Silencing Empowerment, Allowing Bigotry: Auditing the Moderation of Hate Speech on Twitch

Prarabdh Shukla[*1], **Wei Yin Chong**[*2], **Yash Patel**[*1], **Brennan Schaffner**[2],
Danish Pruthi[1], **Arjun Bhagoji**[2,3]
[1]Indian Institute of Science, [2]University of Chicago, [3]Indian Institute of Technology, Bombay
Correspondence: arjunp@iitb.ac.in

Abstract

***Warning**: This paper contains content that may be offensive or upsetting.*

To meet the demands of content moderation, online platforms have resorted to automated systems. Newer forms of real-time engagement (*e.g.*, users commenting on live streams) on platforms like Twitch exert additional pressures on the latency expected of such moderation systems. Despite their prevalence, relatively little is known about the effectiveness of these systems. In this paper, we conduct an audit of Twitch's automated moderation tool (`AutoMod`) to investigate its effectiveness in flagging hateful content. For our audit, we create streaming accounts to act as siloed test beds, and interface with the live chat using Twitch's APIs to send over 107,000 comments collated from 4 datasets. We measure `AutoMod`'s accuracy in flagging blatantly hateful content containing misogyny, racism, ableism and homophobia. Our experiments reveal that a large fraction of hateful messages, up to 94% on some datasets, *bypass moderation*. Contextual addition of slurs to these messages results in 100% removal, revealing `AutoMod`'s reliance on slurs as a moderation signal. We also find that contrary to Twitch's community guidelines, `AutoMod` blocks up to 89.5% of benign examples that use sensitive words in pedagogical or empowering contexts. Overall, our audit points to large gaps in `AutoMod`'s capabilities and underscores the importance for such systems to understand context effectively[1].

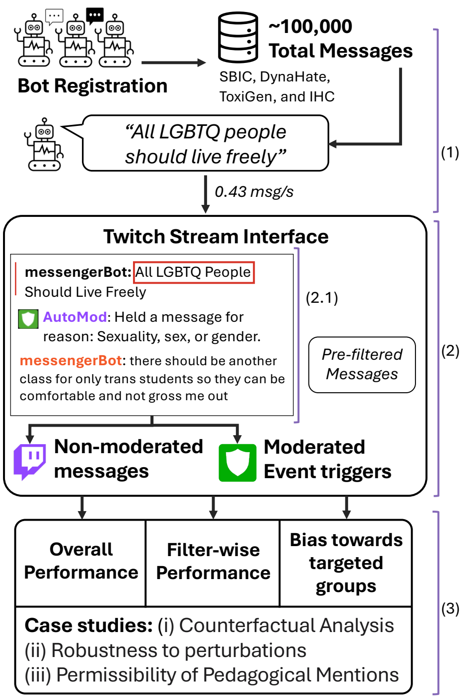


Figure 1: **The audit pipeline**: (**1**) Setting up bots to interact with `AutoMod` and data collation, (**2**) Recording moderation decisions at scale, (**3**) Analysis of `AutoMod` moderation decisions. (**2.1**) shows actual instances of moderated and non-moderated messages from our experiments in the Twitch interface. The moderation decisions clearly show the limitations of `AutoMod`.

1 Introduction

For any online platform to exist without being overrun by hateful, pornographic, abusive, misogynistic and violent content, it must moderate what its users post (Gillespie, 2018). Barring certain regions (Bundesamt für Justiz, 2022), there are few or no legal regulations dictating what is acceptable content on platforms (Schaffner et al., 2024). Moreover, in the Global North, platforms often enjoy legal protections from liability for hosting user-generated content (47 U.S.C. § 230, 1996; EU, 2000). This favorable regulatory framework bestows upon platforms the discretion to moderate content as they wish. The discourse stands divided on the benefits of such freedom, with some crediting it for aiding the growth of online platforms (Kosseff, 2019), while others critique it for enabling the proliferation of harmful content (Wakabayashi, 2019). Platforms codify the behavior ex-

[*] Equal contribution.
[1]Our code and data can be found at `https://github.com/weiyinc11/HateSpeechModerationTwitch`

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 22771–22797
July 27 - August 1, 2025

pected of their users through terms of service and community guidelines. While policies and guidelines may considerably vary across platforms, most promise their users a safe space, free from online harm. For instance, the community guidelines of Twitch, a video streaming platform, state:

> *"Twitch does not permit behavior that is motivated by hatred, prejudice or intolerance, including behavior that promotes or encourages discrimination, denigration, harassment, or violence based on the following protected characteristics: race, ethnicity, color,... We also provide certain protections for age."*

The rhetoric notwithstanding, the pressures exerted on moderation systems have never been higher. Platforms have increasingly begun to integrate automated, usually machine learning-based, systems in their moderation pipelines (Gorwa et al., 2020) due to the scale and velocity of content posted as well as the demanding latency expectations.[2] The practice of content moderation is further challenged by balancing the competing objectives of blocking broadly undesirable content and upholding users' freedom of expression. In response to these pressures, several platforms release aggregate data about the state of harmful content and corresponding actions taken (Twitch, 2024b; X, 2024; Meta, 2024). Despite such transparency efforts, little is known about the underlying algorithms used for moderation and the biases they may introduce.

In this work, we conduct an audit of Twitch's content moderation system. Twitch is a digital platform designed primarily for live streaming content which is created in channels or "streams" that visitors can watch and interact with through a text-based live chat. We focus on *hate speech* content as it is socially important and easier to define than other harmful content such as misinformation (Schaffner et al., 2024). We identify Twitch as a fertile platform for auditing due to three key advantages it offers: (i) Twitch is widely used (TwitchTracker, 2025); (ii) streams on Twitch can be "siloed," allowing us to set up controlled experiments where only the research team can view the content to be tested (iii) the platform provides streamers with a suite of machine learning-based moderation tools collectively called `AutoMod` (Twitch, 2024a), which provides streamers control over different categories of harmful content. Within each category, it offers options to block harmful content targeted at different identity groups, with knobs to vary the extent of moderation (§3.1). Through this audit, we seek to answer the following *key research questions* (§4): (i) How effective is `AutoMod` at flagging commentary rife with hate?; (ii) How specific and effective are individual filters at blocking hate of different kinds and intents?; and (iii) Are moderation rates consistent across different target groups, or are certain groups disproportionately affected? We use case-studies (§5) to nuance these by asking (i) how reliant are the moderation filters on explicit slurs; (ii) how do they respond to sensitive but pedagogical content about marginalized groups?; and (ii) how far can a filter-aware malicious actor bring down their performance? To answer such questions, we develop a framework that allows us to stress test `AutoMod` at scale by launching chatbots in a siloed sandbox. For our study, we use four datasets – a real world comment dataset (SBIC), a real world implicit hate dataset (IHC), a synthetic dataset on implicit hate (ToxiGen) and a synthetic dataset designed to fool hate classifiers (DynaHate). We discuss the hate speech datasets used in §3.2 and our experimental setup in §4.1.

Our audit reveals that Twitch's automated moderation is far from adequate (§4.2), flagging only 22% of hateful content even at its most stringent setting. We observe that hateful samples concerning *Race, Ethnicity and Religion* are least flagged, with only 12.3% of hateful examples being caught. Further, on some datasets, we find that an overwhelming 98% of hate targeted at *Mentally Disabled folks* escapes moderation. On the two implicit hate datasets we use, `AutoMod` is only able to flag 6.8% of hateful examples, implying that the filters rely heavily on slur-based moderation and miss out on implicit hate. Worryingly, empowering or positive phrases about communities (*e.g.* Figure 1) are flagged, and we find `AutoMod` to be quite brittle to semantic-preserving perturbations (See §5).

While it is undeniable that Twitch provides powerful, customizable tools for moderation to its users, our audit serves as an important reminder that these tools must be comprehensively tested and their limitations made clear. Third-party audits like ours can inform the discourse on content moderation

[2] From April 2022 to April 2023, social media platforms saw nearly 150 million new users (4.7/second) (Nyst, 2023).

by grounding the discussion with quantitative evidence regarding the challenge of moderating in a holistic fashion. We hope our methodology inspires and is generalized to audit automated content moderation and other decision-making systems across platforms and data modalities.

2 Related Work

Content Moderation Given its importance, content moderation has been studied extensively (Keller et al., 2020). Prior work has systematically analyzed and critiqued the moderation policies from various online platforms, either focusing on a single platform (Chandrasekharan et al., 2018; Fiesler et al., 2018; Keegan and Fiesler, 2017) or on a industry-wide analysis across many platforms (Schaffner et al., 2024). Other work has focused on specific aspects of moderation, such as user reactions to moderation (Cai et al., 2024; Ribeiro et al., 2023), the effects moderation has on community behavior (Chancellor et al., 2016; Chandrasekharan et al., 2017; Chang and Danescu-Niculescu-Mizil, 2019), and the disproportionate negative effects of moderation on blind users (Lyu et al., 2024). Prior work has also proposed using various AI models to assist in automated content moderation (Kumar et al., 2024; Franco et al., 2023a; Kolla et al., 2024a; Gray and Suzor, 2020). For an in-depth discussion of current content moderation efforts, we refer the reader to (Arora et al., 2023b). Our work complements existing work by auditing real-world automated content moderation algorithms.

Auditing Algorithms Auditing, as defined by (Gaddis, 2018), is a methodology used to deploy randomized controlled experiments in a field setting. When audits target algorithms and computer systems—termed "algorithm audits" (Sandvig et al., 2014; Metaxa et al., 2021b)—the outputs of a system are analyzed when making minor changes to the input, which can lead to insights about the system as a whole. Most often, algorithm audits investigate underlying biases and discriminatory behavior of a system (Edelman and Luca, 2014; Speicher et al., 2018; Chen et al., 2018a; Metaxa et al., 2021a). Other studies have audited whether platforms enforce their policies effectively and fairly, targeting, for example, the political advertising policies on Facebook and Google (Pochat et al., 2022; Matias et al., 2021). Algorithm audits span a wide variety of domains examining several platforms, including housing (rental platforms such as AirBnb) (Edelman and Luca, 2014), ride sharing (Uber) (Chen et al., 2015), healthcare (Obermeyer et al., 2019), employment and hiring (Chen et al., 2018b; Speicher et al., 2018), advertisements (Speicher et al., 2018), and product pricing (Mikians et al., 2012). In a typical algorithm audit such as ours, auditors work with only black-box access to the system, and need to draw conclusions with just that level of access (Cen and Alur, 2024). A concurrently published study (Hartmann et al., 2025) evaluates various moderation APIs (such as OpenAI's moderation API (Markov et al., 2023)) that have recently emerged. The choice of datasets and methodology in this study is similar to ours, further validating our approach. (Hartmann et al., 2025) note that most of the evaluated APIs undermoderate implicit hate and over-moderate pedagogical/empowering content, reclaimed slurs, and counter-speech. They also highlight that moderation APIs often rely on group identity keywords such as "black" when making moderation decisions. We make almost identical observations for `AutoMod` (See §4.2, §5) and together, these results point to the overall gap that exists between the capabilities of SoTA language technology and that used for commercial moderation (See Figure 2).

Live Streaming on Twitch With Twitch's rise in popularity, it has been the subject of several recent studies, including as an emergent political space by modeling the roles of different actor groups (Ruiz Bravo and Roshan, 2022). Another work studies volunteer moderators on Twitch by analyzing their recruitment, motivation and roles in comparison to other online platforms (Seering and Kairam, 2022). Recent work has revealed that waves of attacks (termed "hate raids" by popular media) which were experienced across Twitch in 2021 were targeted on the basis of creator demographics (Han et al., 2023). To the best of our knowledge, we are the first to study automated content moderation on Twitch.

3 Methodology

In this section, we first describe our survey of online platforms. We then describe the datasets we use for the audit, followed by a mathematical description of the audit.

3.1 Exploring Platforms

We set two requirements for choosing a platform to audit: (i) *Harm reduction*: the content posted as a part of the audit should only be visible to a controlled group (*i.e.*, “siloed"); and (ii) *Advanced moderation tools*: a suite of configurable moderation tools (preferably leveraging native machine learning models). The first requirement stems from the ethical requirement to minimize harm to unsuspecting users and not contribute to the existing deluge of hate speech (Twitch, 2024b). ML systems tend to have biases and often have characterizable faults, which makes it important to audit a platform that uses this technology, hence the second requirement. Moreover, while some platforms may employ less advanced moderation systems in the form of blocklists, auditing such systems would only test the comprehensiveness of the lists, not how well the provided tools work. We surveyed the 43 largest platforms that host user-generated content (Schaffner et al., 2024) to check if any meet our requirements, finding two suitable platforms—Reddit and Twitch. We also separately considered Discord as a candidate platform, but we ultimately chose Twitch for this audit due to its moderation system being particularly configurable and also revealing moderation justifications. We compare these three platforms in light of our requirements (Table 3 in Appendix A).

What moderation tools does Twitch offer? There are three main kinds of interactions on Twitch: streamer-to-viewer, viewer-to-streamer, and viewer-to-viewer, with content being moderated across these interactions. The moderation at the streamer-to-viewer level happens via machine learning-based tools that check the audiovisual stream for content that violates Twitch's platform-wide policies. Auditing moderation of audiovisual content is beyond the scope of this paper but presents an interesting direction for future work (§6). In this audit, we focus on moderation of the viewer-to-streamer and viewer-to-viewer interactions, which happen via the live chat interface.

Streamers, as well as designated moderators, can configure Twitch's `AutoMod` tool to handle the potentially high volume of chat content automatically with filters for various types of unwanted content. `AutoMod` uses machine learning (Twitch, 2024a) to detect the unwanted content and surface it in the Moderator view (Figure 5). In addition, streamers can also populate a blocklist of specific words or phrases to restrict from the chat. `AutoMod`'s customisability to vary its detection in levels of strength and content categories makes it an interesting testbed for auditing. For instance, detection sensitivity for *Profanity* and *Discriminatory Content* can be independently set on 5-point scales (See §4.1 for details on how we configured `AutoMod` for our audit study.) Twitch also allows viewers themselves (independent of the streamer) to toggle Chat Filters (Figure 7) to block certain content categories, but these filters are not as granular as those offered to moderators. A Smart Detection option is also in beta, which allows moderation rules to be learned from moderation actions. Since this requires manual moderation, it is beyond the scope of this paper.

3.2 Scope of harmful content: Hate Speech

Amongst the categories of problematic content that Twitch moderates, the majority of moderated content falls into the Sexual Conduct, Harassment, and Hateful Conduct categories (as per Twitch's 2024 Transparency Report (Twitch, 2024b)). We focus our audit on hateful speech, which more specifically falls under the *Discrimination & Slurs* category of Twitch's moderation (Table 4). Speech in this category is often nuanced in its intent and use of language, making for an interesting study. We use four datasets for our experiments: DynaHate (Vidgen et al., 2021b), Social Bias Inference Corpus (SBIC) (Sap et al., 2020), ToxiGen (Hartvigsen et al., 2022) and the Implicit Hate Corpus (IHC) (ElSherief et al., 2021), of which DynaHate and ToxiGen are synthetically generated. Of these, the first three datasets come with annotations specifying the target groups, allowing us to stratify our analysis (§4.2).[3] For SBIC, each example comes with an offensiveness score ranging from 0 to 1, where 0 represents the least offensive rating and 1 represents the most offensive rating. We use a threshold of 1 on the offensiveness score to obtain ground truth labels unless specified otherwise. A more detailed, dataset-wise description is provided in Appendix B.

3.3 Audit Design

Notation and Terminology: We define a black box content moderation system (such as `AutoMod`) as a two-tuple $\mathcal{S} = (\mathcal{F}, \mathcal{C})$. Here, $\mathcal{F} = \{\mathcal{F}_1, \dots, \mathcal{F}_k\}$

[3] We did not find enough examples in IHC for each category, and therefore did not use it for the stratified analysis.

is a set of k black-box moderation functions (filters) and $\mathcal{C} = \{c_1, \ldots, c_k\}$ is the corresponding set of k abstract criteria that map on to real-world concepts such as disability or misogyny, usually derived from the policy that $\mathcal{S}$ aims to enforce. Each $\mathcal{F}_i : T \to \{0, 1\}$ is a function that maps a text $t \sim T$ to either label 0 (benign) or 1 (violation) based on criterion c_i. Here T represents the distribution of all possible textual inputs to the moderation system. The moderation decision for an input t is determined by the *active moderation function* $\mathcal{F}_A$ which corresponds to the set of active filters $\mathcal{C}_A$. $\mathcal{C}_A$ is the set of criteria according to which content needs to be moderated. In AutoMod, each filter $\mathcal{F}_i$ is further parameterized by α, the *filtering level*—which is a discrete measure that modulates the strictness of enforcement.

Filter Choices: In automated content moderation, the elements of the set $\mathcal{C}$ frame the moderation policy of the platform as a whole, while $\mathcal{C}_A$ represents the decisions made by the current streamer or moderator. When we say a particular set $\mathcal{C}_A$ of filters is "turned on," we are referring to the active moderation function $\mathcal{F}_A$ being $\mathbb{1}(\bigcup_{c_i \in \mathcal{C}_A} \mathcal{F}_i = 1)$ which returns 1 when any of the filters returns 1. In this study, we focus on auditing moderation functions of a subset $\tilde{\mathcal{C}} = \{\text{Disability, SSG, Misogyny, RER}\}$[4], corresponding to the broad category of *Discrimination and Slurs* (§3.2). Our analysis in §4.2 proceeds by varying $\mathcal{C}_A \subseteq \tilde{\mathcal{C}}$ (and correspondingly $\mathcal{F}_A$) with respect to different subsets of our base dataset $\mathcal{D}$.

In our experiments we construct $\mathcal{D} = \left(\bigcup_{c_i \in \tilde{\mathcal{C}}} D_{c_i}\right) \bigcup D_{\text{benign}}$, where each text sample in D_{c_i} has *at least one ground-truth label mapping to* c_i. D_{benign} are text samples that do not correspond to any category in $\mathcal{C}$, the set of criteria Twitch allows filtering for. Given the ground-truth labels, we use standard metrics such as precision and recall to measure the effectiveness of the filters at detecting hate speech for the overall dataset, for different categories, and aimed at different target groups within each category.

Filter- and Policy-Informed Text Generation: The base datasets we use contain generic examples of speech, hateful or not, which are not tailored to investigate the robustness of AutoMod's moderation functions and the alignment between Twitch's stated policies and AutoMod operation. In §5, we further test $\mathcal{F}$ with bespoke constructed samples for implicit hate detection, context-awareness and robustness to semantic-preserving inputs.

4 Experiments & Results

In §4.1 we describe the experimental setup that allows us to record moderation decisions from Twitch at scale. We then discuss AutoMod's performance in different contexts in §4.2.

4.1 Setup

Here, we describe the setup and components of our experimental pipeline (Figure 1) in brief (details in Appendix C). For simplicity, unless otherwise indicated, the level for each filter was set to $\alpha = 4$, which is 'Maximum Filtering'. Twitch provides functionality for registering and developing chatbots, typically used by streamers to facilitate stream engagement, such as viewer rewards or stream donations. To send messages programmatically and at scale, we created chatbots that we registered with Twitch's Developer Console using a combination of online burner phone numbers, personal phone numbers, and alias email accounts. Running one instance of the experimental pipeline requires three authenticated bots: a messenger bot, a receiver bot, and a Pubsub bot.

We use the messenger bot to send messages to the chat stream while adhering to Twitch's Chat Rate Limits (Twitch, 2025) to prevent message duplication or omission. We then use a receiver bot to mimic a user viewing the chat stream—and record all messages that were *not* moderated. To observe the moderated messages, we subscribe the third bot to Twitch's Pubsub events, which provide information about the problematic fragments that leads to moderation as well as internal categorical labels such as `Ableism`, `Misogyny`, `Racism`, and `Homophobia` (Figure 6). While these categories are not detailed in the AutoMod documentation, we infer an injective relationship between the content categories in the AutoMod documentation and internal categories from the API usage. With these labels, we are able to investigate AutoMod's stated reasons for moderation and conduct further category-specific analysis in §4.2. In total, we send and record the moderation (in)activity for around $300,000$ messages from December 2024 to January 2025 for our experiments.

[4] SSG stands for *Sex, Sexuality and Gender* and RER stands for *Race, Ethnicity and Religion*.

Challenges Faced: Despite adhering to the rate limit, we speculate that Twitch's safety measures against fraudulent activity (Twitch, 2024b) led to the repeated banning of our chatbots, which forced us to create 30 different developer accounts. Additionally, we observe a *third category of messages* that were not detected by either the receiver or the pubsub bot, indicating a third undocumented destination for messages on Twitch. We suspect that these messages—given that most of them contained specific hateful slurs—are caught at the Service Level in Twitch's moderation pyramid (See Figure 4a) and therefore filtered out *prior to* `AutoMod` which occurs at the Channel Level. We refer to such messages as *pre-filtered* from here on and discuss their effects later in this section.

4.2 Results

In this section, we first present analysis on the overall effectiveness of `AutoMod`. We then discuss filter-wise performance and filter-specificity, followed by target group specific results.

`AutoMod`'s Overall performance: We pass hateful content to Twitch as described in §4.1 and obtain, corresponding to each example, a binary label $Y \in \{0, 1\}$ that indicates if the example was moderated ($Y = 1$) or not ($Y = 0$). Here, $\mathcal{C}_A = \tilde{\mathcal{C}}$, implying all categories under *Discrimination and Slurs* are considered. We then compare these to the ground-truth labels and measure the accuracy, precision, recall or True Positive Rate (TPR), and F1 score. As described in §1, a content moderation system aims to strike a balance between the competing objectives of i) blocking hateful content while ii) safeguarding freedom of speech. $F1_{\text{TPR, TNR}}$ is the harmonic mean between the TPR and the True Negative Rate (TNR). It measures the simultaneous optimization of both objectives. Recall captures performance on objective i). `AutoMod` achieves only 22% recall overall which is quite low when compared to open source classifiers trained on similar data (Sap et al., 2020) and also when compared to zero-shot performance of SoTA language models (see Figure 2). `AutoMod` struggles on even the most evidently hateful real-world data, flagging just 19% of content that SBIC annotators unanimously labeled as hateful. On the two implicit hate datasets, ToxiGen and IHC, we observe a much lower recall (6 and 7% respectively) than the others which indicates that `AutoMod` is poor at recognizing implicit hate and lacks understanding of context (we further validate this in §5). We also evaluate Twitch's Chat Filter (described in §3.1) and our findings (see Appendix D.1) reveal that it behaves identically to `AutoMod` at $\alpha = 4$. Thus, we focus on `AutoMod` for the rest of the paper.

Offensiveness thresholds: For SBIC, we used different thresholds on the offensiveness score to obtain ground truth labels. A higher threshold reflects stricter human agreement for categorizing an example as hateful. As we relax the threshold on the offensiveness score, accounting for varied opinions and more nuanced instances of hate, the recall drops further (Figure 10). Further results at different thresholds are in Appendix D.4.

FN/FP analysis: We speculate that the high FNR ($1-$ Recall) is primarily due to implicit hate. To validate this, we use an LLM to identify swear words in the false negatives (see Appendix D.3) and find that 89.8% of False Negatives are devoid of profanity (*i.e.*, are implicitly hateful). We also note that `AutoMod` performs well with the negative class, as is evidenced by the high TNR. Using a similar analysis on DynaHate — which has a relatively higher FPR — we find that nearly 73% of false positives contain profanity. This analysis hints at `AutoMod`'s heavy reliance on using swear words as a signal for hate (further details in §5).

Conversation level context awareness: While the analysis of false positives and false negatives hint at very limited context-awareness at the message level, we conducted another experiment to ascertain if `AutoMod` possesses any context-awareness at the conversation level (*i.e.*, across different messages passed to the chat). For this, we selected 100 hateful and 100 benign examples from SBIC. We first passed the 100 hate examples alone and observed that 35 were blocked. Next, we interleaved the same 100 hateful examples with the 100 benign examples and observed that the same 35 messages were blocked again, indicating that `AutoMod` does not possess conversation-level context awareness. In addition, we also tested our framework with different permutations of examples to ascertain the reproducibility of our results. We observed that changing the order of input messages does not change the moderation decision for any example.

Filter specificity and effectiveness: We manually categorize hateful examples according to the four filters we analyze: Disability, Misogyny, RER and SSG, thus constructing 4 subsets $\mathcal{D}_{c_i} \subset \mathcal{D}$ (See Appendix F). Quality control analysis on our

Dataset	Accuracy	Precision	Recall	TNR	$\mathbf{F1}_{\mathrm{P,R}}$	$\mathbf{F1}_{\mathrm{TPR,TNR}}$
SBIC (Sap et al., 2020)	0.73	0.42	0.19	0.91	0.26	0.31
DynaHate* (Vidgen et al., 2021a)	0.49	0.54	0.41	0.59	0.47	0.48
ToxiGen* (Hartvigsen et al., 2022)	0.53	0.86	0.07	0.98	0.13	0.13
IHC (ElSherief et al., 2021)	0.52	0.70	0.06	0.97	0.12	0.11
Overall	0.55	0.56	0.22	0.84	0.32	0.35

Table 1: **`AutoMod`'s aggregate performance**. `AutoMod` is reasonably accurate with benign examples, but struggles with the hateful ones. Recall on ToxiGen and IHC datasets is lower than the other datasets implying `AutoMod` is weak at detecting implicit hate. (*) is used to indicate synthetic data. Green indicates the best value and Red indicates the worst value of a metric.

Dataset	**SBIC**		**DynaHate**		**ToxiGen**		**Overall**	
Filter	$\mathcal{R}$ (%)	*Pf* (%)	$\mathcal{R}$ (%)	*Pf* (%)	$\mathcal{R}$ (%)	*Pf* (%)	$\mathcal{R}$ (%)	*Pf* (%)
Disability	22.4	2.0	44.2	4.4	2.9	13.8	10.6	6.1
Misogyny	27.3	0.8	25.8	1.4	3.9	4.6	19.0	1.5
RER	17.3	36.1	20.5	18.8	6.6	21.5	12.3	22.0
SSG	32.0	64.1	25.3	55.3	5.7	44.3	17.5	54.8

Table 2: **Filter-wise recall across datasets**. We report recall ($\mathcal{R}$) and the percentage of examples that were pre-filtered (*Pf*). The *Misogyny* filter is the most effective. The SSG filter relies on pre-filtering more heavily than other filters.

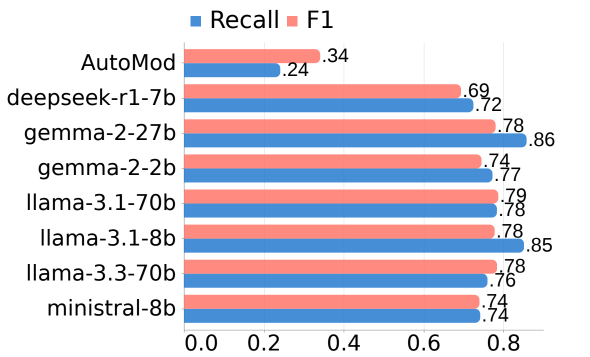


Figure 2: **Comparison of `AutoMod` to State-of-The-Art (SoTA) language models**. Language models were prompted with a zero-shot instruction containing Twitch's community guidelines. For experimental setup, see Appendix D.6.

data subsets is provided in Appendix D.8. The primary objective of this analysis is to assess how well a filter $\mathcal{F}_i$ models the criterion c_i associated with it. Note that we receive `AutoMod`'s internal categories for any moderated message, allowing us to determine which filter led to moderation even when multiple are on.

Filter-wise recall: First, for each subset $\mathcal{D}_{c_i}$, we set $\mathcal{C}_A = \{c_i\}$ to measure the filter-wise recall (Table 2). We observe that the *Misogyny* and *Sexuality and Gender (SSG)* filters have the highest recall. It is also important, however, to consider the pre-filtering rate. The Disability and Misogyny subsets have a very low pre-filtering rate on all datasets. This implies that most examples from these subsets went through the filter and the recall is a good representation of the filter-only performance. For the SSG filter, we observe the opposite — the overall pre-filtering rate is 54.8%, and high across datasets. This implies that for detecting SSG related hate, `AutoMod` relies on pre-filtering more heavily than it does for other kinds of hate speech. A similar inference cannot be made for the RER filter, however, because the pre-filtering rate for the RER filter on DynaHate is nearly half of what it is for SBIC. It is unclear whether the lack of diversity in the data itself causes higher pre-filtering in case of SBIC. We speculate that SBIC has more instances of racial slurs that appear in the pre-filtering blocklist than ToxiGen which mostly consists of implicit hate.

Filter precision: We repeat our experiment with $\mathcal{C}_A = \tilde{\mathcal{C}}$ for each $\mathcal{D}_{c_i}$. This allows us to measure the filter precision $P_{\mathcal{F}_i}$, the specificity of a filter. A naive filter that outputs 1 for every hateful text, will have a high recall yet is undesirable because it moderates examples beyond its criterion — which may not be suitable when pedagogical or empow-

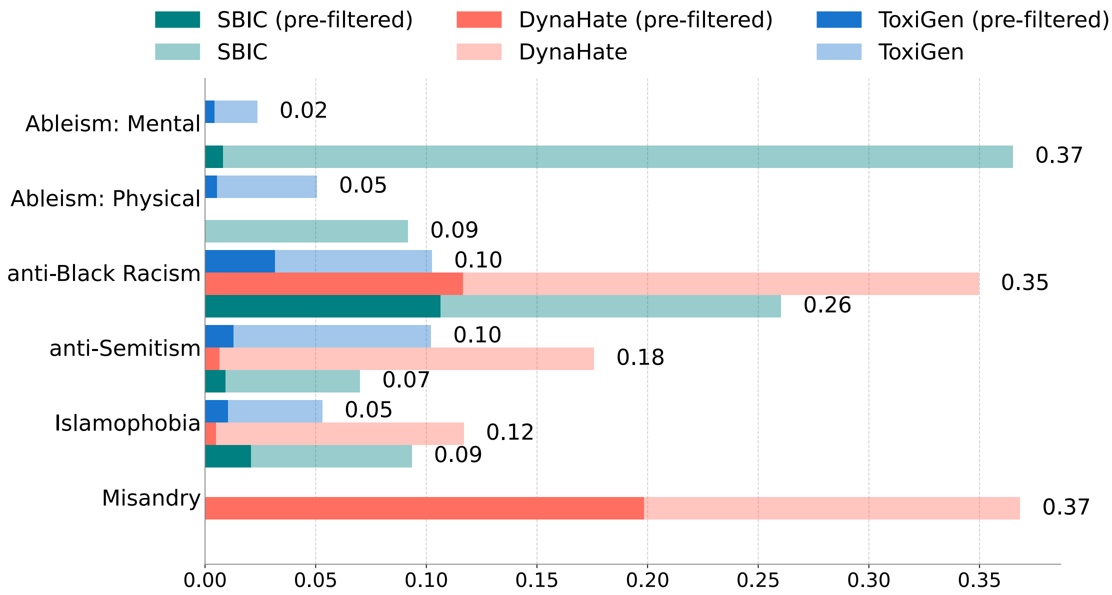


Figure 3: **`AutoMod`'s recall for subsets of hate examples directed towards various communities (All filters under the *Discrimination & Slurs* category turned on)**. The opaque portion of each bar indicates the fraction of messages that are pre-filtered.

ering utterances use n-grams that are frequently seen in hate speech. Measuring the specificity of filters is therefore as important as recall. The filter precision is shown in Figure 8. We observe that the RER filter is the most precise, while Misogyny is the least.

Effects of moderation across target groups: In this analysis we study how `AutoMod` performs in blocking hate related to specific target groups. To obtain hate data specific to different target groups, we use the target group labels in our dataset (mapping details in Appendix F). While each subset may correspond to a filter (*e.g.* anti-Black Racism $\rightarrow$ RER, Mental Ableism $\rightarrow$ Disability), there may also be instances where an example may be relevant to more than one filter. To address this in measuring target-group performance, we evaluate recall with $\mathcal{C}_A = \tilde{\mathcal{C}}$. By turning all filters on, we ensure that the recall under this setting is a fair measure of how well `AutoMod` blocks hate directed towards each of these communities. The overall community-wise recall is shown in Figure 3. We observe that hate speech directed towards Men, Black Folks and Mentally Disabled Folks is more effectively blocked by `AutoMod` than for the other target groups.

Pre-filtering and Filter Level (α)**:** We analyse the pre-filtered examples, and find the use of blocklists for pre-filtering causes disparate performance across target groups, perhaps due to bias in blocklist construction (see Appendix D.7). We also evaluate `AutoMod` at different filter levels (α) and find that for $\alpha > 0$, there is minimal gain ($+1.1\%$) in performance (Appendix D.5).

5 Case Studies: Implicit Hate, Context and Robustness

This section explores three case studies that examine `AutoMod`'s versatility and robustness.

Counterfactual Analysis: Unlike explicit hate speech, implicit hate is harder for systems to detect due to the usage of stereotypes, insinuations, or coded language — which may evade traditional keyword-based filters. Therefore, detecting such content requires a deeper understanding of context, intent, and the subtle ways in which language can perpetuate harm. For this study, we select 110 false negative samples from the SBIC dataset and manually review them to ensure that they are devoid of any slurs, while still remaining offensive. We then programmatically replace terms corresponding to demographic groups with offensive slurs associated with the same groups (*e.g.* replacing the phrase *"Black People"* with the *n*-word). When these counterfactual examples are passed to `AutoMod` (with $\mathcal{C}_A = \tilde{\mathcal{C}}$), we observe 100% recall. Some examples from our counterfactual set are presented in Table 6. This study highlights `AutoMod`'s heavy bias towards using slurs as an indication of hate and poor understanding of context which allows implicit hate to easily evade detection.

Policy Adherence Evaluation: Twitch's Community Guidelines explicitly recognize the importance of context in content moderation:

> *"At Twitch, we allow certain words or terms, which might otherwise violate our policy, to be used in an empowering way or as terms of endearment when such intent is clear."*

To study AutoMod's adherence to this policy, we manually select 20 non-slur sensitive fragments from SBIC and prompt the GPT-4o model to generate statements that incorporate these fragments in a non-offensive manner including usage of the term in an educational or empowering context. The prompt used for generation and some examples are provided in Appendix E.1. We test these examples (with $\mathcal{C}_A$ set to $\tilde{\mathcal{C}}$) with the filtering level varying between *Some Filtering* ($\alpha = 2$) and *Maximum Filtering* ($\alpha = 4$) for each filter. Under the former, AutoMod flags 89.5% of examples while in the latter setting, it flags 98.5% examples. This behavior is in contrast to the aforementioned policy. While a streamer can manually configure AutoMod to allow these sensitive fragments when a stream is likely to elicit their usage in non-offensive contexts, this could give a free pass to bad actors to perpetuate hate. This study further underscores the need for context-aware automated moderation.

Robustness: We measure AutoMod's robustness to minor, semantic-preserving perturbations of input text, given its seeming reliance on direct slurs. For this study, we prompt GPT-4o to make subtle changes (such as adding spaces or punctuation, altering spelling etc.) to 50 manually-chosen sensitive fragments from the SBIC dataset. We employ 6 perturbation methods (listed in Table 7). For each fragment we also generate one example that uses the fragment as is (Unperturbed). The prompt used for generation and the definitions of the perturbation methods are provided in Appendix E.3. We observe that the recall drops from 100% to 4% on some of our perturbations. This indicates that malicious actors can easily evade AutoMod. However, it is worth noting that AutoMod was able to detect all perturbations (except the reverse one) of certain frequent terms such as the *n*-word. This indicates some degree of robustness for highly sensitive terms.

6 Discussion and Future Work

AutoMod exhibits poor recall and F1-scores across datasets, compared to SoTA language models (Figure 2), and even GPT-2 (Sap et al., 2020). Our analysis of the false negatives and positives (§4.2 and Appendix D.3), along with the case studies (§5) collectively explains how the poor performance of AutoMod arises from a lack of contextual understanding of hate speech. Regardless, Twitch's effort in the development of AutoMod is commendable for the flexibility it provides, with most other platforms lacking such tools. Our audit serves to highlight current gaps in AutoMod's capabilities and we make good faith recommendations for Twitch to address these issues for greater user-safety. There are several directions for future work, the most direct of which is to evaluate Twitch's Smart Detection feature for text, and expand the audit to audiovisual content. Considering other platforms and languages, particularly given the availability of multilingual hate speech data (Arora et al., 2023a) is of critical importance. More nuanced audits for black-box systems may be possible by leveraging techniques such as model reconstruction attacks (Tramèr et al., 2016) for reverse engineering and transfer- and query- based black-box adversarial example generation (Demontis et al., 2019; Bhagoji et al., 2018) In conclusion, we hope our study serves as a blueprint for further audits of decision-making systems operating in complex socio-technical environments.

Limitations

While we attempt to be comprehensive in our evaluation of hate speech moderation on Twitch, several key limitations remain. *First*, we only consider a single platform in our study, and do not compare AutoMod to any other deployed text moderation. As of the beginning of this study, we could not find any other suitable candidates. However, alternatives such as Mistral's Moderation tool (Mistral AI Team, 2024) have since emerged which would make for an interesting follow-up study, particularly in the context of other studies of open-source LLMs for moderation (Kumar et al., 2024). A concurrently published study (Hartmann et al., 2025) carries out an evaluation similar to ours for various moderation APIs on similar datasets. *Second*, we largely investigate both hateful and benign messages in a non-conversational context, *i.e.,* messages are passed one-by-one without any simulation of dialogue. More work is needed to curate datasets of dialogue that contain both hateful and benign content. For curation of such datasets, researchers may consider pursuing a DynaHate-style (Vidgen et al., 2021b) creation protocol, which is now feasible thanks to our framework that allows one to query black box systems like `AutoMod` at scale. In addition, investigating in-group/out-group dynamics may uncover differential treatment and moderation of content based on group affiliations. In the case of `AutoMod`, however, we speculate that our results would have remained unchanged as the policy violation case studies already demonstrate a lack of contextual awareness. *Third*, our study is focused entirely on text in English. Considering other languages and modalities would be a critical direction for future work.

Ethical Considerations Statement

Exposure to Offensive Content: In conducting this research, our team dealt with a significant amount of offensive content. All authors were aware of the nature of the work and consented to view such content. It is important to note that no individuals apart from the authors were exposed to this material given our use of siloed experimental streams. Moreover, the harmful content used in this work was sourced from open-source datasets.

Legal Compliance: To conduct this research, we did post content that violates Twitch's terms of service. However, this action was taken within the legal boundaries established by the *Sandvig v. Barr* case (Gilens, Naomi and Williams, Jamie, 2020), which protects such research activity.

Potential for Adverse Impact: We demonstrated how perturbations to offensive text circumvent Twitch's existing moderation systems, and there is a potential adverse impact wherein Twitch users might exploit these examples. However, these techniques and attacks are already well-documented and have been extensively explored in the literature concerning NLP classifiers. If a hateful actor is determined to spread hate on Twitch, it is unlikely that they are unaware of these rather unsophisticated perturbations.

Impact on Twitch Developers: We also recognize the unfortunate, unlikely possibility that our study inadvertently increased the burden for Twitch's internal moderation developers. Twitch has added measures to protect its users, and we do not wish to undermine these efforts. Instead, we intend to highlight existing failures while advocating for careful audits of deployed systems. We have contacted Twitch with our findings prior to the publication of this work, ensuring that they are aware of their system vulnerabilities and can take appropriate action.

Acknowledgments

We thank anonymous reviewers for the constructive suggestions. YP is thankful for the Reliance Foundation Postgraduate Scholarship for supporting his research. GC is grateful to the Careers in Computer Science Mini-Internship Program at the University of Chicago for supporting her research. DP is additionally grateful for Adobe, Inc. for generously supporting his group's research. We note that the findings, conclusions and opinions presented in the paper are those of the authors and do not necessarily reflect the views of the sponsoring organizations or agencies.

References

47 U.S.C. § 230. 1996. Protection for private blocking and screening of offensive material. Available at: `https://www.law.cornell.edu/uscode/text/47/230`.

Arnav Arora, Maha Jinadoss, Cheshta Arora, Denny George, Haseena Dawood Khan, Kirti Rawat, Seema Mathur, Shivani Yadav, Shehla Rashid Shora, Rie Raut, et al. 2023a. The uli dataset: An exercise in experience led annotation of ogbv. *arXiv preprint arXiv:2311.09086*.

Arnav Arora, Preslav Nakov, Momchil Hardalov, Sheikh Muhammad Sarwar, Vibha Nayak, Yoan Dinkov, Dimitrina Zlatkova, Kyle Dent, Ameya Bhatawdekar, Guillaume Bouchard, and Isabelle Augenstein. 2023b. Detecting harmful content on online platforms: What platforms need vs. where research efforts go. *Preprint*, arXiv:2103.00153.

Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. 2018. Practical black-box attacks on deep neural networks using efficient query mechanisms. In *Proceedings of the European conference on computer vision (ECCV)*, pages 154–169.

Bundesamt für Justiz. 2022. Gesetz zur Verbesserung der Rechtsdurchsetzung in sozialen Netzwerken (Netzwerkdurchsetzungsgesetz – NetzDG).

Jie Cai, Aashka Patel, Azadeh Naderi, and Donghee Yvette Wohn. 2024. Content moderation justice and fairness on social media: Comparisons across different contexts and platforms. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, CHI '24, page 1–9. ACM.

Sarah H. Cen and Rohan Alur. 2024. From transparency to accountability and back: A discussion of access and evidence in ai auditing. *Preprint*, arXiv:2410.04772.

Stevie Chancellor, Jessica Annette Pater, Trustin Clear, Eric Gilbert, and Munmun De Choudhury. 2016. #thyghgapp: Instagram content moderation and lexical variation in pro-eating disorder communities. In *Proceedings of CSCW*, CSCW '16, pages 1201–1213, New York, NY, USA. ACM.

Eshwar Chandrasekharan, Umashanthi Pavalanathan, Anirudh Srinivasan, Adam Glynn, Jacob Eisenstein, and Eric Gilbert. 2017. You can't stay here: The efficacy of reddit's 2015 ban examined through hate speech. *In Proceedings of ACM Human Computer Interaction*, 1(Computer-Supported Cooperative Work and Social Computing):31:1–31:22.

Eshwar Chandrasekharan, Mattia Samory, Shagun Jhaver, Hunter Charvat, Amy Bruckman, Cliff Lampe, Jacob Eisenstein, and Eric Gilbert. 2018. The Internet's Hidden Rules: An Empirical Study of Reddit Norm Violations at Micro, Meso, and Macro Scales. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):1–25.

Jonathan P. Chang and Cristian Danescu-Niculescu-Mizil. 2019. Trajectories of blocked community members: Redemption, recidivism and departure. In *Proceedings of WWW*, pages 184–195.

Le Chen, Ruijun Ma, Anikó Hannák, and Christo Wilson. 2018a. Investigating the impact of gender on rank in resume search engines. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–14, New York, NY, USA. Association for Computing Machinery.

Le Chen, Ruijun Ma, Anikó Hannák, and Christo Wilson. 2018b. Investigating the impact of gender on rank in resume search engines. In *Proceedings of the 2018 chi conference on human factors in computing systems*, pages 1–14.

Le Chen, Alan Mislove, and Christo Wilson. 2015. Peeking beneath the hood of uber. In *Proceedings of the 2015 internet measurement conference*, pages 495–508.

Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. 2019. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *28th USENIX security symposium (USENIX security 19)*, pages 321–338.

Benjamin Edelman and Michael Luca. 2014. Digital discrimination: The case of airbnb.com. *SSRN Electronic Journal*.

Mai ElSherief, Caleb Ziems, David Muchlinski, Vaishnavi Anupindi, Jordyn Seybolt, Munmun De Choudhury, and Diyi Yang. 2021. Latent hatred: A benchmark for understanding implicit hate speech. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 345–363, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

EU. 2000. Articles 12-14, european union e-commerece directive.

Casey Fiesler, Jialun Jiang, Joshua McCann, Kyle Frye, and Jed Brubaker. 2018. Reddit rules! Characterizing an Ecosystem of Governance. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 12.

Mirko Franco, Ombretta Gaggi, and Claudio E. Palazzi. 2023a. Analyzing the use of large language models for content moderation with chatgpt examples. In *Proceedings of the 3rd International Workshop on Open Challenges in Online Social Networks*, OASIS '23, page 1–8, New York, NY, USA. Association for Computing Machinery.

Mirko Franco, Ombretta Gaggi, and Claudio E. Palazzi. 2023b. Integrating Content Moderation Systems with Large Language Models. *ACM Trans. Web*.

S.M. Gaddis. 2018. *Audit Studies: Behind the Scenes with Theory, Method, and Nuance*. Methodos Series. Springer International Publishing.

Gilens, Naomi and Williams, Jamie. 2020. Federal Judge Rules It Is Not a Crime to Violate a Website's Terms of Service. [Online; accessed 19. Jan. 2025].

Tarleton Gillespie. 2018. *Custodians of the Internet: Platforms, Content Moderation, and the Hidden Decisions that Shape Social Media*. Yale University Press.

Robert Gorwa, Reuben Binns, and Christian Katzenbach. 2020. Algorithmic content moderation: Technical and political challenges in the automation of platform governance. *Big Data & Society*, 7(1):2053951719897945.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, and et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Joanne E Gray and Nicolas P Suzor. 2020. Playing with machines: Using machine learning to understand automated copyright enforcement at scale. *Big Data & Society*, 7(1):2053951720919963.

Catherine Han, Joseph Seering, Deepak Kumar, Jeffrey T. Hancock, and Zakir Durumeric. 2023. Hate raids on twitch: Echoes of the past, new modalities, and implications for platform governance. *Preprint*, arXiv:2301.03946.

David Hartmann, Amin Oueslati, Dimitri Staufer, Lena Pohlmann, Simon Munzert, and Hendrik Heuer. 2025. Lost in moderation: How commercial content moderation apis over- and under-moderate group-targeted hate speech and linguistic variations. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI '25, page 1–26. ACM.

Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. 2022. ToxiGen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3309–3326, Dublin, Ireland. Association for Computational Linguistics.

Jacob Foster. 2023. tmi.js. [Online; accessed 20. Jan. 2025].

Brian Keegan and Casey Fiesler. 2017. The evolution and consequences of peer producing wikipedia's rules. In *Proceedings of ICWSM*.

Daphne Keller, Paddy Leerssen, et al. 2020. Facts and where to find them: Empirical research on internet platforms and content moderation. *Social media and democracy: The state of the field and prospects for reform*, pages 220–251.

Mahi Kolla, Siddharth Salunkhe, Eshwar Chandrasekharan, and Koustuv Saha. 2024a. Llm-mod: Can large language models assist content moderation? In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, CHI EA '24, New York, NY, USA. Association for Computing Machinery.

Mahi Kolla, Siddharth Salunkhe, Eshwar Chandrasekharan, and Koustuv Saha. 2024b. LLM-Mod: Can Large Language Models Assist Content Moderation? *CHI EA '24: Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 1–8.

Jeff Kosseff. 2019. *The twenty-six words that created the Internet*. Cornell University Press.

Deepak Kumar, Yousef AbuHashem, and Zakir Durumeric. 2024. Watch your language: Investigating content moderation with large language models. *Preprint*, arXiv:2309.14517.

Yao Lyu, Jie Cai, Anisa Callis, Kelley Cotter, and John M. Carroll. 2024. "i got flagged for supposed bullying, even though it was in response to someone harassing me about my disability.": A study of blind tiktokers' content moderation experiences. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24, page 1–15. ACM.

Todor Markov, Chong Zhang, Sandhini Agarwal, Tyna Eloundou, Teddy Lee, Steven Adler, Angela Jiang, and Lilian Weng. 2023. A holistic approach to undesired content detection in the real world. *Preprint*, arXiv:2208.03274.

J. Nathan Matias, Austin Hounsel, and Nick Feamster. 2021. Software-supported audits of decision-making systems: Testing google and facebook's political advertising policies. *Preprint*, arXiv:2103.00064.

Meta. 2024. Community Standards Enforcement Report. [Online; accessed 19. Jan. 2025].

Danaë Metaxa, Michelle A Gan, Su Goh, Jeff Hancock, and James A Landay. 2021a. An image of society: Gender and racial representation and impact in image search results for occupations. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW1):1–23.

Danaë Metaxa, Joon Sung Park, Ronald E. Robertson, Karrie Karahalios, Christo Wilson, Jeff Hancock, and Christian Sandvig. 2021b. Auditing algorithms: Understanding algorithmic systems from the outside in. *Foundations and Trends® in Human–Computer Interaction*, 14(4):272–344.

Jakub Mikians, László Gyarmati, Vijay Erramilli, and Nikolaos Laoutaris. 2012. Detecting price and search discrimination on the internet. In *Proceedings of the 11th ACM workshop on hot topics in networks*, pages 79–84.

Mistral AI Team. 2024. Mistral Moderation API. [Online; accessed 16 Feb. 2025].

Annabelle Nyst. 2023. 134 Social Media Statistics You Need To Know For 2023 — searchenginejournal.com. `https://www.searchenginejournal.com/social-media-statistics/480507/`. [Accessed 17-01-2025].

Ziad Obermeyer, Brian Powers, Christine Vogeli, and Sendhil Mullainathan. 2019. Dissecting racial bias in an algorithm used to manage the health of populations. *Science*, 366(6464):447–453.

Victor Le Pochat, Laura Edelson, Tom Van Goethem, Wouter Joosen, Damon McCoy, and Tobias Lauinger. 2022. An audit of facebook's political ad policy enforcement. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 607–624, Boston, MA. USENIX Association.

Manoel Horta Ribeiro, Justin Cheng, and Robert West. 2023. Automated content moderation increases adherence to community guidelines. *Preprint*, arXiv:2210.10454.

Nadia Ruiz Bravo and Maryam Roshan. 2022. The political turn of twitch – understanding live chat as an emergent political space.

Christian Sandvig, Kevin Hamilton, Karrie Karahalios, and Cédric Langbort. 2014. Auditing algorithms : Research methods for detecting discrimination on internet platforms.

Maarten Sap, Saadia Gabriel, Lianhui Qin, Dan Jurafsky, Noah A. Smith, and Yejin Choi. 2020. Social bias frames: Reasoning about social and power implications of language. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5477–5490, Online. Association for Computational Linguistics.

Brennan Schaffner, Arjun Nitin Bhagoji, Siyuan Cheng, Jacqueline Mei, Jay L Shen, Grace Wang, Marshini Chetty, Nick Feamster, Genevieve Lakier, and Chenhao Tan. 2024. "community guidelines make this the best party on the internet": An in-depth study of online platforms' content moderation policies. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24, page 1–16. ACM.

Joseph Seering and Sanjay R. Kairam. 2022. Who moderates on twitch and what do they do? quantifying practices in community moderation on twitch. *Proc. ACM Hum.-Comput. Interact.*, 7(GROUP).

Till Speicher, Muhammad Ali, Giridhari Venkatadri, Filipe Nunes Ribeiro, George Arvanitakis, Fabrício Benevenuto, Krishna P. Gummadi, Patrick Loiseau, and Alan Mislove. 2018. Potential for discrimination in online targeted advertising. In *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, pages 5–19. PMLR.

Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction {APIs}. In *25th USENIX security symposium (USENIX Security 16)*, pages 601–618.

Twitch. 2024a. How to Use AutoMod. [Online; accessed 19. Jan. 2025].

Twitch. 2024b. Transparency Reports. [Online; accessed 18. Jan. 2025].

Twitch. 2025. Chat & Chatbots. [Online; accessed 19. Jan. 2025].

TwitchTracker. 2025. Twitch channels, games and global statistics.

Bertie Vidgen, Kayla Chatelon, Scott Hale, Helen Margetts, and Dong Nguyen. 2021a. Learning from the worst: Dynamically generated datasets to improve online hate detection. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4216–4231. Association for Computational Linguistics.

Bertie Vidgen, Tristan Thrush, Zeerak Waseem, and Douwe Kiela. 2021b. Learning from the worst: Dynamically generated datasets to improve online hate detection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1667–1682, Online. Association for Computational Linguistics.

D Wakabayashi. 2019. Legal shield for websites rattles under onslaught of hate speech. *The New York Times, Aug*, 6.

X. 2024. Dsa transparency report - october 2024.

This Appendix provides additional information corresponding to each section of the main paper as follows:

1. **Platform Choice and Twitch Moderation Details** (Appendix A): Provides additional details about Twitch Moderation options referenced in §3.1.
2. **Dataset Details** (Appendix B): Detailed descriptions of the 4 datasets of hate speech content introduced in §3.2.
3. **Experimental Setup Details** (Appendix C): Breakdown of the steps followed to send and receive messages from Twitch, expanding upon §4.1.
4. **Further Results** (Appendix D): Building on results in §4.2 with analysis of false negatives and false positives from `AutoMod`, SBIC data threshold, different `AutoMod` filter levels, prefiltering bias, and quality control analysis of filter-specific datasets.
5. **Ablation Details** (Appendix E): Information on the methods used to construct counterfactuals, policy adherence samples and perturbed examples in §5.
6. **Filter- and Community-Specific Subset Extraction** (Appendix F): Dataset-wise breakdown of the procedure followed to obtain subsets for experiments in §4.2.

A Platform Choice and Twitch Moderation Details

As an extension of the Platform Exploration and Twitch Moderation section (§3.1), we provide more details about the investigation guiding our platform choice, the Twitch interface, and its internal documentation.

A.1 Platform Survey

In Table 3, we provide details about the 3 platforms that we choose from in terms of their suitability for the audit. While both Reddit and Discord offer siloed environments for experimentation, neither provides native machine learning models for offensive text detection.

A.2 Twitch Automated Moderation

In Figure 4a, we show the Twitch Moderation Pyramid (Twitch, 2025). As stated in §3.1, we focus on viewer-to-streamer and streamer-to-streamer which are moderated via `AutoMod` and Chat Filters. In Figure 4b, we show the various levels Twitch offers moderators to customize filtering strength.

Table 4 describes Twitch's Official Content Categories and their differentiating definitions as of 04 July, 2024. The audit's investigation focuses on the effectiveness of `AutoMod`'s overall performance with respect to the Discrimination and Slurs content category. More specifically, the datasets used provide ground truth relating to the subsections of Discrimination and Slurs as seen in Table 4, Disability, SSG, Misogyny, and RER.

The dashboard is customizable allowing the streamer to view in real-time their stream's `AutoMod` queue and to manually approve or deny detected messages as shown in Figure 5. The `AutoMod` queue also displays the associated content category so as to inform the streamer for `AutoMod`'s reasoning behind moderation. Highlighted segments also help the user focus on the problematic fragments of moderated messages.

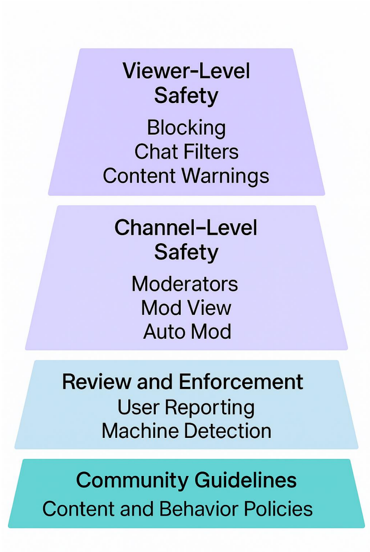
In evaluating the overall performance of `AutoMod` on moderating various categories of hateful content, mere binary evaluations were deemed primary but not comprehensive to effectively determine performance in terms of accuracy and precision. Hence, `AutoMod`'s moderation reasons with respect to Twitch's pre-defined categories of moderation provided more insight. To programmatically access the streamer's `AutoMod` queue, Twitch Pubsub event subscriptions were used to grab API results whenever `AutoMod` logged a message in the queue for moderation. However, we found that `AutoMod`'s internal categories did not match Twitch Documentation and Policies on Content Moderation across the platform. Figure 6 depicts a snapshot of Twitch API result from `automodqueue` event subscription detailing the message and its metadata. This metadata include `topics` and `category` of the moderated message from which we map the categories semantically.

B Dataset Details

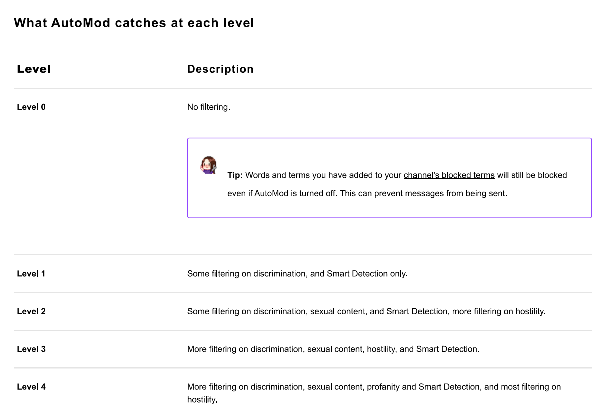
All datasets used are open-source and permitted for research. All of our data is in English.

Platform	Closed environment	Moderation tools
Discord	Private server with permissions for everyone but admin disabled	Image hashing and “ML powered tech" for child sexual abuse material; No native ML models for text, relies on keyword detection
Reddit	Via private subreddits	Needs plugins for ML models and subreddit specific rules for moderation; Well-investigated (Kolla et al., 2024b; Kumar et al., 2024; Franco et al., 2023b)
Twitch	Untagged streams are private	Customisable moderation levels using native ML models; Also has keyword lists and Smart Detection to train model on the actions of human moderators

Table 3: **Comparison of candidate platforms.** We survey various platforms and choose Twitch primarily for its configurability and ML-based moderation tool.



(a) Twitch’s moderation pyramid



What AutoMod catches at each level

Level	Description
Level 0	No filtering. **Tip:** Words and terms you have added to your channel's blocked terms will still be blocked even if AutoMod is turned off. This can prevent messages from being sent.
Level 1	Some filtering on discrimination, and Smart Detection only.
Level 2	Some filtering on discrimination, sexual content, and Smart Detection, more filtering on hostility.
Level 3	More filtering on discrimination, sexual content, hostility, and Smart Detection.
Level 4	More filtering on discrimination, sexual content, profanity and Smart Detection, and most filtering on hostility.

(b) `AutoMod` moderation level (α) distinctions from Level 0 to 4

Figure 4: **Official Twitch documentation of moderation tools:** (a) Different levels at which Twitch moderates content along with their respective methods of moderation (b) Twitch documentation on `AutoMod` moderation levels (referred to as α in this work) from 0 to 4.

B.1 Dynahate

The Dynahate dataset (Vidgen et al., 2021b) contains approximately 41,000 entries, each labeled as either 0 (Not hateful) or 1 (Hateful) and is generated and labelled by trained annotators over four rounds of dynamic data creation. The dataset includes 54% of hateful examples. In our work, we use three columns from the Dynahate dataset: *text*, *label*, and *target*. The ‘target’ column specifies the community towards which the hate is directed, making it particularly useful for dividing the data into filter and community-specific subsets.

B.2 SBIC

The Social Bias Inference Corpus (SBIC) (Sap et al., 2020) is a large-scale dataset with over 150k annotations across 34,000 social media posts, capturing nuanced biases and stereotypes in online language. Its focus on offensive content, implied stereotypes, and target demographics makes it particularly useful for auditing Twitch’s moderation system, especially in areas related to group-based bias and contextual offensiveness. We sample 20k examples from the SBIC training data for our work. Since SBIC provides an average annotator offensiveness score, we set a threshold of 1 (all annotators agreeing on the offensiveness of an example)

Moderation Category	Explanation of Hate Speech Category
Sexual Content	Words or phrases referring to sexual acts and/or anatomy.
Discrimination and Slurs	Includes race, religion, gender-based discrimination. Hate speech falls under this category.
I) Disability	Demonstrating hatred or prejudice based on perceived or actual mental or physical abilities.
II) SSG	Demonstrating hatred or prejudice based on sexual identity, sexual orientation, gender identity, or gender expression.
III)Misogyny	Demonstrating hatred or prejudice against women, including sexual objectification.
IV) RER	Demonstrating hatred or prejudice based on race, ethnicity, or religion.
Hostility	Provocation and bullying, sexual harassment.
Profanity	Expletives, curse words, and vulgarity. This filter especially helps those who wish to keep their community family-friendly.
Smart Detection	Detects unwanted messages (including spam) based on moderation actions taken in your channel.

Table 4: **`AutoMod` Content Categories.** The definition of various categories and sub-categories that `AutoMod` moderates.

to evaluate the overall effectiveness of `AutoMod`, resulting in a hate-to-non-hate ratio of 1:3. For further division into hateful subsets related to specific filters, we use a threshold of 0.5, yielding a total of 8,748 examples. The SBIC dataset contains numerous features, but for our analysis, we utilize the following: *post*, *targetMinority*, *targetCategory*, *offensiveYN* (indicating whether the example is offensive, non-offensive, or ambiguous), and annotator-related columns for aggregating the offensive score.

B.3 ToxiGen

The ToxiGen dataset (Hartvigsen et al., 2022) is a large-scale dataset containing approximately 274k toxic and benign statements about 13 minority groups. This dataset is particularly useful for our work as it consists of synthetically generated implicit hate speech, allowing us to evaluate the effectiveness of filters on implicit hate content. The authors use the GPT-3 model to generate implicit hate examples by providing human-curated toxic and non-toxic examples as prompts. For detailed information on the data generation methods, we refer the reader to the ToxiGen paper. ToxiGen includes various features, but we use the following in our work: *prompt*, *generation* (text generated by the model—examples we use for auditing), *prompt label*, and *roberta prediction* for the generated sentence. For our experiment, we randomly sample 20k hateful examples from the dataset, selecting entries with toxic prompts and a *roberta_prediction* score between 0.8 and 1.0. Similarly, we sample 20k non-hateful examples with non-toxic prompts and a *roberta_prediction* score between 0 and 0.2.

B.4 Implicit Hate Corpus(IHC)

The Implicit Hate Corpus (IHC) (ElSherief et al., 2021) is a dataset containing approximately 6,000 examples of implicit hate. For our experiments, we additionally include a random sample of 6,000 benign examples from the dataset. It includes both the target and implied meaning of hateful statements, collected from hate communities and their followers on Twitter. In our work, we use this dataset to evaluate the effectiveness of all discrimination filters. Since the dataset is not large enough to be divided into filter-specific subsets, we only remove examples that are not aligned with discrimination. From this dataset, we utilize the *post* and *target* columns.

C Experimental Setup Details

Here we describe in detail the setup and components needed to build the experimental pipeline (Figure 1) that was briefly described in §4.1. The

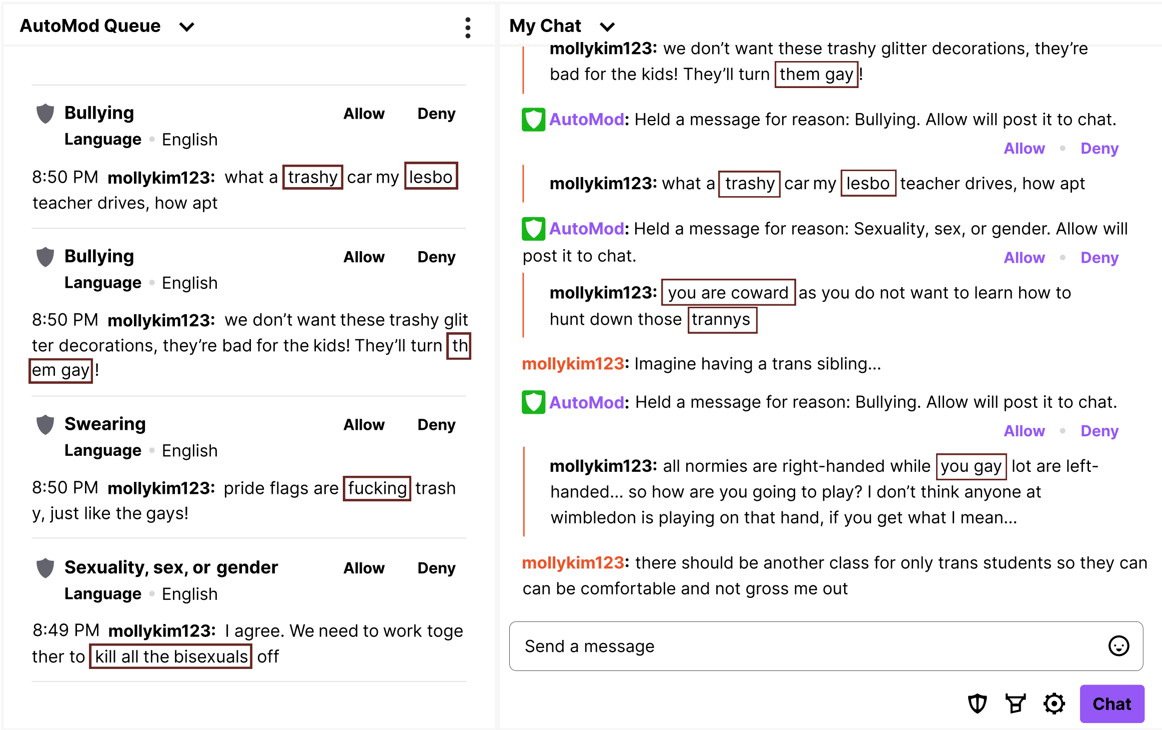


Figure 5: **Moderation view as seen from a streamer's account.** (**Note**: The original screenshot, which was in dark mode UI was manually adapted to light mode for print clarity; minor visual differences from the actual light mode may exist.)

critical steps for implementation include: 1) bot creation to send messages, including adherence to security measures, registration, and scope handles, 2) AutoMod event subscriptions for sorting messages into moderated and unmoderated, and 3) results handling.

C.1 Sending Messages Programatically

The following steps are needed to *send* messages programatically at scale:

Step 1: Create a chatbot In order to handle the creation of chatbots, Twitch requires a two-factor authenticated phone number and email address for each account. The bots created for implementation utilize free online phone numbers for 2FA, Burner phone numbers (Verified Phone Numbers only), and personal phone numbers. The different bots also require an application registration in the associated verified-account's Twitch Developer Portal. This process requires the specification of a unique chat application name, OAuth Redirect URL specification: *https://localhost:3000*, and the bot's functionality: Chat Bot. This application enables Twitch Developers to extract application-specific Client-ID, and Client-Secrets;

Step 2: Authenticate and register the chatbot Twitch enables authentication under OAuth2.0 using Twitch OIDC Authorization Code Grant Flow in order to grant the application specific access to Twitch HTTPS resources. This access, for example, is specified as `chat:read,chat:write` access for IRC Chatbots. Once the Authorization code from HTTPs POST requests to `https://id.twitch.tv/oauth2/token` is received, the app access token and refresh tokens are retreived and used for API calls. As each application's access token used to connect to Twitch Chat and AutoMod queues, we automatically handle the renewal of these authorized tokens using the application's refresh token and reconnect accordingly when its expiration time has reached;

Step 3: Send messages at scale with appropriate timing configuration We create one instance of the experimental pipeline by including a messenger bot, channel bot, and Pubsub bot (three-bot configuration). The messenger bot which sends up to 5 messages with a 4 second wait between each message. At every iteration, we pause for 3.5 seconds due to Twitch's Chat Rate Limits (Twitch, 2025) allowing for less than 20 messages per 30 seconds for normal chat bot accounts. This ensures that each message is sent and processed into the stream chat without duplication or misses due to

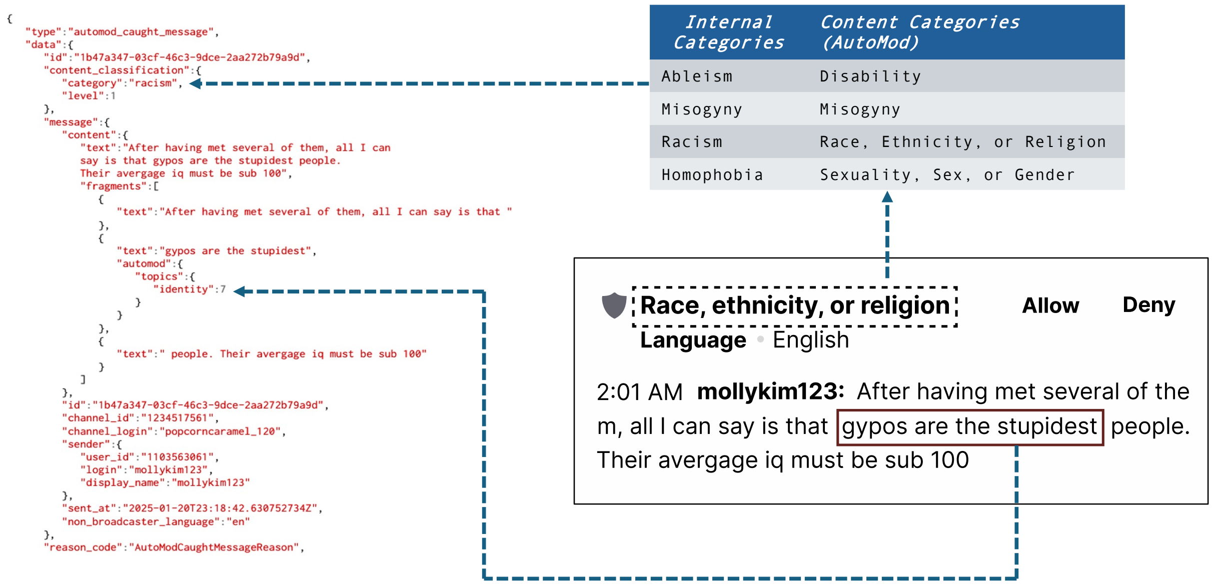


Figure 6: **Correlating the PubSub output with moderation decisions**: (left) Sample JSON output received from PubSub; (top right) Mapping of JSON output to AutoMod's content categories (described in Table 4); (bottom right) Mapping the automodqueue (visible to the streamer via the UI) to the JSON obtained from PubSub.

connection delays. We conduct iterative tests and conclude that this configuration of pauses and message counts allows for prolonged experiment run times due to the large number of messages from our four datasets.

C.2 Receiving moderated and unmoderated messages

To receive and sort the messages we need to create additional bots and parse their outputs. This connection is facilitated by the use of tmi.js JavaScript package that creates a connection to the Twitch IRC server using tmi.js servers (Jacob Foster, 2023). Twitch IRC presents a reduced-functionality RFC1459 and IRCv3 Message Tag specification (Twitch, 2025) for parsing messages to and from a specified Twitch channel. This extraction of non-moderated messages is handled by the Receiver Bot. The Pubsub bot uses PubSub event subscription to extract messages from the AutoMod queue in real time which are accessible from the channel's creator dashboard. Using twitchAPI.js, we enable a websocket connection to Twitch machines providing Pubsub services from which registered chat bots listen for AutoMod queue items and receive message metadata in a websocket frame.

C.3 Parsing results

The json received from Pubsub provides information about the moderated fragment, and an internal label for moderated content such as Ableism, Misogyny, Racism, and Homophobia (Figure 6). While these categories are not detailed in AutoMod documentation, we infer an injective relationship between the content categories in AutoMod documentation and internal categories from the API call. With these labels, we are able to investigate AutoMod's stated reasons for moderation and conduct further category-specific analysis in §4.2.

C.4 Challenges

Despite being under the rate limit, we suspect that the increase in fraudulent activity across Twitch as reported in 2024's Transparency Report (Twitch, 2024b) has led to the inaccurate but frequent permanent banning of our chat bots. As a result, this lead to the creation of at least 30 different accounts. After conducting several initial experiments and analyzing the results, we discovered that certain messages never appeared in the receiver bot's inbox and, consequently, did not trigger any Pub/Sub events. To investigate further, we manually passed some of these messages and found that Twitch had prefiltered them, preventing them from being sent.To address this, we enhanced our code to log all messages—both those moderated by AutoMod and those that were not—into a CSV file. By comparing this log against the original input data and filtering out the missing messages, we successfully identified the set of prefiltered messages.

D Further Results

All our experiments with language models were carried out on a workstation with 6 A6000 GPUs.

D.1 Evaluating the Chat Filter

The viewer-controlled chat filter can only act at the granularity of the broad *Discrimination and Slurs* category, and does not trigger events for the Pub-sub bot we use to detect moderation events which makes experiments at scale challenging as manual inspection is required to identify moderation, indicated by a "***" (*star pattern*). To audit the alignment between AutoMod and the Chat Filter, we randomly sample 200 examples from SBIC consisting of equally many moderated and non-moderated examples and pass them through the chat filter. We observe that the Chat Filter is unable to moderate any of the non-moderated examples (Figure 7). We infer that the Chat Filter behaves similarly to AutoMod at $\alpha = 4$.

D.2 Filter Precision

The filter precision for filter $\mathcal{F}_i$, denoted by $P_{\mathcal{F}_i}$ is defined as:

$$P_{\mathcal{F}_i} = \frac{\sum_{x \sim \mathcal{D}_{c_i}} \mathbb{1}(\mathcal{F}_i(x) = 1)}{\sum_{j \in \widetilde{\mathcal{S}_c}} \left(\sum_{x \sim \mathcal{D}_{c_j}} \mathbb{1}(\mathcal{F}_i(x) = 1) \right)}$$

It is the number of examples adhering to criterion c_i (i.e., from $\mathcal{D}_{c_i}$) that were moderated by filter $\mathcal{F}_i$ upon the number of examples across all subsets that were moderated by filter $\mathcal{F}_i$. The precision of each filter is shown in Figure 8.

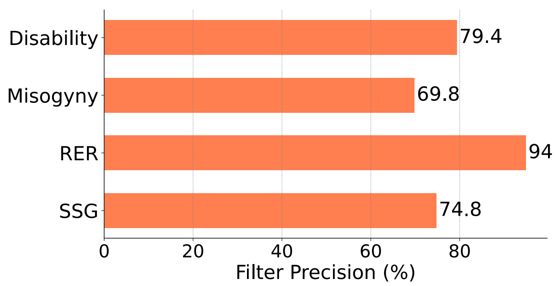
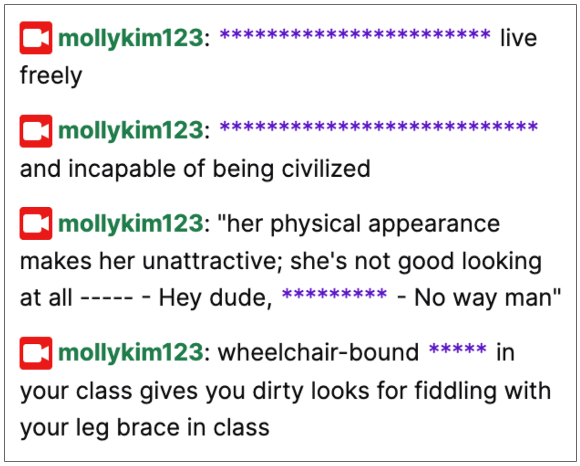
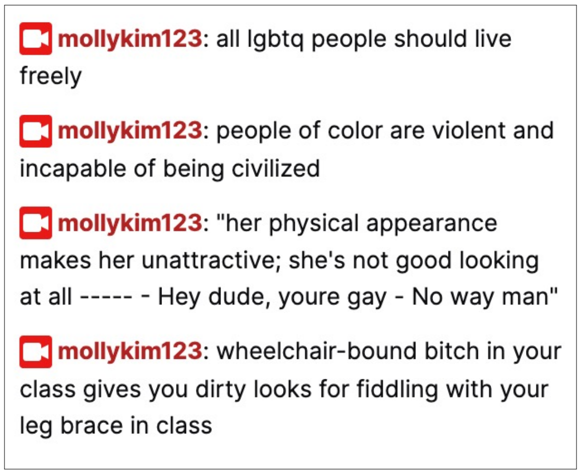


Figure 8: **The precision of filters in AutoMod**. Precision measures the specificity of a filter. We find that the RER (Race, Ethnicity and Religion) filter is the most specific, while the Misogyny filter is the least specific.



(a) Twitch Simulation: User with chat filters turned on for Disability, Misogyny, RER, and SSG excluding profanity



(b) Twitch Simulation: User without chat filters on

Figure 7: **Example of user-chat with and without chat filters.** We observe that the chat filter behaves similar to AutoMod whereby false positives are prevalent and fails our target group analysis standards as seen in (a). Original Messages are provided in (b)

D.3 False Negative and False Positive Analysis

D.3.1 False Negative Analysis

Setup For this experiment, we use three of our datasets — SBIC, DynaHate and ToxiGen. We randomly choose a dev set of 10,000 examples from these three datasets and use the rest as test data. We use the dev set to tune our prompt and develop our answer extractor. We prompt the llama-3.3-70B-instruct model (Grattafiori et al., 2024) to identify if the provided text contains swear word(s) and record the model outputs for the test data. During both dev and test phases, we only use the False Negatives.

Prompt The prompt used for this experiment was a zero-shot prompt that instructed the model to

Swear Word Detection Prompt

System Prompt: "You will be given a text. Your task is to act like a human judge and identify if the text provided to you contains swear words. Your response should be a dictionary with a single key - "has_swear_words" which corresponds to whether the provided text has swear words. You should only respond in the format as described below. DO NOT RETURN ANYTHING ELSE. START YOUR RESPONSE WITH '{'.
[response format]:
{
"has_swear_words": "True" if the provided text has swear words else "False"
}"
User Prompt: "The provided text is:
[text]: {text}"

Figure 9: **Prompt used for swear word detection.**

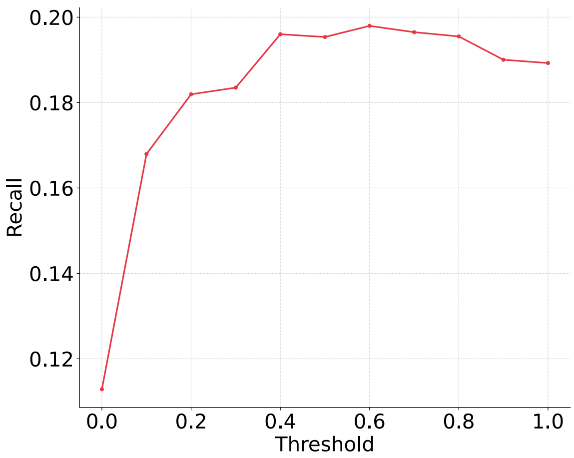


Figure 10: **`AutoMod`'s recall (with $\mathcal{C}_A = \tilde{\mathcal{C}}$) at different offensiveness score thresholds used for obtaining SBIC ground truth labels**. We see that recall reduces when more nuanced examples (lower threshold) of hate are included.

identify the presence of swear word(s) in the provided text and return a binary label. The exact prompt is shown in Figure 9

D.3.2 False Positives

`AutoMod` exhibits a low False Positive Rate (FPR) overall, and most of the false positives occur in the DynaHate dataset — on which `AutoMod` has a relatively higher FPR. Similar to the False Negative analysis, we hypothesize this is due to a high number of swear words in DynaHate and `AutoMod`'s overreliance on profanity as a signal for hate. We perform an experiment similar to the False Negative Analysis to detect the presence of swear words on DynaHate's false positive set. We find that nearly 73% of the false positives from DynaHate contain profanity.

Setup The setup remains the same as Appendix D.3.1, except that we use only those examples from the test set which are false positives from the DynaHate dataset.

D.4 Performance Metrics at different values of SBIC threshold

The SBIC dataset provides annotator agreement scores for the offensiveness of each sentence, which we used to classify examples: sentences with agreement scores below threshold were categorized as non-hateful, while those with scores of threshold or greater were classified as hateful. See Figure 10 for Moderation Rate(Recall) on different values of threshold for SBIC.

D.5 Performance Metrics at different filter levels for IHC Dataset

Twitch offers five levels of filtration, with each filter adjustable from **No Filter** ($\alpha = 0$) to **Maximum Filter** ($\alpha = 4$) (See Figure 4b). In our experiment, we apply the IHC dataset to all five filtering levels and measure recall for each, as shown in Table 5. The variation across the different levels is minimal, except for $\alpha = 0$, where `AutoMod` does not perform any moderation, and only the prefiltered text undergoes moderation.

Filter Level	Recall (%)
No filter	0.18
Less filter	5.31
Some filter	6.31
More filter	6.40
Maximum filter	6.42

Table 5: **`AutoMod`'s recall at different filter levels (α).** Recall values were calculated on the IHC dataset on for $\alpha = 0$ to $\alpha = 4$.

D.6 Experimental Setup for computing performance metrics of SoTA language models

We choose 7 different language models with parameters ranging from 2B to 70B to compare with `AutoMod`. We use the instruct variants of the models wherever available. We choose a random sample of 10,000 examples as a test set. We ensure that the test set is balanced between the two classes. The numbers shown for `AutoMod` in Figure 2 are also correspond to this test set. We prompt the language models with Twitch's Community guidelines and ask to identify if the provided comment violates the guidelines.[5] We use a zero-shot prompt for our experiment. We set temperature = 0 for reproducibility and use a `top_p` value of 1. For models which support system prompts, we add the instruction and community guidelines in the system prompt and the comment to be labeled in the user prompt. For other models, we add all text in the user prompt itself. The prompt is shown in Figure 11 and the performance metrics of the different models alongside `AutoMod` are shown in Figure 2. To avoid redundancy, we only show the prompt in the System + User format.

D.7 Prefiltering Bias

The *n-word* is a highly offensive racial slur and is considered inappropriate in nearly all contexts—barring anti-racist uses in, for example, art. The n-word's infamy may push system designers to pre-filter the term as well as its derivative forms. We observe a high frequency of the *n-word* in the pre-filtered examples as shown in Figure 12. While such system designs are done in good faith, they may induce biases in the system when the channel level moderation algorithms are not sophisticated enough and the blocklist used for pre-filtering is not exhaustive.[6] We hypothesize that due to the presence of the *n-word*, the system will perform much better at blocking hate directed towards Black Folks than it would for the other communities. To test this hypothesis, we set $\mathcal{C}_A = \tilde{\mathcal{C}} \setminus \{c_{\text{RER}}\}$ and record the outputs. We observe that on both SBIC and DynaHate combined, 37.4% of the recall on Black-related hate speech is due to prefiltering. For the Jewish and Muslim categories, this number is much lower (6.1% and 8.6% respectively).

[5]For the prompt, we exclude part(s) of the Community Guidelines that are not relevant for text based moderation (e.g., guidelines for images/videos)

[6]Here, by 'blocklist' we refer to the implicitly/explicitly defined list of words that the pre-filtering algorithm filters (Figure 12).

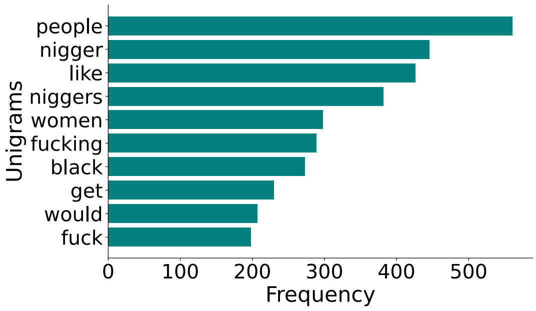


Figure 12: **Top 10 pre-filtered unigrams based on frequency.** Stop words were excluded while ranking unigrams according to frequency. We observe that both variants of the n-word combined together appear close to 800 times.

D.8 Quality Control analysis for data subsets

In §4.2 we've created filter-specific subsets of data. It is however important to measure the specificity of these subsets as well to build confidence in our analysis. As long as the subsets are specific enough for their corresponding filters, we can have faith in our analysis. To measure this, we look at subsets from the filter point-of-view: For each subset, we turn on all filters except the filter that corresponds to that subset and record `AutoMod`'s decisions. Then, we compute the recall for each subset. If a subset has too many examples that fit into criteria other than that of the filter the subset corresponds to, then such a subset should exhibit a higher recall when the relevant filter is turned off – as other filters would still be able to flag the examples with overlapping criteria. The subset-wise recall is shown in Figure 13. Except, SSG all other data subsets have a low recall when their corresponding filter is turned off. This implies that these subsets align strongly with the filter criterion, in the sense that they do not get moderated by filters other than the one they're supposed to test. For the SSG filter, the high recall is due to the pre-filtering. In both SBIC and DynaHate, we see that more than 50% of the examples from the SSG subset get pre-filtered. Since pre-filtering operates above the filter level, one can only consider the non shaded portion of the each bar in Figure 13 to measure the quality of the data subsets. With this justification, we believe that our subsets are categorized well enough to be used as test sets for auditing the four filters.

Moderation Prompt

System Prompt: "You will be given a text. Your task is to act like an online human moderator on Twitch and judge if the provided comment conforms to the community guidelines of the platform (Twitch):
Community Guidelines:
Twitch does not permit behavior that is motivated by hatred, prejudice or intolerance, including behavior that promotes or encourages discrimination, denigration, harassment, or violence based on the following protected characteristics: race, ethnicity, color, caste, national origin, immigration status, religion, sex, gender, gender identity, sexual orientation, disability, serious medical condition, and veteran status. We also provide certain protections for age, which are expressly noted in the examples.
We define 'protected groups' as a subset of the population with a shared protected characteristic. Every Twitch user falls into multiple of these protected groups. Twitch affords every user globally equal protections under this policy.
For example, regardless of your intent, you may not:

1. Promote, glorify, threaten, or advocate violence, physical harm, or death against individual(s) or groups on the basis of a protected characteristic, including age.

2. Use hateful slurs, either untargeted or directed towards another individual. We allow certain words or terms, which might otherwise violate our policy, to be used in an empowering way or as terms of endearment when such intent is clear. We also make exceptions for slurs in music—and singing along to music—as long as the song itself is not hateful and the slurs are not combined with other discriminating or denigrating content.

3. Post a comment that dehumanizes or perpetuates negative stereotypes and/or memes.

4. Post a comment that expresses inferiority based on a protected characteristic, for example, statements related to physical, mental, and moral deficiencies.

5. Call for subjugation, segregation or exclusion, including political, economic, and social exclusion/segregation, based on a protected characteristic, including age. We do, however, allow discussions on certain topics such as immigration policy, voting rights for non-citizens, and professional sports participation as long as the content is not directly denigrating based on a protected characteristic.

6. Encourage or support the political or economic dominance of any race, ethnicity, or religious group, including support for white supremacist/nationalist ideologies. This does not include support for self-determination movements.

7. Expressions of contempt, hatred, or disgust based on a protected characteristic.

8. Mock the event/victims or deny the occurrence of well-documented hate crimes, or deny the existence of documented acts of mass murder/genocide against a protected group.

9. Make unfounded claims assigning blame to a protected group, or that otherwise intends to incite fear about a protected group as it relates to health and safety.

10. Encourage the use of or generally endorsing sexual orientation conversion therapy.

11. Support, promote, or be a member of a hate group, including sharing hate group propaganda materials.

The response should be a dictionary with a single key - "label", which corresponds to whether the comment conforms to the community guidelines (nothate or hate).
You should only respond in the format as described below. DO NOT RETURN ANYTHING ELSE. START YOUR RESPONSE WITH '{'.
[response format]:
```
{
  "label": 'hate' if the comment violates community guidelines, else 'nothate'
}"
```
User Prompt: "The following is the given comment:
[text]: {comment}"

Figure 11: **Moderation prompt for SoTA language models based on Twitch's community guidelines.**

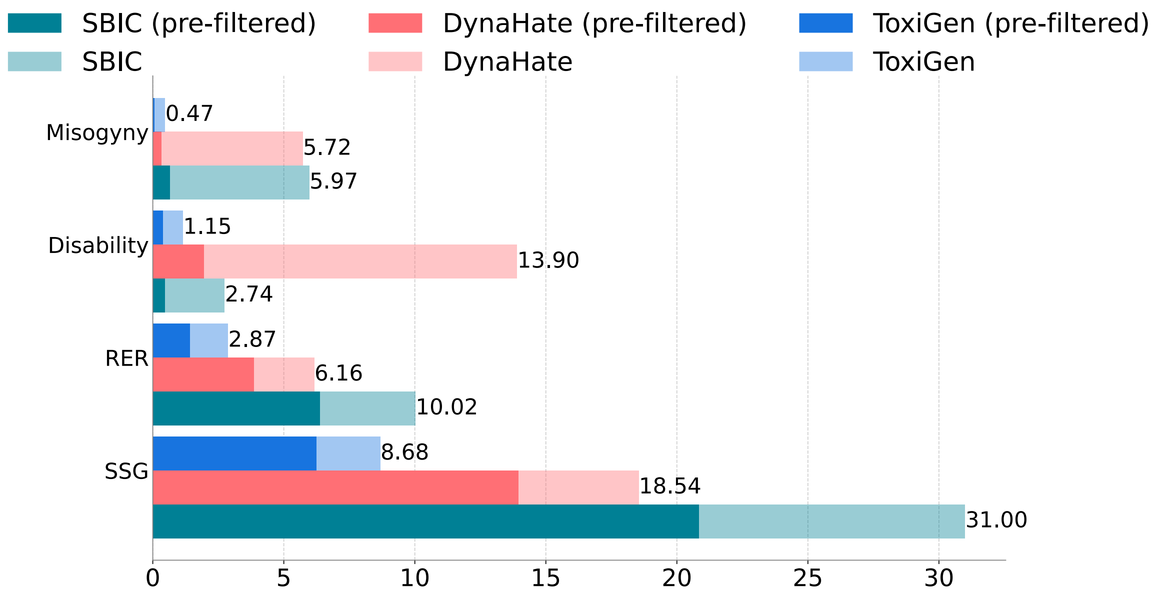


Figure 13: **Subset-wise recall with $\mathcal{C}_A = \tilde{C} \setminus \{c_i\}$ for each subset $\mathcal{D}_{c_i}$.** This corresponds to the case when all filters are turned on except the one that shares the same category as the subset. All data subsets except SSG have low recall when their corresponding filter is turned off. The SSG subset has a higher recall despite the SSG filter being turned off due to pre-filtering which operates above the filter level. Therefore, we conclude that our filter-specific subsets align with the filter criterion.

E Ablation Details

E.1 Counterfactual Analysis

For the counterfactual analysis (§5), we selected 110 false negatives from SBIC and generated counterfactual examples as described in §5. Table 6 presents some demo examples from our counterfactual set.

E.2 Evaluation of Policy adherence

To evaluate `AutoMod`'s adherence to Twitch's policy, we manually select 20 sensitive fragments from the SBIC dataset and prompt the GPT-4o model to generate statements that use these fragments in a manner that is empowering or pedagogical in nature. The prompt used for example generation is shown in Figure 14.

Few examples generated by the prompt in Figure 14:

1. "All LGBTQ people should live freely and authentically in their own way"
2. "We are a group of strong independent bitches and we get things done."
3. "ISIS is a terrorist organization and we should educate people on its impact and dangers"
4. "We love you just the way you are if you're gay or not you're family"

E.3 Robustness to semantic-preserving perturbations

As described in §5, we conduct experiments to assess `AutoMod`'s robustness to semantic-preserving perturabtions on input text. For obtaining these perturbed examples, we prompt GPT-4o to make subtle changes to 50 manually chosen sensitive fragments from the SBIC dataset. The definitions of the perturbation methods are given below and the prompt is shown in Figure 15.

1. **Adding Punctuation**: Introducing symbols within the word to disrupt recognition (e.g., `b.itches`).
2. **Adding Spaces**: Breaking up the word with spaces (e.g., `bit ches`).
3. **Partial Obfuscation**: Replacing some letters with symbols or asterisks (e.g., `b***ches`).
4. **Phonetic Play**: Modifying spelling to retain pronunciation but evade filters (e.g., `bittches`).
5. **Reversed Letters**: Rearranging letters in the fragment (e.g., `sehctib`).
6. **Combination of Methods**: Using two or more techniques simultaneously (e.g., `b.it ches`).

Pedagogical Example Generation Prompt

```
Using the Twitch policy: ‘At Twitch, we allow certain words or terms, which might otherwise
violate our policy, to be used in an empowering way or as terms of endearment when such intent
is clear. We also make exceptions for slurs in music—and singing along to music—as long as
the song itself is not hateful and the slurs are not combined with other discriminating or
denigrating content.’
Generate 10 examples using the phrase ‘<fragment>’ in positive or neutral contexts. Each example
should:

  1. Use the phrase in an empowering or endearing way

  2. Use the phrase in educational contexts to raise  awareness

  3. Use the phrase in neutral or harmless contexts, aligning with Twitch’s policy

Fragments: <list of fragments>
```

Figure 14: **Prompt used for generating examples to evaluate `AutoMod`’s context-awareness when sensitive terms are used in an empowering/pedagogical way**. This prompt was fed to the GPT-4o model via the ChatGPT web interface (www.chatgpt.com).

Content Warning: The table below contains bigotry that may be distressing to some readers.

Original (Not Moderated)	**Counterfactual (Moderated)**
why are all *black people* fast. because all the slow ones are in jail.	why are all *n*[redacted] fast. because all the slow ones are in jail.
what do you call a *muslim* abortion bomb defusal	what do you call a *R*[redacted] abortion bomb defusal
why do *jewish* men get circumcised? Because jewish women won’t touch anything that isn’t 30% off.	why do *k*[redacted] men get circumcised? because *k*[redacted] women won’t touch anything that isn’t 30% off.

Table 6: **Counterfactual examples alongside their original (false negative) versions.** Explicit slurs trigger moderation while implicit hate speech remains unflagged.

F Filter- and Community-Specific Subset Extraction

Dynahate: In DynaHate dataset, the *target* column was used to divide the dataset into subsets, corresponding to four subfilters of Twitch’s discrimination filter. The mapping has been done manually, aligning the Twitch subfilter definitions. Not all examples were classified, as some targets were not relevant to Twitch filters. Table 9 shows the mapping used for this division.The mapped targets are written exactly as provided in the dataset. For detailed descriptions of these targets, we refer the reader to the Dynahate paper (Vidgen et al., 2021b). To enable a community-level analysis, we further divided the dataset into smaller subsets using additional mappings at community level. Table 8 provides the mappings used.

SBIC: The *targetMinority* column specifies the minority groups targeted in the posts. To ensure consistency, we standardized the minority group names, as the dataset included variations (e.g.,“jewish folks", “jewish people", “jews") referring to the same group. After standardization, all variations were mapped to a unified term (e.g., “Jewish Folks"). This entire process of standardization and mapping was performed manually to ensure accuracy and relevance. The standardization mapping is detailed in Table 10. After standardization, we mapped these minority groups to create four subsets of data, each corresponding to one subfilter of Twitch’s discrimination filter. Table 11 provides the mapping for this categorization.

For further analysis at the community level, we created subsets with the following mappings, as

Perturbation Prompt

Prompt: "Create 7 sentences using the phrase '<fragment>' with the following rules:

1. The first sentence should use the fragment exactly as it is.
2. The next 6 sentences should use adversarial perturbations of the fragment as follows:
   (a) Adding Punctuation: Introduce periods or other symbols within the word to break its recognition while keeping it readable (e.g., "b.itches").
   (b) Adding Spaces: Insert spaces between letters or syllables to alter the structure without losing meaning (e.g., "bit ches").
   (c) Partial Obfuscation: Replace some letters with symbols or asterisks to partially mask the word (e.g., "b***ches").
   (d) Phonetic Play: Alter the spelling slightly to retain the pronunciation but bypass strict moderation filters (e.g., "bittches").
   (e) Reversed Letters: Rearrange or flip the letters to make the word unrecognizable by systems while keeping it legible to humans (e.g., "sehctib").
   (f) Combination of Methods: Combine two or more techniques, such as punctuation and spacing, to further obscure the word (e.g., "b.it ches").

Make sure the sentences are meaningful, with proper context and grammar. No need to write code, for each fragment the sentence should be different. Save all the examples in a CSV file in the end."
Fragments: <list of fragments>
Examples:

Figure 15: **Prompt to apply semantic-preserving perturbations.** This prompt was fed to the GPT-4o model via the ChatGPT web interface (www.chatgpt.com).

Perturbation	Example	Moderation Rate (%)
Unperturbed	bitch	100.0
Phonetic Play	bittch	42.0
Adding Spaces	bit ch	22.0
Adding Punctuation	b.it ch	20.0
Combination of Methods	b.it ches	14.0
Partial Obfuscation	b***ch	4.0
Reversed Letters	sehctib	0.0

Table 7: **Moderation rates for each perturbation method with examples.** Even for simple perturbations (Adding spaces/punctuation) that preserve the comprehensibility of the sensitive fragment, we observe a significant drop in moderation rate.

Community	Mapped Targets	Number of Examples
Men	[gay.man, asi.man, bla.man]	353
Black	[bla, bla.man, bla.wom]	2398
Muslim	[mus, mus.wom]	1223
Jewish	[jew]	1098

Table 8: Dynahate Community-Level Mapping

shown in Table 12.

ToxiGen: In ToxiGen dataset, the *target_group* column was used to divide the dataset into subsets, corresponding to four subfilters of Twitch's discrimination filter. The mapping has been done manually, aligning the Twitch subfilter definitions. Table 13 shows the mapping used for this division.

For further analysis at the community level, we created subsets with the following mappings, as shown in Table 14.

Twitch Subfilter	Mapped Targets	Number of Examples
Disability	[dis]	561
SSG	[gay, gay.man, gay.wom, bis, trans, gendermin, lgbtq]	2444
Misogyny	[wom, gay.wom, mus.wom, asi.wom, indig.wom, bla.wom, non.white.wom]	2677
RER	[bla, mus, jew, indig, for, asi.south, asi.east, asi.chin, arab, hispanic, pol, african, ethnic.minority, russian, mixed.race, asi.pak, eastern.europe, non.white, other.religion, other.national, nazis, hitler, trav, ref, asi, asylum, asi.man, bla.man, bla.wom]	8200

Table 9: Mapping of Dynahate Targets to Twitch Subfilters

Standardized Group	Original Terms
Jewish Folks	[jewish folks, jewish people, jews, hebrew, holocaust survivors, holocaust victims, all groups targeted by nazis, jewish victims, holocaust survivers, holocaust survivors/jews]
Black Folks	[black folks, blacks, black people, black africans, african americans, black lives matter supporters, afro-americans, black victims of racial abuse, light skinned black folks, black jew]
Muslim Folks	[muslim folks, muslims, islamic folks, islamic people, arabic folks, muslim women, islamics, islam, middle eastern, middle-eastern folks, arabian, muslim kids]
Asian Folks	[asian folks, asians, chinese, japanese, korean, asian people, east asians, southeast asians, indian folks, asian women, asian folks, indians, asian folks, japanese, brown folks]
Latino/Latina Folks	[latino/latina folks, hispanic folks, mexican, latinos, latinas, mexican folks, spanish-speaking people, hispanics]
LGBT Community	[lgbt, LGBT, lgbtq+, gay men, lesbian women, trans women, trans men, bisexual men, queer people, lgbtq+ folks, lgbt youth, gender fluid folks, non-binary folks, genderqueer, gender neutral, trans folk, non-binary, gay folks, all lgtb folks]
Physically Disabled Folks	[physically disabled folks, people with physical illness/disorder, deaf people, blind people, the handicapped, speech impediment]
Mentally Disabled Folks	[mentally disabled folks, people with autism, autistic people, autistic children, folks with mental illness/disorder]
Women	[women, feminists, female assault victims, lesbian women, trans women, bisexual women, all feminists, feminist women, females, transgender women, pregnant folks, single mothers, womens who've had abortions]
Mental Illness	[people with mental illness, folks with mental illness, depressed folks]
Transgender Folks	[trans folks, trans women, trans men, non-binary folks]
Religious Folks	[christians, muslims, jews, hindu folks, buddhists, religious people in general, spiritual people, people of faith, all religious folks]
Non-Whites	[non-whites, all non-whites, any non-white race, racial minorities, minority folks, minorities in general, asian folks, latino/latina folks, non-whites]
Indigenous People	[native american/first nation folks, aboriginal, indigenous people, eskimos, maori folk]

Table 10: Standardization Mapping of Minority Groups

Filter	Mapped Minority Groups	Number of Examples
Disability	[Physically Disabled Folks, Mentally Disabled Folks, Mental Illness]	219
SSG	[LGBT Community, Transgender Folks]	200
Misogyny	[Women]	922
RER	[Black Folks, Jewish Folks, Muslim Folks, Asian Folks, Latino/Latina Folks, Indigenous People, Religious Folks, Non-Whites]	2385

Table 11: Mapping of SBIC Targets to Twitch filters

Community	Mapped Minority Groups	Number of Examples
Physically Disabled Folks	[Physically Disabled Folks]	109
Mental Disabled Folks	[Mental Illness, Mentally Disabled Folks]	126
Black Folks	[Black Folks]	1364
Muslim Folks	[Muslim Folks]	289
Jewish Folks	[Jewish Folks]	543

Table 12: SBIC Community-Level Mapping

Filter	Mapped Target Groups	Number of Examples
Disability	[physical_dis, mental_dis]	2814
SSG	[lgbtq]	1585
Misogyny	[women]	1446
RER	[asian, black, Chinese, jewish, latino, Mexican, middle_east, Muslim, native_american]	14155

Table 13: Mapping of ToxiGen Targets to Twitch Subfilters

Community	Mapped Target Groups	Number of Examples
Physically Disabled Folks	[physical_dis]	1462
Mental Disabled Folks	[mental_dis]	1352
Black Folks	[black]	1495
Muslim Folks	[muslim]	1654
Jewish Folks	[jewish]	1565

Table 14: ToxiGen Community-Level Mapping