsaw dataset, our objective was to expand the dataset by creating nine additional samples for each original toxic sample while maintaining a balanced distribution. We employed the flan-t5-base model, known for its multilingual capabilities, for this task. However, we encountered a challenge when translating English toxic samples into nine different versions of a single language, such as French. To overcome this challenge, we opted to translate each English toxic sample into nine different languages and subsequently back into English. Notably, we employed the Zero-shot learning technique for this process. In zero-shot learning, the model is trained to perform a task without explicit examples or training data. In our case, we provided prompts to the flan-t5-base model, instructing it to translate each English toxic sample into nine different languages. Crucially, we did not provide any explicit examples for the model to learn from; instead, it generalized its translation capabilities based on the prompt and input provided. This zero-shot learning technique allowed us to effectively generate diverse translations for each English toxic sample without the need for specific training data in each target language. By back-translating these multilingual translations into English, we were able to augment our dataset with additional diverse toxic samples, enhancing the robustness of our evaluation.

We translate toxic samples into French, Spanish, Romanian, Bulgarian, German, Dutch, Portuguese, Albanian, and Russian, followed by translating all samples back into English. The selection of these languages was determined through trial and error, as we experimented with various languages. Flan-t5-base did not consistently respond appropriately in some instances, generating responses with unintelligible characters, leading us to settle on the aforementioned languages.

We used all toxic samples as input for the prompt requesting translation from English into different languages and vice versa. The method is illustrated in Figure 14.

For the ToxiGen dataset, we followed a similar process, but with a slight modification. Instead of utilizing all nine languages available, we restricted our focus to English and French, along with their respective translations. This decision was made due to the limited availability of toxic prompts, which amounted to only 2,000 entries.

5.3.6 Computational Resources

We utilized the cloud computing instance 'Paperspace P6000' equipped with NVIDIA P6000 GPUs. The P6000 features a GPU with 24 GB memory, 30 GB RAM, 8 vCPUs, and supports multi-GPU instances of 2X and 4X. The cost per hour for each task on the P6000 is $1.10.

The Google Perspective API is free and processes each query per second. To expedite the process, we employed five different API keys simultaneously and divided the Test set into five parts, with each part assigned to a separate API key.

5.4 Experimental Results

Generating Toxic Samples based on Jigsaw dataset

As detailed in section (Section 5.3), the proposed technique ($flant5_{\text{PPO-API}}$) and four baselines ($flant5_{Zero-shot}$, $flant5_{LoRA}$, $flant5_{PPO-RoBERTa}$, $flant5_{Back-translation}$), were employed to generate toxic samples.

A total of "146,025" samples were generated by each of the models $flant5_{\text{PPO-API}}$, $flant5_{\text{PPO-RoBERTa}}$, $flant5_{LoRA}$, $flant5_{Zero-shot}$, and $flant5_{\text{Back-translation}}$. However, not all samples are deemed acceptable, as some may be repetitive, nonsensical, or nontoxic, lacking coherence or logical consistency. Consequently, we conducted a

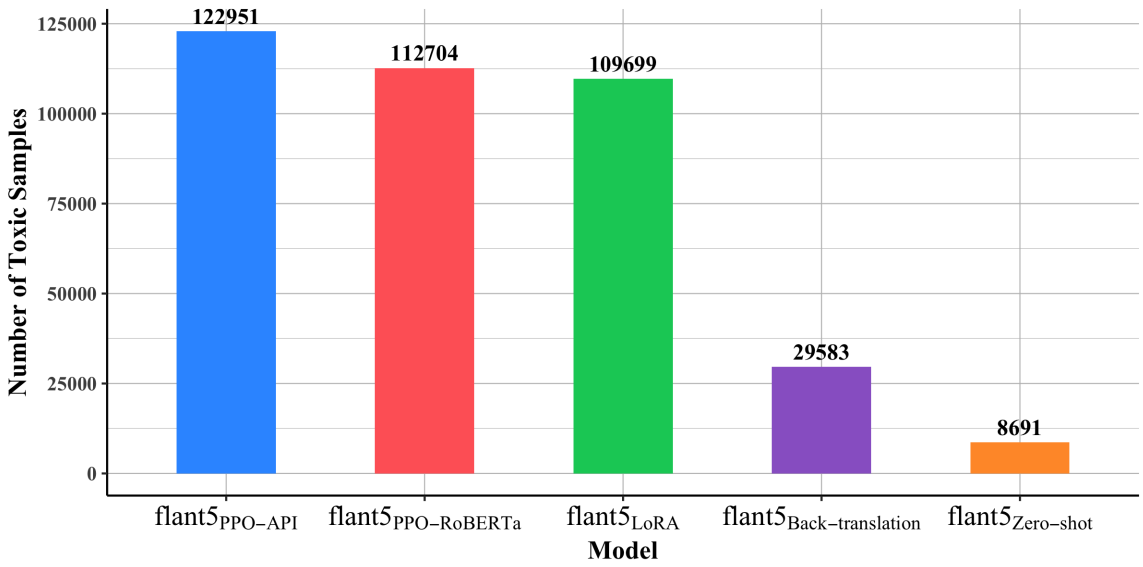


Figure 15: Total Toxic Samples Generated by Each Model: Toxicity Scores ≥ 0.3 (Jigsaw Dataset)

thorough analysis of the generated samples, meticulously selecting only those that are readable and meaningful for both humans and machines. This filtering process ensures the quality and relevance of the data for subsequent tasks and evaluations.

The $flant5_{\text{Zero-shot}}$ model struggled to produce multiple responses per request, leading us to limit each request to one response. Ultimately, it generated 16,225 responses, each corresponding to a single prompt. However, the model encountered difficulty in generating distinct responses. Specifically, for 458 toxic samples, the paraphrased responses were identical. To address this issue, we filtered out the redundant responses, resulting in a final count of "15,763" unique responses.

We leveraged Google Perspective API to evaluate the toxicity of our selected samples. Notably, we also incorporated $HateRoBERTa$ into our analysis to mitigate potential bias, ensuring a comprehensive assessment of toxicity. Interestingly, the results from both evaluators were highly consistent. As such, for the sake of clarity and simplicity in reporting, we focus solely on the findings obtained through Perspective API.

Following evaluation with Perspective API, all samples received toxicity scores ranging from 0 to 1. Subsequent manual inspection revealed an apparent threshold: samples scoring below 0.3 typically exhibited nontoxic characteristics, while those surpassing 0.3 were deemed potentially toxic. As a result, we made the decision to discard samples with scores below 0.3 and focus solely on those with toxicity scores above this threshold. This ensured that only samples exhibiting a significant level of toxicity were included in our analysis.

It is worth noting that during the training phase, we incentivized the model to generate responses with higher toxicity by rewarding samples with scores above 0.7. After experimenting with various thresholds, we determined that a threshold of 0.7 or higher yielded the most satisfactory results during hyperparameter tuning. Consequently, this threshold was chosen as the desired level of toxicity for our study.

Figure 15 below displays the final number of toxic samples generated by each model. The analysis reveals that the proposed model ($flant5_{\text{PPO-API}}$) generates a higher number of toxic samples (122,951) compared to other models. In contrast, $flant5_{\text{Zero-shot}}$ exhibits the poorest performance, generating only 8,691 toxic samples out of 15,763 responses generated. Following this, $flant5_{\text{Back-translation}}$ generates 29,583 toxic samples out of the total 146,025 generated samples. Overall, the instruction-tuned models demonstrate exceptional performance in the task of text data augmentation, with potential for further enhancement through Reinforcement Learning with Human Feedback (RLHF). Back-translation technique did not work well, because toxic language often contains subtle nuances, sarcasm, or contextual references that may not translate accurately or be preserved through this process. As a result, paraphrased versions may fail to capture the original toxicity or convey the intended tone. Moreover, the complexity of toxic language and the need for contextual understanding pose challenges for back-translation models, which may struggle to accurately reproduce the nuanced

Table 14: Composition of Balanced Datasets: Model-Generated Toxic Samples and Random Nontoxic Samples-Jigsaw

Model	Generated Toxic Samples ≥ 0.3	Total Toxic Samples	Nontoxic	Toxic + Nontoxic
$flant5_{\text{Zero-shot}}$	8,691	24,916	24,916	49,832
$flant5_{\text{Back-translation}}$	29,583	45,808	45,808	91,616
$flant5_{LoRA}$	109,699	125,924	125,924	251,848
$flant5_{\text{PPO-RoBERTa}}$	112,704	128,929	128,929	257,858
$flant5_{\text{PPO-API}}$	122,951	139,176	139,176	**278,352**

toxicity present in the original text. Additionally, the effectiveness of back-translation relies on the quality and capabilities of the underlying translation model, which may further limit its suitability for toxic text paraphrasing tasks.

Now, we need to understand how the generated toxic samples (GTS) can impact the performance of classifiers for toxic language detection. Therefore, we first build balanced datasets and then employ some classifiers to test the quality and effectiveness of the balanced dataset. Since we have ensured that the original toxic samples (16,225 toxic samples) are not included in the augmented dataset, we add them to all augmented samples. Subsequently, we randomly select an equal number of nontoxic samples from the toxic dataset in Section 5.3.2 to create a balanced dataset. Note that different models generated different numbers of toxic samples, as illustrated in Figure 15. Therefore, we will have balanced datasets with varying numbers of toxic and nontoxic samples, as shown in Table 14.

Classification of Balanced Datasets Generated from Jigsaw Prompts

After preparing all balanced datasets, our focus shifted to evaluating the quality of the generated samples and determining which dataset could enhance the accuracy of classifiers for toxicity detection. In this phase, we selected four different classifiers; two

Table 15: Classification Results-CNN-Jigsaw

Dataset	Accuracy	Precision	Recall	F1-Score
$Unbalanced_{\text{Jigsaw}}$	94.78%	71.39%	81.25%	76%
$Balanced_{\text{Zero-shot}}$	89.20%	88.42%	90.85%	89.50%
$Balanced_{\text{Back-translation}}$	90.20%	88.85%	91.95%	90.37%
$Balanced_{LoRA}$	93.75%	92.03%	95.79%	93.88%
$Balanced_{\text{PPO-RoBERTa}}$	94.01%	92.22%	96.12%	94.13%
$Balanced_{\text{PPO-API}}$	**95.28%**	**94.23%**	**96.46%**	**95.33%**

Table 16: Classification Results-CNN-FastText-Jigsaw

Dataset	Accuracy	Precision	Recall	F1-Score
$Unbalanced_{\text{Jigsaw}}$	89.98%	50.40%	91.52%	65%
$Balanced_{\text{Zero-shot}}$	90.49%	89.12%	91.63%	90.62%
$Balanced_{\text{Back-translation}}$	91.32%	92.68%	89.72%	91.18%
$Balanced_{LoRA}$	93.35%	90.69%	96.61%	93.56%
$Balanced_{\text{PPO-RoBERTa}}$	94.00%	91.60%	96.31%	94.17%
$Balanced_{\text{PPO-API}}$	**94.23%**	**92.47%**	**96.89%**	**94.35%**

of them are CNN-based, and the other two are transformer-based, to be trained/fine-tuned using the balanced datasets and the original unbalanced toxic dataset. The classifiers include Convolutional Neural Networks (CNN) [55], CNN with FastText embeddings, where FastText embeddings are a vector representation technique created and released by Facebook AI research [113]. Furthermore, our selection of transformer-based models encompassed Bidirectional Encoder Representations from Transformers (BERT) [116], HateBERT [244], a specialized variant of BERT tailored for detecting abusive language in English, and RoBERTa [119], a robust BERT model that employs dynamic masking during pre-training to enhance its performance. Additionally, we incorporated BERTweet [315], a pre-trained language model designed specifically for English Tweets, to further enrich our analysis.

As discussed, we incorporated the primary Jigsaw toxic dataset including 143,346 nontoxic samples, and 16,225 toxic samples to examine the impact of an imbalanced

training set on classifier performance. To mitigate dataset imbalance, we implemented a weighted loss function during training, prioritizing the minority class. This strategy enhances the model's ability to learn from underrepresented data, effectively addressing challenges posed by imbalanced distributions. Traditional methods like oversampling and undersampling, as tested on the Jigsaw dataset, often fall short compared to more sophisticated approaches such as ensemble learning [101]. Undersampling risks losing valuable data and features, potentially degrading model performance, while oversampling may introduce redundancy and overfitting, particularly if not carefully applied. Additionally, both methods may fail to accurately capture the underlying data distribution, leading to biased models and suboptimal performance. Details about the experimental setup of classifiers are mentioned in Chapter 7. The performance of classifiers is then evaluated based on Accuracy, Precision, Recall, and F1-score. All results per classifier are demonstrated in Table 15, Table 16, Table 17, Table 18, Table 19 and Table 20.

The results indicate that the dataset generated by $flant5_{PPO-API}$ significantly enhanced the performance of all classifiers, outperforming alternative versions trained or fine-tuned with different datasets. Classifiers developed using $balanced_{flant5_{PPO-API}}$ achieved the highest metrics in accuracy, precision, recall, and F1-score. Despite addressing the imbalance issue with a weighted loss function, prioritizing F1-score for comparison due to the dataset's imbalance revealed BERT developed by $balanced_{flant5_{\text{PPO-API}}}$ as the top-performing classifier with an outstanding F1-score of **97.28%**. Following closely is HateBERT, fine-tuned by $balanced_{flant5_{\text{PPO-API}}}$, which achieved a notable F1-score of **97.12%**. Conversely, classifiers developed using the $Unbalanced_{\text{Jigsaw}}$ dataset did not yield satisfactory results. Furthermore, classifiers developed with the $Balanced_{\text{PPO-RoBERTa}}$ dataset exhibited good performance, closely trailing $balanced_{flant5_{\text{PPO-API}}}$, but fell short of surpassing our proposed technique.

Table 17: Classification Results-BERT-Jigsaw

Dataset	Accuracy	Precision	Recall	F1-Score
$Unbalanced_{\text{Jigsaw}}$	96.04%	75.75%	89.83%	82.19%
$Balanced_{\text{Zero-shot}}$	94.31%	92.74%	96.14%	94.41%
$Balanced_{\text{Back-translation}}$	95.15%	95.61%	94.66%	95.13%
$Balanced_{LoRA}$	96.70%	96.11%	97.34%	96.72%
$Balanced_{\text{PPO-RoBERTa}}$	96.82%	96.28%	97.41%	96.84%
$Balanced_{\text{PPO-API}}$	**97.27%**	**96.91%**	**97.65%**	**97.28%**

Generating Toxic Samples and Classification with ToxiGen Dataset

We selected 2,000 toxic prompts and utilized various models, including $flant5_{\text{Zero-shot}}$, $flant5_{\text{LoRA}}$, $flant5_{\text{PPO-RoBERTa}}$, $flant5_{\text{PPO-API}}$, and $flant5_{\text{back-translation}}$, to generate toxic samples. It is important to note that we aimed to generate a maximum of 2,000 samples, as only one response was requested per prompt. Subsequently, all generated samples underwent toxicity evaluation using the Perspective API, and only those with a toxicity score of 0.30 or higher were retained. Figure 16 illustrates the distribution of generated toxic samples across different models.

Then, all augmented samples were added to the initially selected toxic samples used as prompts to create a balanced dataset. To achieve balance, an equal number of nontoxic (benign) samples were randomly selected from the ToxiGen dataset. The final results are shown in Table 21. The results indicate that applying reinforcement learning for instruction tuning surpasses other techniques such as zero-shot learning, back-translation, or even simple instruction fine-tuning. Moreover, utilizing the Perspective

Table 18: Classification Results-RoBERTa-Jigsaw

Dataset	Accuracy	Precision	Recall	F1-Score
$Unbalanced_{\text{Jigsaw}}$	96.68%	82.11%	86.19%	84.10%
$Balanced_{\text{Zero-shot}}$	94.27%	93.27%	95.42%	94.33%
$Balanced_{\text{Back-translation}}$	95.68%	95.88%	95.45%	95.67%
$Balanced_{LoRA}$	96.65%	95.53%	97.69%	96.59%
$Balanced_{\text{PPO-RoBERTa}}$	96.65%	95.89%	97.47%	96.67%
$Balanced_{\text{PPO-API}}$	**97.01%**	**96.13%**	**97.97%**	**97.04%**

API as a toxicity evaluator for toxicity reward yields better performance compared to HateRoBERTa. Finally, 1,894 toxic samples were generated by $flant5_{\text{PPO-API}}$, which, combined with the initial 2,000 toxic prompts, resulted in a total of 3,894 toxic samples. Subsequently, a balanced dataset comprising 7,788 samples, both toxic and nontoxic, was created.

After creating balanced datasets from ToxiGen, various classifiers, including CNN, CNN-FastText, BERT, RoBERTa, HateBERT, and BERTweet, were trained on the complete datasets. Each dataset was carefully split into training and testing sets, maintaining an 80% to 20% ratio, respectively. The performance metrics and classification results of these classifiers can be found in Table 22, and Table 23.

The classification results for datasets developed using ToxiGen prompts show that classifiers trained on the $Balanced_{\text{PPO-API}}$ dataset outperform other models. Overall, transformer-based models outperform CNN-based models. Specifically, BERT fine-tuned with the $Balanced_{\text{PPO-API}}$ dataset achieves the best performance with an accuracy of 97.98% and an F1-score of 98.05%. In contrast, the CNN classifier trained

Table 19: Classification Results-HateBERT-Jigsaw

Dataset	Accuracy	Precision	Recall	F1-Score
$Unbalanced_{\text{Jigsaw}}$	96.52%	82.9%	86.11%	84.55%
$Balanced_{\text{Zero-shot}}$	94.32%	93.24%	95.56%	94.39%
$Balanced_{\text{Back-translation}}$	95.35%	95.51%	95.16%	95.34%
$Balanced_{LoRA}$	96.72%	95.88%	97.63%	96.74%
$Balanced_{\text{PPO-RoBERTa}}$	96.75%	96.07%	97.48%	96.77%
$Balanced_{\text{PPO-API}}$	**97.03%**	**96.20%**	**97.98%**	**97.12%**

on the $Balanced_{\text{Zero-shot}}$ dataset performs the worst, with an accuracy of 90.18% and an F1-score of 90.24%.

To conclude, our proposed approach to text data augmentation for constructing a balanced dataset has not only demonstrated its effectiveness in generating a larger quantity of high-quality toxic samples but also led to enhanced performance across a diverse range of evaluation metrics. This augmentation technique, leveraging state-of-the-art language models and reinforcement learning, provides a straightforward and effective strategy for addressing imbalances in toxic datasets, contributing to superior classifier performance in toxicity detection tasks.

However, an important question arises regarding the significance of our improvements. Despite our model consistently outperforming the baseline methods, the progress might seem modest, especially considering that the baselines were already quite effective. It is crucial to emphasize that even small improvements in accuracy and performance have significant implications, particularly in areas such as detecting toxic content in online conversations. Additionally, the ability of our method to create

Table 20: Classification Results-BERTweet-Jigsaw

Dataset	Accuracy	Precision	Recall	F1-Score
$Unbalanced_{\text{Jigsaw}}$	96.41%	82.45%	86.03%	84.21%
$Balanced_{\text{Zero-shot}}$	94.22%	93.19%	95.44%	94.24%
$Balanced_{\text{Back-translation}}$	95.48%	95.49%	95.12%	95.30%
$Balanced_{LoRA}$	96.68%	95.83%	97.60%	96.64%
$Balanced_{\text{PPO-RoBERTa}}$	96.71%	96.67%	97.41%	96.96%
$Balanced_{\text{PPO-API}}$	**97.02%**	**96.26%**	**97.95%**	**97.00%**

Table 21: Composition of Balanced Datasets: Model-Generated Toxic Samples and Random Nontoxic Samples-ToxiGen

Model	Generated Toxic Samples ≥ 0.3	Total Toxic Samples	Nontoxic	Toxic + Nontoxic
$flant5_{\text{Zero-shot}}$	139	2,139	2,139	4,278
$flant5_{\text{Back-translation}}$	341	2,341	2,341	4,682
$flant5_{LoRA}$	804	2,804	2,804	5,608
$flant5_{\text{PPO-RoBERTa}}$	1,682	3,682	3,682	7,364
$flant5_{\text{PPO-API}}$	**1,894**	3,894	3,894	**7,788**

more toxic samples is extremely useful, especially when there are not many toxic examples available.

In terms of the computational aspect, we understand the importance of balancing the complexity of our model with the available resources. Our approach includes methods such as PEFT to reduce computational overhead and make better use of memory during training. Furthermore, we use RLHF to improve our model's performance while keeping computational demands low.

In summary, our model significantly enhances toxicity detection by addressing data imbalances and increasing resilience.

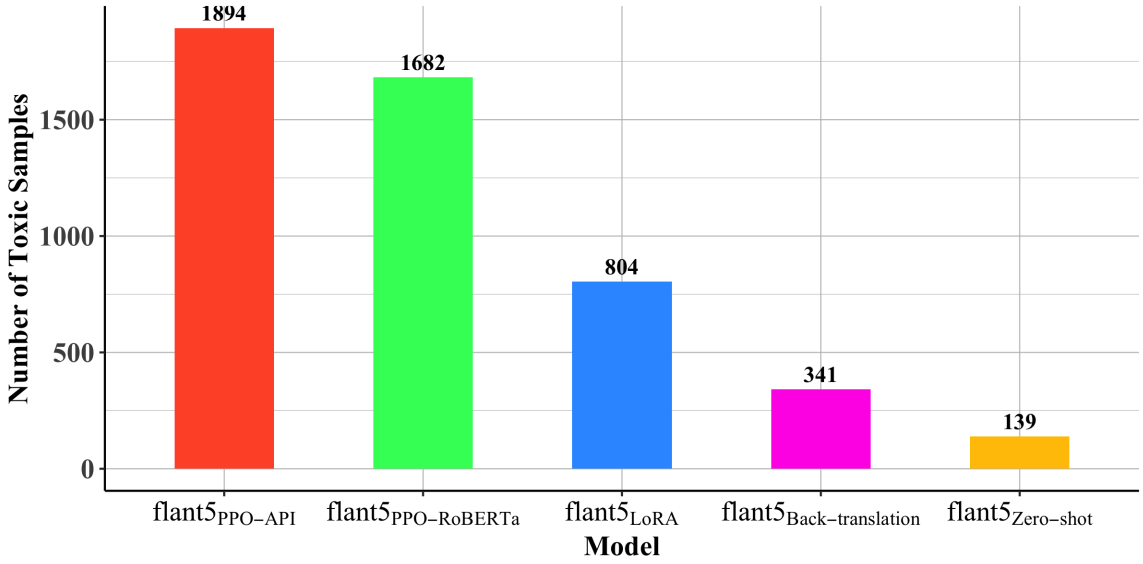


Figure 16: Total Toxic Samples Generated by Each Model: Toxicity Scores ≥ 0.3 (ToxiGen Dataset)

5.5 Summary

In conclusion, this chapter presents a novel approach to sentence-level text data augmentation, employing reinforcement learning guided by human feedback to enhance the performance of the fine-tuned FLAN-T5 model. By prioritizing paraphrasing while maintaining semantic coherence and generating toxic responses, our method effectively tackles the challenges posed by imbalanced datasets in toxic language detection. Central to our approach is the utilization of Proximal Policy Optimization as a reinforcement learning technique, coupled with the Google Perspective API as a toxicity evaluator and the integration of Kullback-Leibler Divergence. These components synergistically produce high-quality toxic responses, yielding a balanced and diverse dataset that outperforms existing data augmentation methods. Through a comprehensive exploration of various methodologies for toxic text generation, including Zero-shot learning, Back-translation, and instruc-tuned FLAN-T5, optimized with Proximal Policy Optimization (PPO) and evaluated using different toxicity evaluators

Table 22: CNN-Based Classification Performance-ToxiGen

Model	Dataset	Accuracy	Precision	Recall	F1-Score
CNN	$Balanced_{\text{Zero-shot}}$	90.18%	89.31%	91.67%	90.24%
	$Balanced_{\text{Back-translation}}$	91.34%	89.97%	93.78%	91.42%
	$Balanced_{LoRA}$	92.40%	91.73%	94.15%	92.61%
	$Balanced_{\text{PPO-RoBERTa}}$	93.36%	91.58%	94.28%	93.71%
	$Balanced_{\text{PPO-API}}$	**94.30%**	**93.86%**	**95.75%**	**94.62%**
CNN-FastText	$Balanced_{\text{Zero-shot}}$	91.20%	90.42%	92.85%	91.50%
	$Balanced_{\text{Back-translation}}$	92.31%	91.27%	93.84%	92.69%
	$Balanced_{LoRA}$	93.75%	91.03%	94.79%	93.88%
	$Balanced_{\text{PPO-RoBERTa}}$	94.01%	94.22%	95.12%	94.24%
	$Balanced_{\text{PPO-API}}$	**95.28%**	**94.10%**	**96.73%**	**95.47%**

such as HateRoBERTa and the Google Perspective API, we have demonstrated the superiority of our proposed framework. Our findings, validated across two distinct toxic datasets, Jigsaw and ToxiGen, consistently point to the efficacy of instruc-tuned FLAN-T5 with PPO and Perspective API as a superior approach. Our approach has yielded impressive results, generating 122,951 toxic samples with a toxicity score exceeding 30%, highlighting its potential to significantly advance toxicity detection models in online conversations. Notably, over 20,000 of the generated samples exhibit toxicity levels exceeding 90%, further emphasizing the impact of our method on enhancing toxicity detection models. However, it is essential to acknowledge the limitations of our approach, which we aim to address in future research endeavors. Specifically, the simplification of the Jigsaw toxic dataset into two broad categories limits the granularity of our analysis.

Acknowledgments

This research is supported by NSERC Discovery Grants (RGPIN-2024-04087) and Canada Research Chairs Program (CRC-2019-00041).

Table 23: Transformer-Based Classification Performance-ToxiGen

Model	Dataset	Accuracy	Precision	Recall	F1-Score
BERT	$Balanced_{\text{Zero-shot}}$	92.46%	91.25%	92.84%	92.39%
	$Balanced_{\text{Back-translation}}$	92.71%	91.44%	93.06%	92.68%
	$Balanced_{LoRA}$	96.88%	95.03%	96.87%	96.75%
	$Balanced_{\text{PPO-RoBERTa}}$	97.52%	97.12%	96.22%	97.43%
	$Balanced_{\text{PPO-API}}$	**97.98%**	**96.23%**	**98.26%**	**98.05**%
RoBERTa	$Balanced_{\text{Zero-shot}}$	92.01%	90.34%	93.59%	92.27%
	$Balanced_{\text{Back-translation}}$	92.68%	91.38%	92.91%	92.46%
	$Balanced_{LoRA}$	95.88%	94.39%	95.97%	95.76%
	$Balanced_{\text{PPO-RoBERTa}}$	96.83%	96.22%	97.12%	96.43%
	$Balanced_{\text{PPO-API}}$	**97.31%**	**96.04%**	**97.59%**	**97.29**%
BERTweet	$Balanced_{\text{Zero-shot}}$	92.27%	93.59%	90.34%	92.16%
	$Balanced_{\text{Back-translation}}$	92.23%	93.83%	91.48%	92.57%
	$Balanced_{LoRA}$	96.05%	94.83%	96.11%	95.98%
	$Balanced_{\text{PPO-RoBERTa}}$	96.94%	96.17%	97.29%	96.88%
	$Balanced_{\text{PPO-API}}$	**97.36%**	**96.54%**	**97.62%**	**97.34**%
HateBERT	$Balanced_{\text{Zero-shot}}$	92.35%	91.22%	92.76%	92.27%
	$Balanced_{\text{Back-translation}}$	92.65%	91.39%	93.01%	92.59%
	$Balanced_{LoRA}$	96.81%	94.98%	96.79%	96.67%
	$Balanced_{\text{PPO-RoBERTa}}$	97.47%	97.02%	96.15%	97.25%
	$Balanced_{\text{PPO-API}}$	**97.85%**	**97.19%**	**98.13%**	**97.92**%

Chapter 6

Real-Time Adaptive Toxicity Detection with Cascaded Classifiers Optimized by Proximal Policy Optimization

This chapter is dedicated to the article entitled *"Real-Time Adaptive Toxicity Detection with Cascaded Classifiers Optimized by Proximal Policy Optimization"*. This article was submitted to the *IEEE Transactions on Computational Social Systems* . The titles, figures, and mathematical formulations have been revised to keep the coherence through the manuscript.

Managing toxic content, such as hate speech and harassment, on online platforms is a critical and complex task due to the impracticality of manual moderation and the sheer volume of data. Current moderation systems employ a combination of human moderators and automated tools using Machine Learning (ML) algorithms. However, these systems often face difficulties in balancing classification accuracy and speed, leading to frequent misclassifications that increase the workload for human moderators and result in longer overall process times. These delays impact user satisfaction, as users who wish to avoid toxic content desire it to be hidden as quickly as possible. To address these challenges, we propose a novel framework for toxicity detection in user-generated content, leveraging cascaded classifiers optimized through deep reinforcement learning. Our framework, *Proximal Policy Optimization-based Cascaded Inference System* (*PPO-CIS*), dynamically assigns classifiers based on their performance and computational cost, utilizing high-throughput classifiers for initial filtering and more accurate classifiers for final decision-making. By employing Proximal Policy Optimization, our deep reinforcement learning (DRL)-based system adapts to varying data volumes and classifier performances, ensuring efficient and accurate content moderation. Extensive evaluations on two different datasets, $Kaggle$ and $ToxiGen$, considering all possible combinations of classifiers with various label determination techniques, including majority voting, soft voting, and sequential output, as well as different policy-based DRL approaches with various reward functions, demonstrate the effectiveness of PPO-CIS. The results show significant improvements in both processing time and detection accuracy. This work paves the way for more scalable and cost-effective content moderation systems, enhancing user satisfaction and reducing the burden on human moderators.

6.1 Problem Statement & Contributions

Every platform that enables user interaction faces the complexities of handling toxicity including hate speech, profanity, offensive language, slurs, personal attacks, defamatory statements, bullying, and harassment [316], [317]. Addressing this issue is difficult because manually removing toxic content is impractical, given the sheer volume, speed, and diversity of online material [1].

Social media content moderation currently relies on a collaborative approach between human moderators and automated tools [318]. Platforms have adopted Machine Learning (ML) and Deep Learning (DL) algorithms to analyze user-generated content (UGC), generating predictive scores and classifying content into different categories [20]–[22]. Depending on these scores or classifications, UGC might be flagged and sent to a human moderator (HM) for final assessment on whether it should remain visible or be hidden [319]. Content moderation systems (CMS) face the daunting task of managing a vast volume of UGC every second. The rapid expansion of social networking services over the past decade has led to an explosion of data from millions of users, making it difficult to efficiently process and moderate the constant influx of information [15]. Social media platforms have seen a significant increase in user numbers over the years, reflecting a corresponding rise in overall usage. For instance, in 2022, Facebook and YouTube boasted nearly 2.85 billion and 2 billion users, respectively, while X (called Twitter) and Reddit had 350 million and 430 million users [16]. X alone records 192 million daily active users, averaging 6,000 tweets per second, 350,000 per minute, and 500 million tweets daily, with numbers surging to over 140,000 tweets per second during events such as natural disasters or breaking news [14], [320]. Fast forward to April 2024, the numbers have grown substantially: Facebook now leads with 3.065 billion monthly active users, followed

by YouTube with 2.504 billion users, while other platforms such as X have also seen significant growth, reaching 661 million users [321]. This surge in user numbers indicates higher engagement on these platforms, with the global average time spent on social media daily now at 2 hours and 24 minutes [322]. Moreover, under recent regulations such as the German NetzDG or the EU Code of Conduct on Hate Speech, platforms are compelled to rapidly remove these contents, driving them to rely on automated systems for proactive and large-scale detection of illegal or problematic material [316].

Due to the substantial potential for incorrect labels (false positive or false negative) leading to the removal of innocent users and their content, there is a noteworthy emphasis on auditing, evaluating, and improving ML/DL methods employed in CMS [1], [323]. Traditional ML methods often struggle with complex data characteristics such as high-dimensionality, imbalances, and noise, which can lead to higher rates of misclassification [54]. Ensemble classifiers, widely adopted for their ability to achieve accurate, robust, and efficient performance in analyzing large and diverse datasets, offer a compelling solution [324].

Ensemble learning, also known as multiple classifier systems or committee-based learning, combines multiple learners to address learning challenges effectively [325]. Given the rapid data generation (high velocity), vast amount of data (large volume), and diverse data types and sources (variety) on social media platforms, ensemble learning stands out as a superior approach for processing, analyzing, and evaluating incoming data compared to single-model approaches [136].

In toxic content detection, ensemble techniques such as bagging [326], boosting [327], and stacking [328] have proven effective. A specific ensemble classifier type, the cascade of binary classifiers, arranges ML/DL models sequentially to enhance detection performance [329], [330]. In this approach, the output of one classifier

serves as the input for the next classifier. This method has been applied in various domains such as object detection and sentiment analysis [143], [331], [332]. Using multiple classifiers in a cascaded manner can significantly reduce storage space and computation time, resulting in faster processing and increased user satisfaction [333]. By cascading several classifiers with increasing complexity, the first few classifiers remove most of the easy negatives, while the more accurate and complex ones at the end of the cascade provide excellent discrimination and yield good overall performance [334].

In the context of toxicity detection, a tree of classifiers can significantly enhance the classification process by balancing accuracy and speed at scale. Given the massive volume and high velocity of the data that platforms receive every second, it is critical to process information both accurately and efficiently. Typically, improving accuracy requires more complex architectures that slow down processing, while simplifying architectures to boost speed can reduce accuracy [335].

To address this challenge, we propose an efficient multi-stage inference system for toxicity detection, utilizing a dynamic cascade of classifiers. Initially, high-throughput classifiers quickly process all data, categorizing it into toxic and nontoxic samples. Toxic samples then proceed to a second stage with highly accurate but lower-throughput classifiers. Depending on the query, at least one classifier will process each sample in each stage, and in some cases, multiple classifiers might be employed for more thorough consideration. This approach optimizes processing time while maintaining accurate classification of toxic content.

However, utilizing multiple ML/DL models for each sample can significantly increase processing time [142]. Selecting the best set of classifiers for the cascade is challenging due to variations in accuracy, processing time, and computational cost among classifiers. To improve the effectiveness and efficiency of these cascades, we

employ deep reinforcement learning (DRL) [53] to identify optimal cascaded models that can analyze user-generated content (UGC) effectively.

By discovering the most cost-effective set of cascaded classifiers for toxicity detection, we introduce an adaptive multi-stage inference system optimized by DRL, specifically Proximal Policy Optimization (PPO) [58]. This system dynamically adjusts classifier selection based on real-time confidence scores, ensuring that each sample is processed by the most suitable classifiers at each stage. To the best of my knowledge, this is the first DRL-based inference system for toxicity detection that successfully balances throughput and accuracy.

The proposed Proximal Policy Optimization-based Cascaded Inference System (PPO-CIS) represents a significant advancement in the field of toxicity detection, being the first framework to leverage deep reinforcement learning (DRL) for dynamically optimizing a cascade of classifiers. Unlike static baselines, which rely on single classifiers or predetermined combinations of classifiers, PPO-CIS dynamically adjusts classifier selection based on real-time confidence scores, optimizing both detection accuracy and efficiency. This adaptive approach ensures superior performance over fixed configurations in real-world applications. Our extensive evaluations on multiple datasets demonstrate that PPO-CIS achieves a unique balance between throughput and accuracy, surpassing existing methods. Specifically, PPO-CIS improves accuracy by approximately 2.10% and increases throughput from 42.74 to 384 samples per second compared to static baselines. When benchmarked against CETRA [139], a state-of-the-art DRL-based system for malware detection, PPO-CIS shows significant improvements, enhancing accuracy by 0.37% and boosting throughput by around 107.57%. These results underscore the framework's effectiveness in rapidly and accurately identifying toxic content, providing a scalable and cost-effective solution for real-time content moderation. This pioneering work not only advances the methodology

for toxicity detection but also establishes a new benchmark in the application of DRL for optimizing classifier cascades, offering a robust, efficient, and adaptive system for large-scale social media platforms.

These results highlight PPO-CIS's capability to achieve higher accuracy and throughput, making it highly effective for various deployment scenarios. Summary of Our Contributions:

- **Dynamic Cascade of Classifiers:** Introduced a novel approach to toxicity detection using a dynamic cascade of classifiers, optimized through Proximal Policy Optimization (PPO), a deep reinforcement learning (DRL) method.
- **Enhanced Classification System:** Developed an innovative inference system where high-throughput classifiers perform initial filtering, followed by highly accurate classifiers for final detection, improving both speed and accuracy.
- **Optimized DRL Reward Function:** Designed a reward function within the DRL framework that minimizes processing time and enhances classification accuracy by reducing false positives and false negatives.
- **Comprehensive Evaluation:** Conducted extensive evaluations on two datasets (Kaggle and ToxiGen), demonstrating significant improvements in processing time and detection accuracy with the proposed PPO-CIS framework.
- **Balance of Throughput and Accuracy:** Achieved a unique balance of throughput and accuracy, addressing the challenge of processing high volumes of data quickly without sacrificing detection precision.
- **Scalable Real-Time Content Moderation:** Provided a scalable, cost-effective solution for real-time content moderation, reducing the burden on human moderators and enhancing user satisfaction by efficiently identifying toxic content.

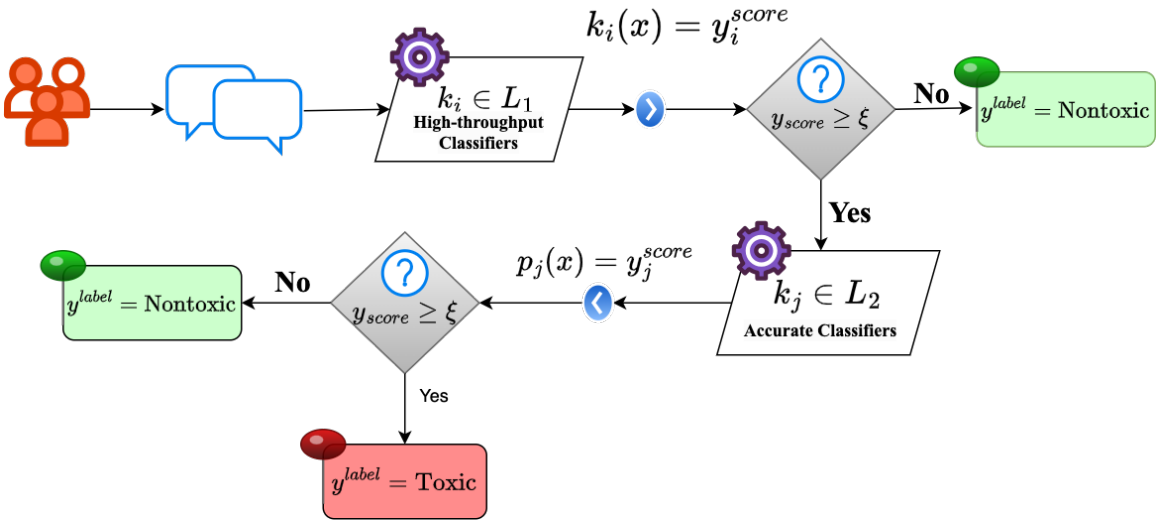


Figure 17: Architecture of Multi-Stage Cascade Inference Systems

The rest of this chapter is structured as follows. The proposed methodology is detailed in Section 6.2. Section 6.3 presents the experimental setup, and Section 6.4 analyzes the experimental results. Finally, Section 6.5 summarizes the key findings and conclusions of the work.

6.2 Methodology

In this section, I present the structure of my proposed approach. I start by constructing a multi-stage cascade of classifiers tailored for toxicity detection, as illustrated in Figure 17 within Section 6.2.1. To optimize the performance of these cascades, we employ Proximal Policy Optimization (PPO) combined with a specially designed reward function, depicted in in Figure 18 (see Section 6.2.2). This methodology aims to achieve higher throughput and more accurate classification of toxic content.

6.2.1 Cascaded Inference Systems

The proposed multi-stage cascade inference system is designed to efficiently and accurately detect toxic content in real-time social media applications. This system utilizes a hierarchical approach with multiple stages of classifiers, where each stage is tailored to balance speed (throughput) and accuracy. The architecture of the proposed CIS is illustrated in Figure 17. The initial stage employs high-throughput classifiers to quickly process the bulk of the data, while the subsequent stages use more accurate classifiers to refine the detection of toxic content. This system can effectively handle the massive volume and velocity of UGC.

Let $K = \{k_1, k_2, \ldots, k_n\}$, with $n \subseteq \mathbb{N}$ represent a set of classifiers developed for toxic content detection, each assessed for prediction quality based on true positive (TP), true negative (TN), false positive (FP), and false negative (FN) counts, alongside resource usage costs such as runtime, CPU, and memory.

- **Accuracy** (α): $\frac{TP+TN}{TP+TN+FP+FN}$
- **Precision** (ρ): $\frac{TP}{TP+FP}$
- **Recall** (σ): $\frac{TP}{TP+FN}$
- **F-measure** (ϕ): $2 \cdot \frac{\rho \cdot \sigma}{\rho+\sigma}$
- **Throughput** (ψ): Number of samples processed per unit of time, crucial for fast predictions.

$$\psi_{k_i} = \frac{\text{total number of samples processed by } k_i}{\text{time period}}$$

- **Individual Instance Latency ($\lambda_{q_{k_i}}$)**: Latency for each query q_i processed by classifier k_i.

- **Average Latency per Classifier (λ_{k_i})**: Average latency across all queries processed by classifier k_i.

- **Additional Costs (ω)**: Costs incurred during processing, such as CPU usage or instance costs. For each classifier k_i, additional costs are denoted as ω_{k_i}.

Therefore, the performance of each classifier $k_i \in K$ is evaluated using the tuple:

$$(\alpha_{k_i}, \rho_{k_i}, \sigma_{k_i}, \phi_{k_i}, \psi_{k_i}, \lambda_{k_i}, \omega_{k_i})$$

These metrics offer a comprehensive comparison of classifier effectiveness and efficiency in toxic content detection, focusing on operational costs per classifier.

Once the classifiers are developed, we select $k_i \in K$ based on their throughput (ψ), accuracy (α), and F-measure (ϕ). We consider classifiers with high throughput, high accuracy, and a balance between accuracy and throughput. This results in three sets of classifiers including Most Accurate classifiers (V_{acc}), Moderately Accurate classifiers (V_{mod}) and High-Throughput classifiers(V_{fast}):

$V_{\text{acc}} = \{k_i \in K \mid \max(\alpha_{k_i})\}$

$V_{\text{mod}} = \{k_j \in K \mid \text{moderate}(\alpha_{k_j}) \text{ and moderate}(\psi_{k_j})\}$

$V_{\text{fast}} = \{k_k \in K \mid \max(\psi_{k_k})\}$

where $V_{\text{acc}} \cap V_{\text{mod}} \cap V_{\text{fast}} = \emptyset$.

In social media applications, real-time processing is critical, as users expect to see their content quickly [16], [67]. Therefore, in the first stage, we use classifiers from

$L_1 = V_{\text{fast}} \cup V_{\text{mod}}$ due to their higher throughput. When the CIS receives a set of samples $D = \{x_1, x_2, \ldots, x_z\}$, with $z \subseteq \mathbb{N}$, including toxic and nontoxic samples, at least one classifier $k_i \in L_1$ will be employed to generate a toxicity score (y_i^{score}).

The nontoxic samples $D_{\text{nontox}} \subseteq D$ are those with a high probability of being nontoxic and y_i^{score} is below a predefined threshold (ξ), while $D_{\text{tox}} = D \setminus D_{\text{nontox}}$ contains potentially toxic samples when the y_i^{score} meets or exceeds the (ξ). If necessary, another classifier $k_j \in L_1$ ($i \neq j$) will be applied to samples in D_{nontox} to confirm their status, ensuring that nontoxic samples are correctly identified and exit the process. Toxic samples are then forwarded to the second stage.

In the second stage, classifiers from $L_2 = V_{\text{acc}} \cup V_{\text{mod}}$ are used, where $L_1 \cap L_2 = \emptyset$. This ensures that no classifier is used in both stages, maintaining the uniqueness of each stage's classifiers. Classifiers in the second stage are more accurate but have lower throughput compared to the first stage. This balance allows for trade-offs between accuracy and speed, depending on the number of samples. Each classifier in L_2 processes the flagged input from the first stage and assigns a refined toxicity score (y_j^{score}). Based on this score, the content is finally classified as either toxic or nontoxic. Content classified as nontoxic at either stage is labeled accordingly and does not proceed further, while content confirmed as toxic in the second stage is labeled as toxic and can be handled or removed as necessary.

This adaptive approach ensures that the system can quickly filter out nontoxic content using high-throughput classifiers and thoroughly analyze potentially toxic content with more accurate classifiers, effectively balancing speed and accuracy.

However, manually selecting the best set of classifiers can be challenging. If $|L_1| = m_1$ classifiers and $|L_2| = m_2$ classifiers, the total number of possible cases where at least one classifier is selected from the union of these two sets $|L_1| + |L_2| = m_1 + m_2$ can be determined using the principle of combinations. The total number of ways to

select any subset of classifiers (including the empty set) from these $m_1 + m_2$ classifiers is $2^{m_1+m_2}$. Since we need at least one classifier to be selected, we must subtract the one case where no classifiers are selected. Thus, the total number of possible cases where at least one classifier is selected from the union of $|L_1| + |L_2| = m_1 + m_2$ is $2^{m_1+m_2} - 1$. This makes it difficult to find an ideal solution for various situations. Furthermore, applying the entire ensemble to every sample is costly and often unnecessary.

6.2.2 DRL-based Cascade Inference Systems

Since each classifier has its own features, in some cases, processing a query with several classifiers can result in a more accurate response. Assume that classifiers in L_1 all have high throughput, which is useful for processing large volumes of data within a second. Among these classifiers, one can have the highest ψ while its α might be less compared to other $k_i \in L_1$. There is another classifier k_j (where $i \neq j$) whose ψ_j is the least throughput in L_1 but still acceptable for analyzing large volumes of data in seconds, and it has the highest α in L_1. There are also other classifiers in L_1 where $\psi < \max(\psi)$ and $\alpha < \max(\alpha)$.

Now, assume that $x \in D$ is processed by $k_1 \in L_1$ and the prediction score k_1^{score} indicates nontoxic content. If another classifier such as k_2 analyzes x, the prediction score might either reinforce the nontoxic label or suggest toxic content. If it is closer to nontoxic, the probability of a nontoxic label for x is higher. If the score is closer to toxic, it is better to call $k_3 \in L_1$ to continue the analysis and get the score with the highest confidence. Finally, those $x \in D$ that are detected as nontoxic with high confidence can be made visible to users and removed from further processing. This reduces the number of samples requiring more processing, allowing us to use classifiers in L_2 with lower throughput but higher accuracy. The same process is repeated in the second stage to get the final decision with a high confidence score.

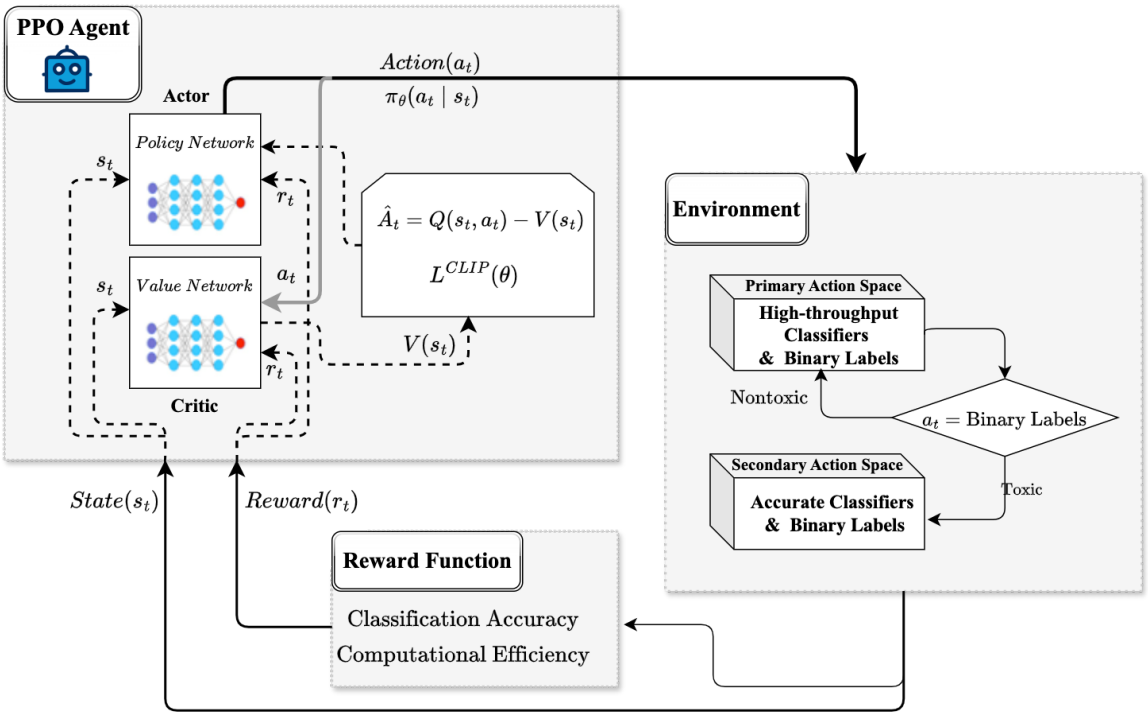


Figure 18: Architecture of the proposed PPO-based cascade inference system for dynamic selection of classifiers

Since manually selecting classifiers is challenging and costly, we use PPO (Proximal Policy Optimization) as a DRL-based approach to handle this multi-decision-making process.

We present PPO-CIS, depicted in Figure 18, a cascade inference system based on deep reinforcement learning using Proximal Policy Optimization (PPO). PPO-CIS is designed for high-throughput and accurate classification of online toxicity, serving as an early phase in content moderation systems. Instead of deploying a single classifier or all at once, the agent dynamically selects which additional classifiers (if any) to call based on the results of previous ones, and moves to the next stage where more accurate but lower-throughput classifiers analyze the samples using the same rule as in the first stage. Using this approach, the classifiers in the second stage, which have lower throughput, receive fewer samples, allowing them to handle those samples more effectively. Below, we describe the states, actions, and rewards of our proposed

PPO-CIS.

States: The state space S consists of all possible scenarios the algorithm may encounter. For the first stage with $|L_1| = m_1$ classifiers, there are $2^{m_1} - 1$ possible selections of at least one classifier, each associated with a confidence score y_1^{score}. Based on this initial confidence score, the classifiers in the second stage $|L_2| = m_2$ are activated. Consequently, the overall environment includes $|S| = 2^{m_1+m_2} - 1$ states. When at least one classifier in the second stage is activated, a second confidence score y_2^{score} is achieved. It is important to note that the order of classifiers applied in the cascade is not fixed but is dynamically selected to optimize performance.

For a cascade consisting of $H \subseteq K$ classifiers, each possible state $s \in S$ is represented by a vector $\beta = \{\beta_1, \beta_2, \ldots, \beta_H\}$, with the value of β_h set by:

$$\beta_h = \begin{cases} [0, 1] & \text{if the classifier is employed} \\ -1 & \text{if the classifier is not employed} \end{cases}$$

Action Space: In the PPO-CIS, the action space is designed to adapt dynamically based on the stage of classification. Initially, the action space A_1 includes selecting one of the m_1 classifiers in L_1 or making a classification decision (toxic or nontoxic), resulting in $|A_1| = m_1 + 2$ possible actions. If the sample is classified as **toxic** in the first stage, the action space changes to A_2, where the actions include selecting one of the classifiers in L_2 or making a final classification decision. This design ensures that the agent can make more granular decisions initially and refine these decisions in subsequent stages.

Thus, the action spaces are defined as follows:

- **Termination Action Space (A_T)**:
  - A_T consists of two actions:

    - ∗ **Classify as Toxic**
    - ∗ **Classify as Nontoxic**

- **First Stage Action Space (A_1)**:

$$A_1 = |L_1| + 2$$

  The available actions for the agent at the first stage include:

  - Applying any classifier $k_i \in L_1$
  - Classifying a sample as nontoxic
  - Classifying a sample as toxic

  If the agent classifies a sample as nontoxic (taking the action of nontoxic $\in A_T$), the process terminates for that sample.

- **Second Stage Action Space (A_2)**:

$$A_2 = |L_2| + 2$$

  The available actions for the agent at the second stage include:

  - Applying any classifier $k_j \in L_2$
  - Classifying a sample as nontoxic
  - Classifying a sample as toxic (which must be reviewed by moderators)

The design of these action spaces enables the agent to dynamically select the most appropriate classifiers and make classification decisions that maximize the confidence in correct labeling while minimizing processing time. By structuring the action spaces

in stages, the system ensures that nontoxic samples are efficiently filtered out, and toxic samples receive thorough and detailed analysis.

Rewards: The reward function in our PPO-CIS is designed to balance the need for accurate classification with the efficiency of processing time and resource utilization. The rewards are structured to reflect the importance of minimizing classification errors and optimizing computational resources. The following factors are considered to define the reward function:

1. Classification Accuracy Reward: Correct classifications (TP and True TN) are rewarded, while incorrect classifications (FP and FN) incur penalties. This ensures that the system prioritizes accuracy. Different weights can be assigned to each type of error based on the organization's policy. For instance, FN (where toxic content is misclassified as nontoxic) might incur a higher penalty due to the potential harm of exposing toxic content to users. The classification reward is defined as follows:

$$R_{\text{class}}(x, y) = \begin{cases} +r & \text{if } TP, TN \\ -\iota r & \text{if } FP, FN \end{cases} \tag{13}$$

where:

- r is the reward for a correct classification (TP or TN).
- ι is a penalty factor ($\iota > 1$) to differentiate the severity of misclassifications. For example, ι can be higher for FN to reflect the higher penalty due to the potential harm of exposing toxic content.
- x is the predicted label.
- y is the true label.

This reward structure allows the agent to prioritize accuracy by rewarding correct

classifications and penalizing incorrect ones, with adjustable weights to reflect the relative importance of different types of errors.

2. Computational Efficiency Reward: The efficiency of processing user-generated content (UGC) is measured in terms of latency. To incentivize quick processing and penalize delays, we propose a reward function that rewards lower latency and applies penalties for higher latency. This ensures that the system can handle large volumes of data efficiently.

The reward function $R_{\text{time}}(t)$ is defined as follows:

$$R_{\text{time}}(t) = \begin{cases} \varphi + \vartheta_1 \cdot \log_2(u - t) & \text{if } 0 \leq t < u \\ \varphi + \vartheta_1 \cdot \log_2(t - u) & \text{if } t \geq u \end{cases} \tag{14}$$

where:

- t is the processing time for a sample.
- u is the upper limit of acceptable processing time.
- φ is a constant offset to ensure a baseline reward, set to $\varphi \geq 1$.
- ϑ_1 is a scaling factor to amplify the differences between rewards and penalties, set to $\vartheta_1 > 1$.

The function is designed to achieve the following:

- **Reward for Low Latency**: When the processing time t is less than the acceptable limit u, the agent receives a reward. The reward increases as t decreases, encouraging the agent to minimize latency.
- **Penalty for High Latency**: When the processing time t exceeds the acceptable limit u, the agent incurs a penalty. The penalty increases as t increases, discouraging the agent from allowing high latency.

The constants φ and ϑ_1 are critical for tuning the reward function. By setting ϑ_1 to a value greater than 1, we significantly amplify the reward for low latency and the penalty for high latency, creating a strong incentive for the agent to optimize processing times effectively.

3. Single Observation Reward: The total reward for each observation combines both classification accuracy and computational efficiency. This ensures the agent is incentivized to achieve both high accuracy and low processing times.

$$R_{\text{input}}(x, y, t) = R_{\text{class}}(x, y) + R_{\text{time}}(t) \tag{15}$$

where:

- $R_{\text{class}}(x, y)$ is the reward for classification accuracy, based on the comparison of the predicted label x with the true label y (see Equation 13.
- $R_{\text{time}}(t)$ is the reward for computational efficiency (see Equation 14.

4. Batch Reward: We also introduce a reward function for a batch of M observations. This allows the system to evaluate accuracy and other metrics for a batch of samples, incorporating throughput as a metric to guide the agent in optimizing processing efficiency for batches of data.

The throughput ψ_M for a batch of M observations is calculated as:

$$\psi_M = \frac{M}{\sum_{i=1}^{M} t_i} \tag{16}$$

where $\sum_{i=1}^{M} t_i$ is the total processing time for the batch.

5. Total Batch Reward: The reward for the batch combines the classification and time rewards for each observation and incorporates the throughput reward:

$$R_M = \sum_{i=1}^{M} (R_{\text{time}}(t_i) + R_{\text{class}}(x_i, y_i)) \tag{17}$$

The total reward for the batch is:

$$R_{\text{tot}} = R_M + \psi_M \tag{18}$$

This reward function ensures that the agent is encouraged to make accurate classifications efficiently, balancing the trade-off between processing speed and computational cost.

To ensure that the agent processes samples within acceptable time ranges, we apply a final adjustment to the batch reward based on the evaluated metric Δ:

$$\hat{R}_{\text{tot}}(\Delta) = \begin{cases} -\vartheta_2 \cdot R_M & \text{if } \Delta \leq l \\ R_M & \text{if } l < \Delta \leq U \\ \vartheta_2 \cdot R_M & \text{if } \Delta > U \end{cases} \tag{19}$$

where:

- U and l are the upper and lower bounds of the acceptable range of the evaluated metric Δ.
- Δ is the value obtained by our approach for a batch of M samples.
- ϑ_2 is a manually defined bonus/penalty factor.

PPO is a powerful RL algorithm that can utilize both single observation rewards and batch rewards to guide the agent's learning process. During each step of the training process, the agent receives an immediate reward for each observation $R_{\text{input}}(x, y, t)$ (Equation 15). This reward informs the agent about the quality of its classification decision and the efficiency of its processing time for each individual sample.

PPO can also be configured to use batch rewards, which provide feedback based on the collective performance over a batch of observations. The total reward for a batch of M observations is R_{tot} (Equation 17). Where R_M is the sum of rewards for each observation in the batch, and ψ_M is the throughput reward. Additionally, the final adjustment ($\hat{R}_{\text{tot}}$ Equation 19 ensures that the agent considers the overall efficiency and accuracy within acceptable ranges.

By incorporating both single and batch rewards, PPO can balance the trade-offs between immediate and long-term performance, encouraging the agent to optimize for both individual and collective metrics. This dual reward structure helps the agent to make decisions that are beneficial in both the short-term (single observations) and long-term (batches), leading to more robust and effective learning.

6.3 Experimental Setup

6.3.1 Toxic Datasets

To evaluate our proposed technique, we used two distinct toxic datasets, detailed in Section 6.3.1.1 and Section 6.3.1.2.

6.3.1.1 Jigsaw Dataset

We utilized a toxic content dataset developed by our team, based on publicly available datasets from Google Jigsaw and Kaggle [271], as described by [3]. This dataset was generated using the instruct-tuning of FLAN-T5 [292] and further optimized through Reinforcement Learning from Human Feedback (RLHF) [288].

The dataset consists of 278,352 samples, equally divided between toxic and nontoxic categories (139,176 samples each). All nontoxic samples were generated and rated by humans. The toxic samples include 16,225 instances generated and rated by humans,

while the remaining 122,951 samples were generated by the optimized FLAN-T5 and rated using the Google Perspective API[1].

The dataset is divided into training, validation, and test sets, with proportions of 70%, 10%, and 20%, respectively (see Table 24. The training and validation sets were used for developing and fine-tuning the classifiers, while the test set was reserved for evaluating the performance of classifiers and developing the DRL-based and cascaded models. In this chapter, we refer to this dataset as "Kaggle" and denote it as D_{kaggle}.

Table 24: Number of Samples per Set for D_{kaggle}

Set	Number of Samples
Train	194,846
Validation	27,835
Test	55,670

6.3.1.2 ToxiGen Dataset

The second dataset, ToxiGen, consists of 274,000 machine-generated statements, covering both toxic and nontoxic content related to 13 different minority groups [163]. We selected 8,000 samples, equally divided into 4,000 toxic and 4,000 nontoxic. This dataset was exclusively used to test the performance of the classifiers and for developing the DRL-based and cascaded classification models. Throughout this chapter, we will refer to this dataset as "toxiGen" and denote it as D_{toxigen}.

6.3.2 Classifiers

Our selection of classifiers for cascaded inference was guided by strategic objectives aimed at optimizing performance and adaptability:

Architectural Diversity: We selected classifiers with diverse architectures

[1]https://perspectiveapi.com/

Table 25: All Models selected for experimentation

Objective	Model	Accuracy	F1-score	Throughput (s)
Accuracy	CNN_{α}	95.04%	95.10%	494
	CNN-fastText$_{\alpha}$	94.17%	94.29%	307
	BERT-base$_{\alpha}$	96.97%	97.22%	7.4
	RoBERTa-base$_{\alpha}$	96.26%	97.00%	7.8
Throughput	CNN_{ψ}	93.27%	932%	1,512
	CNN-fastText$_{\psi}$	91.90%	92.24%	418
	BERT-Tiny$_{\psi}$	95.10%	95.19%	169
	DistilRoBERTa-base$_{\psi}$	96.64%	96.68%	11

tailored for various facets of text processing, including CNN-based models and transformer-based models. This diversity ensures comprehensive coverage across different types of textual data and tasks.

Performance Variability: Each classifier was deliberately chosen to offer distinct trade-offs in performance metrics, such as accuracy and computational efficiency (throughput). This approach allows us to explore different levels of computational demand while maintaining high accuracy levels across different application scenarios.

These objectives underpin our approach to building a robust cascaded inference system capable of effectively handling diverse tasks and adapting to varying computational requirements.

To operationalize these goals, we evaluated four different detectors sourced from recent studies. For each classifier, we developed two variants: one optimized for accuracy and another for high throughput, resulting in a total of eight distinct classifiers.

For developing detectors, we used the training set discussed in Section 6.3.1.1, and the validation set was utilized for hyperparameter tuning. We employed four different sets of classifiers, including CNN, CNN with fastText embeddings [336] for CNN-based models, and BERT and RoBERTa [119] for Transformer-based models. Our objective

was to select the most accurate and the fastest (higher throughput) models. Therefore, we developed different versions with random hyperparameters and evaluated them on the validation set to find the most suitable classifiers. Throughout this chapter, we denote the accurate variants of classifiers as α and the high-throughput variants as ψ. Since all datasets used in this chapter are balanced with an equal number of toxic and nontoxic samples in all sets (train, validation, and test), we used Accuracy (α) and Throughput (ψ) as the evaluation metrics for comparing the performance of classifiers.

CNN-based: For developing accurate variants of CNN and CNN-fastText, we used four 1D convolutional layers with multiple window sizes and output channels. A pooling layer was utilized to reduce the dimensionality of the convolutions, followed by a global max pooling layer to further reduce the dimension. The hidden layers of the model employed the ReLU activation function, while the output layer used the Sigmoid activation function. During training, binary cross-entropy was used as the loss function to measure the difference between predicted and actual labels. Additionally, the Adam optimizer was employed to update the model's parameters during training, which helped speed up convergence and improve accuracy. For CNN-fastText, we compared different variants of fastText embeddings, with "crawl-300d-2M" yielding the highest accuracy. The experimental results for the most accurate CNN and CNN-fastText models are shown in Table 25 the results are based on the validation set of $D^{\text{val}}_{\text{kaggle}}$. Detailed information about the parameters of different variants is provided in Chapter 7.

As shown in Table 25, the throughput for the accurate variants of CNN and CNN-fastText is not high, which is acceptable since our focus was on improving accuracy. To develop high-throughput variants, we modified the architecture, resulting in some loss of accuracy. We applied various techniques, including parameter tuning

and adopting a shallower network topology. While deeper networks can improve accuracy, they are computationally expensive and can slow down the process. Thus, a shallower network with fewer convolutional 1D layers can be a good compromise between accuracy and throughput. We reduced the number of convolutional layers to three and two, while still maintaining good accuracy. The highest throughput was achieved with two convolutional 1D layers and a specific set of hyperparameters detailed in (Chapter 7). The high-throughput variants of CNN-based classifiers are also depicted in Table 25.

Transformer-based: We used the Huggingface transformer library, compatible with TensorFlow 2.15.0, for our work. We experimented with different variants of BERT and RoBERTa to select the most accurate and high-throughput variants for each. The experiments resulted in "bert-base-uncased" and "roberta-base" as the most accurate models, while "BERT-Tiny" [337] and "DistilRoBERTa" were identified as the high-throughput variants. We explored a distinct set of hyperparameters randomly to identify the configurations yielding the desired results. The optimal hyperparameters for each model are detailed in Appendix A (Chapter 7). The results are shown in Table 25.

According to Table 25, $BERT_{\alpha}$ is the most accurate classifier with an accuracy of 96.97%, whereas $fastText_{\psi}$ has the lowest accuracy at 91.90%. When considering throughput, $BERT_{\alpha}$ processes the fewest samples per second, handling only 7.4, while CNN_{ψ} achieves the highest throughput at 1,512 samples per second. Additionally, CNN_{α} offers a balance between accuracy and speed, with an accuracy of 95.04% and a throughput of 494 samples per second.

This analysis highlights that no single classifier excels in both accuracy and throughput, making it challenging to select an ideal solution for toxicity detection. Although CNN_{α} performs moderately well in both aspects, its throughput may

Table 26: Classifier performance on misclassified samples for $D^{\text{test}}_{\text{kaggle}}$

	$\textbf{CNN}_{\alpha}$	$\textbf{fastText}_{\alpha}$	$\textbf{BERT}_{\alpha}$	$\textbf{RoBERTa}_{\alpha}$	$\textbf{CNN}_{\psi}$	$\textbf{fastText}_{\psi}$	$\textbf{BERT}_{\psi}$	$\textbf{RoBERTa}_{\psi}$
$\textbf{CNN}_{\alpha}$	-	14%	93%	91%	12%	16%	92%	91%
$\textbf{fastText}_{\alpha}$	16%	-	93%	90%	13%	9%	92%	90%
$\textbf{BERT}_{\alpha}$	50%	47%	-	32%	48%	42%	32%	29%
$\textbf{RoBERTa}_{\alpha}$	52%	48%	49%	-	51%	42%	54%	31%
$\textbf{CNN}_{\psi}$	11%	10%	92%	91%	-	13%	92%	90%
$\textbf{fastText}_{\psi}$	21%	12%	92%	89%	18%	-	92%	90%
$\textbf{BERT}_{\psi}$	50%	48%	37%	43%	47%	47%	-	40%
$\textbf{RoBERTa}_{\psi}$	48%	43%	43%	26%	46%	40%	49%	-

Table 27: Classifier performance on misclassified samples for $D^{\text{test}}_{\text{toxiGen}}$

	$\textbf{CNN}_{\alpha}$	$\textbf{fastText}_{\alpha}$	$\textbf{BERT}_{\alpha}$	$\textbf{RoBERTa-base}_{\alpha}$	$\textbf{CNN}_{\psi}$	$\textbf{fastText}_{\psi}$	$\textbf{BERT}_{\psi}$	$\textbf{lRoBERTa}_{\psi}$
$\textbf{CNN}_{\alpha}$	-	11%	74%	71%	14%	11%	76%	77%
$\textbf{fastText}_{\alpha}$	35%	-	77%	75%	28%	5%	79%	80%
$\textbf{BERT}_{\alpha}$	55%	46%	-	19%	55%	43%	32%	30%
$\textbf{RoBERTa}_{\alpha}$	55%	47%	30%	-	55%	44%	38%	31%
$\textbf{CNN}_{\psi}$	18%	6%	76%	72%	-	5%	77%	78%
$\textbf{fastText}_{\psi}$	40%	13%	78%	76%	33%	-	80%	81%
$\textbf{BERT}_{\psi}$	59%	50%	34%	30%	59%	47%	-	39%
$\textbf{RoBERTa}_{\psi}$	55%	46%	21%	10%	55%	43%	30%	-

be insufficient during peak times when platforms receive higher volumes of UGC. Moreover, in scenarios with a high ratio of toxic content, CNN_{α} may struggle to process all UGC efficiently, indicating the need for a more robust solution that can handle diverse and high-volume content effectively.

To demonstrate the relative performance and ensure no single classifier dominates, we evaluated all selected classifiers using separate test sets from D_{kaggle} (55,653 samples) and D_{toxigen} (8,000 samples). Notably, these data points were not used during classifier training or validation. As of now, we refer to these datasets as $D^{\text{test}}_{\text{kaggle}}$ and $D^{\text{test}}_{\text{toxigen}}$ to distinguish them from the training and validation sets.

Table 26 and Table 27 compare the classifiers' performance on the $D^{\text{test}}_{\text{kaggle}}$ and the $D^{\text{test}}_{\text{toxigen}}$ dataset, respectively. Each table illustrates the percentage of samples that one classifier misclassified (rows) but another correctly classified (columns).

For example, the $BERT_{\alpha}$ detector accurately identified 93% of comments misclassified by CNN_{α}. In the $D^{\text{test}}_{\text{kaggle}}$ set, 35.20% of samples were correctly classified by all

eight classifiers, with 17.08% being toxic and 18.12% nontoxic. Similarly, in $D^{\text{test}}_{\text{toxigen}}$, 50.22% of samples were correctly classified overall, comprising 35.64% toxic and 14.58% nontoxic samples. As of now, we refer to these datasets as $D^{\text{test}}_{\text{kaggle}}$ and $D^{\text{test}}_{\text{toxigen}}$ to distinguish them from the training and validation sets. In addition, for $D^{\text{test}}_{\text{toxigen}}$, only 50.22% of the samples were correctly classified by all classifiers. Specifically, 35.64% of the toxic samples were correctly identified as toxic, while only 14.58% of the nontoxic samples were correctly identified as nontoxic.

For $D^{\text{test}}_{\text{kaggle}}$, only 45.20% of the samples were correctly classified by all classifiers. Among these, 27.08% of the toxic samples were correctly identified as toxic, while 18.12% of the nontoxic samples were correctly identified as nontoxic.

These findings underscore that no single classifier dominates in accuracy across both datasets, affirming the robustness and comparative performance of each classifier.

6.3.3 Baselines

To evaluate the performance of PPO-CIS, we compare it with several baselines:

Dynamic vs. Static Classifier Selection: PPO-CIS dynamically adjusts classifier selection based on real-time confidence scores to optimize detection accuracy and efficiency. In contrast, static baselines employ predetermined combinations of classifiers within the cascade, with decisions finalized using majority voting, soft voting, or the output from the last classifier in the sequence. This comparison highlights DRL-CIS's adaptability and its performance advantages over fixed configurations in real-world applications.

Comparison with CETRA: We benchmark PPO-CIS against CETRA [139] to evaluate the effectiveness of its reward function, Proximal Policy Optimization (PPO) as the DRL model, and cascaded classifier arrangements with dual variants emphasizing throughput and accuracy. This comparison underscores the superiority

of PPO-CIS's cascaded design in toxicity detection scenarios, showcasing its enhanced adaptability and effectiveness.

In real-time scenarios, we assess PPO-CIS and the baselines across critical metrics including throughput, processing time, and detection accuracy. This evaluation provides insights into the operational efficiency of PPO-CIS for dynamic toxicity detection, highlighting its capability for rapid decision-making and robust performance.

6.3.4 Experimental setting

As detailed in Section 6.3.2, we first trained and validated classifiers using the $D^{\text{train}}_{\text{kaggle}}$ and $D^{\text{val}}_{\text{kaggle}}$ datasets (see Section 6.3.1.1). Subsequently, the $D^{\text{test}}_{\text{kaggle}}$ and the entire $D^{\text{test}}_{\text{toxigen}}$ datasets were utilized to evaluate these classifiers. Each data point was assessed for binary and probabilistic labels, alongside processing times, providing comprehensive metrics such as accuracy, precision, recall, F1-score, throughput per second, and average processing time. These evaluations formed the basis for developing our deep reinforcement learning (DRL) models, using the $D^{\text{test}}_{\text{kaggle}}$ and $D^{\text{test}}_{\text{toxigen}}$ datasets for testing and refinement.

For developing DRL-based models, we split $D^{\text{test}}_{\text{kaggle}}$ and $D^{\text{test}}_{\text{toxigen}}$ into train, test, and validation sets. To compare the performance of all potential cascade inference systems (CIS) with single classifiers, we held out two balanced datasets, $\hat{D}^{\text{test}}_{\text{kaggle}}$ and $\hat{D}^{\text{test}}_{\text{toxigen}}$, containing 5,000 and 2,000 samples respectively. All results reported from this point onwards are based on these datasets.

Our framework was implemented in Python v3.8 using OpenAI Gym [338]. For our DRL agents, we employed Proximal Policy Optimization (PPO) (**??**) for PPO-CIS and Actor-Critic with Experience Replay (ACER) (Section 2.5.2) for CETRA, utilizing the "ChainerRL" library [339]. CETRA, originally designed for dynamic classifier selection in ensemble learning for malware detection, provided a comparative baseline

with five distinct reward functions evaluated separately.

In our implementation of PPO-CIS, we used an "FCSoftmaxPolicyAdam" architecture for the policy network, featuring fully connected layers followed by softmax activation to facilitate decision-making. The "FCVFunction" served as the value function, crucial for estimating expected returns during policy updates. The Adam optimizer was configured with a conservative learning rate of 1×10^{-5} and gradient clipping with a threshold of 0.5, ensuring stable training dynamics by limiting excessive gradient updates.

The neural network architecture included an input layer with 8 neurons to handle the state representation required for decision-making, a hidden layer with 64 neurons, and an output layer with 12 neurons supporting 12 possible actions across two distinct action spaces. These actions encompassed selections among individual classifiers and final toxicity classifications, critical for effective decision-making in cascade inference systems.

To ensure reproducibility and clarity of our proposed method, we specify the hyperparameters used in our experiments in Table 28 and Table 29.

Table 28: PPO-Specific Hyperparameters

Hyperparameter	**Value**
Clip Range (ϵ)	0.2
Learning Rate (lr)	0.0003
Batch Size	256
Number of Epochs	10
Discount Factor (γ)	0.99
GAE Lambda (λ)	0.95
Entropy Coefficient	0.01
Value Function Coefficient	0.5

Our experimental approach prioritized stability and efficiency, with carefully chosen architecture and optimizer configurations tailored to the demands of real-time inference tasks. By detailing these parameters, we aimed to enhance transparency and

Table 29: Reward Function Hyperparameters

Hyperparameter	**Value**
Classification Reward (r)	10
Classification Penalty Factor (ι)	2
Upper Limit of Acceptable Processing Time (u)	1
Constant Offset (φ)	1
Scaling Factor (ϑ_1)	10
Bounds for Final Adjustment (U and l)	$U = 1.5$, $l = 0.5$
Bonus/Penalty Factor (ϑ_2)	1.2

reproducibility while ensuring robust performance in complex decision environments.

The reward function in CETRA is defined as follows:

$$C_{2_{\text{Total}}}(T) = \begin{cases} r & \text{for TP or TN} \\ -1 \times \sum_{t=1}^{T} C_2(t) & \text{for FN or FP} \end{cases} \tag{20}$$

Here, r is a constant, and $C_2(t)$ is the reward function for each step, defined as:

$$C_2(t) = \begin{cases} \frac{t}{d_2} & \text{for } 0 \leq t < d_2 \\ 1 + \log_2\left(\frac{\min(t, t_2(s))}{d_2}\right) & \text{for } d_2 \leq t \end{cases} \tag{21}$$

The reward values were assigned differently in five separate experiments. In Experiment 1, TP and TN both received a reward value of $C_{2,j}$, whereas FP and FN both received a penalty of $-C_{2,j}$. Experiment 2 maintained the same rewards for TP and TN at $C_{2,j}$, but increased the penalties for FP and FN to $-10C_{2,j}$. In Experiment 3, a uniform reward of 1 was assigned to both TP and TN, with FP and FN penalties remaining at $-C_{2,j}$. Experiment 4 increased the rewards for TP and TN to 10, while keeping the FP and FN penalties at $-C_{2,j}$. Finally, Experiment 5 further increased the rewards for TP and TN to 100, with FP and FN penalties still at $-C_{2,j}$. In our experiment, d_2 was set to 0.5 because both datasets include a balanced number of toxic and nontoxic samples.

6.3.5 Computational Resources

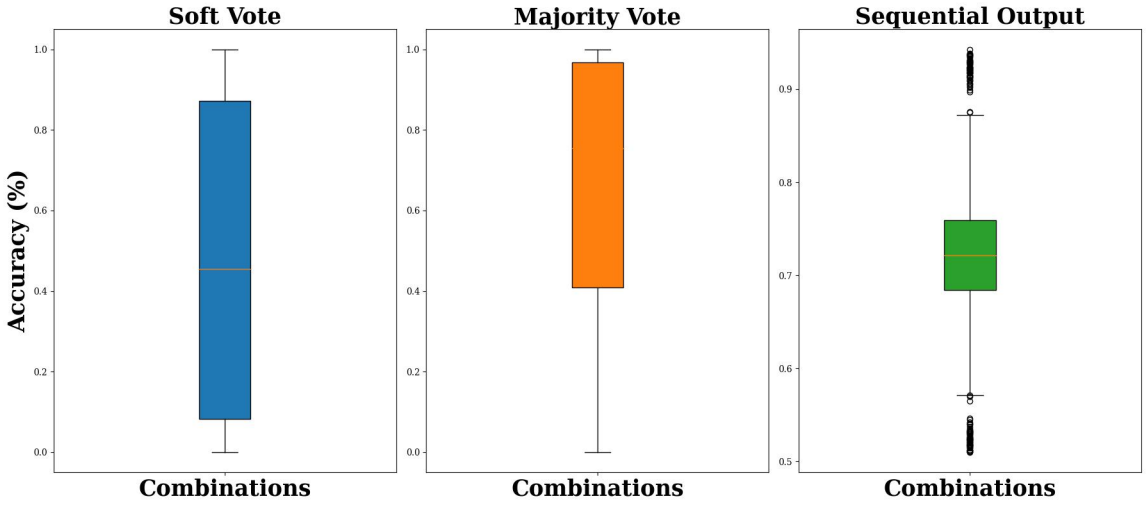
For our experiments, we utilized the 'Paperspace P6000' cloud computing instance, which is equipped with NVIDIA P6000 GPUs. Each P6000 GPU offers 24 GB of memory, 30 GB of RAM, and 8 vCPUs, providing robust computational power for our tasks. This setup supports multi-GPU configurations, including 2x and 4x instances, allowing for scalability and parallel processing. The cost for utilizing the P6000 instance is $1.10 per hour.

This computing environment was consistently used for both developing and training the classifiers as well as for implementing and evaluating the DRL-based approaches. The consistent usage of this environment ensures that our experiments have a reliable and stable computational foundation, allowing for accurate performance comparison and benchmarking.

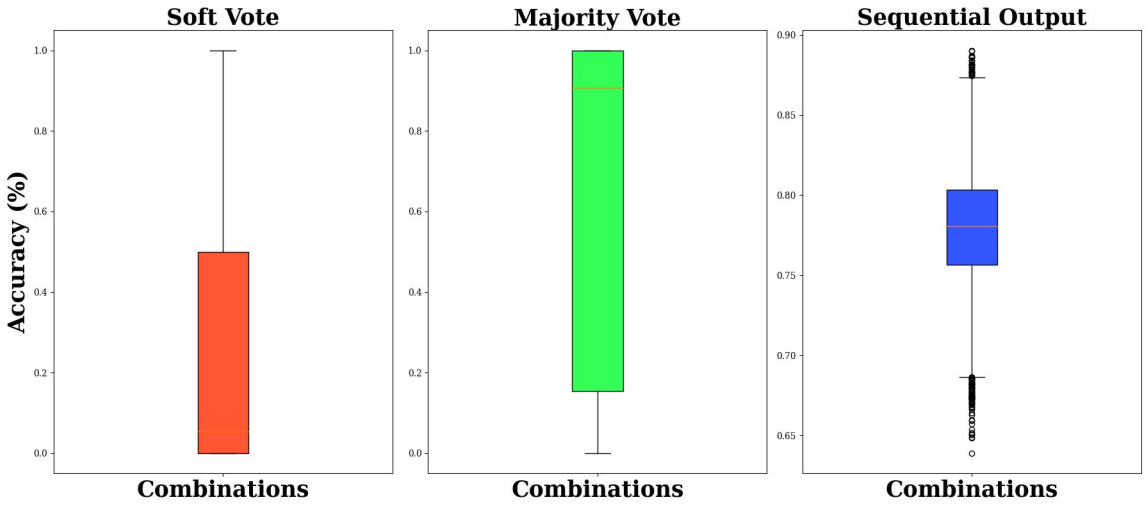
6.4 Experimental Results

In this section, we present the results from testing all baseline models. We start by evaluating the performance of individual classifiers on the $\hat{D}^{\text{test}}_{\text{kaggle}}$ and $\hat{D}^{\text{test}}_{\text{toxigen}}$ datasets, as shown in Table 30. Subsequently, Section 6.4 details the performance of various classifier combinations arranged in cascades. This analysis highlights the effectiveness of different cascaded configurations and identifies the optimal setups for toxicity detection. The best-performing cascades are then compared with the DRL-based approaches. Finally, Section 6.4 discusses the experimental results for CETRA with different reward functions and the proposed PPO-CIS.

According to Table 30, for both datasets, $BERT_{\alpha}$ is the most accurate classifier but has the lowest throughput. Conversely, CNN_{ψ} demonstrated the highest throughput as expected, but with lower accuracy.

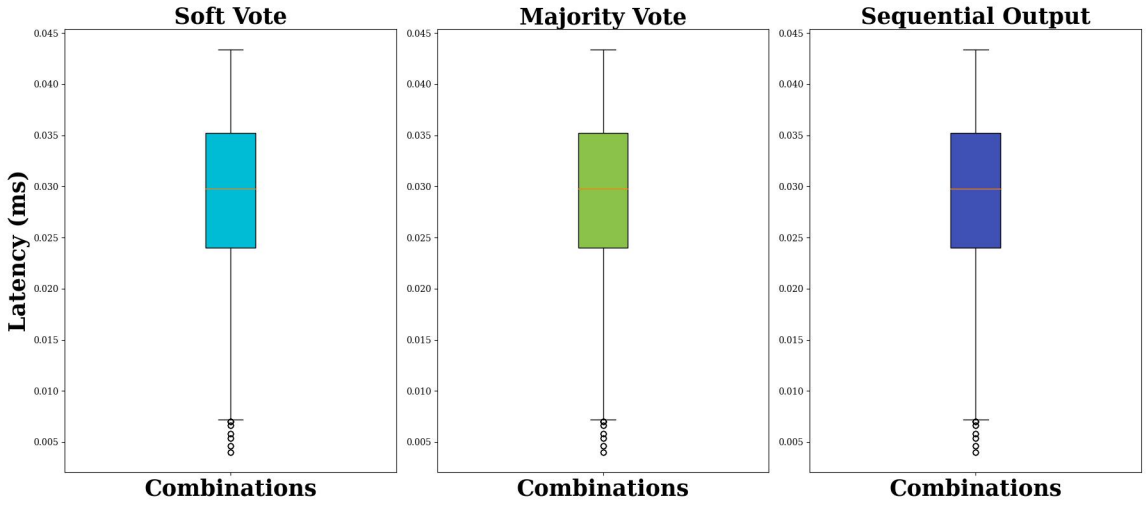


(a) $\hat{D}^{\text{test}}_{\text{kaggle}}$

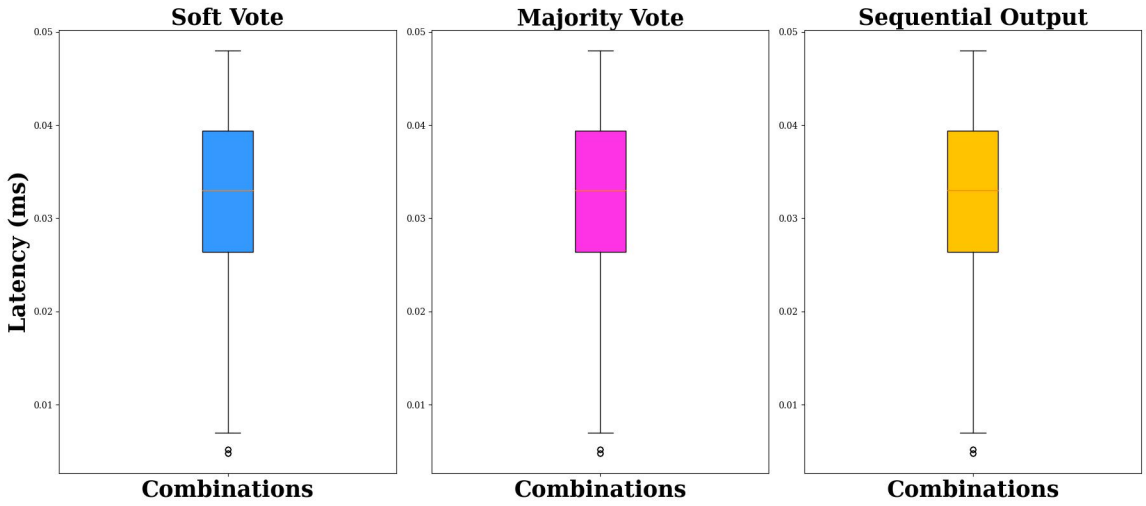


(b) $\hat{D}^{\text{test}}_{\text{toxigen}}$

Figure 19: Comparison of accuracy distributions for different combinations of classifiers using different label determination techniques on the a) $\hat{D}^{\text{test}}_{\text{kaggle}}$ and b) $\hat{D}^{\text{test}}_{\text{toxigen}}$ datasets.



(a) $\hat{D}^{\text{test}}_{\text{kaggle}}$



(b) $\hat{D}^{\text{test}}_{\text{toxigen}}$

Figure 20: Comparison of latency distributions for different combinations of classifiers using different label determination techniques on the a) $\hat{D}^{\text{test}}_{\text{kaggle}}$ and b) $\hat{D}^{\text{test}}_{\text{toxigen}}$ datasets.

Table 30: Performance Metrics of Classifiers on $\hat{D}^{\text{test}}_{\text{kaggle}}$ and $\hat{D}^{\text{test}}_{\text{toxigen}}$

Dataset	Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Throughput (s)	Latency (s)
$\hat{D}^{\text{test}}_{\text{kaggle}}$	CNN_{α}	93.18%	94.17%	95.42%	94.79%	148	0.0067
	$fastText_{\alpha}$	92.23%	90.19%	94.78%	92.43%	64.43	0.01552
	$BERT_{\alpha}$	**93.22%**	92.38%	94.20%	93.28%	**4.595**	0.2176
	$RoBERTa_{\alpha}$	92.67%	90.85%	93.43%	92.12%	4.73	0.2113
	CNN_{ψ}	**90.33%**	90.27%	94.30%	92.09%	**269.58**	0.00371
	$fastText_{\psi}$	91.07%	92.25%	93.84%	93.08%	121.48	0.00823
	$BERT_{\psi}$	91.00%	87.27%	96.00%	91.43%	7.843	0.1275
	$RoBERTa_{\psi}$	91.56%	89.48%	94.20%	91.78%	5.41	0.1848
$\hat{D}^{\text{test}}_{\text{toxigen}}$	CNN_{α}	93.54%	89.46%	91.25%	90.35%	28.01	0.0357
	$fastText_{\alpha}$	91.52%	90.42%	93.25%	91.82%	19.93	0.050175
	$BERT_{\alpha}$	**93.91%**	90.52%	95.45%	92.92%	**4.10**	0.2442
	$RoBERTa_{\alpha}$	92.85%	89.79%	94.98%	92.31%	4.23	0.2366
	CNN_{ψ}	**90.09%**	88.34%	92.78%	90.83%	**52.56**	0.01903
	$fastText_{\psi}$	90.19%	89.36%	92.98%	91.07%	34.81	0.0287
	$BERT_{\psi}$	91.82%	88.76%	93.75%	91.18%	5.75	0.174
	$RoBERTa_{\psi}$	92.76%	93.14%	92.30%	92.73%	4.56	0.21925

Cascade of Classifiers

Since we have 8 classifiers, where 4 (CNN-based) have higher throughput and the other 4 (transformer-based) have lower throughput but higher accuracy, the low-throughput classifiers are not capable of processing the large amount of data received per second. However, we tested all possible combinations of classifiers and recorded the total $(\alpha_{P_i}, \rho_{P_i}, \sigma_{P_i}, \phi_{P_i}, \psi_{P_i}, \lambda_{P_i})$ for each set, as explained in Section 6.2.1. Please note that the computational cost ω_{P_i} is considered zero in our experiments. The results for $\hat{D}^{\text{test}}$kaggle are shown in Table 31, and for $\hat{D}^{\text{test}}$toxiGen in Table 32.

The combination of 8 classifiers, using different voting methods (majority and soft), or the last response (sequential output), results in large number of possible cases when considering at least 2 classifiers for each dataset. Therefore, we only report the competitive combinations that result in higher accuracy and throughput compared to a single classifier.

Figure 19 illustrates the accuracy achieved by combining at least two classifiers in

a cascaded manner using various label determination techniques on different datasets. For the $\hat{D}^{\text{test}}_{\text{kaggle}}$ dataset, the Soft Vote method shows a wide accuracy range (0.0 to 1.0) with a median of 0.5, indicating high variability. The Majority Vote method has a similar wide range but a higher median of 0.6, also showing significant variability. The Sequential Output method, however, has a more concentrated accuracy range (0.5 to 0.9) with a median of 0.72, demonstrating higher consistency and reliability.

For the $\hat{D}^{\text{test}}_{\text{toxigen}}$ dataset, the Soft Vote method also shows a wide accuracy range (0.0 to 1.0) with a median of 0.35, indicating high variability and less consistent performance. The Majority Vote method again displays a broad range of accuracies (0.0 to 1.0) with a higher median accuracy of 0.65, but still shows significant variability. In contrast, the Sequential Output method demonstrates a more concentrated distribution of accuracies, ranging from 0.65 to 0.98, with a median accuracy near 0.75, indicating less variability and higher consistency. Overall, the Sequential Output technique outperforms both the Soft Vote and Majority Vote methods in terms of median accuracy and consistency for both datasets.

The box plots in Figure 20 illustrate the latency distributions for different combinations of classifiers using Soft Vote, Majority Vote, and Sequential Output label determination techniques on the $\hat{D}^{\text{test}}_{\text{kaggle}}$ and $\hat{D}^{\text{test}}_{\text{toxigen}}$ datasets.

For the Kaggle dataset (Figure 20a), the Soft Vote method shows latencies ranging from approximately 0.015 ms to 0.045 ms, with a median latency around 0.03 ms, indicating moderate variability in latency across different classifier combinations. The Majority Vote method exhibits a similar latency range, from approximately 0.015 ms to 0.045 seconds, with a median latency slightly above 0.03 ms, suggesting comparable variability. The Sequential Output method also demonstrates latencies in the same range, with a median latency around 0.03 ms, showing consistent performance across classifier combinations.

Table 31: Best-Performing Classifier Combinations on $\hat{D}^{\text{test}}_{\text{kaggle}}$

No.	Combination	Label Determination	Accuracy$_\alpha$ (%)	Precision$_\rho$ (%)	Recall$_\sigma$ (%)	F1-Score$_\phi$ (%)	Throughput$_\psi$ (s)	Latency$_\lambda$ (s)
1	$fastText_\psi(0), CNN_\psi \| \| fastText_\psi(1), CNN_\psi$	Sequential	**90.32%**	89.13%	96.08%	92.47%	**723**	0.0013
2	$fastText_\alpha(0), BERT_\psi \| \| fastText_\alpha(1), BERT_\psi$	Sequential	93.16	92.95	93.40	93.17	192.22	0.0052
3	$BERT_\alpha(1), BERT_\psi$	Sequential	92.39%	92.39%	92.46%	92.42%	86.21	0.0116
4	$CNN_\alpha(1), fastText_\alpha(1), BERT_\alpha(1), RoBERTa_\alpha(1), CNN_\psi \| \| CNN_\alpha(0), fastText_\alpha(0), BERT_\alpha(0), RoBERTa_\alpha(0), fastText_\psi$	Sequential	**95.03%**	95.12%	98.22%	**96.60%**	**340**	0.0224
5	$CNN_\alpha(1), fastText_\alpha(0), BERT_\psi$	Majority	92.01%	93.59%	97.79%	95.64%	311.64	0.0032
6	$CNN_\alpha(0), BERT_\alpha(0), fastText_\psi(1), BERT_\psi(0), RoBERTa_\psi(1)$	Majority	91.35%	91.28%	95.42%	93.31%	51.02	0.0196
7	$CNN_\alpha(0), CNN_\psi(1), fastText_\psi(1), BERT_\psi(1), RoBERTa_\psi \| \| CNN_\alpha(1), CNN_\psi(0), fastText_\psi(0), BERT_\psi(0), RoBERTa_\psi$	Majority	94.94	93.79	96.50%	95.13%	82	0.012
8	$BERT_\alpha(0), RoBERTa_\alpha(1), fastText_\psi(1), BERT_\psi \| \| BERT_\alpha(1), RoBERTa_\alpha(0), fastText_\psi(0), BERT_\psi$	Majority	**96.14%**	96.09%	96.38%	**96.19%**	**34.97**	0.0286
9	$CNN_\psi(1), fastText_\psi(1), BERT_\psi(1), RoBERTa_\psi(1) \| \| CNN_\alpha(0), fastText_\alpha(0), BERT_\alpha(0), RoBERTa_\alpha(0), CNN_\psi$	Majority	**96.15%**	96.10%	97.25	**96.64%**	**42.74**	0.0234
10	$BERT_\alpha(1), CNN_\psi(1), fastText_\psi \| \| BERT_\alpha(0), CNN_\psi(0), fastText_\psi$	Soft	92.63%	92.63%	91.30%	96.17%	53.19	0.0188
11	$fastText_\alpha(0), BERT_\alpha(1), CNN_\psi(1), fastText_\psi(0), BERT_\psi \| \| fastText_\alpha(1), BERT_\alpha(0), CNN_\psi(1), fastText_\psi(1), BERT_\psi$	Soft	93.10%	92.38%	94.20%	93.28%	40.65	0.0246
12	$CNN_\alpha(0), fastText_\alpha(0), BERT_\alpha(1), RoBERTa_\alpha(1), CNN_\psi(0), fastText_\psi \| \| CNN_\alpha(0), fastText_\alpha(1), BERT_\alpha(0), RoBERTa_\alpha(0), CNN_\psi(0), fastText_\psi(0) > RoBERTa_\psi(0), BERT_\psi(0)$	Soft	**84.08%**	92.67%	83.20%	87.68%	**28.09**	0.0356
13	$CNN_\alpha(1), fastText_\alpha(1), BERT_\alpha(1), RoBERTa_\alpha(1), CNN_\psi(1), fastText_\psi(1), RoBERTa_\psi \| CNN_\alpha(1), fastText_\alpha(0), BERT_\alpha(1), RoBERTa_\alpha(1), CNN_\psi(1), fastText_\psi(0), RoBERTa_\psi(1)$	Soft	91.74%	85.32%	92.89%	88.94%	43.04	0.0234

For the ToxiGen dataset (Figure 20b), the Soft Vote method shows a latency range from approximately 0.01 ms to 0.05 ms, with a median latency around 0.03 ms, indicating some variability in latency. The Majority Vote method has a latency range similar to the Soft Vote method, but with a slightly lower median latency around 0.03 ms, demonstrating similar variability. The Sequential Output method shows a latency range from approximately 0.01 ms to 0.05 ms, with a median latency slightly above 0.03 ms, indicating consistent performance with less variability compared to the other methods.

Overall, all three label determination techniques exhibit similar latency distributions for both the Kaggle and Toxigen datasets. The median latencies are consistent around 0.03 ms, with some variability across different classifier combinations. The

Table 32: Best-Performing Classifier Combinations on $\hat{D}^{\text{test}}_{\text{toxiGen}}$ dataset

No.	Combination	Label Determination	Accuracy$_\alpha$ (%)	Precision$_\rho$ (%)	Recall$_\sigma$ (%)	F1-Score$_\phi$ (%)	Throughput$_\psi$ (s)	Latency$_\lambda$ (s)
1	$CNN_\alpha(1), BERT_\psi$	Sequential	92.74%	92.18%	96.96%	94.51%	138.88	0.0072
2	$fasText_\alpha(1), BERT_\alpha(0), RoBERTa_\alpha(0), RoBERTa_\psi$	Sequential	90.48%	87.27%	94.30%	90.65%	64.93	0.0154
3	$CNN_\alpha(1), BERT_\alpha(0), RoBERTa_\alpha(0),$ $BERT_\psi(0), RoBERTa_\psi$	Sequential	90.90%	89.48%	92.46%	90.94%	29.59	0.0338
4	$CNN_\alpha(1), fastText_\alpha(1), BERT_\alpha(1), RoBERTa_\alpha(1),$ CNN_ψ|| $CNN_\alpha(0), fastText_\alpha(0), RoBERTa_\alpha(0), CNN_\psi$	Sequential	91.43%	92.14%	95.91%	93.99%	36.50	0.0274
5	$fastText_\psi(0), CNN_\psi(1), CNN_\alpha(1),$ $fastText_\psi(1), CNN_\alpha(0), CNN_\psi(0)$	Sequential	**92.74%**	93.85%	96.07%	94.95%	**556**	0.00179
6	$CNN_\alpha(0), fastText_\alpha(0), fastText_\psi(1)$	Sequential	93.50%	93.14%	94.09%	93.61%	**301.8**	0.003315
7	$CNN_\alpha(1), CNN_\psi(1), RoBERTa_\psi(1)$	Majority	93.73%	92.80%	95.05%	93.91%	217	0.0046
8	$CNN_\alpha(1), fastText_\alpha(0), BERT_\alpha(0), RoBERTa_\alpha$	Majority	93.87%	92.39%	95.93%	94.12%	208	0.0048
9	$BERT_\alpha(0), RoBERTa_\alpha(0), CNN_\psi(1), RoBERTa_\psi(1)$	Majority	**96.06%**	93.12%	95.85%	94.43%	38.17	0.0262
10	$CNN_\alpha(0), fastText_\alpha(1), fastText_\psi(1), BERT_\psi(1),$ $RoBERTa_\psi(0)\|\|CNN_\alpha(1), fastText_\alpha(0),$ $fastText_\psi(0), BERT_\psi(0), RoBERTa_\psi$	Majority	**96.84%**	94.98%	97.45%	96.20%	30.49	0.0328
11	$BERT_\alpha(0), RoBERTa_\alpha(1), CNN_\psi(1), fastText_\psi(1),$ $RoBERTa_\psi\|\|BERT_\alpha(1), RoBERTa_\alpha(0), CNN_\psi(0),$ $fastText_\psi(0), RoBERTa_\psi$	Majority	**97.10%**	96.43%	97.66%	**97.07%**	28.86	0.0324
12	$CNN_\alpha(1), fastText_\alpha(0), RoBERTa_\alpha(1)$	Soft	90.94	90.92	95.25	92.98%	149.5	0.0066
13	$CNN_\alpha(0), BERT_\alpha(1), RoBERTa_\alpha(1), BERT_\psi$|| $CNN_\alpha(1), BERT_\alpha(0), RoBERTa_\alpha(0), BERT_\psi$	Soft	88.70%	86.73%	95.20%	90.77%	43.86	0.0228
14	$CNN_\alpha(0), RoBERTa_\alpha(1), fastText_\psi(0),$ $RoBERTa_\psi$|| $fastText_\alpha(0), BERT_\alpha(1), CNN_\psi(0), BERT_\psi$	Soft	89.72%	87.27%	92.59%	89.85%	43.10	0.0232
15	$BERT_\alpha(1), RoBERTa_\alpha(1), CNN_\psi(1), RoBERTa_\psi$|| $BERT_\alpha(0), RoBERTa_\alpha(0), FastText_\psi(0), BERT_\psi$	Soft	93.68%	93.56%	97.02%	95.26%	37.02	0.027

choice of label determination technique does not significantly impact the latency, indicating comparable performance in terms of latency across the different methods.

Based on the results presented in Table 31 and Table 32, several key insights can be drawn about the performance of different classifier combinations in terms of throughput, accuracy, and F1-score.

In Table 31, the highest throughput of 723 samples per second is achieved by the No. 1combination. It is significantly higher than the throughput of single classifiers (CNN_ψ and $fastText_\psi$), and it is notably higher than that of transformer-based classifiers applied alone. The combinations No. 5 and No. 2 follow with throughputs of 311 and 192 samples per second, respectively. These combinations are among the best in terms of throughput, but have comparatively lower accuracy.

The highest accuracy and F1-score are achieved by combinations using Majority

Vote for label determination, specifically No. 8 and No. 9. Despite their high accuracy, these combinations handle fewer than 50 samples per second, suggesting that single classifiers might be preferable in scenarios that require higher throughput. The combination No. 4 stands out with an accuracy of 95.03% and an F1-score of 96.60%, coupled with a more reasonable throughput of 340 samples per second, making it a balanced choice compared to No. 8 and No. 9.

Transformer-based models ($BERT_\alpha$, $BERT_\psi$, $RoBERTa_\alpha$, $RoBERTa_\psi$) in the cascade significantly reduce throughput, especially at primary stages. Applying CNN-based classifiers at primary stages boosts throughput. For instance, in combination No. 5, all samples are initially processed by CNN_α. Toxic samples are then processed by $fastText_\alpha$, and only those classified as nontoxic undergo further analysis by $BERT_\psi$. This staged filtering improves throughput by reducing the number of samples needing in-depth processing by $BERT_\psi$. One of the most surprising results was achieved by No. 3, where $BERT_\alpha$ was applied first, followed by $BERT_\psi$. The throughput significantly improved to 86, even though the throughput for both individual models was less than 10.

Soft Vote methods do not significantly enhance accuracy or throughput. Furthermore, long cascades, such as those in combinations No. 11, No. 12, and No. 13 fail to improve accuracy and instead reduce throughput, indicating that more classifiers do not necessarily lead to better performance.

In Table 32, combination No. 5 achieves the highest throughput (ψ) of 556 samples per second by avoiding transformer-based classifiers. The lowest throughput (ψ) of 28.86 samples per second is observed in combination No. 11, which uses highly accurate but slower classifiers, such as $BERT_\alpha$, at the initial stage. Notably, this combination also achieves the highest accuracy with majority vote.

Combinations using majority voting, such as No. 10 and No. 11, yield higher

accuracy and F1-scores. For instance, No. 10 achieves an accuracy of 96.84% and an F1-score of 96.20%, with a throughput of 30.49 samples per second. No. 11 achieves the highest accuracy of 97. 10% and an F1 score of 97. 07%, although it has a lower throughput of 28.86 samples per second.

On the other hand, soft voting combinations, such as No. 12 and No. 13, tend to exhibit lower accuracy and F1-scores, indicating that soft voting may reduce performance compared to majority voting. However, No. 15 achieves the highest accuracy and lower throughput among soft vote models.

Overall, balancing accuracy and throughput is crucial. Using fast classifiers like CNNs in the initial stages can improve throughput without significantly compromising accuracy. For example, in combination No. 5 in Table 32, all samples are processed by CNN_{α}, then toxic samples by $fastText_{\alpha}$, and only nontoxic samples undergo further analysis by $BERT_{\psi}$. This staged filtering reduces the number of samples needing in-depth processing, thereby improving throughput.

DRL-based Inference Systems

After developing top-performing classifiers in a cascaded manner, we demonstrated that this approach could improve processing time for low-throughput classifiers. In addition, high-throughput combinations still exhibited lower accuracy compared to those with higher accuracy but very low throughput.

A key insight from our results is that the order of classifiers is crucial. By applying fast classifiers at the initial stages and slow classifiers in the final stages, we can achieve higher throughput. Nevertheless, caution is required because adding many classifiers does not necessarily lead to better performance and may even negatively impact it. This underscores the proposal for dynamic selection of classifiers.

Moreover, the results presented in Section 6.4 indicated that good results could

be achieved by using cascades of classifiers. The key to success lies in dynamically selecting classifiers to enhance both accuracy and throughput. This objective can be accomplished by designing a reward function that considers both accuracy and throughput.

Therefore, the next step is to explore dynamic classifier selection within the cascade. This approach will guide the selection process towards our goal of simultaneously increasing accuracy and throughput.

To evaluate PPO-CIS for toxicity detection, we compared it with CETRA across three key aspects: (a) the reward function used, (b) the dynamic selection of classifiers in a cascaded manner, where high-throughput classifiers are prioritized in the initial stages and more accurate classifiers are employed in subsequent stages, and (c) the DRL model applied, where CETRA uses ACER and we have implemented PPO.

Note that all five reward functions were tested for both DRL approaches (ACER and PPO). The final results for each dataset are presented in Table 33 and Table 34. All results are reported based on the test set for each dataset, which includes 5,000 samples for $\hat{D}^{\text{test}}_{\text{kaggle}}$ and 2000 samples for $\hat{D}^{\text{test}}_{\text{toxigen}}$.

Please note that we use $CETRA - CIS_i$ for $i = 1$ to 5 to denote models that use the CETRA structure, employing ACER as the DRL model, with all 8 classifiers at the same stage. The index $i = 1$ to 5 refers to the specific reward functions proposed by [139]. Additionally, we use $ACER - CIS$ to refer to our proposed CIS, which includes two space actions, where ACER is applied as the DRL model instead of PPO, along with our proposed reward function. We also use $ACER - CIS_i$ for $i = 1$ to 5 to denote models tested with the five different reward functions. Similarly, $PPO - CIS$ refers to our proposed CIS with PPO as the DRL model, incorporating its own reward function and two space actions. Furthermore, $PPO - CIS_i$ for $i = 1$ to 5 denotes the proposed PPO-CIS models tested with the reward functions proposed by [139].

Table 33: Performance Comparison of Inference Systems for High-Throughput Toxicity Detection on the $\hat{D}^{\text{test}}_{\text{kaggle}}$

Model	**Action Space**	**Accuracy**$_{\alpha}$ (%)	**Precision**$_{\rho}$ (%)	**Recall**$_{\sigma}$ (%)	**F1-Score**$_{\phi}$ (%)	**Throughput**$_{\psi}$ (samples/s)	**Latency**$_{\lambda}$ (s)
$CETRA-CIS_1$	1	94.88%	93.59%	92.20%	92.89%	63.72	0.0314
$CETRA-CIS_2$	1	95.65%	93.24%	96.79%	94.99%	97.12	0.0206
$CETRA-CIS_3$	1	94.38%	94.31%	95.91%	95.11%	83.33	0.0240
$CETRA-CIS_4$	1	95.48%	94.65%	95.79%	95.22%	111.52	0.0179
$CETRA-CIS_5$	1	94.17%	93.78%	95.29%	94.53%	172.41	0.0116
$ACER-CIS$	2	97.76%	97.60%	97.82%	97.71%	213	0.0094
$ACER-CIS_1$	2	95.80%	95.13%	96.70%	95.90%	107.70	0.0186
$ACER-CIS_2$	2	97.70%	96.83%	98.38%	97.60%	105	0.0190
$ACER-CIS_3$	2	96.55%	95.88%	96.93%	96.40%	110	0.0182
$ACER-CIS_4$	2	95.05%	95.38%	95.94%	95.66%	119	0.0168
$ACER-CIS_5$	2	96.12%	95.45%	96.00%	95.72%	160	0.0125
$PPO-CIS$	2	98.17%	97.96%	98.10%	98.03%	384	0.0052
$PPO-CIS_1$	2	96.03%	92.23%	96.95%	94.54%	146	0.0137
$PPO-CIS_2$	2	97.81%	96.40%	98.43%	97.41%	185	0.0108
$PPO-CIS_3$	2	97.05%	96.88%	97.34%	97.11%	251.89	0.0079
$PPO-CIS_4$	2	97.80%	95.13%	98.00%	96.54%	168	0.0119
$PPO-CIS_5$	2	96.50%	95.83%	97.38%	96.60%	203	0.0099

Table 34: Inference Systems Comparison for Toxicity Detection on $\hat{D}^{\text{test}}_{\text{toxiGen}}$ Dataset

Model	**Action Space**	**Accuracy**$_{\alpha}$ (%)	**Precision**$_{\rho}$ (%)	**Recall**$_{\sigma}$ (%)	**F1-Score**$_{\phi}$ (%)	**Throughput**$_{\psi}$ (samples/s)	**Latency**$_{\lambda}$ (s)
$CETRA-CIS_1$	1	92.89%	92.89%	92.89%	92.89%	47.11	0.0425
$CETRA-CIS_2$	1	94.35%	94.14%	94.58%	94.36%	86.90	0.0230
$CETRA-CIS_3$	1	93.04%	92.98%	93.08%	93.03%	79.56	0.0251
$CETRA-CIS_4$	1	93.68%	93.60%	93.65%	93.62%	90.57	0.0221
$CETRA-CIS_5$	1	92.96%	91.83%	93.04%	92.43%	100.3	0.0199
$ACER-CIS$	2	96.87%	96.20%	96.92%	96.56%	206	0.0097
$ACER-CIS_1$	2	95.34%	95.27%	95.39%	95.33%	144	0.0139
$ACER-CIS_2$	2	94.89%	93.92%	95.17%	94.54%	120	0.0167
$ACER-CIS_3$	2	94.01%	93.35%	94.90%	94.12%	127	0.0157
$ACER-CIS_4$	2	95.75%	95.08%	96.65%	95.86%	135	0.0148
$ACER-CIS_5$	2	95.45%	94.78%	96.33%	95.55%	152	0.0132
$PPO-CIS$	2	97.03%	96.81%	97.95%	97.38%	296	0.0068
$PPO-CIS_1$	2	95.10%	93.93%	96.05%	94.98%	113	0.0177
$PPO-CIS_2$	2	96.25%	95.81%	96.37%	96.09%	125	0.0160
$PPO-CIS_3$	2	95.64%	93.99%	96.01%	94.99%	143	0.0140
$PPO-CIS_4$	2	95.92%	93.63%	96.57%	95.08%	182	0.0110
$PPO-CIS_5$	2	95.21%	94.95%	95.84%	95.39%	195	0.0103

We compared the performance of various inference systems for high-throughput toxicity detection on the $\hat{D}^{\text{test}}_{\text{kaggle}}$ dataset. According to the results in Table 33, the $CETRA-CIS$ models demonstrate a good balance between accuracy and throughput. The $CETRA-CIS_1$ model achieves an accuracy of 94.88% with a throughput of 63.72 samples per second and a latency of 0.0314 seconds. Among the $CETRA-CIS$ models, $CETRA-CIS_5$ has the highest throughput at 172.41 samples per second, but it compromises slightly on accuracy, achieving 94.17%. The $CETRA-CIS_4$ model provides a slightly higher accuracy of 95.48% with a throughput of 111.52 samples per second and a lower latency of 0.0179 seconds, making it a well-rounded option.

The $ACER-CIS$ models generally exhibit higher accuracy and throughput compared to $CETRA-CIS$ models. The $ACER-CIS$ model stands out with an accuracy of 97.76%, a high throughput of 213 samples per second, and a very low latency of 0.0094 seconds. This model is particularly effective for scenarios requiring high accuracy and fast processing times. Similarly, the $ACER-CIS_2$ model achieves an accuracy of 97.70% and a throughput of 105 samples per second with a latency of 0.0190 seconds.

Our proposed approach, the $PPO-CIS$ model, offers the highest accuracy among all models tested. The $PPO-CIS$ model achieves an impressive accuracy of 98.17%, with a throughput of 384 samples per second and a latency of 0.0052 seconds. This model is highly efficient, providing both high accuracy and fast processing times. The $PPO-CIS_3$ model also performs well, with an accuracy of 97.05%, a throughput of 251.89 samples per second, and a latency of 0.0079 seconds.

According to the results in Table 34, the $CETRA-CIS$ models demonstrate varied performance in terms of accuracy and throughput. The $CETRA-CIS_1$ model achieves an accuracy of 92.89% with a throughput of 47.11 samples per second and

a latency of 0.0425 seconds. Among the $CETRA-CIS$ models, $CETRA-CIS_5$ has the highest throughput at 100.3 samples per second, but it compromises slightly on accuracy, achieving 92.96%. The $CETRA-CIS_4$ model offers a balance with an accuracy of 93.68% and a throughput of 90.57 samples per second, with a latency of 0.0221 seconds.

The $ACER-CIS$ models generally exhibit higher accuracy and throughput compared to $CETRA-CIS$ models. The $ACER-CIS$ model stands out with an accuracy of 96.87%, a high throughput of 206 samples per second, and a very low latency of 0.0097 seconds. Similarly, the $ACER-CIS_3$ model achieves an accuracy of 94.01% and a throughput of 127 samples per second with a latency of 0.0157 seconds.

The $PPO-CIS$ models maintain their high performance, with the $PPO-CIS$ model achieving an accuracy of 97.03%, a throughput of 296 samples per second, and a latency of 0.0068 seconds. The $PPO-CIS_5$ model also performs well, with an accuracy of 95.21%, a throughput of 195 samples per second, and a latency of 0.0103 seconds.

To compare the throughput and accuracy achieved by all models, including random combinations with various label determination methods and different DRL-based approaches, we visualized the results for both datasets, as shown in Figure 21. The size of each bubble represents the latency, illustrating the trade-off between processing speed and accuracy across different models and combinations. The results indicate that the PPO-CIS model consistently outperforms other models on both datasets, achieving the highest levels of accuracy and throughput.

In summary, the proposed Cascade Inference System (CIS) with two action spaces and a new reward function that considers both accuracy and throughput, particularly when using Proximal Policy Optimization (PPO) as the deep reinforcement learning (DRL) model, delivers the best overall performance in terms of both accuracy and

throughput. The CIS using ACER as the DRL model ($ACER - CIS$) offers a commendable balance of high accuracy and low latency, demonstrating the effectiveness of the proposed CIS and reward function.

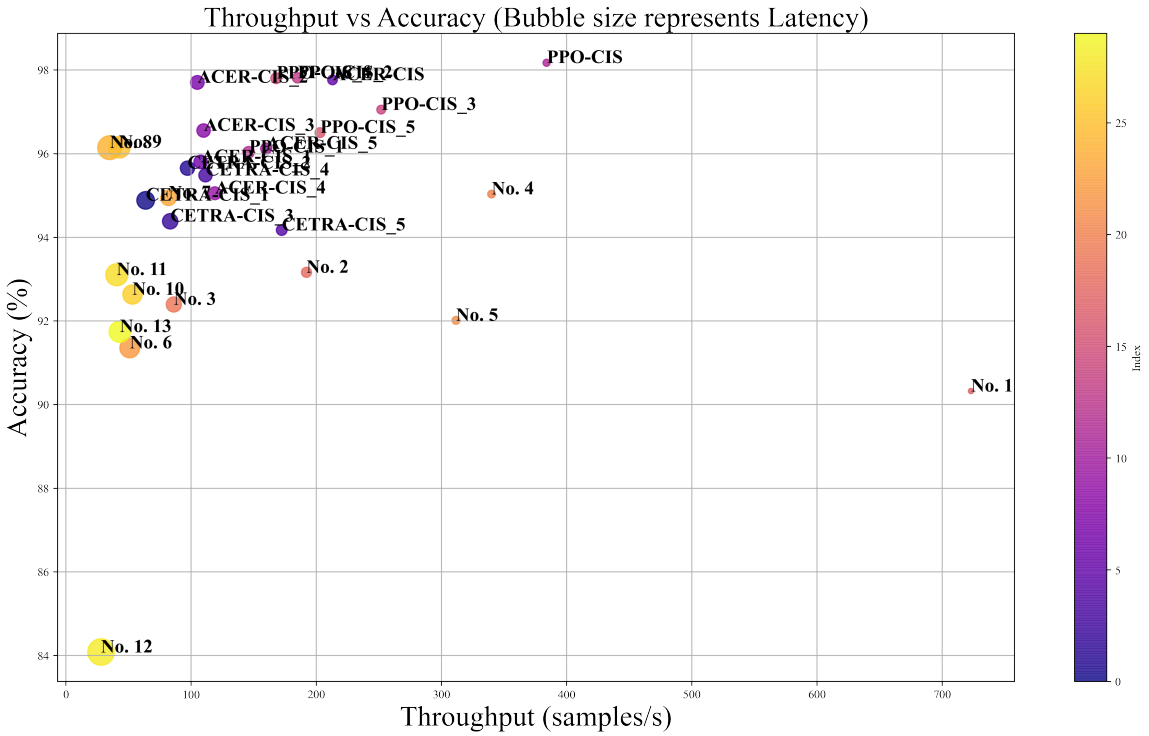
6.5 Summary

In this study, we aimed to develop a robust and efficient inference system for moderating toxic content on social media platforms. Our experimental analysis revealed that no single classifier excels in all performance aspects, highlighting the diversity in processing times and accuracy across various classifier architectures. This underscores the complexity of balancing accuracy with throughput, a critical factor for real-time applications such as content moderation.

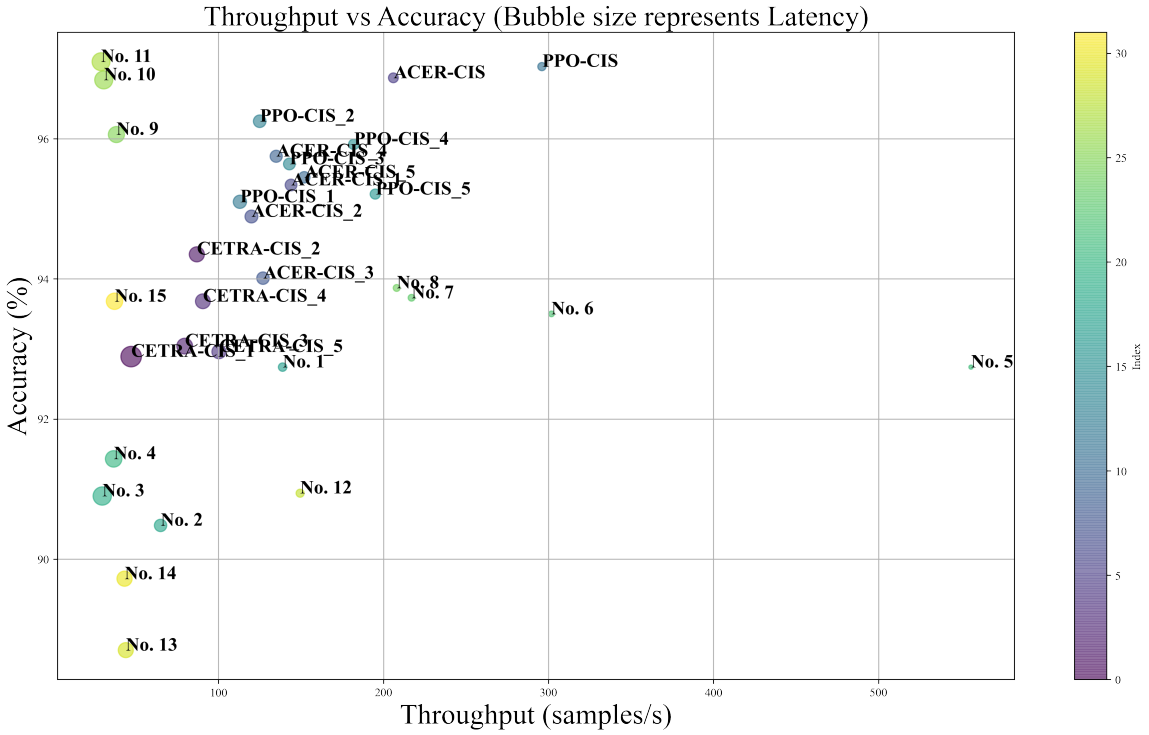
Our investigations suggested that while highly accurate classifiers often suffer from lower throughput, those prioritizing speed may compromise accuracy. To address this, we explored a cascaded arrangement of classifiers, which proved beneficial in leveraging the strengths of multiple systems to enhance overall performance. However, identifying the optimal combination of classifiers remains a non-trivial challenge that significantly impacts operational efficiency and effectiveness.

To overcome these limitations, we introduced a deep reinforcement learning approach, specifically the Proximal Policy Optimization (PPO) guided Cascaded Inference System (PPO-CIS). This system dynamically selects an appropriate cascade of classifiers based on the given context, adjusting via a reward mechanism that considers both accuracy and throughput to optimize decision-making continuously.

Comparative evaluations across two datasets, demonstrated that PPO-CIS consistently outperforms other models, achieving the highest levels of accuracy and throughput. The results show that PPO-CIS is a promising solution for effective and efficient toxicity detection, outperforming individual classifiers and other combinations.



(a) $\hat{D}^{\text{test}}_{\text{kaggle}}$



(b) $\hat{D}^{\text{test}}_{\text{toxigen}}$

Figure 21: Comparison of Throughput vs Accuracy for Various Content Moderation Inference Systems tested on different datasets: (a) $\hat{D}^{\text{test}}_{\text{kaggle}}$ and (b) $\hat{D}^{\text{test}}_{\text{toxigen}}$.

Implementing such a system not only aids in reducing the workload for human moderators but also contributes positively to user engagement and safety. By ensuring a faster and more accurate content moderation process, our approach helps maintain a healthier online environment.

Acknowledgments

This research is supported by NSERC Discovery Grants (RGPIN-2024-04087) and Canada Research Chairs Program (CRC-2019-00041).

Chapter 7

Conclusion and Future Work

Concluding Remarks

This thesis has explored the complexities and challenges associated with detecting and mitigating toxic language on social media platforms. Through a comprehensive multi-method approach, several key contributions have been made to enhance the effectiveness and efficiency of toxicity detection systems.

Firstly, we introduced the innovative Profit-Driven Simulation (PDS) Framework, which effectively determines the most profitable toxicity classifier for various social media environments characterized by differing levels of toxicity. By evaluating toxicity classifiers based on computational cost, real-time data processing capabilities, and the impact of false positives and negatives on user engagement and ad-generated revenue, the PDS framework identifies the optimal classifier for each scenario. Our simulation results demonstrated that some classifiers with higher accuracy, AUC, and F-scores were not necessarily appropriate for all environments, highlighting the necessity of balancing accuracy and throughput to enhance user satisfaction and engagement, ultimately leading to increased profits for social media companies. The PDS framework provides a valuable tool for researchers and practitioners to better understand and

address the complexities of toxic behavior on social media platforms.

Secondly, we developed AugmenToxic, a novel method for enhancing sentence-level text data that specifically targets toxic language. Leveraging reinforcement learning from human feedback with Proximal Policy Optimization (PPO), this approach enhances the performance of the fine-tuned FLAN-T5 model. By generating high-quality toxic responses while maintaining semantic coherence, AugmenToxic effectively addresses the issue of imbalanced datasets. Our extensive evaluations demonstrated that this method outperforms existing data augmentation techniques, generating a balanced and diverse dataset that significantly improves toxicity detection models.

Lastly, we proposed the Proximal Policy Optimization-based Cascaded Inference System (PPO-CIS), a novel adaptive multi-stage inference system for real-time toxicity detection. This system dynamically selects the most appropriate classifiers from a cascade, balancing accuracy and processing efficiency. By integrating deep reinforcement learning with PPO, PPO-CIS optimizes decision-making based on context, continuously improving performance. Comparative evaluations revealed that PPO-CIS consistently outperforms other models in both accuracy and throughput, making it a promising solution for effective and efficient toxicity detection. Implementing such a system not only aids in reducing the workload of human moderators, but also contributes positively to user engagement and safety.

In conclusion, this thesis has contributed to the development of more scalable, cost-effective, and adaptive toxicity detection systems in social media. These advancements not only enhance the safety and inclusiveness of online environments but also reduce the burden on human moderators, paving the way for more robust and equitable content moderation practices.

Future research directions

While the contributions of this thesis represent significant advancements in toxicity detection, several areas for future research remain.

For the PDS Framework, real-world validation with actual content and connections is crucial to further enhance its effectiveness. Future work will involve integrating the PDS framework into A/B testing within social media models to optimize the overall user experience and evaluate the system's real-world effectiveness. In addition, developing other classifiers using various datasets can also enhance accuracy.

Regarding AugmenToxic, future research will focus on generating samples for individual toxic labels to create a more detailed and balanced dataset, thereby enabling a deeper understanding and modeling of various forms of toxic language. Additionally, exploring the performance of other FLAN-T5 variants or larger models could yield further insights and improvements. Addressing the need for human processing of generated responses remains a significant challenge and will be a priority in future iterations of our research. Our study represents a substantial advancement in the development of robust toxicity detection models for online conversations. By acknowledging current limitations and outlining future research directions, we aim to contribute meaningfully to the field of natural language processing and online moderation.

For PPO-CIS, future enhancements will involve developing an advanced version of PPO-CIS with varying action space sizes, as the current model only utilizes two action spaces of the same size. Additionally, we aim to incorporate user behavior patterns into our framework. User activity varies significantly across platforms; for instance, users with large followings receive more comments and consequently face a higher ratio of toxic comments, whereas users with smaller, family-oriented networks encounter fewer toxic interactions. By integrating user activity patterns into the

agent's decision-making process, we can tailor policies to account for these differences, potentially enhancing the system's efficiency and accuracy. This user-centric approach could lead to more personalized and effective content moderation, ultimately improving the overall user experience and safety on social media platforms.

By addressing these areas in future research, we aim to continue advancing the field of toxicity detection, ultimately contributing to the creation of safer and more engaging online environments.

Despite the advancements made, several limitations and challenges require further attention:

Bias and Discrimination: Machine learning models can exhibit biases in modeling, training, and usage, potentially resulting in discrimination against specific social subgroups such as Black users, women, and LGBTQI+ communities. Addressing these biases is crucial for fair and equitable content moderation.

Complexity of Language: The proposed techniques may be limited by the nature of language itself. Sarcasm, irony, and other forms of figurative language can be challenging for algorithms to detect accurately, leading to misclassifications.

High False Positives and Negatives: Incorrectly identifying content as toxic (false positives) can suppress legitimate discourse, while failing to identify toxic content (false negatives) can allow harmful interactions to persist. Both types of errors can significantly impact user experience and trust.

Context Dependence: The meaning of a word or phrase can change based on the context in which it is used. Toxicity detection systems often struggle to understand the broader conversation or context, leading to misclassification.

Multilingual and Multimodal Content: Social media platforms host content in numerous languages and formats (text, images, videos). Developing models that can handle this diversity effectively is a significant challenge. Most current models

are primarily trained on English-language data, limiting their effectiveness in other languages.

Adversarial Attacks: Users who wish to bypass content moderation systems can use adversarial tactics, such as misspellings, deliberate obfuscations, or coded language, to evade detection. This requires constant updates to detection algorithms to keep up with new strategies.

Evolving Nature of Toxicity: The definitions and norms around what constitutes toxic or offensive content can change over time and vary across cultures. Detection systems must be adaptable to these changes to remain effective.

By addressing these areas in future research, the field of toxicity detection can be further advanced, ultimately contributing to the creation of safer and more engaging online environments.

Bibliography

[1] A. Bodaghi, B. C. M. Fung, and K. A. Schmitt, "Technological Solutions to Online Toxicity: Potential and Pitfalls," *IEEE Technology and Society Magazine*, vol. 42, no. 4, pp. 57–65, Dec. 2023, ISSN: 1937-416X. DOI: 10.1109/MTS.2023.3340235.

[2] A. Bodaghi, B. C. M. Fung, and J. Shahen, "A Profit-driven Simulation (PDS) Framework for Comparison of Deep Learning Models for Real-time Toxicity Detection in Social Media," *ACM Transactions on Knowledge Discovery from Data*, pp. 1–31, 2023.

[3] A. Bodaghi, B. C. M. Fung, and K. A. Schmitt, "AugmenToxic: Leveraging Reinforcement Learning to Optimize LLM Instruction Fine-Tuning for Data Augmentation to Enhance Toxicity Detection," *ACM Trans. Web*, no. Special Issue on Advances in Social Media Technologies and Analysis, Oct. 2024, ISSN: 1559-1131. DOI: 10.1145/3700791. [Online]. Available: https://dl.acm.org/doi/10.1145/3700791.

[4] A. Bodaghi, B. C. M. Fung, and K. A. Schmitt, "Optimizing Real-Time Toxicity Detection with Deep Reinforcement Learning-Driven Cascaded Classifiers," *IEEE Transactions on Computational Social Systems*, p. 27, 2024.

[5] D. Nafus, "'Patches don't have gender': What is not open in open source software," *New Media & Society*, vol. 14, no. 4, pp. 669–683, Jun. 2012, ISSN:

1461-4448. DOI: 10.1177/1461444811422887. [Online]. Available: https://doi.org/10.1177/1461444811422887 (visited on 04/28/2022).

[6] E. a. Vogels, *The State of Online Harassment*, Jan. 2021. [Online]. Available: https://www.pewresearch.org/internet/2021/01/13/the-state-of-online-harassment/ (visited on 04/19/2022).

[7] V. Maslej-Krešňáková, M. Sarnovský, P. Butka, and K. Machová, "Comparison of Deep Learning Models and Various Text Pre-Processing Techniques for the Toxic Comments Classification," en, *Applied Sciences*, vol. 10, no. 23, p. 8631, Jan. 2020, Number: 23 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2076-3417. DOI: 10.3390/app10238631. [Online]. Available: https://www.mdpi.com/2076-3417/10/23/8631 (visited on 04/30/2024).

[8] G. Jigsaw, *Jigsaw Toxic Comment Classification Challenge*, en, 2017. [Online]. Available: https://kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge (visited on 04/26/2022).

[9] R. Yan, Y. Li, D. Li, Y. Wang, Y. Zhu, and W. Wu, "A Stochastic Algorithm Based on Reverse Sampling Technique to Fight Against the Cyberbullying," en, *ACM Transactions on Knowledge Discovery from Data*, vol. 15, no. 4, pp. 1–22, Aug. 2021, ISSN: 1556-4681, 1556-472X. DOI: 10.1145/3441455. [Online]. Available: https://dl.acm.org/doi/10.1145/3441455 (visited on 08/25/2023).

[10] L. Abualigah, Y. Y. Al-Ajlouni, M. S. Daoud, M. Altalhi, and H. Migdady, "Fake news detection using recurrent neural network based on bidirectional LSTM and GloVe," en, *Social Network Analysis and Mining*, vol. 14, no. 1, p. 40, Feb. 2024, ISSN: 1869-5469. DOI: 10.1007/s13278-024-01198-w. [Online]. Available: https://doi.org/10.1007/s13278-024-01198-w (visited on 05/01/2024).

[11] R. Beniwal and A. Maurya, "Toxic Comment Classification Using Hybrid Deep Learning Model," 2021. DOI: 10.1007/978-981-15-8677-4_38.

[12] A. Hern, "Social network giants pledge to tackle abuse of women online," en-GB, *The Guardian*, Jul. 2021, ISSN: 0261-3077. [Online]. Available: https://www.theguardian.com/society/2021/jul/01/social-networks-facebook-google-twitter-tiktok-pledge-to-tackle-abuse-of-women-online (visited on 04/19/2022).

[13] F.-z. El-Alami, S. Ouatik El Alaoui, and N. En Nahnahi, "A multilingual offensive language detection method based on transfer learning from transformer fine-tuning model," en, *Journal of King Saud University - Computer and Information Sciences*, Jul. 2021, ISSN: 1319-1578. DOI: 10.1016/j.jksuci.2021.07.013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157821001804 (visited on 10/04/2021).

[14] R. Ranjan, "Modeling and Simulation in Performance Optimization of Big Data Processing Frameworks," *IEEE Cloud Computing*, vol. 1, no. 4, pp. 14–19, Nov. 2014, ISSN: 2325-6095. DOI: 10.1109/MCC.2014.84.

[15] J. Patel, "An Effective and Scalable Data Modeling for Enterprise Big Data Platform," in *2019 IEEE International Conference on Big Data (Big Data)*, Dec. 2019, pp. 2691–2697. DOI: 10.1109/BigData47090.2019.9005614. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9005614 (visited on 04/08/2024).

[16] R. Urbaniak, M. Ptaszyński, P. Tempska, G. Leliwa, M. Brochocki, and M. Wroczyński, "Personal attacks decrease user activity in social networking platforms," *Computers in Human Behavior*, vol. 126, p. 106 972, Jan. 2022, ISSN: 0747-5632. DOI: 10.1016/j.chb.2021.106972. [Online]. Available: https:

//www.sciencedirect.com/science/article/pii/S0747563221002958 (visited on 07/10/2024).

[17] P. Fortuna and S. Nunes, "A Survey on Automatic Detection of Hate Speech in Text," *ACM Computing Surveys*, vol. 51, no. 4, 85:1–85:30, Jul. 2018, ISSN: 0360-0300. DOI: 10.1145/3232676. [Online]. Available: https://doi.org/10.1145/3232676 (visited on 03/29/2023).

[18] A. Sheth, V. L. Shalin, and U. Kursuncu, "Defining and detecting toxicity on social media: Context and knowledge are key," en, *Neurocomputing*, vol. 490, pp. 312–318, Jun. 2022, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2021.11.095. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231221018087 (visited on 03/29/2023).

[19] I. D. Kivlichan, Z. Lin, J. Liu, and L. Vasserman, *Measuring and Improving Model-Moderator Collaboration using Uncertainty Estimation*, arXiv:2107.04212 [cs], Jul. 2021. DOI: 10.48550/arXiv.2107.04212. [Online]. Available: http://arxiv.org/abs/2107.04212 (visited on 04/19/2024).

[20] M. Bickert and B. Fishman, *Hard Questions: What Are We Doing to Stay Ahead of Terrorists?* en-US, Nov. 2018. [Online]. Available: https://about.fb.com/news/2018/11/staying-ahead-of-terrorists/ (visited on 04/11/2024).

[21] B. Etim, "The Times Sharply Increases Articles Open for Comments, Using Google's Technology," en-US, *The New York Times*, Jun. 2017, ISSN: 0362-4331. [Online]. Available: https://www.nytimes.com/2017/06/13/insider/have-a-comment-leave-a-comment.html (visited on 08/01/2024).

[22] A. Lees, V. Q. Tran, Y. Tay, *et al.*, *A New Generation of Perspective API: Efficient Multilingual Character-level Transformers*, arXiv:2202.11176 [cs], Feb.

2022. [Online]. Available: http://arxiv.org/abs/2202.11176 (visited on 04/11/2024).

[23] A. Sheth, V. L. Shalin, and U. Kursuncu, "Defining and detecting toxicity on social media: Context and knowledge are key," en, *Neurocomputing*, Dec. 2021, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2021.11.095. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231221018087 (visited on 03/30/2022).

[24] E. W. Pamungkas, V. Basile, and V. Patti, "Towards multidomain and multilingual abusive language detection: A survey," *Personal and Ubiquitous Computing*, Aug. 2021, ISSN: 1617-4917. DOI: 10.1007/s00779-021-01609-1. [Online]. Available: https://doi.org/10.1007/s00779-021-01609-1 (visited on 04/27/2022).

[25] N. Djuric, J. Zhou, R. Morris, M. Grbovic, V. Radosavljevic, and N. Bhamidipati, "Hate Speech Detection with Comment Embeddings," in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW '15 Companion, New York, NY, USA: Association for Computing Machinery, May 2015, pp. 29–30, ISBN: 978-1-4503-3473-0. DOI: 10.1145/2740908.2742760. [Online]. Available: https://doi.org/10.1145/2740908.2742760 (visited on 03/30/2023).

[26] T. Davidson, D. Bhattacharya, and I. Weber, "Racial Bias in Hate Speech and Abusive Language Detection Datasets," in *Proceedings of the Third Workshop on Abusive Language Online*, Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 25–35. DOI: 10.18653/v1/W19-3504. [Online]. Available: https://aclanthology.org/W19-3504 (visited on 04/28/2023).

[27] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang, "Abusive Language Detection in Online User Content," in *Proceedings of the 25th International Conference on World Wide Web*, ser. WWW '16, Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, Apr. 2016, pp. 145–153, ISBN: 978-1-4503-4143-1. DOI: 10.1145/2872427.2883062. [Online]. Available: https://doi.org/10.1145/2872427.2883062 (visited on 03/30/2023).

[28] M. Abulaish, A. Kamal, and M. J. Zaki, "A Survey of Figurative Language and Its Computational Detection in Online Social Networks," *ACM Transactions on the Web*, vol. 14, no. 1, 3:1–3:52, Feb. 2020, ISSN: 1559-1131. DOI: 10.1145/3375547. [Online]. Available: https://doi.org/10.1145/3375547 (visited on 03/30/2023).

[29] H. Hosseini, S. Kannan, B. Zhang, and R. Poovendran, *Deceiving Google's Perspective API Built for Detecting Toxic Comments*, arXiv:1702.08138 [cs], Feb. 2017. DOI: 10.48550/arXiv.1702.08138. [Online]. Available: http://arxiv.org/abs/1702.08138 (visited on 03/30/2023).

[30] J. S. Huffaker, J. K. Kummerfeld, W. S. Lasecki, and M. S. Ackerman, "Crowdsourced Detection of Emotionally Manipulative Language," en, in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, Honolulu HI USA: ACM, Apr. 2020, pp. 1–14, ISBN: 978-1-4503-6708-0. DOI: 10.1145/3313831.3376375. [Online]. Available: https://dl.acm.org/doi/10.1145/3313831.3376375 (visited on 03/30/2023).

[31] T. Gillespie, "Content moderation, AI, and the question of scale," en, *Big Data & Society*, vol. 7, no. 2, p. 2 053 951 720 943 234, Jul. 2020, Publisher: SAGE Publications Ltd, ISSN: 2053-9517. DOI: 10.1177/2053951720943234.

[Online]. Available: https://doi.org/10.1177/2053951720943234 (visited on 03/31/2023).

[32] H. Hosseinmardi, R. I. Rafiq, R. Han, Q. Lv, and S. Mishra, "Prediction of cyberbullying incidents in a media-based social network," in *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, Aug. 2016, pp. 186–192. DOI: 10.1109/ASONAM.2016.7752233.

[33] J. Liu, A. Cohen, R. Pasunuru, Y. Choi, H. Hajishirzi, and A. Celikyilmaz, *Don't throw away your value model! Making PPO even better via Value-Guided Monte-Carlo Tree Search decoding*, arXiv:2309.15028 [cs], Oct. 2023. DOI: 10.48550/arXiv.2309.15028. [Online]. Available: http://arxiv.org/abs/2309.15028 (visited on 12/17/2023).

[34] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, *The Curious Case of Neural Text Degeneration*, arXiv:1904.09751 [cs], Feb. 2020. DOI: 10.48550/arXiv.1904.09751. [Online]. Available: http://arxiv.org/abs/1904.09751 (visited on 12/14/2023).

[35] P. Gupta, C. Jiao, Y.-T. Yeh, S. Mehri, M. Eskenazi, and J. P. Bigham, *InstructDial: Improving Zero and Few-shot Generalization in Dialogue through Instruction Tuning*, en, arXiv:2205.12673 [cs], Oct. 2022. [Online]. Available: http://arxiv.org/abs/2205.12673 (visited on 05/02/2024).

[36] H. Xu, M. Chen, L. Huang, S. Vucetic, and W. Yin, *X-Shot: A Unified System to Handle Frequent, Few-shot and Zero-shot Learning Simultaneously in Classification*, en, arXiv:2403.03863 [cs], Mar. 2024. [Online]. Available: http://arxiv.org/abs/2403.03863 (visited on 05/02/2024).

[37] Y. Song, T. Wang, P. Cai, S. K. Mondal, and J. P. Sahoo, "A Comprehensive Survey of Few-shot Learning: Evolution, Applications, Challenges, and

Opportunities," *ACM Computing Surveys*, vol. 55, no. 13s, 271:1–271:40, Jul. 2023, ISSN: 0360-0300. DOI: 10.1145/3582688. [Online]. Available: https://doi.org/10.1145/3582688 (visited on 05/02/2024).

[38] S. Rahman, S. Khan, and F. Porikli, "A Unified Approach for Conventional Zero-Shot, Generalized Zero-Shot, and Few-Shot Learning," *IEEE Transactions on Image Processing*, vol. 27, no. 11, pp. 5652–5667, Nov. 2018, Conference Name: IEEE Transactions on Image Processing, ISSN: 1941-0042. DOI: 10.1109/TIP.2018.2861573. [Online]. Available: https://ieeexplore.ieee.org/document/8423721 (visited on 05/02/2024).

[39] J. Wei, M. Bosma, V. Y. Zhao, *et al.*, *Finetuned Language Models Are Zero-Shot Learners*, arXiv:2109.01652, Feb. 2022. DOI: 10.48550/arXiv.2109.01652. [Online]. Available: http://arxiv.org/abs/2109.01652 (visited on 11/04/2024).

[40] Y.-L. Sung, J. Cho, and M. Bansal, "VL-ADAPTER: Parameter-Efficient Transfer Learning for Vision-and-Language Tasks," en, in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, LA, USA: IEEE, Jun. 2022, pp. 5217–5227, ISBN: 978-1-66546-946-3. DOI: 10.1109/CVPR52688.2022.00516. [Online]. Available: https://ieeexplore.ieee.org/document/9878858/ (visited on 12/19/2023).

[41] D. Yu, S. Naik, A. Backurs, *et al.*, *Differentially Private Fine-tuning of Language Models*, arXiv:2110.06500 [cs, stat], Jul. 2022. [Online]. Available: http://arxiv.org/abs/2110.06500 (visited on 12/19/2023).

[42] N. Houlsby, A. Giurgiu, S. Jastrzebski, *et al.*, *Parameter-Efficient Transfer Learning for NLP*, arXiv:1902.00751 [cs, stat], Jun. 2019. DOI: 10.48550/arXiv.1902.00751. [Online]. Available: http://arxiv.org/abs/1902.00751 (visited on 12/19/2023).

[43] R. K. Mahabadi, J. Henderson, and S. Ruder, *Compacter: Efficient Low-Rank Hypercomplex Adapter Layers*, arXiv:2106.04647 [cs], Nov. 2021. DOI: 10.48550/arXiv.2106.04647. [Online]. Available: http://arxiv.org/abs/2106.04647 (visited on 12/19/2023).

[44] E. J. Hu, Y. Shen, P. Wallis, *et al.*, *LoRA: Low-Rank Adaptation of Large Language Models*, arXiv:2106.09685 [cs], Oct. 2021. DOI: 10.48550/arXiv.2106.09685. [Online]. Available: http://arxiv.org/abs/2106.09685 (visited on 12/19/2023).

[45] M. Weyssow, X. Zhou, K. Kim, D. Lo, and H. Sahraoui, *Exploring Parameter-Efficient Fine-Tuning Techniques for Code Generation with Large Language Models*, arXiv:2308.10462 [cs], Aug. 2023. [Online]. Available: http://arxiv.org/abs/2308.10462 (visited on 12/21/2023).

[46] C. P. Andriotis and K. G. Papakonstantinou, "Managing engineering systems with large state and action spaces through deep reinforcement learning," *Reliability Engineering & System Safety*, vol. 191, p. 106 483, Nov. 2019, ISSN: 0951-8320. DOI: 10.1016/j.ress.2019.04.036. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0951832018313309 (visited on 07/09/2024).

[47] Z. Wang, V. Bapst, N. Heess, *et al.*, *Sample Efficient Actor-Critic with Experience Replay*, arXiv:1611.01224 [cs], Jul. 2017. DOI: 10.48550/arXiv.1611.01224. [Online]. Available: http://arxiv.org/abs/1611.01224 (visited on 07/09/2024).

[48] S. Casper, X. Davies, C. Shi, *et al.*, *Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback*, en, arXiv:2307.15217 [cs], Sep. 2023. [Online]. Available: http://arxiv.org/abs/2307.15217 (visited on 08/07/2024).

[49] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep Reinforcement Learning from Human Preferences," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://papers.nips.cc/paper_files/paper/2017/hash/d5e2c0adad503c91f91df240d0cd4e49-Abstract.html (visited on 08/07/2024).

[50] D. Bill and T. Eriksson, *Fine-tuning a LLM using Reinforcement Learning from Human Feedback for a Therapy Chatbot Application*, eng. ROYAL INSTITUTE OF TECHNOLOGY, 2023. [Online]. Available: https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-331920 (visited on 12/21/2023).

[51] J. Dai, X. Pan, R. Sun, *et al.*, *Safe RLHF: Safe Reinforcement Learning from Human Feedback*, arXiv:2310.12773 [cs], Oct. 2023. DOI: 10.48550/arXiv.2310.12773. [Online]. Available: http://arxiv.org/abs/2310.12773 (visited on 08/07/2024).

[52] L. Arnold, S. Rebecchi, S. Chevallier, and H. Paugam-Moisy, "An Introduction to Deep Learning," Apr. 2011. [Online]. Available: https://www.semanticscholar.org/paper/An-Introduction-to-Deep-Learning-Arnold-Rebecchi/7fc7bb4eec2f27b8f0a0c7fb2a4112c7a7a7abed (visited on 07/09/2024).

[53] V. Francois-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An Introduction to Deep Reinforcement Learning," *Foundations and Trends® in Machine Learning*, vol. 11, no. 3-4, pp. 219–354, 2018, arXiv:1811.12560 [cs, stat], ISSN: 1935-8237, 1935-8245. DOI: 10.1561/2200000071. [Online]. Available: http://arxiv.org/abs/1811.12560 (visited on 07/09/2024).

[54] H. Dong, Z. Ding, and S. Zhang, Eds., *Deep Reinforcement Learning: Fundamentals, Research and Applications*, en. Singapore: Springer Singapore, 2020,

ISBN: 9789811540943 9789811540950. DOI: 10.1007/978-981-15-4095-0. [Online]. Available: http://link.springer.com/10.1007/978-981-15-4095-0 (visited on 07/09/2024).

[55] K. O'Shea and R. Nash, *An Introduction to Convolutional Neural Networks*, arXiv:1511.08458 [cs], Dec. 2015. DOI: 10.48550/arXiv.1511.08458. [Online]. Available: http://arxiv.org/abs/1511.08458 (visited on 07/09/2024).

[56] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Human-level control through deep reinforcement learning," en, *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, Publisher: Nature Publishing Group, ISSN: 1476-4687. DOI: 10.1038/nature14236. [Online]. Available: https://www.nature.com/articles/nature14236 (visited on 07/06/2024).

[57] S. Amarjyoti, *Deep Reinforcement Learning for Robotic Manipulation-The state of the art*, en, arXiv:1701.08878 [cs], Jan. 2017. [Online]. Available: http://arxiv.org/abs/1701.08878 (visited on 07/09/2024).

[58] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal Policy Optimization Algorithms*, arXiv:1707.06347 [cs], Aug. 2017. DOI: 10.48550/arXiv.1707.06347. [Online]. Available: http://arxiv.org/abs/1707.06347 (visited on 07/09/2024).

[59] J. Bai, Y. Li, M. Zheng, *et al.*, "A Sinh Cosh optimizer," *Knowledge-Based Systems*, vol. 282, p. 111 081, Dec. 2023, ISSN: 0950-7051. DOI: 10.1016/j.knosys.2023.111081. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705123008316 (visited on 05/01/2024).

[60] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy Gradient Methods for Reinforcement Learning with Function Approximation," in *Advances*

in Neural Information Processing Systems, vol. 12, MIT Press, 1999. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1999/hash/464d828b85b0bed98e80ade0a5c43b0f-Abstract.html (visited on 07/09/2024).

[61] J. Zhu, F. Wu, and J. Zhao, "An Overview of the Action Space for Deep Reinforcement Learning," en, in *2021 4th International Conference on Algorithms, Computing and Artificial Intelligence*, Sanya China: ACM, Dec. 2021, pp. 1–10, ISBN: 978-1-4503-8505-3. DOI: 10.1145/3508546.3508598. [Online]. Available: https://dl.acm.org/doi/10.1145/3508546.3508598 (visited on 07/12/2024).

[62] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951, Publisher: Institute of Mathematical Statistics, ISSN: 0003-4851. [Online]. Available: https://www.jstor.org/stable/2236703 (visited on 08/07/2024).

[63] V. Mnih, A. P. Badia, M. Mirza, *et al.*, *Asynchronous Methods for Deep Reinforcement Learning*, arXiv:1602.01783 [cs], Jun. 2016. DOI: 10.48550/arXiv.1602.01783. [Online]. Available: http://arxiv.org/abs/1602.01783 (visited on 07/09/2024).

[64] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, *Trust Region Policy Optimization*, arXiv:1502.05477 [cs], Apr. 2017. DOI: 10.48550/arXiv.1502.05477. [Online]. Available: http://arxiv.org/abs/1502.05477 (visited on 07/09/2024).

[65] J. Wiener and N. Bronson, *Facebook's Top Open Data Problems - Meta Research*, en, Oct. 2014. [Online]. Available: https://research.facebook.com/blog/2014/10/facebook-s-top-open-data-problems/ (visited on 09/21/2022).

[66] E. Weltevrede, A. Helmond, and C. Gerlitz, "The Politics of Real-time: A Device Perspective on Social Media Platforms and Search Engines," en, *Theory, Culture & Society*, vol. 31, no. 6, pp. 125–150, Nov. 2014, Publisher: SAGE Publications Ltd, ISSN: 0263-2764. DOI: 10.1177/0263276414537318. [Online]. Available: https://doi.org/10.1177/0263276414537318 (visited on 08/18/2023).

[67] D. Kirkpatrick, *Google: 53% of mobile users abandon sites that take over 3 seconds to load*, en-US, 2016. [Online]. Available: https://www.marketingdive.com/news/google-53-of-mobile-users-abandon-sites-that-take-over-3-seconds-to-load/426070/ (visited on 04/20/2022).

[68] R. W. Gehl, "The archive and the processor: The internal logic of Web 2.0," en, *New Media & Society*, vol. 13, no. 8, pp. 1228–1244, Dec. 2011, Publisher: SAGE Publications, ISSN: 1461-4448. DOI: 10.1177/1461444811401735. [Online]. Available: https://doi.org/10.1177/1461444811401735 (visited on 08/18/2023).

[69] C. N. d. Santos, I. Melnyk, and I. Padhi, *Fighting Offensive Language on Social Media with Unsupervised Text Style Transfer*, arXiv:1805.07685 [cs], May 2018. DOI: 10.48550/arXiv.1805.07685. [Online]. Available: http://arxiv.org/abs/1805.07685 (visited on 06/10/2024).

[70] M. A. H. Wadud, M. M. Kabir, M. F. Mridha, M. A. Ali, M. A. Hamid, and M. M. Monowar, "How can we manage Offensive Text in Social Media - A Text Classification Approach using LSTM-BOOST," en, *International Journal of Information Management Data Insights*, vol. 2, no. 2, p. 100 095, Nov. 2022, ISSN: 2667-0968. DOI: 10.1016/j.jjimei.2022.100095. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667096822000386 (visited on 09/20/2022).

[71] A.-M. Founta, C. Djouvas, D. Chatzakou, *et al.*, *Large Scale Crowdsourcing and Characterization of Twitter Abusive Behavior*, arXiv:1802.00393 [cs], Apr. 2018. DOI: 10.48550/arXiv.1802.00393. [Online]. Available: http://arxiv.org/abs/1802.00393 (visited on 04/08/2023).

[72] H. Qayyum, B. Z. H. Zhao, I. D. Wood, M. Ikram, M. A. Kaafar, and N. Kourtellis, *A deep dive into the consistently toxic 1% of Twitter*, arXiv:2202.07853 [cs], Feb. 2022. [Online]. Available: http://arxiv.org/abs/2202.07853 (visited on 09/22/2022).

[73] S. Jhaver, C. Boylston, D. Yang, and A. Bruckman, "Evaluating the Effectiveness of Deplatforming as a Moderation Strategy on Twitter," *Proceedings of the ACM on Human-Computer Interaction*, vol. 5, no. CSCW2, 381:1–381:30, Oct. 2021. DOI: 10.1145/3479525. [Online]. Available: https://doi.org/10.1145/3479525 (visited on 09/22/2022).

[74] M. Saveski, B. Roy, and D. Roy, "The Structure of Toxic Conversations on Twitter," en, in *Proceedings of the Web Conference 2021*, Ljubljana Slovenia: ACM, Apr. 2021, pp. 1086–1097, ISBN: 978-1-4503-8312-7. DOI: 10.1145/3442381.3449861. [Online]. Available: https://dl.acm.org/doi/10.1145/3442381.3449861 (visited on 09/22/2022).

[75] B. Radfar, K. Shivaram, and A. Culotta, "Characterizing Variation in Toxic Language by Social Context," en, *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 14, pp. 959–963, May 2020, ISSN: 2334-0770. [Online]. Available: https://ojs.aaai.org/index.php/ICWSM/article/view/7366 (visited on 10/12/2022).

[76] A. Vasalou, A. N. Joinson, and J. Pitt, "Constructing my online self: Avatars that increase self-focused attention," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '07, New York, NY, USA:

Association for Computing Machinery, Apr. 2007, pp. 445–448, ISBN: 978-1-59593-593-9. DOI: 10.1145/1240624.1240696. [Online]. Available: https://doi.org/10.1145/1240624.1240696 (visited on 04/10/2024).

[77] A. Giachanou and P. Rosso, "The Battle Against Online Harmful Information: The Cases of Fake News and Hate Speech," en, in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, Virtual Event Ireland: ACM, Oct. 2020, pp. 3503–3504, ISBN: 978-1-4503-6859-9. DOI: 10.1145/3340531.3412169. [Online]. Available: https://dl.acm.org/doi/10.1145/3340531.3412169 (visited on 03/28/2023).

[78] S. Kim, S. R. Colwell, A. Kata, M. H. Boyle, and K. Georgiades, "Cyberbullying Victimization and Adolescent Mental Health: Evidence of Differential Effects by Sex and Mental Health Problem Type," *Journal of Youth and Adolescence*, vol. 47, no. 3, pp. 661–672, Mar. 2018, ISSN: 1573-6601. DOI: 10.1007/s10964-017-0678-4. [Online]. Available: https://doi.org/10.1007/s10964-017-0678-4 (visited on 04/28/2022).

[79] E. M. Selkie, J. L. Fales, and M. A. Moreno, "Cyberbullying Prevalence Among US Middle and High School–Aged Adolescents: A Systematic Review and Quality Assessment," *Journal of Adolescent Health*, vol. 58, no. 2, pp. 125–133, Feb. 2016, ISSN: 1054-139X. DOI: 10.1016/j.jadohealth.2015.09.026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1054139X15003821 (visited on 04/28/2022).

[80] K. Cross, "Toward a formal sociology of online harassment," en, *Human Technology*, vol. 15, no. 3, pp. 326–346, Nov. 2019, Number: 3, ISSN: 1795-6889. [Online]. Available: https://ht.csr-pub.eu/index.php/ht/article/view/274 (visited on 08/01/2024).

[81] D. M. Douglas, "Doxing: A conceptual analysis," en, *Ethics and Information Technology*, vol. 18, no. 3, pp. 199–210, Sep. 2016, ISSN: 1572-8439. DOI: 10.1007/s10676-016-9406-0. [Online]. Available: https://doi.org/10.1007/s10676-016-9406-0 (visited on 08/01/2024).

[82] M. S. Jahan and M. Oussalah, "A systematic review of Hate Speech automatic detection using Natural Language Processing," *arXiv:2106.00742 [cs]*, May 2021, arXiv: 2106.00742. [Online]. Available: http://arxiv.org/abs/2106.00742 (visited on 04/27/2022).

[83] H. Yao, Y. Chen, Q. Ye, X. Jin, and X. Ren, "Refining Language Models with Compositional Explanations," in *Advances in Neural Information Processing Systems*, vol. 34, Curran Associates, Inc., 2021, pp. 8954–8967. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/hash/4b26dc4663ccf960c8538d595d0a1d3a-Abstract.html (visited on 08/01/2024).

[84] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010, Conference Name: IEEE Transactions on Knowledge and Data Engineering, ISSN: 1558-2191. DOI: 10.1109/TKDE.2009.191. [Online]. Available: https://ieeexplore.ieee.org/document/5288526 (visited on 08/01/2024).

[85] Z. Wang and A. Culotta, "Identifying Spurious Correlations for Robust Text Classification," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, Online: Association for Computational Linguistics, Nov. 2020, pp. 3431–3440. DOI: 10.18653/v1/2020.findings-emnlp.308. [Online]. Available: https://aclanthology.org/2020.findings-emnlp.308 (visited on 04/26/2023).

[86] X. Ferrer, T. v. Nuenen, J. M. Such, M. Coté, and N. Criado, "Bias and Discrimination in AI: A Cross-Disciplinary Perspective," *IEEE Technology and Society Magazine*, vol. 40, no. 2, pp. 72–80, Jun. 2021, Conference Name: IEEE Technology and Society Magazine, ISSN: 1937-416X. DOI: 10.1109/MTS.2021.3056293. [Online]. Available: https://ieeexplore.ieee.org/document/9445793 (visited on 10/27/2023).

[87] F. Faal, J. Yu, and K. Schmitt, "Domain Adaptation Multi-task Deep Neural Network for Mitigating Unintended Bias in Toxic Language Detection:" en, in *Proceedings of the 13th International Conference on Agents and Artificial Intelligence*, Online Streaming, — Select a Country —: SCITEPRESS - Science and Technology Publications, 2021, pp. 932–940, ISBN: 978-989-758-484-8. DOI: 10.5220/0010266109320940. [Online]. Available: https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0010266109320940 (visited on 04/03/2023).

[88] E. W. Pamungkas, V. Basile, and V. Patti, "Towards multidomain and multilingual abusive language detection: A survey," en, *Personal and Ubiquitous Computing*, vol. 27, no. 1, pp. 17–43, Feb. 2023, ISSN: 1617-4917. DOI: 10.1007/s00779-021-01609-1. [Online]. Available: https://doi.org/10.1007/s00779-021-01609-1 (visited on 10/24/2023).

[89] D. Hardage and P. Najafirad, "Hate and Toxic Speech Detection in the Context of Covid-19 Pandemic using XAI: Ongoing Applied Research," en, Jul. 2020. [Online]. Available: https://openreview.net/forum?id=7HP_0BgVX7v (visited on 08/25/2023).

[90] P. Malik, A. Aggrawal, and D. K. Vishwakarma, "Toxic Speech Detection using Traditional Machine Learning Models and BERT and fastText Embedding with Deep Neural Networks," in *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, Apr. 2021, pp. 1254–1259. DOI: 10.1109/ICCMC51019.2021.9418395. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9418395 (visited on 04/30/2024).

[91] T. Wijesiriwardene, H. Inan, U. Kursuncu, *et al.*, "ALONE: A Dataset for Toxic Behavior Among Adolescents on Twitter," in *Social Informatics*, S. Aref, K. Bontcheva, M. Braghieri, *et al.*, Eds., ser. Lecture Notes in Computer Science, event-place: Cham, Springer International Publishing, 2020, pp. 427–439, ISBN: 978-3-030-60975-7. DOI: 10.1007/978-3-030-60975-7_31.

[92] R. Mall, M. Nagpal, J. Salminen, H. Almerekhi, S.-G. Jung, and B. J. Jansen, "Four Types of Toxic People: Characterizing Online Users' Toxicity over Time," en, in *Proceedings of the 11th Nordic Conference on Human-Computer Interaction: Shaping Experiences, Shaping Society*, Tallinn Estonia: ACM, Oct. 2020, pp. 1–11, ISBN: 978-1-4503-7579-5. DOI: 10.1145/3419249.3420142. [Online]. Available: https://dl.acm.org/doi/10.1145/3419249.3420142 (visited on 05/24/2024).

[93] D. Androcec, "Machine learning methods for toxic comment classification: A systematic review," *Acta Universitatis Sapientiae, Informatica*, vol. 12, pp. 205–216, Dec. 2020. DOI: 10.2478/ausi-2020-0012.

[94] O. Makhnytkina, A. Matveev, D. Bogoradnikova, I. Lizunova, A. Maltseva, and N. Shilkina, "Detection of Toxic Language in Short Text Messages," en, in *Speech and Computer*, A. Karpov and R. Potapova, Eds., Cham: Springer

International Publishing, 2020, pp. 315–325, ISBN: 978-3-030-60276-5. DOI: 10.1007/978-3-030-60276-5_31.

[95] J. Salminen, M. Hopf, S. A. Chowdhury, S.-g. Jung, H. Almerekhi, and B. J. Jansen, "Developing an online hate classifier for multiple social media platforms," *Human-centric Computing and Information Sciences*, vol. 10, no. 1, p. 1, Jan. 2020, ISSN: 2192-1962. DOI: 10.1186/s13673-019-0205-6. [Online]. Available: https://doi.org/10.1186/s13673-019-0205-6 (visited on 04/17/2024).

[96] Rahul, H. Kajla, J. Hooda, and G. Saini, "Classification of Online Toxic Comments Using Machine Learning Algorithms," in *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, IEEE, May 2020, pp. 1119–1123. DOI: 10.1109/ICICCS48265.2020.9120939. [Online]. Available: https://ieeexplore.ieee.org/document/9120939 (visited on 04/30/2024).

[97] M. A. Saif, A. N. Medvedev, M. A. Medvedev, and T. Atanasova, "Classification of online toxic comments using the logistic regression and neural networks models," *AIP Conference Proceedings*, vol. 2048, no. 1, p. 060 011, Dec. 2018, ISSN: 0094-243X. DOI: 10.1063/1.5082126. [Online]. Available: https://doi.org/10.1063/1.5082126 (visited on 04/30/2024).

[98] A. Bonetti, M. Martínez-Sober, J. C. Torres, J. M. Vega, S. Pellerin, and J. Vila-Francés, "Comparison between Machine Learning and Deep Learning Approaches for the Detection of Toxic Comments on Social Networks," en, *Applied Sciences*, vol. 13, no. 10, p. 6038, Jan. 2023, Number: 10 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2076-3417. DOI: 10.3390/app13106038. [Online]. Available: https://www.mdpi.com/2076-3417/13/10/6038 (visited on 08/21/2023).

[99] S. Akuma, T. Lubem, and I. T. Adom, "Comparing Bag of Words and TF-IDF with different models for hate speech detection from live tweets," en, *International Journal of Information Technology*, vol. 14, no. 7, pp. 3629–3635, Dec. 2022, ISSN: 2511-2112. DOI: 10.1007/s41870-022-01096-4. [Online]. Available: https://doi.org/10.1007/s41870-022-01096-4 (visited on 08/21/2023).

[100] Y. HaCohen-Kerner, D. Miller, and Y. Yigal, "The influence of preprocessing on text classification using a bag-of-words representation," en, *PLOS ONE*, vol. 15, no. 5, e0232525, May 2020, Publisher: Public Library of Science, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0232525. [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0232525 (visited on 08/21/2023).

[101] V. Rupapara, F. Rustam, H. F. Shahzad, A. Mehmood, I. Ashraf, and G. S. Choi, "Impact of SMOTE on Imbalanced Text Features for Toxic Comments Classification Using RVVC Model," en, *IEEE Access*, vol. 9, pp. 78 621–78 634, 2021, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3083638. [Online]. Available: https://ieeexplore.ieee.org/document/9440474/ (visited on 04/30/2024).

[102] B. Vidgen and L. Derczynski, "Directions in abusive language training data, a systematic review: Garbage in, garbage out," en, *PLOS ONE*, vol. 15, no. 12, e0243300, Dec. 2020, Publisher: Public Library of Science, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0243300. [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0243300 (visited on 04/17/2024).

[103] Anjum and R. Katarya, "Hate speech, toxicity detection in online social media: A recent survey of state of the art and opportunities," en, *International Journal of Information Security*, vol. 23, no. 1, pp. 577–608, Feb. 2024, ISSN:

1615-5270. DOI: 10.1007/s10207-023-00755-2. [Online]. Available: https://doi.org/10.1007/s10207-023-00755-2 (visited on 04/18/2024).

[104] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, "Deep learning and its applications to machine health monitoring," *Mechanical Systems and Signal Processing*, vol. 115, pp. 213–237, Jan. 2019, ISSN: 0888-3270. DOI: 10.1016/j.ymssp.2018.05.050. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0888327018303108 (visited on 05/03/2022).

[105] C. E. R. Salim and D. Suhartono, "A Systematic Literature Review of Different Machine Learning Methods on Hate Speech Detection," *JOIV : International Journal on Informatics Visualization*, vol. 4, no. 4, pp. 213–218, Dec. 2020, ISSN: 2549-9904. DOI: 10.30630/joiv.4.4.476. [Online]. Available: http://joiv.org/index.php/joiv/article/view/476 (visited on 05/03/2022).

[106] A. Vaidya, F. Mai, and Y. Ning, "Empirical Analysis of Multi-Task Learning for Reducing Identity Bias in Toxic Comment Detection," en, *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 14, pp. 683–693, May 2020, ISSN: 2334-0770. DOI: 10.1609/icwsm.v14i1.7334. [Online]. Available: https://ojs.aaai.org/index.php/ICWSM/article/view/7334 (visited on 08/21/2023).

[107] Z. Wang and B. Zhang, "Toxic Comment Classification Based on Bidirectional Gated Recurrent Unit and Convolutional Neural Network," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 21, no. 3, 51:1–51:12, Dec. 2021, ISSN: 2375-4699. DOI: 10.1145/3488366. [Online]. Available: https://doi.org/10.1145/3488366 (visited on 08/21/2023).

[108] M. Ibrahim, M. Torki, and N. El-Makky, "Imbalanced Toxic Comments Classification Using Data Augmentation and Deep Learning," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*,

Dec. 2018, pp. 875–878. DOI: 10.1109/ICMLA.2018.00141. [Online]. Available: https://ieeexplore.ieee.org/document/8614166 (visited on 04/18/2024).

[109] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, en. MIT Press, Nov. 2016, Google-Books-ID: omivDQAAQBAJ, ISBN: 978-0-262-33737-3.

[110] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning*, ser. ICML '08, New York, NY, USA: Association for Computing Machinery, Jul. 2008, pp. 160–167, ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390177. [Online]. Available: https://doi.org/10.1145/1390156.1390177 (visited on 06/14/2022).

[111] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient Estimation of Word Representations in Vector Space*, arXiv:1301.3781 [cs], Sep. 2013. DOI: 10.48550/arXiv.1301.3781. [Online]. Available: http://arxiv.org/abs/1301.3781 (visited on 04/30/2024).

[112] J. Pennington, R. Socher, and C. Manning, "GloVe: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, A. Moschitti, B. Pang, and W. Daelemans, Eds., Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. [Online]. Available: https://aclanthology.org/D14-1162 (visited on 04/30/2024).

[113] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, *Learning Word Vectors for 157 Languages*, arXiv:1802.06893 [cs], Mar. 2018. DOI: 10.48550/arXiv.1802.06893. [Online]. Available: http://arxiv.org/abs/1802.06893 (visited on 01/05/2024).

[114] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017, Place: Cambridge, MA Publisher: MIT Press. DOI: 10.1162/tacl_a_00051. [Online]. Available: https://aclanthology.org/Q17-1010 (visited on 08/25/2023).

[115] H. H. MOHAMMED, E. DOGDU, A. K. GÖRÜR, and R. CHOUPANI, "Multi-Label Classification of Text Documents Using Deep Learning," in *2020 IEEE International Conference on Big Data (Big Data)*, IEEE, Dec. 2020, pp. 4681–4689. DOI: 10.1109/BigData50022.2020.9378266. [Online]. Available: https://ieeexplore.ieee.org/document/9378266 (visited on 05/24/2024).

[116] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv:1810.04805 [cs]*, May 2019, arXiv: 1810.04805. [Online]. Available: http://arxiv.org/abs/1810.04805 (visited on 05/03/2022).

[117] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations," en, Sep. 2019. DOI: 10.48550/arXiv.1909.11942. [Online]. Available: https://arxiv.org/abs/1909.11942v6 (visited on 10/31/2022).

[118] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, *DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter*, arXiv:1910.01108 [cs], Feb. 2020. DOI: 10.48550/arXiv.1910.01108. [Online]. Available: http://arxiv.org/abs/1910.01108 (visited on 04/30/2024).

[119] Y. Liu, M. Ott, N. Goyal, *et al.*, *RoBERTa: A Robustly Optimized BERT Pretraining Approach*, arXiv:1907.11692 [cs], Jul. 2019. DOI: 10.48550/arXiv.1907.11692. [Online]. Available: http://arxiv.org/abs/1907.11692 (visited on 07/15/2024).

[120] A. G. D'Sa, I. Illina, and D. Fohr, "BERT and fastText Embeddings for Automatic Detection of Toxic Speech," in *2020 International Multi-Conference on: "Organization of Knowledge and Advanced Technologies" (OCTA)*, IEEE, Feb. 2020, pp. 1–5. DOI: 10.1109/OCTA49274.2020.9151853. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9151853 (visited on 05/01/2024).

[121] Z. Zhao, Z. Zhang, and F. Hopfgartner, "A Comparative Study of Using Pre-trained Language Models for Toxic Comment Classification," in *Companion Proceedings of the Web Conference 2021*, ser. WWW '21, New York, NY, USA: Association for Computing Machinery, Jun. 2021, pp. 500–507, ISBN: 978-1-4503-8313-4. DOI: 10.1145/3442442.3452313. [Online]. Available: https://doi.org/10.1145/3442442.3452313 (visited on 04/30/2024).

[122] D. Dessì, D. R. Recupero, and H. Sack, "An Assessment of Deep Learning Models and Word Embeddings for Toxicity Detection within Online Textual Comments," en, *Electronics*, vol. 10, no. 7, p. 779, Jan. 2021, Number: 7 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2079-9292. DOI: 10.3390/electronics10070779. [Online]. Available: https://www.mdpi.com/2079-9292/10/7/779 (visited on 05/04/2022).

[123] T. Chu, K. Jue, and M. Wang, "Comment abuse classification with deep learning," *Von https://web. stanford. edu/class/cs224n/reports/2762092. pdf abgerufen*, 2016.

[124] F. Alkomah and X. Ma, "A Literature Review of Textual Hate Speech Detection Methods and Datasets," en, *Information*, vol. 13, no. 6, p. 273, Jun. 2022, Number: 6 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2078-2489. DOI: 10.3390/info13060273. [Online]. Available: https://www.mdpi.com/2078-2489/13/6/273 (visited on 04/18/2024).

[125] F. Museng, A. Jessica, N. Wijaya, A. Anderies, and I. A. Iswanto, "Systematic Literature Review: Toxic Comment Classification," en, in *2022 IEEE 7th International Conference on Information Technology and Digital Applications (ICITDA)*, Yogyakarta, Indonesia: IEEE, Nov. 2022, pp. 1–7, ISBN: 978-1-66546-136-8. DOI: 10.1109/ICITDA55840.2022.9971338. [Online]. Available: https://ieeexplore.ieee.org/document/9971338/ (visited on 04/17/2024).

[126] M. S. Jahan and M. Oussalah, "A systematic review of hate speech automatic detection using natural language processing," *Neurocomputing*, vol. 546, p. 126 232, Aug. 2023, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2023.126232. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231223003557 (visited on 07/29/2024).

[127] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735 (visited on 07/15/2024).

[128] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, IJCNN 2005, vol. 18, no. 5, pp. 602–610, Jul. 2005, ISSN: 0893-6080. DOI: 10.1016/j.neunet.2005.06.042. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608005001206 (visited on 07/15/2024).

[129] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches*, arXiv:1409.1259 [cs, stat], Oct. 2014. DOI: 10.48550/arXiv.1409.1259. [Online]. Available: http://arxiv.org/abs/1409.1259 (visited on 07/15/2024).

[130] L. Breiman, "Bagging predictors," en, *Machine Learning*, vol. 24, no. 2, pp. 123–140, Aug. 1996, ISSN: 1573-0565. DOI: 10.1007/BF00058655. [Online]. Available: https://doi.org/10.1007/BF00058655 (visited on 04/17/2024).

[131] R. E. Schapire, "The strength of weak learnability," en, *Machine Learning*, vol. 5, no. 2, pp. 197–227, Jun. 1990, ISSN: 1573-0565. DOI: 10.1007/BF00116037. [Online]. Available: https://doi.org/10.1007/BF00116037 (visited on 04/16/2024).

[132] K. Poojitha, A. S. Charish, M. A. K. Reddy, and S. Ayyasamy, *Classification of social media Toxic comments using Machine learning models*, arXiv:2304.06934 [cs], Apr. 2023. [Online]. Available: http://arxiv.org/abs/2304.06934 (visited on 04/18/2024).

[133] G. K. Pitsilis, H. Ramampiaro, and H. Langseth, "Detecting Offensive Language in Tweets Using Deep Learning," *Applied Intelligence*, vol. 48, no. 12, pp. 4730–4742, Dec. 2018, arXiv:1801.04433 [cs], ISSN: 0924-669X, 1573-7497. DOI: 10.1007/s10489-018-1242-y. [Online]. Available: http://arxiv.org/abs/1801.04433 (visited on 04/19/2024).

[134] M. K. A. Aljero and N. Dimililer, "A Novel Stacked Ensemble for Hate Speech Recognition," en, *Applied Sciences*, vol. 11, no. 24, p. 11 684, Jan. 2021, Number: 24 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2076-3417. DOI: 10.3390/app112411684. [Online]. Available: https://www.mdpi.com/2076-3417/11/24/11684 (visited on 04/17/2024).

[135] F. R. S. Nascimento, G. D. C. Cavalcanti, and M. Da Costa-Abreu, "Unintended bias evaluation: An analysis of hate speech detection and gender bias mitigation on social media using ensemble learning," *Expert Systems with Applications*, vol. 201, p. 117 032, Sep. 2022, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2022.

117032. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741742200447X (visited on 04/17/2024).

[136] S. Agarwal, A. Sonawane, and C. R. Chowdary, "Accelerating automatic hate speech detection using parallelized ensemble learning models," *Expert Systems with Applications*, vol. 230, p. 120 564, Nov. 2023, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2023.120564. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417423010667 (visited on 04/17/2024).

[137] J. Melton, A. Bagavathi, and S. Krishnan, "DeL-haTE: A Deep Learning Tunable Ensemble for Hate Speech Detection," en, in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Miami, FL, USA: IEEE, Dec. 2020, pp. 1015–1022, ISBN: 978-1-72818-470-8. DOI: 10.1109/ICMLA51294.2020.00165. [Online]. Available: https://ieeexplore.ieee.org/document/9356174/ (visited on 04/17/2024).

[138] S. A. Kokatnoor and B. Krishnan, "Twitter Hate Speech Detection using Stacked Weighted Ensemble (SWE) Model," in *2020 Fifth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, Nov. 2020, pp. 87–92. DOI: 10.1109/ICRCICN50933.2020.9296199. [Online]. Available: https://ieeexplore.ieee.org/document/9296199 (visited on 04/17/2024).

[139] O. Lavie, G. Katz, and A. Shabtai, *Cost Effective Transfer of Reinforcement Learning Policies*, en, SSRN Scholarly Paper, Rochester, NY, Jan. 2023. DOI: 10.2139/ssrn.4341615. [Online]. Available: https://papers.ssrn.com/abstract=4341615 (visited on 01/18/2024).

[140] G. Bansal, B. Nushi, E. Kamar, E. Horvitz, and D. S. Weld, *Is the Most Accurate AI the Best Teammate? Optimizing AI for Teamwork*, arXiv:2004.13102 [cs],

Feb. 2021. DOI: 10.48550/arXiv.2004.13102. [Online]. Available: http://arxiv.org/abs/2004.13102 (visited on 04/19/2024).

[141] V. K. Singh, U. Shrivastava, L. Bouayad, B. Padmanabhan, A. Ialynytchev, and S. K. Schultz, "Machine learning for psychiatric patient triaging: An investigation of cascading classifiers," *Journal of the American Medical Informatics Association*, vol. 25, no. 11, pp. 1481–1487, Nov. 2018, ISSN: 1527-974X. DOI: 10.1093/jamia/ocy109. [Online]. Available: https://doi.org/10.1093/jamia/ocy109 (visited on 04/12/2024).

[142] Y. Birman, S. Hindi, G. Katz, and A. Shabtai, "Cost-effective ensemble models selection using deep reinforcement learning," en, *Information Fusion*, vol. 77, pp. 133–148, Jan. 2022, ISSN: 1566-2535. DOI: 10.1016/j.inffus.2021.07.011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253521001524 (visited on 04/06/2023).

[143] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," en, in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, Kauai, HI, USA: IEEE Comput. Soc, 2001, pp. I–511–I–518, ISBN: 978-0-7695-1272-3. DOI: 10.1109/CVPR.2001.990517. [Online]. Available: http://ieeexplore.ieee.org/document/990517/ (visited on 04/10/2024).

[144] C. D. Sutton, "11 - Classification and Regression Trees, Bagging, and Boosting," in *Handbook of Statistics*, ser. Data Mining and Data Visualization, C. R. Rao, E. J. Wegman, and J. L. Solka, Eds., vol. 24, Elsevier, Jan. 2005, pp. 303–329. DOI: 10.1016/S0169-7161(04)24011-1. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169716104240111 (visited on 04/16/2024).

[145] Y. Yuan, Z. L. Yu, Z. Gu, *et al.*, "A novel multi-step Q-learning method to improve data efficiency for deep reinforcement learning," *Knowledge-Based Systems*, vol. 175, pp. 107–117, Jul. 2019, ISSN: 0950-7051. DOI: 10.1016/j.knosys.2019.03.018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705119301431 (visited on 04/24/2024).

[146] Y. Wang, S. Geng, and H. Gao, "A proactive decision support method based on deep reinforcement learning and state partition," *Knowledge-Based Systems*, vol. 143, pp. 248–258, Mar. 2018, ISSN: 0950-7051. DOI: 10.1016/j.knosys.2017.11.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095070511730504X (visited on 04/24/2024).

[147] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement Learning: A Survey," en, *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, May 1996, ISSN: 1076-9757. DOI: 10.1613/jair.301. [Online]. Available: https://www.jair.org/index.php/jair/article/view/10166 (visited on 04/24/2024).

[148] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, Nov. 2017, Conference Name: IEEE Signal Processing Magazine, ISSN: 1558-0792. DOI: 10.1109/MSP.2017.2743240. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8103164 (visited on 07/06/2024).

[149] R. S. Sutton and A. G. Barto, *Reinforcement Learning, second edition: An Introduction*, en. MIT Press, Nov. 2018, Google-Books-ID: uWV0DwAAQBAJ, ISBN: 978-0-262-35270-3.

[150] X. Li, Y.-N. Chen, L. Li, J. Gao, and A. Celikyilmaz, *End-to-End Task-Completion Neural Dialogue Systems*, arXiv:1703.01008 [cs], Feb. 2018. DOI:

10.48550/arXiv.1703.01008. [Online]. Available: http://arxiv.org/abs/1703.01008 (visited on 07/06/2024).

[151] D. Silver, A. Huang, C. J. Maddison, *et al.*, "Mastering the game of Go with deep neural networks and tree search," en, *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016, Publisher: Nature Publishing Group, ISSN: 1476-4687. DOI: 10.1038/nature16961. [Online]. Available: https://www.nature.com/articles/nature16961 (visited on 07/06/2024).

[152] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," en, *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, Publisher: Nature Publishing Group, ISSN: 1476-4687. DOI: 10.1038/nature14539. [Online]. Available: https://www.nature.com/articles/nature14539 (visited on 07/06/2024).

[153] S. Levine, C. Finn, T. Darrell, and P. Abbeel, *End-to-End Training of Deep Visuomotor Policies*, arXiv:1504.00702 [cs], Apr. 2016. DOI: 10.48550/arXiv.1504.00702. [Online]. Available: http://arxiv.org/abs/1504.00702 (visited on 07/06/2024).

[154] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, *Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection*, en, arXiv:1603.02199 [cs], Aug. 2016. [Online]. Available: http://arxiv.org/abs/1603.02199 (visited on 07/06/2024).

[155] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, *Robust Deep Reinforcement Learning with Adversarial Attacks*, arXiv:1712.03632 [cs], Dec. 2017. DOI: 10.48550/arXiv.1712.03632. [Online]. Available: http://arxiv.org/abs/1712.03632 (visited on 07/06/2024).

[156] L. Binxiang, Z. Gang, and S. Ruoying, "A Deep Reinforcement Learning Malware Detection Method Based on PE Feature Distribution," in *2019 6th*

International Conference on Information Science and Control Engineering (ICISCE), Dec. 2019, pp. 23–27. DOI: 10.1109/ICISCE48695.2019.00014. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9107644 (visited on 07/06/2024).

[157] B. Dhingra, L. Li, X. Li, *et al.*, *Towards End-to-End Reinforcement Learning of Dialogue Agents for Information Access*, arXiv:1609.00777 [cs], Apr. 2017. DOI: 10.48550/arXiv.1609.00777. [Online]. Available: http://arxiv.org/abs/1609.00777 (visited on 07/06/2024).

[158] Y. Li, *Deep Reinforcement Learning: An Overview*, arXiv:1701.07274 [cs], Nov. 2018. DOI: 10.48550/arXiv.1701.07274. [Online]. Available: http://arxiv.org/abs/1701.07274 (visited on 07/06/2024).

[159] Y. Xia, D. He, T. Qin, *et al.*, *Dual Learning for Machine Translation*, en, arXiv:1611.00179 [cs], Nov. 2016. [Online]. Available: http://arxiv.org/abs/1611.00179 (visited on 07/06/2024).

[160] D. Bahdanau, P. Brakel, K. Xu, *et al.*, *An Actor-Critic Algorithm for Sequence Prediction*, arXiv:1607.07086 [cs], Mar. 2017. DOI: 10.48550/arXiv.1607.07086. [Online]. Available: http://arxiv.org/abs/1607.07086 (visited on 07/06/2024).

[161] D. M. Ziegler, N. Stiennon, J. Wu, *et al.*, *Fine-Tuning Language Models from Human Preferences*, arXiv:1909.08593 [cs, stat], Jan. 2020. DOI: 10.48550/arXiv.1909.08593. [Online]. Available: http://arxiv.org/abs/1909.08593 (visited on 07/09/2024).

[162] F. Faal, K. Schmitt, and J. Y. Yu, "Reward Modeling for Mitigating Toxicity in Transformer-based Language Models," *Applied Intelligence*, vol. 53, no. 7, pp. 8421–8435, Apr. 2023, arXiv:2202.09662 [cs], ISSN: 0924-669X, 1573-7497.

DOI: 10.1007/s10489-022-03944-z. [Online]. Available: http://arxiv.org/abs/2202.09662 (visited on 12/22/2023).

[163] T. Hartvigsen, S. Gabriel, H. Palangi, M. Sap, D. Ray, and E. Kamar, *ToxiGen: A Large-Scale Machine-Generated Dataset for Adversarial and Implicit Hate Speech Detection*, arXiv:2203.09509 [cs], Jul. 2022. DOI: 10.48550/arXiv.2203.09509. [Online]. Available: http://arxiv.org/abs/2203.09509 (visited on 07/15/2024).

[164] A. Singh, "Detecting Toxicity in a Diverse Online Conversation using Reinforcement Learning," M.S. thesis, University of Maryland, Baltimore County, 2020. [Online]. Available: https://search.proquest.com/openview/b7af8344c480abe7ba0a9b705dce511c/1?pq-origsite=gscholar&cbl=18750&diss=y (visited on 07/19/2024).

[165] A. T. Aind, A. Ramnaney, and D. Sethia, "Q-Bully: A Reinforcement Learning based Cyberbullying Detection Framework," in *2020 International Conference for Emerging Technology (INCET)*, Jun. 2020, pp. 1–6. DOI: 10.1109/INCET49848.2020.9154092. [Online]. Available: https://ieeexplore.ieee.org/document/9154092 (visited on 07/27/2024).

[166] V. S. Raj, C. N. Subalalitha, L. Sambath, F. Glavin, and B. R. Chakravarthi, "ConBERT-RL: A policy-driven deep reinforcement learning based approach for detecting homophobia and transphobia in low-resource languages," *Natural Language Processing Journal*, vol. 6, p. 100 040, Mar. 2024, ISSN: 2949-7191. DOI: 10.1016/j.nlp.2023.100040. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2949719123000377 (visited on 07/06/2024).

[167] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence*

Research, vol. 16, pp. 321–357, Jun. 2002, arXiv:1106.1813 [cs], ISSN: 1076-9757. DOI: 10.1613/jair.953. [Online]. Available: http://arxiv.org/abs/1106.1813 (visited on 04/18/2023).

[168] A. Fernández, S. García, M. Galar, R. C. Prati, B. Krawczyk, and F. Herrera, *Learning from Imbalanced Data Sets*, en. Cham: Springer International Publishing, 2018, ISBN: 978-3-319-98073-7 978-3-319-98074-4. DOI: 10.1007/978-3-319-98074-4. [Online]. Available: http://link.springer.com/10.1007/978-3-319-98074-4 (visited on 11/09/2023).

[169] M. Koziarski, "CSMOUTE: Combined Synthetic Oversampling and Undersampling Technique for Imbalanced Data Classification," in *2021 International Joint Conference on Neural Networks (IJCNN)*, ISSN: 2161-4407, Jul. 2021, pp. 1–8. DOI: 10.1109/IJCNN52387.2021.9533415. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9533415 (visited on 05/24/2024).

[170] Y. Sun, M. S. Kamel, A. K. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," en, *Pattern Recognition*, vol. 40, no. 12, pp. 3358–3378, Dec. 2007, ISSN: 00313203. DOI: 10.1016/j.patcog.2007.04.009. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0031320307001835 (visited on 11/10/2023).

[171] K. M. Hasib, M. S. Iqbal, F. M. Shah, *et al.*, "A Survey of Methods for Managing the Classification and Solution of Data Imbalance Problem," *Journal of Computer Science*, vol. 16, no. 11, pp. 1546–1557, Nov. 2020, arXiv:2012.11870 [cs], ISSN: 1549-3636. DOI: 10.3844/jcssp.2020.1546.1557. [Online]. Available: http://arxiv.org/abs/2012.11870 (visited on 11/09/2023).

[172] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "RUSBoost: A Hybrid Approach to Alleviating Class Imbalance," *IEEE Transactions on*

Systems, Man, and Cybernetics - Part A: Systems and Humans, vol. 40, no. 1, pp. 185–197, Jan. 2010, Conference Name: IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, ISSN: 1558-2426. DOI: 10.1109/TSMCA.2009.2029559. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5299216 (visited on 11/14/2023).

[173] L. Nanni, C. Fantozzi, and N. Lazzarini, "Coupling different methods for overcoming the class imbalance problem," en, *Neurocomputing*, vol. 158, pp. 48–61, Jun. 2015, ISSN: 09252312. DOI: 10.1016/j.neucom.2015.01.068. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0925231215001411 (visited on 11/10/2023).

[174] W.-C. Lin, C.-F. Tsai, Y.-H. Hu, and J.-S. Jhang, "Clustering-based undersampling in class-imbalanced data," *Information Sciences*, vol. 409-410, pp. 17–26, Oct. 2017, ISSN: 0020-0255. DOI: 10.1016/j.ins.2017.05.008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025517307235 (visited on 11/10/2023).

[175] Z. Sun, Q. Song, X. Zhu, H. Sun, B. Xu, and Y. Zhou, "A novel ensemble method for classifying imbalanced data," *Pattern Recognition*, vol. 48, no. 5, pp. 1623–1637, May 2015, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2014.11.014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320314004841 (visited on 11/10/2023).

[176] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano, "An empirical comparison of repetitive undersampling techniques," in *2009 IEEE International Conference on Information Reuse & Integration*, Aug. 2009, pp. 29–34. DOI: 10.1109/IRI.2009.5211614. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5211614 (visited on 11/10/2023).

[177] P. Hart, "The condensed nearest neighbor rule (Corresp.)," *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 515–516, May 1968, Conference Name: IEEE Transactions on Information Theory, ISSN: 1557-9654. DOI: 10.1109/TIT.1968.1054155. [Online]. Available: https://ieeexplore.ieee.org/document/1054155 (visited on 11/10/2023).

[178] D. L. Wilson, "Asymptotic Properties of Nearest Neighbor Rules Using Edited Data," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-2, no. 3, pp. 408–421, Jul. 1972, Conference Name: IEEE Transactions on Systems, Man, and Cybernetics, ISSN: 2168-2909. DOI: 10.1109/TSMC.1972.4309137. [Online]. Available: https://ieeexplore.ieee.org/document/4309137 (visited on 11/10/2023).

[179] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: One-sided selection," in *Icml*, Issue: 1, vol. 97, Citeseer, 1997, p. 179. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=ebc3914181d76c817f0e35f788b7c4c0f80abb07 (visited on 11/10/2023).

[180] I. Tomek, "Two Modifications of CNN," in *IEEE Transactions on Systems, Man, and Cybernetics*, ISSN: 0018-9472, 2168-2909 Issue: 11 Journal Abbreviation: IEEE Trans. Syst., Man, Cybern., vol. SMC-6, Nov. 1976, pp. 769–772. DOI: 10.1109/TSMC.1976.4309452. [Online]. Available: http://ieeexplore.ieee.org/document/4309452/ (visited on 11/10/2023).

[181] G. Lemaître, F. Nogueira, and C. K. Aridas, "Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning," *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017, ISSN: 1533-7928. [Online]. Available: http://jmlr.org/papers/v18/16-365.html (visited on 11/10/2023).

[182] D. Mediratta and N. Oswal, *Detect Toxic Content to Improve Online Conversations*, arXiv:1911.01217 [cs], Oct. 2019. [Online]. Available: http://arxiv.org/abs/1911.01217 (visited on 11/10/2023).

[183] D. Devi, S. K. Biswas, and B. Purkayastha, "A Review on Solution to Class Imbalance Problem: Undersampling Approaches," in *2020 International Conference on Computational Performance Evaluation (ComPE)*, Shillong, India: IEEE, Jul. 2020, pp. 626–631, ISBN: 978-1-72816-644-5. DOI: 10.1109/ComPE49325.2020.9200087. [Online]. Available: https://ieeexplore.ieee.org/document/9200087/ (visited on 11/10/2023).

[184] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Editorial: Special issue on learning from imbalanced data sets," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 1–6, Jun. 2004, ISSN: 1931-0145. DOI: 10.1145/1007730.1007733. [Online]. Available: https://doi.org/10.1145/1007730.1007733 (visited on 04/18/2023).

[185] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 463–484, Jul. 2012, Conference Name: IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), ISSN: 1558-2442. DOI: 10.1109/TSMCC.2011.2161285. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5978225 (visited on 11/14/2023).

[186] M. S. Shelke, P. R. Deshmukh, and V. K. Shandilya, "A review on imbalanced data handling using undersampling and oversampling technique," *International Journal of Recent Trends in Engineering & Research (IJRTER)*, vol. 3, no. 4, pp. 444–449, 2017, ISSN: 2455-1457. DOI: 10.23883/IJRTER.2017.

3168.0UWXM. [Online]. Available: https://scholar.archive.org/work/teyyts4dwnfpnlyovsrbsfxw6a/access/wayback/http://www.ijrter.com/papers/volume-3/issue-4/a-review-on-imbalanced-data-handling-using-undersampling-and-oversampling-technique.pdf (visited on 11/10/2023).

[187] S. Hu, Y. Liang, L. Ma, and Y. He, "MSMOTE: Improving Classification Performance When Training Data is Imbalanced," in *2009 Second International Workshop on Computer Science and Engineering*, vol. 2, IEEE, Oct. 2009, pp. 13–17. DOI: 10.1109/WCSE.2009.756. [Online]. Available: https://ieeexplore.ieee.org/document/5403368 (visited on 11/10/2023).

[188] J. Stefanowski and S. Wilk, "Selective Pre-processing of Imbalanced Data for Improving Classification Performance," en, in *Data Warehousing and Knowledge Discovery*, I.-Y. Song, J. Eder, and T. M. Nguyen, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2008, pp. 283–292, ISBN: 978-3-540-85836-2. DOI: 10.1007/978-3-540-85836-2_27.

[189] M. Molinara, M. Ricamato, and F. Tortorella, "Facing Imbalanced Classes through Aggregation of Classifiers," in *14th International Conference on Image Analysis and Processing (ICIAP 2007)*, Sep. 2007, pp. 43–48. DOI: 10.1109/ICIAP.2007.4362755. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4362755 (visited on 11/12/2023).

[190] M. Bach, A. Werner, and M. Palt, "The Proposal of Undersampling Method for Learning from Imbalanced Datasets," *Procedia Computer Science*, Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 23rd International Conference KES2019, vol. 159, pp. 125–134, Jan. 2019, ISSN: 1877-0509. DOI: 10.1016/j.procs.2019.09.167. [Online]. Available: https:

//www.sciencedirect.com/science/article/pii/S1877050919313456 (visited on 11/10/2023).

[191] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Information Sciences*, vol. 250, pp. 113–141, Nov. 2013, ISSN: 0020-0255. DOI: 10.1016/j.ins.2013.07.007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025513005124 (visited on 11/10/2023).

[192] H. Ali, M. M. Salleh, R. Saedudin, K. Hussain, and M. F. Mushtaq, "Imbalance class problems in data mining: A review," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 14, no. 3, pp. 1560–1571, 2019. [Online]. Available: https://www.academia.edu/download/99950055/12240.pdf (visited on 05/24/2024).

[193] M. Joshi, V. Kumar, and R. Agarwal, "Evaluating boosting algorithms to classify rare classes: Comparison and improvements," in *Proceedings 2001 IEEE International Conference on Data Mining*, Nov. 2001, pp. 257–264. DOI: 10.1109/ICDM.2001.989527. [Online]. Available: https://ieeexplore.ieee.org/document/989527 (visited on 05/24/2024).

[194] C. Elkan, "The foundations of cost-sensitive learning," in *International joint conference on artificial intelligence*, Issue: 1, vol. 17, Lawrence Erlbaum Associates Ltd, 2001, pp. 973–978. [Online]. Available: http://cseweb.ucsd.edu/~elkan/rescale.pdf (visited on 12/07/2023).

[195] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997, Publisher: Elsevier. [Online].

Available: https://www.sciencedirect.com/science/article/pii/S002200009791504X (visited on 12/07/2023).

[196] M. Temraz and M. T. Keane, "Solving the class imbalance problem using a counterfactual method for data augmentation," *Machine Learning with Applications*, vol. 9, p. 100 375, Sep. 2022, ISSN: 2666-8270. DOI: 10.1016/j.mlwa.2022.100375. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666827022000652 (visited on 12/07/2023).

[197] R. Longadge and S. Dongre, *Class Imbalance Problem in Data Mining Review*, arXiv:1305.1707 [cs], May 2013. [Online]. Available: http://arxiv.org/abs/1305.1707 (visited on 11/14/2023).

[198] M. Wasikowski and X.-w. Chen, "Combating the Small Sample Class Imbalance Problem Using Feature Selection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1388–1400, Oct. 2010, Conference Name: IEEE Transactions on Knowledge and Data Engineering, ISSN: 1558-2191. DOI: 10.1109/TKDE.2009.187. [Online]. Available: https://ieeexplore.ieee.org/document/5276797 (visited on 11/14/2023).

[199] M. Bach and A. Werner, "Cost-Sensitive Feature Selection for Class Imbalance Problem," en, in *Information Systems Architecture and Technology: Proceedings of 38th International Conference on Information Systems Architecture and Technology – ISAT 2017*, L. Borzemski, J. Świątek, and Z. Wilimowska, Eds., ser. Advances in Intelligent Systems and Computing, Cham: Springer International Publishing, 2018, pp. 182–194, ISBN: 978-3-319-67220-5. DOI: 10.1007/978-3-319-67220-5_17.

[200] C. Zhang, K. C. Tan, H. Li, and G. S. Hong, "A Cost-Sensitive Deep Belief Network for Imbalanced Classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 1, pp. 109–122, Jan. 2019, Conference Name:

IEEE Transactions on Neural Networks and Learning Systems, ISSN: 2162-2388. DOI: 10.1109/TNNLS.2018.2832648.

[201] A. Singh and A. Jain, "An efficient credit card fraud detection approach using cost-sensitive weak learner with imbalanced dataset," en, *Computational Intelligence*, vol. 38, no. 6, pp. 2035–2055, 2022, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/coin.12555, ISSN: 1467-8640. DOI: 10.1111/coin.12555. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/coin.12555 (visited on 12/12/2023).

[202] I. D. Mienye and Y. Sun, "A survey of ensemble learning: Concepts, algorithms, applications, and prospects," *IEEE Access*, vol. 10, pp. 99 129–99 149, 2022, Publisher: IEEE. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9893798/ (visited on 05/17/2024).

[203] A. A. Khan, O. Chaudhari, and R. Chandra, "A review of ensemble learning and data augmentation models for class imbalanced problems: Combination, implementation and evaluation," *Expert Systems with Applications*, vol. 244, p. 122 778, Jun. 2024, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2023.122778. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417423032803 (visited on 04/26/2024).

[204] A. Onan, "GTR-GA: Harnessing the power of graph-based neural networks and genetic algorithms for text augmentation," *Expert Systems with Applications*, vol. 232, p. 120 908, Dec. 2023, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2023.120908. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417423014100 (visited on 12/07/2023).

[205] H. Q. Abonizio, E. C. Paraiso, and S. Barbon, "Toward Text Data Augmentation for Sentiment Analysis," en, *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 5, pp. 657–668, Oct. 2022, ISSN: 2691-4581. DOI: 10.1109/TAI.2021.

3114390. [Online]. Available: https://ieeexplore.ieee.org/document/9543519/ (visited on 10/30/2023).

[206] M. Bayer, M.-A. Kaufhold, and C. Reuter, "A Survey on Data Augmentation for Text Classification," *ACM Computing Surveys*, vol. 55, no. 7, 146:1–146:39, Dec. 2022, ISSN: 0360-0300. DOI: 10.1145/3544558. [Online]. Available: https://doi.org/10.1145/3544558 (visited on 12/07/2023).

[207] C. Coulombe, *Text Data Augmentation Made Simple By Leveraging NLP Cloud APIs*, arXiv:1812.04718 [cs], Dec. 2018. DOI: 10.48550/arXiv.1812.04718. [Online]. Available: http://arxiv.org/abs/1812.04718 (visited on 12/07/2023).

[208] G. Rizos, K. Hemker, and B. Schuller, "Augment to Prevent: Short-Text Data Augmentation in Deep Learning for Hate-Speech Classification," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, ser. CIKM '19, New York, NY, USA: Association for Computing Machinery, Nov. 2019, pp. 991–1000, ISBN: 978-1-4503-6976-3. DOI: 10.1145/3357384.3358040. [Online]. Available: https://doi.org/10.1145/3357384.3358040 (visited on 12/12/2023).

[209] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in *Advances in Neural Information Processing Systems*, vol. 26, Curran Associates, Inc., 2013. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html (visited on 12/12/2023).

[210] S. Sharifirad, B. Jafarpour, and S. Matwin, "Boosting Text Classification Performance on Sexist Tweets by Text Augmentation and Text Generation Using a Combination of Knowledge Graphs," in *Proceedings of the 2nd Workshop*

on Abusive Language Online (ALW2), D. Fišer, R. Huang, V. Prabhakaran, R. Voigt, Z. Waseem, and J. Wernimont, Eds., Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 107–114. DOI: 10.18653/v1/W18-5114. [Online]. Available: https://aclanthology.org/W18-5114 (visited on 12/12/2023).

[211] J. Quijas, "Analysing the effects of data augmentation and free parameters for text classification with recurrent convolutional neural networks," 2017. [Online]. Available: https://www.semanticscholar.org/paper/Analysing-the-effects-of-data-augmentation-and-free-Quijas/7dd727962dfe8a303a191aed177a3d6c89b6ab5b (visited on 05/09/2024).

[212] M. Jungiewicz and A. Smywiński-Pohl, "Towards textual data augmentation for neural networks: Synonyms and maximum loss," EN, *Computer Science*, vol. Vol. 20 (1), pp. 57–83, 2019, ISSN: 1508-2806. DOI: 10.7494/csci.2019.20.1.3023. [Online]. Available: http://yadda.icm.edu.pl/baztech/element/bwmeta1.element.baztech-56c51ccc-d18a-410e-aa00-f0e3340ae317 (visited on 05/09/2024).

[213] T. Niu and M. Bansal, *Automatically Learning Data Augmentation Policies for Dialogue Tasks*, arXiv:1909.12868 [cs], Sep. 2019. DOI: 10.48550/arXiv.1909.12868. [Online]. Available: http://arxiv.org/abs/1909.12868 (visited on 05/09/2024).

[214] J. Zhu, F. Gao, L. Wu, *et al.*, *Soft Contextual Data Augmentation for Neural Machine Translation*, en, arXiv:1905.10523 [cs], May 2019. [Online]. Available: http://arxiv.org/abs/1905.10523 (visited on 05/09/2024).

[215] G. G. Şahin and M. Steedman, *Data Augmentation via Dependency Tree Morphing for Low-Resource Languages*, arXiv:1903.09460 [cs], Mar. 2019. DOI:

10.48550/arXiv.1903.09460. [Online]. Available: http://arxiv.org/abs/1903.09460 (visited on 05/09/2024).

[216] H. Kim, D. Woo, S. J. Oh, J.-W. Cha, and Y.-S. Han, *ALP: Data Augmentation using Lexicalized PCFGs for Few-Shot Text Classification*, arXiv:2112.11916 [cs], Dec. 2021. DOI: 10.48550/arXiv.2112.11916. [Online]. Available: http://arxiv.org/abs/2112.11916 (visited on 05/09/2024).

[217] V. Kumar, A. Choudhary, and E. Cho, *Data Augmentation using Pre-trained Transformer Models*, arXiv:2003.02245 [cs], Jan. 2021. DOI: 10.48550/arXiv.2003.02245. [Online]. Available: http://arxiv.org/abs/2003.02245 (visited on 05/09/2024).

[218] R. Gupta, *Data augmentation for low resource sentiment analysis using generative adversarial networks*, arXiv:1902.06818 [cs, stat], Feb. 2019. DOI: 10.48550/arXiv.1902.06818. [Online]. Available: http://arxiv.org/abs/1902.06818 (visited on 05/09/2024).

[219] S. Witteveen and M. Andrews, "Paraphrasing with Large Language Models," in *Proceedings of the 3rd Workshop on Neural Generation and Translation*, arXiv:1911.09661 [cs], 2019, pp. 215–220. DOI: 10.18653/v1/D19-5623. [Online]. Available: http://arxiv.org/abs/1911.09661 (visited on 05/09/2024).

[220] D. R. Beddiar, M. S. Jahan, and M. Oussalah, "Data expansion using back translation and paraphrasing for hate speech detection," *Online Social Networks and Media*, vol. 24, p. 100 153, Jul. 2021, ISSN: 2468-6964. DOI: 10.1016/j.osnem.2021.100153. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2468696421000355 (visited on 10/30/2023).

[221] G. G. Şahin and M. Steedman, "Data Augmentation via Dependency Tree Morphing for Low-Resource Languages," in *Proceedings of the 2018 Conference*

on Empirical Methods in Natural Language Processing, E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, Eds., Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 5004–5009. DOI: 10.18653/v1/D18-1545. [Online]. Available: https://aclanthology.org/D18-1545 (visited on 12/12/2023).

[222] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019. [Online]. Available: https://insightcivic.s3.us-east-1.amazonaws.com/language-models.pdf (visited on 12/11/2023).

[223] T. Wullach, A. Adler, and E. Minkov, "Fight Fire with Fire: Fine-tuning Hate Detectors using Large Samples of Generated Hate Speech," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, Eds., Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 4699–4705. DOI: 10.18653/v1/2021.findings-emnlp.402. [Online]. Available: https://aclanthology.org/2021.findings-emnlp.402 (visited on 11/14/2023).

[224] A. G. D'Sa, I. Illina, D. Fohr, D. Klakow, and D. Ruiter, "Exploring Conditional Language Model Based Data Augmentation Approaches for Hate Speech Classification," en, in *Text, Speech, and Dialogue*, K. Ekštein, F. Pártl, and M. Konopík, Eds., vol. 12848, Series Title: Lecture Notes in Computer Science, Cham: Springer International Publishing, 2021, pp. 135–146, ISBN: 978-3-030-83526-2 978-3-030-83527-9. DOI: 10.1007/978-3-030-83527-9_12. [Online]. Available: https://link.springer.com/10.1007/978-3-030-83527-9_12 (visited on 11/14/2023).

[225] Y. Kim, S. Park, Y. Namgoong, and Y.-S. Han, "ConPrompt: Pre-training a Language Model with Machine-Generated Data for Implicit Hate Speech

Detection," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 10 964–10 980. DOI: 10.18653/v1/2023.findings-emnlp.731. [Online]. Available: https://aclanthology.org/2023.findings-emnlp.731 (visited on 05/05/2024).

[226] M. Juuti, T. Gröndahl, A. Flanagan, and N. Asokan, "A little goes a long way: Improving toxic language classification despite data scarcity," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, T. Cohn, Y. He, and Y. Liu, Eds., Online: Association for Computational Linguistics, Nov. 2020, pp. 2991–3009. DOI: 10.18653/v1/2020.findings-emnlp.269. [Online]. Available: https://aclanthology.org/2020.findings-emnlp.269 (visited on 11/14/2023).

[227] T. Wullach, A. Adler, and E. Minkov, *Towards Hate Speech Detection at Large via Deep Generative Modeling*, arXiv:2005.06370 [cs], May 2020. [Online]. Available: http://arxiv.org/abs/2005.06370 (visited on 12/12/2023).

[228] R. Liu, G. Xu, C. Jia, W. Ma, L. Wang, and S. Vosoughi, "Data Boost: Text Data Augmentation Through Reinforcement Learning Guided Conditional Generation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, arXiv:2012.02952 [cs], 2020, pp. 9031–9041. DOI: 10.18653/v1/2020.emnlp-main.726. [Online]. Available: http://arxiv.org/abs/2012.02952 (visited on 11/14/2023).

[229] A. Lee, X. Bai, I. Pres, M. Wattenberg, J. K. Kummerfeld, and R. Mihalcea, *A Mechanistic Understanding of Alignment Algorithms: A Case Study on DPO and Toxicity*, en, arXiv:2401.01967 [cs], Jan. 2024. [Online]. Available: http://arxiv.org/abs/2401.01967 (visited on 05/05/2024).

[230] C. F. G. D. Santos and J. P. Papa, "Avoiding Overfitting: A Survey on Regularization Methods for Convolutional Neural Networks," en, *ACM Computing Surveys*, vol. 54, no. 10s, pp. 1–25, Jan. 2022, ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3510413. [Online]. Available: https://dl.acm.org/doi/10.1145/3510413 (visited on 05/01/2024).

[231] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding Data Augmentation for Classification: When to Warp?" In *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, Nov. 2016, pp. 1–6. DOI: 10.1109/DICTA.2016.7797091. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7797091 (visited on 05/01/2024).

[232] Y. Meng, J. Huang, Y. Zhang, and J. Han, *Generating Training Data with Language Models: Towards Zero-Shot Language Understanding*, arXiv:2202.04538 [cs], Oct. 2022. DOI: 10.48550/arXiv.2202.04538. [Online]. Available: http://arxiv.org/abs/2202.04538 (visited on 05/17/2024).

[233] X. Qi, Y. Zeng, T. Xie, *et al.*, *Fine-tuning Aligned Language Models Compromises Safety, Even When Users Do Not Intend To!* en, arXiv:2310.03693 [cs], Oct. 2023. [Online]. Available: http://arxiv.org/abs/2310.03693 (visited on 05/01/2024).

[234] F. Liu, Y. Liu, L. Shi, *et al.*, *Exploring and Evaluating Hallucinations in LLM-Powered Code Generation*, en, arXiv:2404.00971 [cs], Apr. 2024. [Online]. Available: http://arxiv.org/abs/2404.00971 (visited on 05/01/2024).

[235] L. Huang, W. Yu, W. Ma, *et al.*, *A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions*, en, arXiv:2311.05232 [cs], Nov. 2023. [Online]. Available: http://arxiv.org/abs/2311.05232 (visited on 05/01/2024).

[236] A. S. Parihar, S. Thapa, and S. Mishra, "Hate Speech Detection Using Natural Language Processing: Applications and Challenges," in *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI)*, Jun. 2021, pp. 1302–1308. DOI: 10.1109/ICOEI51242.2021.9452882. [Online]. Available: https://ieeexplore.ieee.org/document/9452882 (visited on 10/26/2023).

[237] D. Tran, J. Liu, M. W. Dusenberry, *et al.*, *Plex: Towards Reliability using Pretrained Large Model Extensions*, arXiv:2207.07411 [cs, stat], Jul. 2022. [Online]. Available: http://arxiv.org/abs/2207.07411 (visited on 04/26/2023).

[238] L. Weidinger, J. Mellor, M. Rauh, *et al.*, *Ethical and social risks of harm from Language Models*, arXiv:2112.04359 [cs], Dec. 2021. [Online]. Available: http://arxiv.org/abs/2112.04359 (visited on 10/29/2023).

[239] M. Abdar, F. Pourpanah, S. Hussain, *et al.*, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," en, *Information Fusion*, vol. 76, pp. 243–297, Dec. 2021, ISSN: 1566-2535. DOI: 10.1016/j.inffus.2021.05.008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253521001081 (visited on 04/25/2023).

[240] N. A. Palomares and V. S. Wingate, "Victims' Goal Understanding, Uncertainty Reduction, and Perceptions in Cyberbullying: Theoretical Evidence From Three Experiments," en, *Journal of Computer-Mediated Communication*, vol. 25, no. 4, pp. 253–273, Jul. 2020, ISSN: 1083-6101. DOI: 10.1093/jcmc/zmaa005. [Online]. Available: https://academic.oup.com/jcmc/article/25/4/253/5858237 (visited on 10/29/2023).

[241] A. Luccioni and Y. Bengio, "On the Morality of Artificial Intelligence [Commentary]," en, *IEEE Technology and Society Magazine*, vol. 39, no. 1, pp. 16–25, Mar. 2020, ISSN: 0278-0097, 1937-416X. DOI: 10.1109/MTS.2020.2967486.

[Online]. Available: https://ieeexplore.ieee.org/document/9035523/ (visited on 10/31/2023).

[242] M. Sap, D. Card, S. Gabriel, Y. Choi, and N. A. Smith, "The Risk of Racial Bias in Hate Speech Detection," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 1668–1678. DOI: 10.18653/v1/P19-1163. [Online]. Available: https://aclanthology.org/P19-1163 (visited on 10/29/2023).

[243] G. Jigsaw, *Jigsaw Toxic Comment Classification Challenge*, 2017. [Online]. Available: https://kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge (visited on 04/26/2022).

[244] T. Caselli, V. Basile, J. Mitrović, and M. Granitzer, "HateBERT: Retraining BERT for Abusive Language Detection in English," in *Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021)*, A. Mostafazadeh Davani, D. Kiela, M. Lambert, B. Vidgen, V. Prabhakaran, and Z. Waseem, Eds., Online: Association for Computational Linguistics, Aug. 2021, pp. 17–25. DOI: 10.18653/v1/2021.woah-1.3. [Online]. Available: https://aclanthology.org/2021.woah-1.3 (visited on 10/31/2023).

[245] M. Díaz, I. Kivlichan, R. Rosen, *et al.*, "CrowdWorkSheets: Accounting for Individual and Collective Identities Underlying Crowdsourced Dataset Annotation," en, in *2022 ACM Conference on Fairness, Accountability, and Transparency*, Seoul Republic of Korea: ACM, Jun. 2022, pp. 2342–2351, ISBN: 978-1-4503-9352-2. DOI: 10.1145/3531146.3534647. [Online]. Available: https://dl.acm.org/doi/10.1145/3531146.3534647 (visited on 10/29/2023).

[246] J. Dzieza, *AI is a lot of work*, en, Jun. 2023. [Online]. Available: https://nymag.com/intelligencer/article/ai-artificial-intelligence-humans-technology-business-factory.html (visited on 11/27/2023).

[247] I. Olasov, *Offensive political dog whistles: You know them when you hear them. Or do you?* en, Nov. 2016. [Online]. Available: https://www.vox.com/the-big-idea/2016/11/7/13549154/dog-whistles-campaign-racism (visited on 11/27/2023).

[248] H. Adam, A. Balagopalan, E. Alsentzer, F. Christia, and M. Ghassemi, "Mitigating the impact of biased artificial intelligence in emergency decision-making," en, *Communications Medicine*, vol. 2, no. 1, pp. 1–6, Nov. 2022, Number: 1 Publisher: Nature Publishing Group, ISSN: 2730-664X. DOI: 10.1038/s43856-022-00214-4. [Online]. Available: https://www.nature.com/articles/s43856-022-00214-4 (visited on 10/30/2023).

[249] S. Ali, M. H. Saeed, E. Aldreabi, *et al.*, "Understanding the Effect of Deplatforming on Social Networks," in *Proceedings of the 13th ACM Web Science Conference 2021*, ser. WebSci '21, New York, NY, USA: Association for Computing Machinery, Jun. 2021, pp. 187–195, ISBN: 978-1-4503-8330-1. DOI: 10.1145/3447535.3462637. [Online]. Available: https://doi.org/10.1145/3447535.3462637 (visited on 10/24/2023).

[250] S. Jhaver, D. S. Appling, E. Gilbert, and A. Bruckman, ""Did You Suspect the Post Would be Removed?": Understanding User Reactions to Content Removals on Reddit," *Proceedings of the ACM on Human-Computer Interaction*, vol. 3, no. CSCW, 192:1–192:33, Nov. 2019. DOI: 10.1145/3359294. [Online]. Available: https://doi.org/10.1145/3359294 (visited on 04/29/2022).

[251] S. Susan and A. Kumar, "The balancing trick: Optimized sampling of imbalanced datasets—A brief survey of the recent State of the

Art," en, *Engineering Reports*, vol. 3, no. 4, e12298, 2021, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/eng2.12298, ISSN: 2577-8196. DOI: 10.1002/eng2.12298. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/eng2.12298 (visited on 11/06/2023).

[252] J. Prusa, T. M. Khoshgoftaar, D. J. Dittman, and A. Napolitano, "Using Random Undersampling to Alleviate Class Imbalance on Tweet Sentiment Data," in *2015 IEEE International Conference on Information Reuse and Integration*, Aug. 2015, pp. 197–202. DOI: 10.1109/IRI.2015.39. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7300975 (visited on 11/06/2023).

[253] V. Ganganwar, "An overview of classification algorithms for imbalanced datasets," *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, pp. 42–47, Jan. 2012.

[254] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Systems with Applications*, vol. 73, pp. 220–239, May 2017, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2016.12.035. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417416307175 (visited on 11/06/2023).

[255] S. Wang, W. Liu, J. Wu, L. Cao, Q. Meng, and P. J. Kennedy, "Training deep neural networks on imbalanced data sets," in *2016 International Joint Conference on Neural Networks (IJCNN)*, Vancouver, BC, Canada: IEEE, Jul. 2016, pp. 4368–4374, ISBN: 978-1-5090-0620-5. DOI: 10.1109/IJCNN.2016.7727770. [Online]. Available: http://ieeexplore.ieee.org/document/7727770/ (visited on 04/18/2023).

[256] C. Newton, *The secret lives of Facebook moderators in America*, en, Feb. 2019. [Online]. Available: https://www.theverge.com/2019/2/25/18229714/cognizant-facebook-content-moderator-interviews-trauma-working-conditions-arizona (visited on 10/30/2023).

[257] V. Elliott and T. Parmar, *The despair and darkness of people will get to you*, en-US, Jul. 2020. [Online]. Available: https://restofworld.org/2020/facebook-international-content-moderators/ (visited on 10/30/2023).

[258] S. Moscone, *The Beauty of Wattpad*, en, 2018. [Online]. Available: https://vocal.media/journal/the-beauty-of-wattpad (visited on 10/30/2023).

[259] L. Aratani, "Concern as US media hit with wave of layoffs amid rise of disinformation," en-GB, *The Guardian*, Dec. 2022, ISSN: 0261-3077. [Online]. Available: https://www.theguardian.com/media/2022/dec/10/media-layoffs-cnn-buzzfeed-gannett-recount-protocol (visited on 10/30/2023).

[260] P. Suciu, *X Is The Biggest Source Of Fake News And Disinformation, EU Warns*, en, Section: Social Media, Sep. 2023. [Online]. Available: https://www.forbes.com/sites/petersuciu/2023/09/26/x-is-the-biggest-source-of-fake-news-and-disinformation-eu-warns/ (visited on 10/30/2023).

[261] J. Cohen and U. o. S. California, *Analysis finds hate speech has significantly increased on Twitter*, en, Apr. 2023. [Online]. Available: https://phys.org/news/2023-04-analysis-speech-significantly-twitter.html (visited on 10/30/2023).

[262] R. Cao and R. K.-W. Lee, "HateGAN: Adversarial Generative-Based Data Augmentation for Hate Speech Detection," in *Proceedings of the 28th International Conference on Computational Linguistics*, D. Scott, N. Bel, and C. Zong, Eds.,

Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 6327–6338. DOI: 10.18653/v1/2020.coling-main.557. [Online]. Available: https://aclanthology.org/2020.coling-main.557 (visited on 05/01/2024).

[263] F.-z. El-Alami, S. Ouatik El Alaoui, and N. En Nahnahi, "A multilingual offensive language detection method based on transfer learning from transformer fine-tuning model," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 8, Part B, pp. 6048–6056, Sep. 2022, ISSN: 1319-1578. DOI: 10.1016/j.jksuci.2021.07.013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157821001804 (visited on 08/05/2024).

[264] G. Contardo, "Machine learning under budget constraints," en, Ph.D. dissertation, Université Pierre et Marie Curie - Paris VI, Jul. 2017. [Online]. Available: https://tel.archives-ouvertes.fr/tel-01677223 (visited on 04/27/2022).

[265] W. Li, "A Content-Based Approach for Analysing Cyberbullying on Sina Weibo," in *Proceedings of the 2nd International Conference on Information Management and Management Sciences*, ser. IMMS '19, New York, NY, USA: Association for Computing Machinery, Aug. 2019, pp. 33–37, ISBN: 978-1-4503-7144-5. DOI: 10.1145/3357292.3357294. [Online]. Available: https://doi.org/10.1145/3357292.3357294 (visited on 04/11/2023).

[266] J. P. Chang, J. Cheng, and C. Danescu-Niculescu-Mizil, "Don't Let Me Be Misunderstood:Comparing Intentions and Perceptions in Online Discussions," in *Proceedings of The Web Conference 2020*, New York, NY, USA: Association for Computing Machinery, Apr. 2020, pp. 2066–2077, ISBN: 978-1-4503-7023-3.

[Online]. Available: https://doi.org/10.1145/3366423.3380273 (visited on 04/29/2022).

[267] W. Wang, L. Chen, K. Thirunarayan, and A. P. Sheth, "Cursing in English on twitter," in *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, event-place: Baltimore Maryland USA, ACM, Feb. 2014, pp. 415–425, ISBN: 978-1-4503-2540-0. DOI: 10.1145/2531602.2531734. [Online]. Available: https://dl.acm.org/doi/10.1145/2531602.2531734 (visited on 04/28/2022).

[268] A. De Salve, P. Mori, B. Guidi, L. Ricci, and R. D. Pietro, "Predicting Influential Users in Online Social Network Groups," en, *ACM Transactions on Knowledge Discovery from Data*, vol. 15, no. 3, pp. 1–50, Jun. 2021, ISSN: 1556-4681, 1556-472X. DOI: 10.1145/3441447. [Online]. Available: https://dl.acm.org/doi/10.1145/3441447 (visited on 08/25/2023).

[269] C. A. Steed, T. E. Potok, R. M. Patton, J. R. Goodall, C. Maness, and J. Senter, *Interactive Visual Analysis of High Throughput Text Streams*, 2012.

[270] R. Zafarani, L. Tang, and H. Liu, "User Identification Across Social Media," en, *ACM Transactions on Knowledge Discovery from Data*, vol. 10, no. 2, pp. 1–30, Oct. 2015, ISSN: 1556-4681, 1556-472X. DOI: 10.1145/2747880. [Online]. Available: https://dl.acm.org/doi/10.1145/2747880 (visited on 08/25/2023).

[271] C. J. Adams, J. Sorensen, J. Elliott, *et al.*, *Toxic Comment Classification Challenge*, en, 2017. [Online]. Available: https://kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge (visited on 11/15/2023).

[272] L. Manikonda, Y. Hu, and S. Kambhampati, *Analyzing User Activities, Demographics, Social Network Structure and User-Generated Content on Instagram*, arXiv:1410.8099 [physics], Oct. 2014. [Online]. Available: http://arxiv.org/abs/1410.8099 (visited on 10/11/2022).

[273] T. Fawcett, "An introduction to ROC analysis," en, *Pattern Recognition Letters*, ROC Analysis in Pattern Recognition, vol. 27, no. 8, pp. 861–874, Jun. 2006, ISSN: 0167-8655. DOI: 10.1016/j.patrec.2005.10.010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016786550500303X (visited on 02/26/2023).

[274] J. Risch and R. Krestel, "Toxic Comment Detection in Online Discussions," en, in *Deep Learning-Based Approaches for Sentiment Analysis*, ser. Algorithms for Intelligent Systems, B. Agarwal, R. Nayak, N. Mittal, and S. Patnaik, Eds., Singapore: Springer, 2020, pp. 85–109, ISBN: 9789811512162. DOI: 10.1007/978-981-15-1216-2_4. [Online]. Available: https://doi.org/10.1007/978-981-15-1216-2_4 (visited on 11/02/2023).

[275] C. Chelmis and D.-S. Zois, "Dynamic, Incremental, and Continuous Detection of Cyberbullying in Online Social Media," *ACM Transactions on the Web*, vol. 15, no. 3, 14:1–14:33, May 2021, ISSN: 1559-1131. DOI: 10.1145/3448014. [Online]. Available: https://doi.org/10.1145/3448014 (visited on 09/27/2023).

[276] S. Rojas-Galeano, "On Obstructing Obscenity Obfuscation," en, *ACM Transactions on the Web*, vol. 11, no. 2, pp. 1–24, May 2017, ISSN: 1559-1131, 1559-114X. DOI: 10.1145/3032963. [Online]. Available: https://dl.acm.org/doi/10.1145/3032963 (visited on 11/02/2023).

[277] H. He and E. A. Garcia, "Learning from Imbalanced Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, Sep.

2009, Conference Name: IEEE Transactions on Knowledge and Data Engineering, ISSN: 1558-2191. DOI: 10.1109/TKDE.2008.239. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5128907 (visited on 11/06/2023).

[278] K. Madukwe, X. Gao, and B. Xue, "In Data We Trust: A Critical Analysis of Hate Speech Detection Datasets," in *Proceedings of the Fourth Workshop on Online Abuse and Harms*, Online: Association for Computational Linguistics, Nov. 2020, pp. 150–161. DOI: 10.18653/v1/2020.alw-1.18. [Online]. Available: https://aclanthology.org/2020.alw-1.18 (visited on 04/24/2023).

[279] M. A. Tahir, J. Kittler, and F. Yan, "Inverse random under sampling for class imbalance problem and its application to multi-label classification," *Pattern Recognition*, vol. 45, no. 10, pp. 3738–3750, Oct. 2012, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2012.03.014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320312001471 (visited on 11/06/2023).

[280] H. El Saadi, A. F. Al-Sadek, and M. W. Fakhr, "Informed under-sampling for enhancing patient specific epileptic seizure detection," *International Journal of Computer Applications*, vol. 57, no. 16, pp. 41–46, 2012, Publisher: Foundation of Computer Science. [Online]. Available: https://www.academia.edu/download/66547918/pxc3883733.pdf (visited on 12/07/2023).

[281] A. Gosain and S. Sardana, "Handling class imbalance problem using oversampling techniques: A review," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, IEEE, Sep. 2017, pp. 79–85. DOI: 10.1109/ICACCI.2017.8125820. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8125820 (visited on 12/07/2023).

[282] S. Choirunnisa and J. Lianto, "Hybrid Method of Undersampling and Oversampling for Handling Imbalanced Data," in *2018 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, IEEE, Nov. 2018, pp. 276–280. DOI: 10.1109/ISRITI.2018.8864335. [Online]. Available: https://ieeexplore.ieee.org/document/8864335 (visited on 05/24/2024).

[283] S. Hido, H. Kashima, and Y. Takahashi, "Roughly balanced bagging for imbalanced data," en, *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 2, no. 5-6, pp. 412–426, 2009, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sam.10061, ISSN: 1932-1872. DOI: 10.1002/sam.10061. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sam.10061 (visited on 11/07/2023).

[284] S. S. Rawat and A. K. Mishra, *Review of Methods for Handling Class-Imbalanced in Classification Problems*, arXiv:2211.05456 [cs], Nov. 2022. DOI: 10.48550/arXiv.2211.05456. [Online]. Available: http://arxiv.org/abs/2211.05456 (visited on 12/07/2023).

[285] Haibo He and E. Garcia, "Learning from Imbalanced Data," en, *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009, ISSN: 1041-4347. DOI: 10.1109/TKDE.2008.239. [Online]. Available: http://ieeexplore.ieee.org/document/5128907/ (visited on 10/03/2023).

[286] P. Liu, X. Wang, C. Xiang, and W. Meng, "A survey of text data augmentation," in *2020 International Conference on Computer Communication and Network Security (CCNS)*, IEEE, 2020, pp. 191–195. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9240734/ (visited on 05/24/2024).

[287] C. Shorten, T. M. Khoshgoftaar, and B. Furht, "Text Data Augmentation for Deep Learning," *Journal of Big Data*, vol. 8, no. 1, p. 101, Jul. 2021, ISSN: 2196-1115. DOI: 10.1186/s40537-021-00492-0. [Online]. Available: https://doi.org/10.1186/s40537-021-00492-0 (visited on 12/07/2023).

[288] J. Filar and K. Vrieze, *Competitive Markov Decision Processes*, en. Springer Science & Business Media, Dec. 2012, Google-Books-ID: uXDjBwAAQBAJ, ISBN: 978-1-4612-4054-9.

[289] Zhang, Y. a. Baldridge, J. a. He, and Luheng, *PAWS: Paraphrase Adversaries from Word Scrambling*, original-date: 2019-03-12T23:00:22Z, 2019. [Online]. Available: https://github.com/google-research-datasets/paws (visited on 11/15/2023).

[290] H. Naveed, A. U. Khan, S. Qiu, *et al.*, *A Comprehensive Overview of Large Language Models*, arXiv:2307.06435 [cs], Apr. 2024. DOI: 10.48550/arXiv.2307.06435. [Online]. Available: http://arxiv.org/abs/2307.06435 (visited on 05/05/2024).

[291] L. Ouyang, J. Wu, X. Jiang, *et al.*, *Training language models to follow instructions with human feedback*, arXiv:2203.02155 [cs], Mar. 2022. DOI: 10.48550/arXiv.2203.02155. [Online]. Available: http://arxiv.org/abs/2203.02155 (visited on 05/05/2024).

[292] H. W. Chung, L. Hou, S. Longpre, *et al.*, *Scaling Instruction-Finetuned Language Models*, arXiv:2210.11416 [cs], Dec. 2022. [Online]. Available: http://arxiv.org/abs/2210.11416 (visited on 10/02/2023).

[293] S. Zhang, L. Dong, X. Li, *et al.*, *Instruction Tuning for Large Language Models: A Survey*, arXiv:2308.10792 [cs], Oct. 2023. [Online]. Available: http://arxiv.org/abs/2308.10792 (visited on 12/19/2023).

[294] J. Kim, J. H. Lee, S. Kim, *et al.*, *Memory-Efficient Fine-Tuning of Compressed Large Language Models via sub-4-bit Integer Quantization*, arXiv:2305.14152 [cs], Oct. 2023. DOI: 10.48550/arXiv.2305.14152. [Online]. Available: http://arxiv.org/abs/2305.14152 (visited on 01/08/2024).

[295] R. Zhang, J. Han, C. Liu, *et al.*, *LLaMA-Adapter: Efficient Fine-tuning of Language Models with Zero-init Attention*, arXiv:2303.16199 [cs], Jun. 2023. [Online]. Available: http://arxiv.org/abs/2303.16199 (visited on 12/19/2023).

[296] M. Ghasemi, M. Zare, A. Zahedi, M.-A. Akbari, S. Mirjalili, and L. Abualigah, "Geyser Inspired Algorithm: A New Geological-inspired Meta-heuristic for Real-parameter and Constrained Engineering Optimization," en, *Journal of Bionic Engineering*, vol. 21, no. 1, pp. 374–408, Jan. 2024, ISSN: 2543-2141. DOI: 10.1007/s42235-023-00437-8. [Online]. Available: https://doi.org/10.1007/s42235-023-00437-8 (visited on 05/01/2024).

[297] J. O. Agushaka, A. E. Ezugwu, and L. Abualigah, "Gazelle optimization algorithm: A novel nature-inspired metaheuristic optimizer," en, *Neural Computing and Applications*, vol. 35, no. 5, pp. 4099–4131, Feb. 2023, ISSN: 1433-3058. DOI: 10.1007/s00521-022-07854-6. [Online]. Available: https://doi.org/10.1007/s00521-022-07854-6 (visited on 05/02/2024).

[298] J. O. Agushaka, A. E. Ezugwu, and L. Abualigah, "Dwarf Mongoose Optimization Algorithm," *Computer Methods in Applied Mechanics and Engineering*, vol. 391, p. 114 570, Mar. 2022, ISSN: 0045-7825. DOI: 10.1016/j.cma.2022.114570. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045782522000019 (visited on 05/02/2024).

[299] G. Hu, Y. Guo, G. Wei, and L. Abualigah, "Genghis Khan shark optimizer: A novel nature-inspired algorithm for engineering optimization," *Advanced Engineering Informatics*, vol. 58, p. 102 210, Oct. 2023, ISSN: 1474-0346. DOI: 10.

1016/j.aei.2023.102210. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474034623003385 (visited on 05/01/2024).

[300] M. Kale and A. Rastogi, "Text-to-Text Pre-Training for Data-to-Text Tasks," in *Proceedings of the 13th International Conference on Natural Language Generation*, B. Davis, Y. Graham, J. Kelleher, and Y. Sripada, Eds., Dublin, Ireland: Association for Computational Linguistics, Dec. 2020, pp. 97–102. [Online]. Available: https://aclanthology.org/2020.inlg-1.14 (visited on 11/15/2023).

[301] K. Anuranjana, "DiscoFlan: Instruction Fine-tuning and Refined Text Generation for Discourse Relation Label Classification," in *Proceedings of the 3rd Shared Task on Discourse Relation Parsing and Treebanking (DISRPT 2023)*, C. Braud, Y. J. Liu, E. Metheniti, *et al.*, Eds., Toronto, Canada: The Association for Computational Linguistics, Jul. 2023, pp. 22–28. DOI: 10.18653/v1/2023.disrpt-1.2. [Online]. Available: https://aclanthology.org/2023.disrpt-1.2 (visited on 11/15/2023).

[302] C.-Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries," in *Text Summarization Branches Out*, Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: https://aclanthology.org/W04-1013 (visited on 11/15/2023).

[303] X. Chi and Y. Xiang, "Augmenting Paraphrase Generation with Syntax Information Using Graph Convolutional Networks," *Entropy*, vol. 23, no. 5, p. 566, May 2021, ISSN: 1099-4300. DOI: 10.3390/e23050566. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8147394/ (visited on 11/15/2023).

[304] H. Palivela, "Optimization of paraphrase generation and identification using language models in natural language processing," *International Journal of*

Information Management Data Insights, vol. 1, no. 2, p. 100 025, Nov. 2021, ISSN: 2667-0968. DOI: 10.1016/j.jjimei.2021.100025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667096821000185 (visited on 11/15/2023).

[305] S. Banerjee and A. Lavie, "METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments," in *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, J. Goldstein, A. Lavie, C.-Y. Lin, and C. Voss, Eds., Ann Arbor, Michigan: Association for Computational Linguistics, Jun. 2005, pp. 65–72. [Online]. Available: https://aclanthology.org/W05-0909 (visited on 05/10/2024).

[306] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," *arXiv preprint arXiv:1904.09675*, 2019. [Online]. Available: https://arxiv.org/abs/1904.09675 (visited on 05/10/2024).

[307] H. Saadany and C. Orasan, *BLEU, METEOR, BERTScore: Evaluation of Metrics Performance in Assessing Critical Translation Errors in Sentiment-oriented Text*, en, Sep. 2021. DOI: 10.26615/978-954-452-071-7_006. [Online]. Available: https://arxiv.org/abs/2109.14250v1 (visited on 05/10/2024).

[308] A. V. Glazkova and D. A. Morozov, "Applying Transformer-Based Text Summarization for Keyphrase Generation," en, *Lobachevskii Journal of Mathematics*, vol. 44, no. 1, pp. 123–136, Jan. 2023, ISSN: 1818-9962. DOI: 10.1134/S1995080223010134. [Online]. Available: https://doi.org/10.1134/S1995080223010134 (visited on 05/10/2024).

[309] M. Denkowski and A. Lavie, "Meteor Universal: Language Specific Translation Evaluation for Any Target Language," in *Proceedings of the Ninth Workshop on Statistical Machine Translation*, O. Bojar, C. Buck, C. Federmann, *et al.*, Eds., Baltimore, Maryland, USA: Association for Computational Linguistics, Jun. 2014, pp. 376–380. DOI: 10.3115/v1/W14-3348. [Online]. Available: https://aclanthology.org/W14-3348 (visited on 05/10/2024).

[310] Jigsaw, *What do Perspective's scores mean?* en, Feb. 2021. [Online]. Available: https://medium.com/jigsaw/what-do-perspectives-scores-mean-113b37788a5d (visited on 11/15/2023).

[311] B. Vidgen, T. Thrush, Z. Waseem, and D. Kiela, *Learning from the Worst: Dynamically Generated Datasets to Improve Online Hate Detection*, arXiv:2012.15761 [cs], Jun. 2021. [Online]. Available: http://arxiv.org/abs/2012.15761 (visited on 11/16/2023).

[312] L. von Werra, Y. Belkada, L. Tunstall, *et al.*, *TRL: Transformer Reinforcement Learning*, 2020. [Online]. Available: https://github.com/huggingface/trl.

[313] S. Welleck, I. Kulikov, S. Roller, E. Dinan, K. Cho, and J. Weston, *Neural Text Generation with Unlikelihood Training*, arXiv:1908.04319 [cs, stat], Sep. 2019. [Online]. Available: http://arxiv.org/abs/1908.04319 (visited on 01/04/2024).

[314] A. Fan, M. Lewis, and Y. Dauphin, *Hierarchical Neural Story Generation*, arXiv:1805.04833 [cs], May 2018. DOI: 10.48550/arXiv.1805.04833. [Online]. Available: http://arxiv.org/abs/1805.04833 (visited on 01/04/2024).

[315] D. Q. Nguyen, T. Vu, and A. T. Nguyen, *BERTweet: A pre-trained language model for English Tweets*, arXiv:2005.10200 [cs], Oct. 2020. DOI: 10.48550/

arXiv.2005.10200. [Online]. Available: http://arxiv.org/abs/2005.10200 (visited on 05/10/2024).

[316] R. Gorwa, R. Binns, and C. Katzenbach, "Algorithmic content moderation: Technical and political challenges in the automation of platform governance," en, *Big Data & Society*, vol. 7, no. 1, p. 2 053 951 719 897 945, Jan. 2020, Publisher: SAGE Publications Ltd, ISSN: 2053-9517. DOI: 10.1177/2053951719897945. [Online]. Available: https://doi.org/10.1177/2053951719897945 (visited on 04/09/2024).

[317] Z. Waseem, T. Davidson, D. Warmsley, and I. Weber, "Understanding Abuse: A Typology of Abusive Language Detection Subtasks," in *Proceedings of the First Workshop on Abusive Language Online*, Z. Waseem, W. H. K. Chung, D. Hovy, and J. Tetreault, Eds., Vancouver, BC, Canada: Association for Computational Linguistics, Aug. 2017, pp. 78–84. DOI: 10.18653/v1/W17-3012. [Online]. Available: https://aclanthology.org/W17-3012 (visited on 07/15/2024).

[318] Y. Gerrard and H. Thornham, "Content moderation: Social media's sexist assemblages," en, *New Media & Society*, vol. 22, no. 7, pp. 1266–1286, Jul. 2020, Publisher: SAGE Publications, ISSN: 1461-4448. DOI: 10.1177/1461444820912540. [Online]. Available: https://doi.org/10.1177/1461444820912540 (visited on 04/11/2024).

[319] S. T. Roberts, *Behind the screen: content moderation in the shadows of social media*, en. New Haven: Yale University Press, 2019, OCLC: on1055263168, ISBN: 978-0-300-23588-3.

[320] J. Sani and N. Oseji, "UTILIZING THE POTENTIALS OF BIG DATA IN LIBRARY ENVIRONMENTS IN NIGERIAN FOR RECOMMENDER SERVICES," *Library Philosophy and Practice (e-journal)*, Jan. 2022. [Online]. Available: https://digitalcommons.unl.edu/libphilprac/7467.

[321] S. J. Dixon, *Biggest social media platforms 2024*, en, May 2024. [Online]. Available: https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/ (visited on 07/08/2024).

[322] B. Dean, *Social Network Usage & Growth Statistics: How Many People Use Social Media in 2024?* en-US, Dec. 2023. [Online]. Available: https://backlinko.com/social-media-users (visited on 07/08/2024).

[323] B. Ganesh and J. Bright, "Countering Extremists on Social Media: Challenges for Strategic Communication and Content Moderation," en, *Policy & Internet*, vol. 12, no. 1, pp. 6–19, 2020, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/poi3.236, ISSN: 1944-2866. DOI: 10.1002/poi3.236. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/poi3.236 (visited on 04/09/2024).

[324] R. Polikar, "Ensemble Learning," en, in *Ensemble Machine Learning: Methods and Applications*, C. Zhang and Y. Ma, Eds., New York, NY: Springer, 2012, pp. 1–34, ISBN: 978-1-4419-9326-7. DOI: 10.1007/978-1-4419-9326-7_1. [Online]. Available: https://doi.org/10.1007/978-1-4419-9326-7_1 (visited on 04/15/2024).

[325] Z.-H. Zhou, "Ensemble Learning," en, in *Machine Learning*, Z.-H. Zhou, Ed., Singapore: Springer, 2021, pp. 181–210, ISBN: 9789811519673. DOI: 10.1007/978-981-15-1967-3_8. [Online]. Available: https://doi.org/10.1007/978-981-15-1967-3_8 (visited on 04/15/2024).

[326] F. Husain, *Arabic Offensive Language Detection Using Machine Learning and Ensemble Machine Learning Approaches*, arXiv:2005.08946 [cs], May 2020. [Online]. Available: http://arxiv.org/abs/2005.08946 (visited on 04/15/2024).

[327] A. M. U. D. Khanday, S. T. Rabani, Q. R. Khan, and S. H. Malik, "Detecting twitter hate speech in COVID-19 era using machine learning and ensemble learning techniques," *International Journal of Information Management Data Insights*, vol. 2, no. 2, p. 100 120, Nov. 2022, ISSN: 2667-0968. DOI: 10.1016/j.jjimei.2022.100120. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667096822000635 (visited on 07/08/2024).

[328] A. C. Mazari, N. Boudoukhani, and A. Djeffal, "BERT-based ensemble learning for multi-aspect hate speech detection," en, *Cluster Computing*, vol. 27, no. 1, pp. 325–339, Feb. 2024, ISSN: 1573-7543. DOI: 10.1007/s10586-022-03956-x. [Online]. Available: https://doi.org/10.1007/s10586-022-03956-x (visited on 04/12/2024).

[329] M. Chen, Z. Xu, K. Weinberger, O. Chapelle, and D. Kedem, "Classifier Cascade for Minimizing Feature Evaluation Cost," en, in *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, ISSN: 1938-7228, PMLR, Mar. 2012, pp. 218–226. [Online]. Available: https://proceedings.mlr.press/v22/chen12c.html (visited on 05/10/2023).

[330] V. C. Raykar, B. Krishnapuram, and S. Yu, "Designing efficient cascaded classifiers: Tradeoff between accuracy and cost," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '10, New York, NY, USA: Association for Computing Machinery, Jul. 2010, pp. 853–860, ISBN: 978-1-4503-0055-1. DOI: 10.1145/1835804.1835912. [Online]. Available: https://doi.org/10.1145/1835804.1835912 (visited on 04/22/2024).

[331] J. I. A. Salas, P. Mirabal, and A. Ballester-Espinosa, "Cascade of Biased Two-class Classifiers for Multi-class Sentiment Analysis," 2021. [Online]. Available: https://www.semanticscholar.org/paper/Cascade-

of - Biased - Two - class - Classifiers - for - Salas - Mirabal / 436fde9146659d530e64168f80f9d316d2f9c8ec (visited on 04/10/2024).

[332] H. Calvo and O. Juárez Gambino, "Cascading Classifiers for Twitter Sentiment Analysis with Emotion Lexicons," en, in *Computational Linguistics and Intelligent Text Processing*, A. Gelbukh, Ed., Cham: Springer International Publishing, 2018, pp. 270–280, ISBN: 978-3-319-75487-1. DOI: 10.1007/978-3-319-75487-1_21.

[333] S. Paisitkriangkrai, "Robust object detection with efficient features and effective classifiers," PhD Thesis, UNSW Sydney, 2011. [Online]. Available: https://unsworks.unsw.edu.au/entities/publication/cd8df6ea-322a-46f0-b485-d094c6c649fd/full (visited on 04/15/2024).

[334] G. Forman and S. Rajaram, "Scaling up text classification for large file systems," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '08, New York, NY, USA: Association for Computing Machinery, Aug. 2008, pp. 239–246, ISBN: 978-1-60558-193-4. DOI: 10.1145/1401890.1401923. [Online]. Available: https://doi.org/10.1145/1401890.1401923 (visited on 04/15/2024).

[335] A. Vandierendonck, "On the Utility of Integrated Speed-Accuracy Measures when Speed-Accuracy Trade-off is Present," en-US, *Journal of Cognition*, vol. 4, no. 1, Mar. 2021, ISSN: 2514-4820. DOI: 10.5334/joc.154. [Online]. Available: https://journalofcognition.org/articles/10.5334/joc.154 (visited on 07/09/2024).

[336] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, *Advances in Pre-Training Distributed Word Representations*, arXiv:1712.09405 [cs], Dec. 2017. DOI: 10.48550/arXiv.1712.09405. [Online]. Available: http://arxiv.org/abs/1712.09405 (visited on 07/15/2024).

[337] X. Jiao, Y. Yin, L. Shang, *et al.*, *TinyBERT: Distilling BERT for Natural Language Understanding*, arXiv:1909.10351 [cs], Oct. 2020. DOI: 10.48550/arXiv.1909.10351. [Online]. Available: http://arxiv.org/abs/1909.10351 (visited on 07/15/2024).

[338] G. Brockman, V. Cheung, L. Pettersson, *et al.*, *OpenAI Gym*, arXiv:1606.01540 [cs], Jun. 2016. DOI: 10.48550/arXiv.1606.01540. [Online]. Available: http://arxiv.org/abs/1606.01540 (visited on 07/18/2024).

[339] A. Nandy and M. Biswas, "Reinforcement Learning with Keras, TensorFlow, and ChainerRL," en, in *Reinforcement Learning : With Open AI, TensorFlow and Keras Using Python*, A. Nandy and M. Biswas, Eds., Berkeley, CA: Apress, 2018, pp. 129–153, ISBN: 978-1-4842-3285-9. DOI: 10.1007/978-1-4842-3285-9_5. [Online]. Available: https://doi.org/10.1007/978-1-4842-3285-9_5 (visited on 07/18/2024).

Appendix A

In this section, we outline the experimental configurations for the classifiers developed in this research. We divided all datasets, including five balanced datasets generated using TDA and the original imbalanced dataset, into training and testing sets. The training set consists of 70% of the data, with 10% allocated for the validation set, and the remaining 20% designated for the test set, applied to both the Jigsaw and ToxiGen datasets. To ensure consistency across classifiers and datasets, we used a random-state approach for the train-test split, resulting in identical training and testing sets for all classifiers. All six classifiers received the same training and testing sets to maintain uniformity in the experimental setup.

Table 35: Experimental Configurations for CNN-based Classifiers based on Jigsaw Dataset

Model	Dataset	Epoch	Batch size	Learning rate	Dropout rate	Filters	Max Features	Max Length
CNN	$Unbalanced_{\text{Jigsaw}}$	35	128	5.00E-05	0.5	128	100000	400
	$Balanced_{\text{Zero-shot}}$	25	128	5.00E-03	0.3	128	100000	400
	$Balanced_{\text{Back-translation}}$	18	512	2.00E-05	0.5	128	100000	400
	$Balanced_{LoRA}$	28	512	2.00E-05	0.3	128	100000	400
	$Balanced_{\text{PPO-RoBERTa}}$	15	128	2.00E-04	0.5	300	100000	400
	$Balanced_{\text{PPO-API}}$	10	512	1.00E-05	0.5	300	100000	400
CNN-fasText	$Unbalanced_{\text{Jigsaw}}$	19	32	1.00E-03	0.4	128	100000	400
	$Balanced_{\text{Zero-shot}}$	17	256	5.00E-05	0.5	256	100000	400
	$Balanced_{\text{Back-translation}}$	33	128	3.00E-04	0.3	128	100000	400
	$Balanced_{LoRA}$	6	256	2.00E-04	0.4	300	100000	400
	$Balanced_{\text{PPO-RoBERTa}}$	22	256	5.00E-04	0.5	128	100000	400
	$Balanced_{\text{PPO-API}}$	25	512	5.00E-05	0.5	128	100000	400

Table 36: Experimental Configurations for CNN-based Classifiers based on ToxiGen Dataset

Model	Dataset	Epoch	Batch size	Learning rate	Dropout rate	Filters	Max Features	Max Length
CNN	$Unbalanced_{\text{ToxiGen}}$	31	512	5.00E-05	0.5	300	100000	400
	$Balanced_{\text{Zero-shot}}$	26	256	5.00E-03	0.5	300	100000	400
	$Balanced_{\text{Back-translation}}$	20	128	5.00E-05	0.5	300	100000	400
	$Balanced_{LoRA}$	22	512	5.00E-05	0.4	128	100000	400
	$Balanced_{\text{PPO-RoBERTa}}$	27	256	5.00E-05	0.4	128	100000	400
	$Balanced_{\text{PPO-API}}$	17	512	5.00E-05	0.4	128	100000	400
CNN-fasText	$Unbalanced_{\text{Jigsaw}}$	12	256	2.00E-03	0.5	128	100000	400
	$Balanced_{\text{Zero-shot}}$	15	256	5.00E-05	0.5	300	100000	400
	$Balanced_{\text{Back-translation}}$	24	128	4.00E-04	0.5	300	100000	400
	$Balanced_{LoRA}$	18	64	4.00E-04	0.5	300	100000	400
	$Balanced_{\text{PPO-RoBERTa}}$	15	64	5.00E-05	0.5	300	100000	400
	$Balanced_{\text{PPO-API}}$	23	64	5.00E-05	0.5	300	100000	400

CNN-based: CNN and CNN-fastText for balanced datasets

We started the training process with a data cleaning and preprocessing phase as the initial step. This process includes the removal of URLs, punctuation, digits, stopwords, and the normalization of cases, acronyms, and abbreviations. Subsequently, the tokenization process takes place, with text comments either truncated or padded to achieve a fixed length before being transformed into vector words.

The model architecture incorporates four 1D convolutional layers with multiple

window sizes and output channels. A pooling layer is employed to reduce the dimensionality of the convolutions, followed by a global max pooling step to further diminish the dimension. ReLU activation functions are used in the hidden layers, while the output layer employs a Sigmoid activation function. During training, the binary cross-entropy loss function measures the disparity between predicted and actual labels. The Adam optimizer is applied to update the model's parameters, facilitating faster convergence and improved accuracy.

To expedite the training process, we integrated pre-trained fastText word embeddings into our CNN model, resulting in the CNN-fastText variant. Specifically, we utilized the "crawl-300d-2M" embeddings, consisting of 2 million word vectors trained on Common Crawl using a continuous bag of words (CBOW) with position weights, character n-grams of length 5, and a window of size 5 and 10 negatives. The CNN model convolves over these embedded vectors, which then undergo max pooling to reduce dimensionality. The final classification is executed using fully connected layers with a Sigmoid activation function. Incorporating pre-trained word embeddings significantly reduced the training time of our CNN model. We systematically tested hyperparameters to optimize results for each dataset. A detailed overview of the final hyperparameter settings is presented in Table 39 for datasets developed using prompts from the Jigsaw dataset and in Table 36 for those developed using the ToxiGen dataset.

Transformer-based: BERT, RoBERTa, HateBERT, and BERTweet

We used the Huggingface transformer library, compatible with Tensorflow 2.14.0, for our work. To identify toxic content, we employed the bert-base-uncased, roberta-base, HateBERT, and bertweet-base variants. Text data preparation involved the use of tokenizers, which convert raw text into tokens compatible with BERT/RoBERTa. We

Table 37: Experimental Configurations for Transformer-based Classifiers: Jigsaw-based Datasets

Model	Dataset	Epoch	Batch size	Learning rate	Dropout rate	Max Length
BERT	$Unbalanced_{\text{Jigsaw}}$	2	16	2e-5	0.5	400
	$Balanced_{\text{Zero-shot}}$	2	8	3e-5	0.4	400
	$Balanced_{\text{Back-translation}}$	2	8	3e-5	0.4	400
	$Balanced_{LoRA}$	1	8	4e-5	0.5	400
	$Balanced_{\text{PPO-RoBERTa}}$	1	2	2e-5	0.4	400
	$Balanced_{\text{PPO-API}}$	1	4	2e-5	0.5	400
RoBERTa	$Unbalanced_{\text{Jigsaw}}$	2	16	4e-5	0.3	400
	$Balanced_{\text{Zero-shot}}$	2	8	3e-5	0.4	400
	$Balanced_{\text{Back-translation}}$	3	8	3e-5	0.3	400
	$Balanced_{LoRA}$	1	16	3e-5	0.4	400
	$Balanced_{\text{PPO-RoBERTa}}$	1	8	3e-5	0.5	400
	$Balanced_{\text{PPO-API}}$	1	2	3e-5	0.4	400
Hate-BERT	$Unbalanced_{\text{Jigsaw}}$	2	16	4e-5	0.3	400
	$Balanced_{\text{Zero-shot}}$	2	128	3e-5	0.4	400
	$Balanced_{\text{Back-translation}}$	2	64	2e-5	0.4	400
	$Balanced_{LoRA}$	1	16	2e-5	0.4	400
	$Balanced_{\text{PPO-RoBERTa}}$	1	4	2e-5	0.5	400
	$Balanced_{\text{PPO-API}}$	1	4	2e-5	0.5	400
BERTweet	$Unbalanced_{\text{Jigsaw}}$	1	16	4e-5	0.3	400
	$Balanced_{\text{Zero-shot}}$	1	256	3e-5	0.5	400
	$Balanced_{\text{Back-translation}}$	1	64	2e-5	0.5	400
	$Balanced_{LoRA}$	1	8	2e-5	0.5	400
	$Balanced_{\text{PPO-RoBERTa}}$	1	4	2e-5	0.5	400
	$Balanced_{\text{PPO-API}}$	1	4	2e-5	0.5	400

explored a distinct set of hyperparameters randomly to identify the configuration yielding the most accurate results. The optimal hyperparameters for each model developed via different datasets are detailed via Table 40 and Table 38.

Appendix B

In this section, we describe the experimental setup for the classifiers developed in this research. We utilized the training and validation sets from D_{kaggle} for all classifier

Table 38: Experimental Configurations for Transformer-based Classifiers: ToxiGen-based Datasets

Model	Dataset	Epoch	Batch size	Learning rate	Dropout rate	Max Length
BERT	$Balanced_{\text{Zero-shot}}$	2	128	2e-5	0.5	400
	$Balanced_{\text{Back-translation}}$	2	64	2e-5	0.5	400
	$Balanced_{LoRA}$	1	16	2e-5	0.5	400
	$Balanced_{\text{PPO-RoBERTa}}$	1	2	2e-5	0.4	400
	$Balanced_{\text{PPO-API}}$	1	2	2e-5	0.5	400
RoBERTa	$Balanced_{\text{Zero-shot}}$	2	128	2e-5	0.4	400
	$Balanced_{\text{Back-translation}}$	1	64	2e-5	0.3	400
	$Balanced_{LoRA}$	1	16	2e-5	0.5	400
	$Balanced_{\text{PPO-RoBERTa}}$	1	4	2e-5	0.5	400
	$Balanced_{\text{PPO-API}}$	1	4	2e-5	0.4	400
HateBERT	$Balanced_{\text{Zero-shot}}$	2	256	2e-5	0.4	400
	$Balanced_{\text{Back-translation}}$	2	128	2e-5	0.3	400
	$Balanced_{LoRA}$	1	16	2e-5	0.5	400
	$Balanced_{\text{PPO-RoBERTa}}$	1	2	2e-5	0.5	400
	$Balanced_{\text{PPO-API}}$	1	2	2e-5	0.5	400
BERTweet	$Unbalanced_{\text{Zero-shot}}$	2	128	2e-5	0.5	400
	$Balanced_{\text{Back-translation}}$	2	64	2e-5	0.5	400
	$Balanced_{LoRA}$	1	64	2e-5	0.5	400
	$Balanced_{\text{PPO-RoBERTa}}$	1	2	2e-5	0.5	400
	$Balanced_{\text{PPO-API}}$	1	2	2e-5	0.5	400

development. To ensure consistency across the classifiers, we applied a random-state approach for the train-test split, which resulted in identical training, validation, and testing sets for each classifier. This uniformity ensured that all eight classifiers were evaluated under the same conditions.

Further details on the development process and parameter settings for the various classifiers are provided in Chapter 7 and Chapter 7.

CNN-based: CNN, CNN-fastText

We began with data cleaning and preprocessing, including the removal of URLs, punctuation, digits, and stop-words, and normalization. The text was tokenized,

Table 39: Experimental Configurations for CNN-based Classifiers

Model	Hidden Layers	Epochs	Batch Size	Learning Rate	Dropout Rate	Filters	Max Features	Max Length
CNN_{α}	4	15	512	5.00E-05	0.4	300	100000	400
CNN_{ψ}	2	23	512	5.00E-05	0.5	300	100000	400
$fastText_{\alpha}$	4	5	256	0.0002	0.4	300	100000	400
$fastText_{\psi}$	2	14	256	5.00E-7	0.5	128	100000	400

Table 40: Experimental Configurations for Transformer-based Classifiers

Model	Model Name	Epochs	Batch Size	Learning Rate	Weight Decay	Max Length
$BERT_{\alpha}$	$bert-base-uncased$	1	8	2E-5	0.01	300
$BERT_{\psi}$	$google_bert_uncased_L-2_H-128\_A-2$	3	32	3E-5	0.01	300
$RoBERTa_{\alpha}$	$roberta-base$	1	4	2E-5	0.01	300
$DistilRoBERTa_{\psi}$	$distilroberta-base$	2	16	3E-5	0.01	300

truncated or padded to a fixed length, and converted into word vectors.

The model was tested with different configurations. For accuracy, the optimal variant used four 1D convolutional layers, while the high-throughput variant achieved better performance with two 1D layers. We applied pooling and global max pooling to reduce dimensionality, used ReLU activations in hidden layers, and a Sigmoid activation in the output layer. Binary cross-entropy loss and the Adam optimizer were employed for training.

To accelerate training, we incorporated pre-trained fastText embeddings ("crawl-300d-2M") into the CNN model, resulting in the CNN-fastText variant. These embeddings, trained on Common Crawl with CBOW, character n-grams, and negative sampling, improved training efficiency. Hyperparameter settings are detailed in Table 39.

Transformer-based: BERT, RoBERTa, TinyBERT, DistilRoBERTa

We utilized the Hugging Face Transformers library with TensorFlow 2.15.0 to evaluate various models for toxic content detection. Our assessment included several BERT-based models: bert-base-uncased, bert-large-uncased, ALBERT (albert-base-v2), RoBERTa (roberta-base), GroNLP's hateBERT, and vinai's BERTweet (vinai/bertweet-base), along with their distilled variants and 24 BERT miniature models.

For preprocessing, we employed model-specific tokenizers to convert raw text into tokens compatible with each model. This ensured that the data was appropriately formatted for effective processing.

Our testing revealed that bert-base-uncased delivered the best balance of accuracy and acceptable throughput among the BERT variants. Among the BERT miniatures, BERT-Tiny achieved the highest throughput. For RoBERTa models, roberta-base provided the most accurate results, while DistilRoBERTa demonstrated the highest throughput.

We thoroughly explored various hyperparameters to optimize both accuracy and throughput for each model. The detailed hyperparameter settings and results are provided in Table 40.

This comprehensive evaluation enabled us to select the models that offer the best performance for both accuracy and efficiency in toxic content detection.