A Low-Cost Human-in-the-Loop Investigation of Toxicity on GitHub at Scale

Rahat Rizvi Rahman
Virginia Commonwealth University
Richmond, USA
rahmanr12@vcu.edu

Mia Mohammad Imran
Missouri University of Science and Technology
Rolla, USA
imranm@mst.edu

Kostadin Damevski
Virginia Commonwealth University
Richmond, USA
kdamevski@vcu.edu

Abstract

Toxic interactions in open source discussions can alienate contributors and threaten project sustainability, yet prior empirical studies of GitHub toxicity have been limited in scale, raising questions about their generalizability. Scaling up is difficult because toxicity on GitHub is often implicit and context-dependent, making both fully manual annotation and LLM-based labeling unreliable.

We present a human-in-the-loop (HITL) annotation methodology that makes large-scale, domain-calibrated toxicity labeling practical. A single call to a small, local LLM produces both a toxicity prediction and a set of interpretable event category scores. A lightweight Random Forest validator then uses those scores to flag the small subset of conversations most likely to be mislabeled, directing human review only where it is needed. The validator outperforms confidence-based and multi-LLM baselines while adding low annotation cost.

We apply this pipeline to over 124,000 GitHub issue and pull request conversations. Using the resulting dataset, we evaluate key findings from prior small-scale research, confirming some and qualifying others, and present new insights into the prevalence, characteristics, and dynamics of toxicity across diverse open source projects.

CCS Concepts

• **Software and its engineering → Open source model**; **Programming teams**.

Keywords

Toxicity, Bug Report, Empirical Study, Open Source Software

ACM Reference Format:
Rahat Rizvi Rahman, Mia Mohammad Imran, and Kostadin Damevski. 2026. A Low-Cost Human-in-the-Loop Investigation of Toxicity on GitHub at Scale. In *Proceedings of the 41st IEEE/ACM International Conference on Automated Software Engineering (ASE '26), October 12–16, 2026, Munich, Germany.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3832783.3837505

1 Introduction

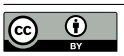
Open source collaboration on GitHub has enabled substantial innovation, but it has also created space for toxic discourse in project discussions. Recent research suggests that toxicity on GitHub differs in important ways from toxicity on other online platforms [16, 39, 58]. Rather than appearing primarily as overt hostility, it often takes subtler, more technical forms, such as dismissive language, passive-aggressive comments, condescending code reviews, and exclusionary feedback toward newcomers. These behaviors can discourage participation, especially among less experienced developers, which in turn threatens project sustainability and the health of the open source ecosystem.

Empirically studying toxicity on GitHub is also methodologically challenging. Toxicity in conversations is shaped by interaction structure and context, not surface text alone [4]. Moreover, toxicity detection remains inherently subjective: judgments vary across annotators, perspectives, and context, including when LLMs are used as evaluators [6]. As a result, even frontier LLMs do not reliably detect GitHub toxicity. For instance, when we evaluated Nemotron-3-Super [7], a large frontier model with 120B parameters, on Imran et al.'s [30] dataset of 898 human-annotated GitHub conversations, it correctly classified all 696 non-toxic instances but missed half of the toxic ones (detecting only 101 out of 202), revealing a strong bias toward predicting non-toxicity. Such results suggest that generic model judgments miss the implicit, context-dependent forms of toxicity common in software engineering discourse.

Several well-known studies have examined toxicity on GitHub, but most have been small in scale. Miller et al. [39] analyzed 100 issues and found that toxicity is often directed at individuals rather than code, and appears more frequently in discussions involving repeated reporters. Ferreira et al. [14] annotated 205 locked issues and observed that many toxic discussions emerge from miscommunication or disagreement about code and tend to escalate over time. These studies identified important patterns, but their limited scale constrains generalization.

Scaling this line of work is difficult because strong indicators of toxicity are rare. For example, locked issues account for only 0.22% of GitHub issues [52]. This creates a two-sided gap: the field lacks large-scale, GitHub-calibrated toxicity labels, and producing such labels is expensive. Full manual annotation is prohibitively costly, while generic LLM labeling can be unreliable when toxicity depends on community-specific norms and discussion context [35].

Recent work shows that LLMs can support toxicity detection on GitHub [2, 40] and can help model conversational dynamics for forecasting derailment in OSS discussions [30]. At the same time, these studies show that performance depends heavily on prompt design, contextual framing, and task formulation, rather than on simply applying a powerful model out of the box. In addition, frontier LLMs can be expensive to deploy at scale, making annotation efficiency a methodological requirement, not just an engineering concern.



ASE '26, Munich, Germany

ACM ISBN 979-8-4007-2882-2/2026/10
https://doi.org/10.1145/3832783.3837505

In this paper, we address this gap with two contributions: a methodological one and an empirical one. On the methodological side, we introduce a human-in-the-loop (HITL) annotation pipeline that produces GitHub-calibrated toxicity labels at low cost. A single call to a small, local LLM assigns a first-pass toxicity label and simultaneously produces a set of interpretable event category scores that characterize the conversation's tone. A lightweight Random Forest validator then uses those scores to flag the small subset of conversations most likely to be mislabeled, sending only those for human review. This design avoids repeated prompting and multi-pass labeling, reduces annotation cost, and adapts to GitHub-specific communication patterns, building on HITL methodology from Wang et al. [64].

On the empirical side, we apply this pipeline to annotate over 124K GitHub issue and pull request discussions for toxicity. The paper is organized accordingly: Section 2 describes the pipeline's design, implementation, and evaluation, while Sections 3–5 use the resulting dataset to revisit prior findings at scale and present new insights into toxicity dynamics. The main contributions of this paper are:

- Development and evaluation of a HITL annotation pipeline for GitHub toxicity that combines single-pass local-model annotation with a lightweight validation model, requiring only one LLM call per conversation and reducing human review to roughly 1.5% of the dataset.
- A large-scale dataset of over 124K GitHub issue and pull request discussions annotated for toxicity.
- Empirical validation of toxicity hypotheses from prior small-scale studies on a substantially larger dataset.
- New insights into toxicity dynamics, including project governance, contributor behavior, and language ecosystem differences.

Our findings can inform the design of automated moderation tools, improve community guidelines, and support initiatives that promote a healthier open source environment. The dataset and methodology can also serve as a foundation for future research in computational social science and software engineering.

2 Human-in-the-Loop Annotation of Toxicity on GitHub

Annotating GitHub discussions for toxicity at scale requires balancing automation with reliability. LLMs can provide efficient first-pass labels, but their outputs may be inaccurate or systematically biased relative to human judgment [28, 38, 69–71], motivating hybrid strategies that combine automated labeling with targeted human review [46, 64].

This section presents our human-in-the-loop annotation methodology. To avoid circularity, we develop and evaluate the validator on previously annotated GitHub datasets before applying it to our newly collected large-scale 2024 corpus (Section 3).

Our pipeline has three stages. First, an LLM generates an initial binary toxicity prediction along with event category scores that characterize the conversation's tone and dynamics (§2.1). A separate machine learning validator then estimates whether the LLM output is likely incorrect, using the event category scores to identify unreliable annotations (§2.2). Finally, human annotators review only those instances flagged by the validator, concentrating manual effort on the conversations most likely to be mislabeled.

Our pipeline is thus human-in-the-loop in the annotation-verification sense established in prior work on human–LLM collaborative labeling [46, 64]. The LLM is not retrained during annotation, rather, human judgment enters the pipeline at two points: 1) human-annotated data trains the validator that decides which LLM outputs require review, and 2) human annotators serve as the final label authority for every flagged instance.

2.1 First-Pass LLM Toxicity Annotation

The first stage of the pipeline assigns an initial toxicity label to each GitHub conversation using the Ministral model (*ministral3:14b*) [37]. We selected this model because it is open weight under Apache 2.0, supports long contexts of up to 128k tokens, and is small enough to process large collections of conversations efficiently on available hardware. Larger alternatives such as *Llama 3.3:70B* were impractically slow in our setting, whereas smaller alternatives such as *Ministral-3:8B* were less accurate.

With temperature set to 0, the model produces two primary outputs for each conversation: a binary toxicity prediction (1 if clear toxic language is present, 0 otherwise) and a set of eight event category scores (described below) that decompose the conversation along dimensions known to make toxicity annotation difficult in software engineering contexts. The event category scores are not used to classify toxicity directly; rather, they serve as features for a separate validation model that identifies conversations where the LLM is likely to produce unreliable labels (Section 2.2.4). We also use another prompt to detect the toxicity prediction with a confidence score (0–10) for the generated toxicity and a brief explanation, retained for baseline comparisons.

2.1.1 Prompt Structure. The prompt guides the model through a step-by-step toxicity assessment with three main elements:

- **Toxicity Definition:** A definition tailored to GitHub conversations, drawing on the toxicity categories of Miller et al. [39] (e.g., insulting, entitled, arrogant) and the incivility markers of Ferreira et al. [16] and Ehsani et al. [13] (e.g., frustration, mocking, dismissiveness). A conversation is toxic if it contains rude, disrespectful, dismissive, antagonistic, profane, mocking, or hostile language that could make a reasonable contributor feel unwelcome. The definition distinguishes language directed at people from strong negative language directed at project artifacts [39, 56], noting that harsh technical criticism can be non-toxic if it remains professional.
- **Binary Classification Task:** Instructions to assign a toxicity prediction (0/1), a confidence score (0–10), and a brief explanation.
- **Event Category Scoring:** Instructions to score each of eight event categories from 0 to 10 based on explicit text evidence. These categories (Table 1) were derived from prior studies of toxicity, incivility, and conflict in open source communities, and capture dimensions along which conversations become difficult to annotate—e.g., strong process criticism mixed with resolution language, or escalating tension without overt abuse. The prompt emphasizes that category scoring and toxicity prediction are

Table 1: Event category features scored by the LLM (0–10), used by the validator to identify conversations likely to be hard for the LLM to annotate.

Category	Definition
`person_hostility`	Hostile language directed at a person or group (insults, name-calling), excluding criticism of code or process [24, 39, 56].
`process_hostility`	Strongly negative language directed at code, tools, processes, or decisions rather than people. Common in OSS and largely independent of toxicity [39, 56].
`dismissiveness`	Language that shuts down or refuses to engage with another participant's contribution or concern [16, 63].
`escalation`	Conversation becomes more tense or confrontational over time without overt insults, slurs, or profanity [13, 14, 39].
`subtle_criticism`	Mild or indirect criticism or disapproval that is noticeable but not strongly explicit [13, 14].
`hostility_density`	Concentration of person-directed hostile language relative to conversation length [57].
`resolution_language`	Signs of agreement, de-escalation, or collaborative closure, indicating heated exchanges were normal debate [19].
`behavior_callout`	Commentary explicitly naming problematic behavior (e.g., "that's rude"), indicating community self-policing [17].

separate tasks, and that a high score on any single category does not automatically indicate toxicity.

2.2 Annotation Validation Model

Although the first-pass LLM outputs were generally reasonable, they remained prone to error in cases common in software engineering discourse, such as project-specific figurative language, lengthy discussions containing stack traces or logs, and emotionally charged technical exchanges that are not necessarily toxic [28]. We therefore introduce a separate validation model that identifies conversations where the LLM annotation is most likely to be unreliable. Crucially, the validator does not relabel conversations or detect toxicity directly. Instead, it learns to recognize *confusing* conversations—those where competing signals (e.g., strong process criticism alongside resolution language, or subtle hostility mixed with collaborative tone) make the correct label hard to determine automatically—and flags them for human review.

2.2.1 Validator Features. The validator uses the eight event category scores produced by the LLM prompt as its feature set. These scores, each ranging from 0 to 10, characterize different dimensions of the conversation's tone and interpersonal dynamics. Table 1 summarizes the categories and the prior work motivating each one.

Using the event category scores as validator features offers several advantages over externally computed features. The scores are derived directly from the LLM's reading of the conversation, so they reflect the same evidence the model used for its toxicity prediction. They decompose the conversation into interpretable dimensions that can reveal *conflicting signals*, for example, a conversation with high process hostility but also high resolution language is inherently ambiguous and more likely to be misclassified. Because the scores are produced alongside the toxicity prediction in a single LLM call, they add no additional inference cost.

Table 2: Validator performance (5-fold cross-validation) at selected thresholds (τ). *Inc.* = Incorrect class (LLM ≠ Human); *Cor.* = Correct class.

	Inc.			Cor.			Macro	Weighted
τ	P	R	F1	P	R	F1	F1	F1
0.1	0.76	0.32	0.45	0.94	0.99	0.96	0.71	0.92
0.3	0.70	0.54	0.61	0.96	0.98	0.97	0.79	0.94
0.5	0.68	0.69	0.68	0.97	0.97	0.97	0.83	0.94
0.7	0.60	0.76	0.67	0.98	0.95	0.96	0.82	0.94
0.9	0.42	0.86	0.56	0.99	0.89	0.93	0.75	0.90

2.2.2 Training Procedure. We trained a Random Forest validation model on Imran et al.'s manually annotated 898 GitHub conversations (202 toxic, 696 non-toxic) [30]. Because the dataset is imbalanced (approximately 22% toxic), we applied SMOTE (Synthetic Minority Over-sampling Technique) to the training folds to balance the class distribution during model training. We evaluated the model using stratified 5-fold cross-validation to preserve the class distribution in each fold and to obtain more robust performance estimates across the full dataset. We implemented the model using the *scikit-learn* and *imbalanced-learn* libraries [47].

2.2.3 Validator Evaluation. Table 2 presents the validator's 5-fold cross-validation performance at different thresholds τ, reporting metrics for the *Incorrect* class (LLM ≠ Human) and *Correct* class (LLM = Human). High Incorrect recall is desirable, as it indicates the validator effectively flags LLM mistakes for human review. As τ increases, Incorrect Recall rises while Precision decreases, reflecting a trade-off between catching errors and minimizing unnecessary review. At $\tau = 0.5$, the model achieves the best balance: Macro F1 of 0.83, Incorrect F1 of 0.68, and Weighted F1 of 0.94.

To understand which features are most informative for the validator, we computed Random Forest feature importances (mean decrease in impurity) averaged across the cross-validation folds. The most influential feature is `escalation` (importance 0.373), followed by `subtle_criticism` (0.249) and `process_hostility` (0.158). Together, these three features account for over 78% of the total importance, indicating that the validator relies most heavily on signals of indirect tension and tone rather than overt hostility. Notably, `person_hostility` (0.039) and `behavior_callout` (0.006) contribute the least, suggesting that conversations with clear interpersonal hostility or explicit norm enforcement are relatively easy for the LLM to classify correctly, whereas ambiguity arises primarily in conversations with escalating but implicit tension.

2.2.4 Baseline Comparison. To assess whether the event category features genuinely help identify difficult-to-classify conversations, we compare our validator against several baselines (Table 3). The three Random Forest variants share the same setup: SMOTE oversampling and stratified 5-fold cross-validation on the 898-conversation dataset, with the best threshold reported. The remaining baselines are evaluated on the full dataset without cross-validation, since they involve no trained model.

Confidence Threshold. The simplest baseline flags predictions whose self-reported confidence falls below a threshold. This performed poorly: for thresholds 0-7 it flagged no incorrect predictions (Macro F1 = 0.488, equivalent to the majority class). Even at the best threshold of 9, Macro F1 reached only 0.592 with Incorrect Recall

Table 3: Comparison of validator approaches for identifying incorrect LLM annotations. RF models use SMOTE and stratified 5-fold cross-validation; other baselines are evaluated on the full 898-conversation dataset. Best threshold per approach is shown. *LLM Calls*: inference calls per conversation. *Human Review*: estimated percentage of the dataset requiring manual annotation.

Approach	τ	Incorrect (LLM ≠ Human)			Correct (LLM = Human)			Macro	LLM	Human
		Prec.	Rec.	F1	Prec.	Rec.	F1	F1	Calls	Review (%)
Confidence threshold	9	0.400	0.143	0.211	0.959	0.989	0.974	0.592	1	1.7% (15/898)
RF (conf. + pred.)	0.5	0.238	0.571	0.336	0.977	0.910	0.943	0.639	1	11.2% (101/898)
RF (all 10 features)	0.5	0.602	0.679	0.639	0.969	0.957	0.963	0.801	1	9.8% (88/898)
2nd LLM disagreement	n/a	0.046	0.051	0.048	0.909	0.899	0.904	0.476	2	9.7% (87/898)
Validator (ours)	**0.5**	**0.675**	**0.692**	**0.684**	**0.971**	**0.968**	**0.969**	**0.827**	**1**	**8.9% (80/898)**

of 0.143, flagging just 15 of 898 instances and missing most LLM errors—indicating that the model's confidence is poorly calibrated for this task.

RF on Confidence + Prediction. To test whether a learned decision boundary helps, we trained a Random Forest on the confidence score and binary prediction. This achieved a Macro F1 of 0.64 at $\tau = 0.5$, flagging 101 of 898 instances—an improvement over simple thresholding, but with weak Incorrect-class performance (F1 = 0.34). The model tends to overfit on the binary prediction: because the prediction is correct most of the time, the Random Forest relies on it as a shortcut rather than learning the conversational patterns that characterize genuinely difficult cases.

RF on All 10 Features. Adding the eight event category scores to the confidence and prediction (10 features total) tests whether these direct LLM outputs carry complementary information. This model reaches Macro F1 of 0.80 at $\tau = 0.5$, flagging 88 instances—but still underperforms our validator (Macro F1 = 0.83, 80 instances). The same overfitting dynamic persists: the Random Forest assigns disproportionate importance to the prediction feature, which acts as a proxy for the majority class and crowds out the subtler category-based signals most informative for detecting annotation difficulty.

Second LLM Disagreement. We also tested flagging conversations where two LLMs disagree. We ran Qwen (*Qwen3-14b*) [67], an instruction-tuned model with a different architecture, on the same 898 conversations using the same prompt. Despite doubling the inference cost, this baseline performed worst overall (Macro F1 = 0.476, Incorrect F1 = 0.048): although Qwen disagreed with Ministral on 87 conversations, this disagreement failed to identify most incorrect annotations.

Table 3 confirms that raw confidence is a weak signal: the LLM tends to express high confidence even when wrong, likely because toxic and non-toxic software engineering conversations share surface features (e.g., strong language about code). Including the prediction and confidence alongside the category scores does not help—these features dilute the category-based signal because the classifier overfits on the prediction, which is correct roughly 90% of the time and dominates the learned splits. Using the eight category scores alone yields the strongest results (Macro F1 = 0.827, Incorrect Recall = 0.692), flagging nearly five times as many LLM errors as the confidence threshold while requiring no additional inference cost.

2.2.5 Validator Independence. The performance drop when prediction and confidence are included with the event scores (Macro F1 = 0.801) compared to using the event scores alone (Macro F1 = 0.827) suggests that the event scores capture information about annotation difficulty beyond the LLM's direct toxicity judgment. Although both outputs come from the same model, they answer structurally different questions. The prediction provides a direct binary judgment of toxicity, whereas the event scores decompose the conversation into dimensions that can make that judgment difficult, such as strong process criticism paired with resolution language. The feature importance analysis in Section 2.2.3 supports this interpretation: the validator relies primarily on indirect signals of tension, especially escalation and subtle criticism, rather than on overt hostility features most aligned with the toxicity label. This indicates that the validator succeeds by learning which conversational patterns make labeling hard, not simply by reproducing the original prediction.

2.2.6 Pipeline Annotation Cost. A key advantage of our pipeline is its low annotation cost. Each conversation requires exactly one LLM call that produces the toxicity prediction and all eight event category scores in a single structured output. The Random Forest validator adds negligible overhead—training takes under one second and scoring the full 124,757-conversation corpus takes only seconds on commodity hardware. Human effort is concentrated on the small fraction of conversations the validator flags; on our large-scale corpus, this amounted to roughly 1.5% of all conversations (Section 3). As the *Human Review* column in Table 3 shows, our validator achieves the best Macro F1 while keeping human annotation effort minimal—the same single-call inference budget as the confidence threshold, but with substantially better error detection.

2.3 Generalization to External Dataset

To evaluate generalization, we applied our pipeline to the dataset of Raman et al. [52], which has no overlap with Imran et al.'s data (different sampling strategy and time period, 2012-2018). Raman et al. targeted GitHub issues locked as "too heated" or containing maintainer reactions referencing "attitude." We used the 314 instances (71 toxic, 243 non-toxic) available at the comment level [9].

On these 314 conversations, Ministral predicted 297 correctly, yielding a weighted F1 of 0.96. Of the 17 incorrect predictions, the validator flagged 10 for review. The remaining 7 undetected errors—3 of which involved toxic conversations—all exhibited subjective or borderline toxicity where annotator judgment could reasonably differ (e.g., [62], annotated as non-toxic, which is debatable). These findings suggest that the HITL methodology generalizes effectively across datasets with differing sampling and annotation criteria.

Table 4: Statistics of the GitHub conversations sample.

Metric	Value
Total Repositories	340
Pull Request Conversations	39,891
Issue Conversations	84,866
Total Sampled Conversations	124,757
Median Comments per Conv.	3
Median Conversation Length	41 words
Maximum Conversation Length	29,078 words
Time Frame Start	2024-01-01 00:54:06 UTC
Time Frame End	2025-01-31 23:47:57 UTC
Total Days	396

3 Large-Scale Data Collection and Annotation

Having established and evaluated the annotation pipeline, we now apply it to a newly collected large-scale GitHub corpus. This section describes the construction of the 2024 dataset used in the remainder of the paper: repository sampling and conversation collection, application of the first-pass LLM annotator and validator, and production of the final labeled dataset.

3.1 Large-Scale Corpus Collection

To construct a broad and realistic sample of GitHub discussions, we collected conversations from active, collaborative, and software-relevant repositories. Our sampling strategy follows the guidance of Kalliamvakou et al., who argue that empirical studies on GitHub should prioritize repositories with meaningful collaborative activity while avoiding personal, inactive, or non-software projects [32].

We began by selecting repositories with at least 1,000 stars as a signal of community interest, at least 100 total issues (open or closed), and at least one commit within the 10 months preceding our collection window starting 1 January 2024. We further required that each repository have either a locked conversation or an explicit code of conduct, since these suggest active moderation: locked conversations indicate maintainer intervention in heated discussions, while a code of conduct reflects an explicit attempt to define acceptable behavior. The first author then manually reviewed the candidate list and excluded clearly non-software repositories, such as programming problem collections and curated lists.

Using the GitHub GraphQL API [22], we collected all public English-language issue and pull request conversations created from January 2024 through January 2025 from each of the 340 selected repositories. We retained only conversations with at least one comment beyond the initial description (i.e., at least two total comments), ensuring the corpus captures actual interaction rather than single-post reports.

Table 4 summarizes the resulting corpus. We collected 124,757 conversations from 340 repositories, comprising 39,891 pull request and 84,866 issue discussions. Conversation length varies substantially, from very short exchanges to threads of up to 29,078 words, with a median of 3 comments and 41 words.

The 124,757 conversations involve 100,907 unique participants. For 10,247 of them, author association data was not available. Of the remaining 90,660, 88,674 carry exactly one role. Among them, 76,466 appear as None (external, unaffiliated participants), 10,550 as Contributors, 1,338 as Members, 289 as Collaborators, and 31 as Owners. The remaining 1,986 carry multiple roles, reflecting that GitHub author associations are repository-specific and a participant's role may differ across repositories. For example, an Owner of one repository may appear as None when commenting in another.

3.2 Applying the HITL Annotation Pipeline

We applied the pipeline described in Section 2 to the collected corpus. Because the validator was trained on Imran et al.'s dataset, which was sampled from GitHub prior to our 2024 collection window, there is no overlap between the training data and the corpus annotated here. Each conversation was processed by the Ministral-based annotator, producing a binary toxicity prediction along with event category scores.

We then ran the Random Forest validator on the LLM-labeled corpus using the threshold selected in Section 2.2 ($\tau = 0.5$). The validator flagged 1,874 conversations (approximately 1.5% of the corpus) for manual review.

Two of the authors, one a Ph.D. holder in software engineering and one a Ph.D. candidate, both with prior experience studying toxicity in GitHub conversations, manually reviewed the 1,874 flagged conversations. They used toxicity definitions, category taxonomies, and examples from prior GitHub toxicity studies (Miller et al. [39], Ferreira et al. [16], Imran et al. [30], and Raman et al. [52]) to calibrate their judgments. For each flagged conversation, the assigned annotator reviewed the discussion together with the Ministral prediction and explanation, assigned a binary toxicity label, and recorded a brief justification. The manual review required approximately 16 person hours of effort.

To assess consistency between the two annotators, we conducted an inter-annotator agreement study on a shared subset of 200 flagged conversations, drawn from the 1,874, that both annotators reviewed independently. The resulting Cohen's κ was 0.78, indicating substantial agreement. Disagreements were resolved through discussion, justifying the division of the remaining 1,674 flagged conversations equally between the two annotators for individual review.

Manual review led to 169 label changes. Ministral had originally classified 805/124,757 conversations as toxic and 123,952/124,757 as non-toxic. The corrections were predominantly in one direction: 155 conversations were relabeled from non-toxic to toxic, while 14 were changed from toxic to non-toxic, indicating that the LLM's primary failure mode among flagged conversations was under-detection of toxicity. The two annotators made 78 and 91 corrections respectively, suggesting that the flagged cases were distributed relatively evenly across the two subsets.

3.3 Validation on Unflagged Conversations

The validator is designed to flag conversations where the LLM is likely to have made an error, but it may also miss some errors among the unflagged majority. To estimate the rate of such missed errors, we drew a stratified random sample of 5,000 conversations, comprising 4,800 conversations predicted non-toxic by the model and 200 predicted toxic, from the 122,883 conversations that the validator did *not* flag for review.

Two annotators independently reviewed an initial subset of 500 sampled conversations using the same annotation protocol applied to the flagged set. They achieved a κ of 0.978 on this subset, then

resolved disagreements through discussion and reached full agreement. The remaining conversations were then divided between the annotators for individual review. For each conversation, the annotator determined whether the model's original toxicity label was correct.

Across the 5,000 unflagged conversations, the reviewers found 11 labeling errors in total. Among the 4,800 conversations predicted non-toxic by the LLM, 4 were actually toxic, yielding an LLM false-negative rate of 0.083% with a 95% confidence interval of 0.023% to 0.213%. Among the 200 conversations predicted toxic by the LLM, 7 were actually non-toxic, yielding an LLM false-positive rate of 3.5% in the predicted-toxic stratum, with a 95% confidence interval of 1.42% to 7.07%. Overall, this suggests that the validator misses very few toxic conversations among the unflagged majority. While the point estimate of the false-positive rate is 3.5%, the wide confidence interval (1.42% to 7.07%) indicates that this estimate should be interpreted with caution, as it is based on a small number of observed misclassifications (n = 7). Given the extreme rarity of toxic content in GitHub conversations and the resulting class imbalance, these results suggest that the validator performs reasonably well on the unflagged set.

In 3 of the 4 false-negative cases (toxic threads mislabeled as non-toxic), the conversation involved a disagreement in tone around an issue or PR that was closed without resolution. The 7 false-positive cases (non-toxic threads mislabeled as toxic) were more varied: three had comments where suggestions were discarded, two involved a user expressing impatience over an unresolved bug, one contained spam comments, and one involved name-calling.

Extrapolating the observed error rates from the manual recheck to the full set of unflagged conversations yields an estimated 129 additional labeling errors in the corpus beyond those corrected through the validator, for an estimated overall labeling accuracy of 99.90%. These results indicate that the validator successfully concentrates the majority of LLM errors into the flagged set, with a residual error rate of 0.105% among unflagged conversations.

As an additional baseline, we evaluated ToxiShield, a recent SE-specific toxicity detector whose BERT-based module outperformed prior techniques [2]. We trained the detection module with the authors' public code and dataset, using BERT as the backbone and following their original setup. We then applied the trained model to our manually verified sample of 5,000 unflagged GitHub conversations. Because our complete pipeline includes targeted human correction, it is not directly comparable to fully automated toxicity detectors. We therefore compare ToxiShield with the automated portion of our pipeline, before human correction, as a positioning baseline rather than a full detector benchmark.

ToxiShield achieved a toxic-class F1 of 0.148, with macro, micro, and weighted F1 scores of 0.531, 0.843, and 0.803, respectively. In contrast, the automated portion of our HITL pipeline, consisting of Ministral annotation followed by validator-based filtering with no human correction, achieved a toxic-class F1 of 0.972 and macro, micro, and weighted F1 scores of 0.986, 0.998, and 0.998. ToxiShield tended to overpredict toxicity, often flagging isolated frustration or negative wording without considering the surrounding discussion context. This pattern is consistent with its reported error modes, including context misreads, self-deprecating humor, and technical jargon or inline code interpreted as hostile [2].

3.4 Final Dataset

After manual correction, the final dataset contains 124,757 conversations: 946 labeled toxic and 123,811 labeled non-toxic. This corpus serves as the basis for the empirical analyses in the remainder of the paper.

The large-scale deployment confirms the practical value of the HITL pipeline: the first-pass annotator provided efficient coverage of the full corpus, the validator reduced the review burden to roughly 1.5% of conversations, and targeted manual review corrected the residual errors.

4 Validating Prior Findings at Scale

In this section, we use our large-scale toxicity dataset to test hypotheses introduced in prior work on toxicity in software engineering and online discussions. We selected these papers because they present empirically grounded, testable claims about toxicity dynamics, many of which were based on smaller datasets or limited contexts. Raman et al. [52] and Miller et al. [39] analyzed toxicity trends on GitHub. Imran et al. [30] and Sarker et al. [55] introduced annotated datasets of toxic GitHub discussions. Lastly, Saveski et al. [57] provided structural insights from toxicity on X (Twitter), which we treat as transferable to GitHub's threaded discussions.

4.1 Participants of Toxic Conversations

Prior work by Miller et al. [39] and Imran et al. [30] found that external participants (e.g., users who are not developers of the project), rather than active project contributors, play a significant role in driving toxicity in GitHub issue/PR discussions. Imran et al., using a dataset of 202 toxic and 696 non-toxic threads, found that 76.0% of toxic conversations were initiated by external participants and made 52.79% of the comments in toxic threads. In contrast, non-toxic discussions had a higher engagement from project contributors (66.62%), indicating their role in fostering constructive discussions.

GitHub assigns roles to participants in issue/PR discussions based on their involvement level within a repository. We used the following five categories of GitHub participants that we found in our corpus: None, Member, Contributor, Collaborator, and Owner [10]. Participants labeled as "None" have no affiliation with the project, while Members belong to the owning organization. Contributors are people who have previously committed code, Collaborators are invited contributors, and Owners have administrative control. Following Imran et al.'s approach, we grouped *Contributor, Collaborator, Member,* and *Owner* as project contributors, and *None* participants as external participants, i.e., unaffiliated with the repository.

We analyzed our dataset of 124,757 GitHub issue/PR conversations and categorized participants by their level of involvement in a repository. Overall, external participants contributed 44.48% of all comments, while project contributors accounted for 55.52%. In toxic conversations, however, this distribution shifts: external participants contributed 57.56% of comments, while project contributors contributed only 42.44%. When examining toxicity levels by the proportion of project contributor comments in a discussion, we found a non-monotonic pattern (Table 5). Toxicity initially rises as contributor participation increases, peaking at 1.12% in the 40%–60% range, before declining at higher levels: 0.81% at 60%–80%, 0.71%

Table 5: Toxic conversation rates across discussions with varying levels of project contributor participation.

Comments by Project Contributors (%)	Toxic Conversations (%)
0%–20%	0.95% (203/21339)
20%–40%	1.02% (194/19002)
40%–60%	1.12% (370/32911)
60%–80%	0.81% (79/9712)
80%–99%	0.71% (16/2256)
100%	0.21% (84/39537)

at 80%–99%, and reaching its lowest rate of 0.21% when project contributors made 100% of the comments. This inverted-U pattern suggests that discussions with moderate contributor presence, i.e., where external and internal participants interact most, may be particularly prone to friction, while discussions dominated by project contributors are associated with substantially less toxicity.

Observation #1

Our findings partially support Miller et al. and Imran et al.'s observation that greater project contributor engagement correlates with lower toxicity. External participants made 44.48% of all comments, but 57.56% in toxic conversations. However, the relationship is non-monotonic: toxicity peaks at moderate contributor participation (40%–60%) before declining at higher levels, suggesting that mixed-participation discussions may introduce friction, while predominantly contributor-driven discussions are the most civil.

4.2 Toxicity and Mention Graph

Saveski et al. [57] examined the relationship between conversation structure, specifically reply trees, and toxicity in discussions on X (formerly Twitter). They represented conversations as reply trees, where the root node was the initial tweet that initiated the discussion, and the subsequent nodes were replies. They hypothesized that larger reply trees tend to be more toxic and found a positive correlation between tree size and toxicity. They measured toxicity as the fraction of toxic tweets within a conversation and analyzed how toxicity levels varied with conversation size, defined as the total number of tweets. Their findings suggested that as reply structures expanded, discussions became increasingly toxic.

GitHub issue and PR discussions do not support threaded replies in the way that X does. To evaluate whether a similar relationship between conversation depth and toxicity exists on GitHub, we constructed mention graphs for each conversation in our dataset. In our approach, each unique commenter is represented as a node, with a directed edge from commenter A to commenter B if A explicitly mentions B using the "@B" syntax in a comment. The longest path length in a mention graph is defined as the deepest chain of mentions within a conversation. Unlike reply trees, which capture hierarchical reply structures based on direct responses, mention graphs represent explicit participant references within discussions, independent of the reply structure. While most toxic conversations appear in discussions with shallow mention depth, likely because they are much more common overall, the rate of toxicity increases with greater depth. As shown in Table 6, toxicity increases with mention depth: it remains below 1.05% for path lengths 0–2, rises to 2.46% at depth 3, and further increases to 3.11% for deeper chains (≥ 4).

Table 6: Toxic conversations by depth of mention chains.

Deepest Path Length	Toxic Conversations (%)
0	0.71% (611/86179)
1	0.77% (241/31478)
2	1.05% (61/5827)
3	2.46% (25/1016)
≥ 4	3.11% (8/257)

Observation #2

Our findings support the general observation that deeper mention chains tend to be more toxic. Although such deep chains are relatively rare, conversations with longer mention paths (≥ 4) exhibit higher toxicity rates (3.11%) compared to shallow discussions. This suggests that increasingly complex interaction patterns are more prone to toxic behavior, consistent with prior research.

4.3 Prevalence and Trends of Toxic Conversations Over Time

Previous work by Raman et al. found that toxic conversations are rare in GitHub discussions, occurring in about 0.6% of cases [52]. They conducted a longitudinal study of 1,732,124 GitHub conversations collected from 2012 to 2018. They initially manually annotated a small subset of 611 issues (of which 167 were toxic threads) and trained a machine learning classifier to detect toxicity in GitHub. After applying the classifier to the larger dataset, they concluded that toxic conversations are infrequent. Their analysis also found that toxicity had a declining trend over time.

In contrast, we use a HITL-constructed dataset drawn from a more recent corpus of GitHub issue and PR discussions. In our dataset, 946 conversations (0.76%) were toxic, slightly higher than the 0.6% reported by Raman et al. However, this comparison should be interpreted with caution: the two studies use different detection methodologies (HITL with LLM annotation vs. a trained ML classifier), different sampling strategies, and different repository pools, so the difference in rates may reflect methodological variation rather than a true temporal shift. That said, a recent 2024 GitHub survey offers suggestive—though not directly comparable—corroboration, noting that 64.23% of developers either experienced or witnessed negative interactions, up from 60% in 2017 [23, 72]. Together, these data points leave open the possibility that toxicity has not declined as Raman et al. observed in their earlier period, though stronger longitudinal evidence is needed to draw firm conclusions.

To further examine recent trends in toxicity, we conducted a month-by-month analysis of GitHub conversations from January 2024 through January 2025. In Figure 1, we show the monthly toxicity frequency measurements, calculated as the proportion of toxic conversations each month. As shown in Figure 1, toxicity remains consistently low but exhibits mild fluctuations over time. The toxicity rate ranges from approximately 0.62% to 1.08%, with most months clustered between 0.7% and 0.8%. Notably, a slight increase is observed toward the end of 2024, with peaks around October (1.08%), December (1.02%), and January 2025 (1.05%), suggesting a modest upward trend rather than complete stability.

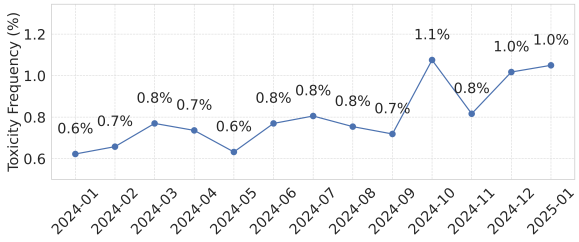


Figure 1: Monthly percentage of toxic conversations in GitHub discussions (Jan. 2024 – Jan. 2025).

Observation #3

Our findings remain consistent with Raman et al., who reported that toxic conversations are relatively rare in GitHub discussions [52]. Our observed average toxicity rate of approximately 0.76% is higher than their reported 0.6%, though as noted above, this difference may partly reflect methodological variation rather than a true temporal shift. The monthly trend indicates mild temporal variation, with a small upward shift toward the end of the observed period.

4.4 Relation Between Toxicity and Number of Comments

Previous work by Imran et al. [30] found that toxic conversations on GitHub tend to be longer, with a median of 11 comments, compared to 6 in non-toxic discussions (out of 202 toxic and 696 non-toxic threads). They found that toxicity often appeared within the first 10 comments and that over half of toxic discussions contained multiple toxic comments. This suggests that toxicity arises more often in longer discussions and contributes to prolonging them.

To assess whether this trend holds in our dataset, we analyzed 124,757 GitHub issue/PR conversations, comparing the 946 toxic ones against 123,811 non-toxic ones. At scale, the association between toxicity and conversation length is considerably weaker than previously reported. The median number of comments in toxic conversations was 4, compared to 3 in non-toxic ones—a gap of just one comment, far smaller than the 11 vs. 6 reported by Imran et al. The distributional difference is somewhat more visible at the 75th percentile (8 comments for toxic vs. 5 for non-toxic), and toxic conversations exhibited greater variability in length (SD of 7.64 vs. 4). Still, the overall effect is modest, suggesting that while toxicity and conversation length are associated, the relationship is weaker than small-scale studies indicated.

Observation #4

At scale, the association between toxicity and conversation length is much weaker than Imran et al. reported (median of 4 vs. 3, compared to their 11 vs. 6). The directional trend holds, but it suggests that conversation length only modestly correlates with toxicity rather than strongly.

4.5 Pull Request Acceptance and Toxicity in Discussions

Previous work by Sarker et al. examined the relationship between pull request (PR) acceptance and toxicity in OSS development [55].

Table 7: Toxicity rates by programming language in languages with ≥10,000 issues and ≥20 projects.

Primary Language	Repo Count	Issues Count	Toxic Issues Count	Toxicity Ratio (%)
Python	34	12,407	132	1.06
C++	20	15,329	125	0.82
Go	30	12,189	93	0.76
JavaScript	37	13,852	90	0.65
TypeScript	74	28,862	166	0.58

They analyzed a sampled subset from a collection of 6.3 million PRs from 1,155 projects. Their study found that accepted PRs are less likely to encounter toxicity, while rejected PRs tend to have a higher likelihood of toxic interactions. Specifically, they found that 2.47% of accepted PRs contained at least one toxic comment, compared to 3.26% of rejected PRs.

To evaluate whether a similar relationship holds in our dataset, we analyzed 39,891 PR conversations and measured toxicity across accepted and non-accepted PRs. Our findings align with the hypothesis that accepted PRs are less toxic. Of the 24,656 accepted PRs, 57 (0.23%) contained toxic comments, whereas 155 (1.19%) of the 12,999 non-accepted PRs were toxic – over five times higher.

From a toxic PR perspective, we found that 25.56% of all toxic PRs were accepted, whereas 62.01% of all non-toxic PRs were accepted. This reinforces the idea that toxic interactions are more prevalent in rejected PRs, while accepted PRs experience low toxicity.

Observation #5

Our findings support the hypothesis previously observed that accepted PRs are less likely to encounter toxicity. In our corpus, toxicity rates in non-accepted PRs (1.19%) were higher than accepted PRs (0.23%). Additionally, toxic PRs had a lower acceptance rate (25.56%) compared to non-toxic ones (62.01%).

5 Novel Findings on Toxicity Dynamics

Our large-scale dataset provides new opportunities to study the relationship between toxicity and project and contributor dynamics on GitHub. We describe each finding below.

5.1 Programming Language and Toxicity Rates

While programming languages are not themselves the source of toxicity, they often correlate with broader community norms and dynamics. Different programming language ecosystems attract distinct profiles, project domains, and cultural norms [3, 59].

We examined toxicity rates in repositories grouped by their primary programming language as provided through GitHub REST API. We limited our analysis to languages represented by at least 20 repositories, each with a minimum of 10,000 issues and PRs, to ensure representativeness and statistical reliability. Table 7 summarizes the results. Python and C++ repositories exhibit comparatively higher toxicity rates (1.06% and 0.82%, respectively). Go and JavaScript show moderate levels of toxicity (0.76% and 0.65%), while TypeScript has the lowest observed toxicity rate at 0.58%.

To assess whether the observed differences across ecosystems are statistically reliable, we applied a Kruskal-Wallis test on per-repository toxicity rates within each language group. The test yielded $H = 9.004$, $p = 0.061$, marginally non-significant at $\alpha = 0.05$.

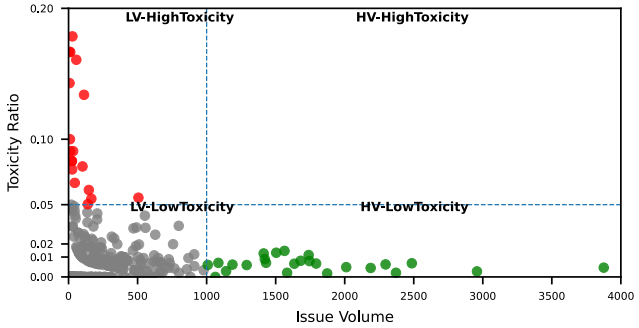


Figure 2: Repository groups by issue activity and proportion of toxic discussions.

The ordering of ecosystems (Python > C++ > Go > JavaScript > TypeScript) remains consistent with the aggregate rates in Table 7.

Observation #6

Toxicity levels vary across programming language ecosystems. Python and C++ repositories show relatively higher toxicity rates, Go and JavaScript fall in a moderate range, and TypeScript exhibits the lowest toxicity. However, the differences do not reach statistical significance ($p = 0.061$).

5.2 Toxicity and Project Governance

Community smells in open source software projects often reflect structural or social dysfunctions that hinder collaboration and inclusivity. Effects such as the "Prima Donna" and "Lone Wolf" are commonly associated with weak or poorly defined governance structures [1, 3, 60, 61]. Prior work also suggests that stronger governance, including active moderation [27], codes of conduct [36], and organizational support [55], can reduce harmful interactions. Motivated by this literature, we examine how governance-related characteristics align with toxicity in our dataset.

We partition repositories along two interpretable dimensions: new issue volume during the study period and the proportion of toxic issues. We set fixed thresholds on these axes by manually inspecting the empirical distribution rather than applying learned clustering or automatic optimization, assigning repositories to four quadrants. This rule-based grouping preserves interpretability and makes categories easy to examine substantively. We retain all repositories, since highly toxic cases are central to the analysis rather than measurement noise. Figure 2 shows the full dataset partitioned by issue volume and toxicity ratio.

Figure 2 reveals three populated quadrants. The largest, Low Volume-Low Toxicity (LV-LT), contains 292 repositories with modest activity and low toxicity. High Volume-Low Toxicity (HV-LT) contains 27 repositories with high issue activity but still low toxicity. Low Volume-High Toxicity (LV-HT) contains 21 repositories with lower activity but substantially higher toxicity. No repositories fall into the High Volume-High Toxicity quadrant, suggesting that sustained toxicity is uncommon among highly active repositories in our sample.

To better understand elevated toxicity in LV-HT, we examine governance through administrator participation in toxic discussions. Since maintainer-level data are unavailable via the GitHub REST API, we proxy administrators using users with write or higher access (Owners, Members, and Collaborators), excluding Contributors, who lack write, push, or merge permissions.

Clear differences emerge across groups. HV-LT repositories have more administrators participation, with a median of 23, compared with 4 in LV-LT and 1 in LV-HT. This suggest that LV-HT repositories are typically managed by smaller and less active core teams We also examine repository Health Percentage, which reflects the presence of elements such as a code of conduct, contributing guidelines, a license, and issue and pull request templates. LV-HT has the lowest median health percentage at 75.0%, compared with 87.0% for HV-LT and 85.0% for LV-LT.

Ownership structure provides further context. In HV-LT, 100% of repositories are organization-owned. In LV-LT, 87.4% are organization-owned and 12.6% are user-owned. LV-HT has the lowest share of organization-owned repositories at 66.7%, with 33.3% owned by individual users.

Although we do not directly measure governance, these groups appear to reflect distinct governance styles. HV-LT repositories resemble structured, team-based governance models, such as Apache-style projects [41]. In contrast, LV-HT repositories align more closely with centralized models led by a small core of maintainers [20]. LV-LT may reflect lightweight or emerging governance structures [44].

Observation #7

Repositories with less structured governance tend to exhibit higher toxicity. In the Low Issue Volume–High Toxicity (LV-HT) group, the median number of admins was 1 (vs. 23 in HV-LT) and the median GitHub Health Score was 75.0% (vs. 87.0%). Together, this suggests that GitHub projects with more structured governance and larger teams tend to experience lower levels of conversational toxicity.

5.3 Behavior of Highly Active GitHub Users

To understand the behavior of highly active GitHub users in toxic discussions, we analyzed individuals who had participated in at least 10 toxic GitHub issues in our dataset. There were 16 such users, each appearing in 10–24 toxic conversations. We examined their repository affiliations, administrative roles (i.e., users with write or higher repository access), and dual roles as both participants and moderators in contentious discussions.

All 16 were either repository administrators (15/16) or contributors (1/16) to at least one repository, and most were core project members. On average, they participated in 10,005 issues (median of 388), with the most active user involved in 3,734 issues. Together, they appeared in 236 toxic threads (24.95%, 236/946) and authored 540 comments across those conversations. Two authors independently annotated these comments to assess: (i) whether users authored toxic comments; and (ii) whether they attempted moderation in response to toxicity.

The most common pattern was the overlap between writing toxic comments and issuing moderation responses. All 15 administrators did both, often with moderation that was itself toxic in tone. This challenges the assumption that administrators take "appropriate and fair corrective action" [11, 53], suggesting instead that authority

figures engage in the same behaviors they are expected to moderate. The one non-administrator also authored toxic comments.

Qualitative analysis showed that toxic and moderating behaviors were often intertwined. Administrators sometimes responded to user queries with hostility or sarcasm framed as moderation. For example, one maintainer replied to issue spam with, *"Stop doing this. You are repeating creating this. If you keep doing this, I have to ask the [NAME] to ban you. Last warning!"* Another closed a pull request stating, *"Closing this PR as I don't enjoy being a guinea pig,"* citing template violations. In another case, a moderator dismissed a user's parser concerns with, *"Have you ever written a parser of anything in your life? [...] That alone tells me all I need to know about your level of competence. Good luck to you."* These examples show that moderation can take a punitive, condescending, or dismissive tone. In some cases, it escalated hostility rather than reducing it. One particularly aggressive comment stated, *"Shut up. Go install Windows 10 instead of tirelessly begging for features... Get a life."* Another dismissive comment read, *"[...] Nothing much of rocket science here. [...]"* Language like this, especially when paired with actions and threats such as locking conversations or blocking users, does not reflect the communication standards typically expected in responsible OSS governance.

Observation #8

A focused analysis of 16 highly active GitHub users (15 administrators, 1 contributor) shows that all authored both toxic and moderation comments. This challenges the view of administrators as fair corrective enforcers and suggests leadership can itself be a source of toxicity.

6 Effect Sizes and the Impact of Manual Review

To assess the downstream impact of manual review, we computed effect sizes for five quantitative estimates derived from Observations #1 and #3–#5 under two label sets, the manually corrected labels (946 toxic conversations) and the LLM-only labels (805 toxic conversations). Manual review changed 169 labels in total, relabeling 155 conversations from non-toxic to toxic and 14 from toxic to non-toxic, for a net increase of 141 toxic conversations. We omit the other observations from this analysis for various reasons. For Observation #2, the toxicity rates by mention depth in Table 6 already convey the size of the effect. Observations #6 and #7 are assessed at the repository level in Section 5, and Observation #8 describes a qualitative pattern among named individuals.

We use Cliff's δ for two-group comparisons, Spearman ρ for rank-based correlations, and Cramér's V for categorical associations. Cliff's δ ranges from −1 to +1 and expresses how often values in toxic conversations exceed those in non-toxic ones. Positive values indicate that toxic conversations score higher on the measure, and V measures association strength on a 0–1 scale. Following conventional thresholds, $|\delta|$ values of roughly 0.15, 0.33, and 0.47 mark the onset of small, medium, and large effects. For ρ and V the corresponding values are 0.1, 0.3, and 0.5. We also include the overall toxicity prevalence (#3a), a descriptive proportion rather than an association measure. The trend estimate (#3b) is computed over monthly rates and all other estimates over conversations. Table 8 reports each effect size under the corrected labels, together with its change relative to the LLM-only labels, computed as corrected minus LLM-only.

Table 8: Effect sizes for five quantitative estimates derived from Observations #1 and #3–#5, computed on the manually corrected labels (946 toxic conversations). Change is the corrected-label estimate minus the LLM-only estimate (805 toxic conversations); pp = percentage points.

Obs.	Finding	Metric	Effect Size	Change
#1	Developer engagement	Cliff's δ	−0.232	+0.020
#3a	Toxicity prevalence	Proportion	0.76%	+0.11 pp
#3b	Toxicity trend	Spearman ρ	+0.775	−0.050
#4	Conversation length	Cliff's δ	+0.195	+0.118
#5	PR acceptance	Cramér's V	0.061	+0.004

Most effects are small, as expected given that toxicity is rare and no single factor drives it. The strongest conversation-level association is the participant-type finding (Observation #1, Cliff's $\delta = -0.232$, a small effect). Among non-tied random toxic/non-toxic pairs, the non-toxic conversation has higher developer participation ~62% of the time. The toxicity trend (Observation #3b, Spearman $\rho = +0.775$) is large by these thresholds. Rank correlation captures how consistently the monthly rates rise rather than how steeply, so this estimate is compatible with the modest upward shift described in Section 4.3. It also rests on only 13 monthly data points and should be interpreted with caution. For PR acceptance (#5), non-accepted PRs are 5 times more likely to be toxic, yet Cramér's $V = 0.061$ falls below the small-effect threshold. This occurs because toxicity remains rare in both groups, so a large relative risk corresponds to a small absolute association.

Manual review materially changed two estimates. For the prevalence estimate (#3a), both label sets are scored on the same 124,757 conversations, so McNemar's test is the appropriate comparison. The corrections were strongly asymmetric, with 155 conversations gained and 14 lost, yielding a matched-pairs odds ratio of 11.07 (95% CI (confidence interval) of $[6.41, 19.13]$, $p < 0.001$). The interval lies entirely above 1, indicating a systematic shift. For the conversation-length estimate (#4), the effect is 2.5× larger after manual review (Cliff's δ rises from 0.077 to 0.195), crossing from below to above the small-effect threshold. The reason is that the conversations the LLM missed came disproportionately from longer discussions. Without manual review, we would have reported a substantially weaker association between conversation length and toxicity. The remaining three estimates shift only slightly under relabeling, so their interpretations are unchanged.

7 Related Work

Our work builds upon prior work in two main areas: toxicity analysis in OSS development and HITL approaches in data annotation using a LLM. Below we discuss these two key areas.

7.1 Toxicity in OSS

Toxicity-free OSS has been a long-standing goal of the community [8, 18, 19]. The study of toxicity and negative interactions in OSS ecosystems has received significant attention in recent years. Researchers have explored various forms of negative engagement across multiple communication channels, including issue trackers, pull requests, chat discussions, and code reviews. Early research

predominantly focused on sentiment analysis in OSS communication, identifying positive and negative expressions. Over time, the focus has expanded towards more nuanced aspects, such as granular-level emotions [28, 29, 43, 45], incivility [13, 15–17, 50], and toxicity [12, 31, 39, 52, 54–56]. Prior work also shows that LLMs can miss subtle emotional dynamics in software engineering contexts, such as in pair programming conversations [51].

Researchers have continued to explore this area further and investigate even more nuanced aspects. Li et al. found that codes of conduct are used to govern offensive forms of speech [36]. Qiu et al. noted interpersonal conflict in code reviews [49]. Qiu et al. later proposed a dashboard that overviews community health, that includes toxicity score of the comments [48]. Trinkenreich et al. conducted a study on understanding the link between the motivation of OSS developers and the sense of belonging to the community [63]. Rahman et al.'s survey of 171 developers revealed that half of the developers (56%) have encountered workplace uncivil discourse and the majority of this group (83.70%) experiences it at least once a month [50]. Gunawardena et al. observed that destructive code reviews have a negative impact [24]. Ferreira et al. noted inappropriate feedback in the Linux kernel mailing list [16]. Imran et al. experimented on proactive toxicity detection in GitHub conversations [30]. In a 2020 study, Raman et al. conducted a longitudinal study on finding toxicity trends [52]. In a recent study, Sarker et al. conducted a large-scale study on GitHub toxicity using their ToxiCR tool as a toxicity identifier [55]. Unlike those studies, we proposed a novel human-in-the-loop approach to annotate toxicity in GitHub and attempted to scale up previous observations on toxicity in OSS.

7.2 Human-in-the-Loop Data Annotation Strategies

The human-in-the-loop paradigm has emerged as a crucial methodology for improving the quality and reliability of data annotation for machine learning tasks. Several studies have demonstrated the effectiveness of combining human expertise with automated systems to create more accurate and contextually appropriate labeled datasets [26, 42, 64–66, 68]. With the advancement of LLMs in recent years, the integration of LLMs into HITL pipelines has been widely attempted [5, 34, 46, 64–66]. Studies show that ChatGPT can outperform crowd workers and significantly reduce manual effort [21]. Some studies have found that integrating LLMs as preliminary annotators in HITL pipelines can increase annotation throughput while maintaining comparable quality to fully manual approaches [25, 33]. Unlike previous studies, ours is one of the first large-scale toxicity analyses on GitHub that leverages generative LLMs for toxicity annotation to conduct an empirical study.

8 Limitations

Construct Validity. Our reliance on LLM-generated toxicity annotations may introduce biases or misclassifications despite human verification; the HITL validation model mitigates this by routing likely errors to manual review. Our definition of toxicity follows prior research but may not capture all harmful interactions; we address this by incorporating a range of toxicity categories grounded in established literature. Finally, our approach considers only textual content, ignoring images, code snippets, or external links that might provide additional context. Toxicity annotation also involves inherent subjectivity; we mitigate this through a guideline-based protocol grounded in prior GitHub toxicity studies, annotators with domain expertise, and a Cohen's κ of 0.78 on the shared annotation pilot, indicating substantial agreement.

Internal Validity. The validation model's performance is constrained by the quality and representativeness of its training data (898 conversations); we mitigate this by evaluating across multiple classification thresholds and on an independent dataset. The training data also predates our 2024 corpus, raising a potential distribution-shift concern; we address this through two complementary checks: external validation on Raman et al.'s independently sampled dataset (weighted F1 = 0.96, Section 2.2) and a manual audit of 5,000 unflagged conversations from the 2024 corpus itself, which found only 11 errors (overall labeling accuracy 99.90%, Section 3.3). Our repository selection criteria (star count, activity level) may introduce selection bias, potentially overlooking toxicity patterns in smaller or less active projects.

External Validity. Our findings may not generalize beyond GitHub to platforms such as GitLab or Bitbucket, though we select repositories across diverse domains and programming languages to ensure broad coverage within GitHub. The dataset's time frame (January 2024 to January 2025) provides a snapshot rather than a longitudinal view; future work could assess temporal changes over longer periods. Differences in annotation methodologies between our study and prior work may also affect direct comparisons; further replication is needed to confirm the stability of these trends.

9 Conclusions and Future Work

Our large-scale investigation offers new insights into toxicity in GitHub discussions. Using a human-in-the-loop annotation pipeline, we analyzed over 124,757 conversations from 340 repositories, producing one of the most comprehensive toxicity datasets in OSS research to date.

We found that toxicity is slightly more prevalent (0.76%) than previously reported (0.6%). Our analysis validates several prior findings at scale: toxicity peaks at moderate contributor participation before declining at higher levels, increases in deeply threaded conversations, and is more common in longer discussions and rejected pull requests. We also highlight novel findings about the relationship between toxicity and project dynamics. We observed variation in toxicity across language ecosystems, with Python and C++ showing higher rates than TypeScript or JavaScript, though these differences did not reach statistical significance. Repositories with weaker governance, i.e., fewer admins, limited organizational backing, and lower health scores, exhibited substantially higher toxicity.

Future work will extend our study longitudinally to capture evolving patterns in toxicity, compare community-specific norms across GitHub ecosystems, and develop adaptive intervention mechanisms based on context and contributor history. We also plan to investigate how toxicity affects contributor retention and project sustainability.

Data Availability Statement

Our replication package is available on Zenodo at https://doi.org/10.5281/zenodo.21231251.

References

[1] Nuri Almarimi, Ali Ouni, Moataz Chouchen, Islem Saidani, and Mohamed Wiem Mkaouer. 2020. On the detection of community smells using genetic programming-based ensemble classifier chain. In *Proceedings of the 15th International Conference on Global Software Engineering*. 43–54.

[2] Md Awsaf Alam Anindya, Showvik Biswas, Anindya Iqbal, Jaydeb Sarker, and Amiangshu Bosu. 2026. ToxiShield: Promoting Inclusive Developer Communication through Real-Time Toxicity Filtering. *Proceedings of the ACM on Software Engineering* 3, FSE (2026), 2767–2789.

[3] Giusy Annunziata, Carmine Ferrara, Stefano Lambiase, Fabio Palomba, Gemma Catolino, Filomena Ferrucci, and Andrea De Lucia. 2024. An Empirical Study on the Relation Between Programming Languages and the Emergence of Community Smells. In *2024 50th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 268–275.

[4] Atijit Anuchitanukul, Julia Ive, and Lucia Specia. 2022. Revisiting Contextual Toxicity Detection in Conversations. *Journal of Data and Information Quality* 15, 1, Article 6 (Dec. 2022), 22 pages. https://doi.org/10.1145/3561390

[5] Ekaterina Artemova, Akim Tsvigun, Dominik Schlechtweg, Natalia Fedorova, Konstantin Chernyshev, Sergei Tilga, and Boris Obmoroshev. 2025. Hands-On Tutorial: Labeling with LLM and Human-in-the-Loop. *31st International Conference on Computational Linguistics (COLING)* (2025).

[6] Berk Atil, Rebecca J. Passonneau, and Ninareh Mehrabi. 2026. Robust Persona-Aware Toxicity Detection with Prompt Optimization and Learned Ensembling. arXiv:2601.02337 [cs.CL] https://arxiv.org/abs/2601.02337

[7] Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, et al. 2025. NVIDIA Nemotron 3: Efficient and Open Intelligence. *arXiv preprint arXiv:2512.20856* (2025).

[8] Kevin Daniel André Carillo, Josianne Marsan, and Bogdan Negoita. 2016. Towards developing a theory of toxicity in the context of free/open source software & peer production communities. *SIGOPEN 2016* (2016).

[9] CMUSTRUDEL. 2021. Raman et al. Dataset. https://github.com/CMUSTRUDEL/toxicity-detector/blob/master/data/training/train_comments.csv Accessed: 2025-07-15.

[10] GitHub Docs. [n. d.]. CommentAuthorAssociation Enum. https://docs.github.com/en/graphql/reference/enums#commentauthorassociation

[11] Coraline Ada Ehmke. 2017. Contributor Covenant: A Code of Conduct for Open Source Projects. https://www.contributor-covenant.org/version/1/4/code-of-conduct/. Accessed: 2025-07-19.

[12] Ramtin Ehsani, Preetha Chatterjee, et al. 2024. Analyzing Toxicity in Open Source Software Communications Using Psycholinguistics and Moral Foundations Theory. *NLBSE* (2024).

[13] Ramtin Ehsani, Mia Mohammad Imran, Robert Zita, Kostadin Damevski, and Preetha Chatterjee. 2024. Incivility in Open Source Projects: A Comprehensive Annotated Dataset of Locked GitHub Issue Threads. In *2024 IEEE/ACM 21st International Conference on Mining Software Repositories (MSR)*. IEEE, 515–519.

[14] Isabella Ferreira, Bram Adams, and Jinghui Cheng. 2022. How heated is it? understanding GitHub locked issues. In *Proceedings of the 19th International Conference on Mining Software Repositories (MSR '22)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3524842.3527957

[15] Isabella Ferreira, Bram Adams, and Jinghui Cheng. 2022. How Heated is it? Understanding GitHub Locked Issues. In *19th International Conference on Mining Software Repositories*.

[16] Isabella Ferreira, Jinghui Cheng, and Bram Adams. 2021. The" shut the f** k up" phenomenon: Characterizing incivility in open source code review discussions. *Proceedings of the ACM on Human-Computer Interaction* CSCW2 (2021).

[17] Isabella Ferreira, Ahlaam Rafiq, and Jinghui Cheng. 2024. Incivility detection in open source code review and issue discussions. *Journal of Systems and Software* (2024).

[18] Anna Filippova and Hichang Cho. 2015. Mudslinging and manners: Unpacking conflict in free and open source software. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. 1393–1403.

[19] Anna Filippova and Hichang Cho. 2016. The effects and antecedents of conflict in free and open source software development. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*.

[20] Karl Fogel. 2005. *Producing open source software: How to run a successful free software project.* " O'Reilly Media, Inc.".

[21] Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. 2023. ChatGPT outperforms crowd workers for text-annotation tasks. *Proceedings of the National Academy of Sciences of the United States of America* 120, 30 (2023), e2305016120.

[22] GitHub. 2025. GitHub GraphQL API. https://docs.github.com/en/graphql Accessed: 2025-02-09.

[23] GitHub, Inc., Kenyatta Forbes, Kevin Xu, Jeffrey Luszcz, Margaret Tucker, Eva Maxfield Brown, Peter Cihon, Mike Linksvayer, Ashley Wolf, Lukas Spieß, Kevin Crosby, and Jason Meridth. 2024. GitHub Open Source Survey 2024. https://doi.org/10.5281/zenodo.13989018

[24] Sanuri Dananja Gunawardena, Peter Devine, Isabelle Beaumont, Lola Piper Garden, Emerson Murphy-Hill, and Kelly Blincoe. 2022. Destructive criticism in software code review impacts inclusion. *Proceedings of the ACM on Human-Computer Interaction* 6, CSCW2 (2022), 1–29.

[25] Xingwei He, Zhenghao Lin, Yeyun Gong, A-Long Jin, Hang Zhang, Chen Lin, Jian Jiao, Siu Ming Yiu, Nan Duan, and Weizhu Chen. 2024. AnnoLLM: Making Large Language Models to Be Better Crowdsourced Annotators. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*. 165–190.

[26] Andreas Holzinger. 2016. Interactive machine learning for health informatics: when do we need the human-in-the-loop? *Brain informatics* 3, 2 (2016), 119–131.

[27] Jane Hsieh, Joselyn Kim, Laura Dabbish, and Haiyi Zhu. 2023. " Nip it in the Bud": Moderation Strategies in Open Source Software Projects and the Role of Bots. *Proceedings of the ACM on Human-Computer Interaction* 7, CSCW2 (2023), 1–29.

[28] Mia Mohammad Imran, Preetha Chatterjee, and Kostadin Damevski. 2024. Uncovering the Causes of Emotions in Software Developer Communication Using Zero-shot LLMs. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (Lisbon, Portugal) *(ICSE '24)*. Association for Computing Machinery, New York, NY, USA, Article 182, 13 pages. https://doi.org/10.1145/3597503.3639223

[29] Mia Mohammad Imran, Yashasvi Jain, Preetha Chatterjee, and Kostadin Damevski. 2022. Data augmentation for improving emotion recognition in software engineering communication. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 1–13.

[30] Mia Mohammad Imran, Robert Zita, Rahat Rizvi Rahman, Preetha Chatterjee, and Kostadin Damevski. 2026. Toxicity Ahead: Forecasting Conversational Derailment on GitHub. In *Proceedings of the 48th IEEE/ACM International Conference on Software Engineering (ICSE '26)*. Association for Computing Machinery, New York, NY, USA.

[31] Jack Jamieson, Naomi Yamashita, and Eureka Foong. 2024. Predicting open source contributor turnover from value-related discussions: An analysis of GitHub issues. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*.

[32] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. 2014. The promises and perils of mining GitHub. In *Proceedings of the 11th Working Conference on Mining Software Repositories* (Hyderabad, India) *(MSR 2014)*. Association for Computing Machinery, New York, NY, USA, 92–101. https://doi.org/10.1145/2597073.2597074

[33] Marzena Karpinska and Mohit Iyyer. 2023. Large Language Models Effectively Leverage Document-level Context for Literary Translation, but Critical Errors Persist. In *Proceedings of the Eighth Conference on Machine Translation*. 419–451.

[34] Hannah Kim, Kushan Mitra, Rafael Li Chen, Sajjadur Rahman, and Dan Zhang. 2024. MEGAnno+: A Human-LLM Collaborative Annotation System. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. 168–176.

[35] Mahi Kolla, Siddharth Salunkhe, Eshwar Chandrasekharan, and Koustuv Saha. 2024. LLM-Mod: Can Large Language Models Assist Content Moderation?. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI EA '24)*. Association for Computing Machinery, New York, NY, USA, Article 217, 8 pages. https://doi.org/10.1145/3613905.3650828

[36] Renee Li, Pavitthra Pandurangan, Hana Frluckaj, and Laura Dabbish. 2021. Code of conduct conversations in open source software projects on github. *Proceedings of the ACM on Human-computer Interaction* 5, CSCW1 (2021), 1–31.

[37] Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, et al. 2026. Ministral 3. *arXiv preprint arXiv:2601.08584* (2026).

[38] Zhiwei Liu, Kailai Yang, Qianqian Xie, Tianlin Zhang, and Sophia Ananiadou. 2024. EmoLLMs: A Series of Emotional Large Language Models and Annotation Tools for Comprehensive Affective Analysis. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Barcelona, Spain) *(KDD '24)*. Association for Computing Machinery, New York, NY, USA, 5487–5496. https://doi.org/10.1145/3637528.3671552

[39] Courtney Miller, Sophie Cohen, Daniel Klug, Bogdan Vasilescu, and Christian KaUstner. 2022. "Did you miss my comment or what?": understanding toxicity in open source discussions. In *Proceedings of the 44th International Conference on Software Engineering (ICSE '22)*. Association for Computing Machinery, New York, NY, USA, 710–722. https://doi.org/10.1145/3510003.3510111

[40] Shyamal Mishra and Preetha Chatterjee. 2024. Exploring ChatGPT for Toxicity Detection in GitHub. In *Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER'24)*. Association for Computing Machinery, New York, NY, USA, 6–10. https://doi.org/10.1145/3639476.3639777

[41] Audris Mockus, Roy T Fielding, and James D Herbsleb. 2002. Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 11, 3 (2002), 309–346.

[42] Robert Munro Monarch. 2021. *Human-in-the-Loop Machine Learning: Active learning and annotation for human-centered AI.* Simon and Schuster.

[43] Nicole Novielli, Fabio Calefato, and Filippo Lanubile. 2014. Towards discovering the role of emotions in stack overflow. In *Proceedings of the 6th international workshop on social software engineering.*

[44] Siobhán O'mahony and Fabrizio Ferraro. 2007. The emergence of governance in an open source community. *Academy of Management Journal* 50, 5 (2007), 1079–1106.

[45] Marco Ortu, Bram Adams, Giuseppe Destefanis, Parastou Tourani, Michele Marchesi, and Roberto Tonelli. 2015. Are bullies more productive? Empirical study of affectiveness vs. issue fixing time. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories.* IEEE.

[46] Nick Pangakis and Sam Wolken. 2025. Keeping Humans in the Loop: Human-Centered Automated Annotation with Generative AI. In *International AAAI Conference on Web and Social Media.*

[47] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[48] Huilian Sophie Qiu, Anna Lieb, Jennifer Chou, Megan Carneal, Jasmine Mok, Emily Amspoker, Bogdan Vasilescu, and Laura Dabbish. 2023. Climate Coach: A Dashboard for Open-Source Maintainers to Overview Community Dynamics. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23).* Association for Computing Machinery, New York, NY, USA, 1–18. https://doi.org/10.1145/3544548.3581317

[49] Huilian Sophie Qiu, Bogdan Vasilescu, Christian Kästner, Carolyn Egelman, Ciera Jaspan, and Emerson Murphy-Hill. 2022. Detecting interpersonal conflict in issues and code review: cross pollinating open-and closed-source approaches. In *Proceedings of the 2022 ACM/IEEE 44th International Conference on Software Engineering: Software Engineering in Society.* 41–55.

[50] Md Shamimur Rahman, Zadia Codabux, and Chanchal K Roy. 2024. Do Words Have Power? Understanding and Fostering Civility in Code Review Discussion. *Proceedings of the ACM on Software Engineering* 1, FSE (2024), 1632–1655.

[51] Rahat Rizvi Rahman, Seung Pyo Kang, Se Jin Jang, Seungmoo Lee, Chungyean Cho, and Kostadin Damevski. 2026. When Pair Programming Gets Emotional: A Case Study of What LLMs Still Miss. *International Journal of Software Engineering and Knowledge Engineering* (2026), 1–15.

[52] Naveen Raman, Minxuan Cao, Yulia Tsvetkov, Christian Kästner, and Bogdan Vasilescu. 2020. Stress and burnout in open source: Toward finding, understanding, and mitigating unhealthy interactions. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results.* 57–60.

[53] Aimee Kendall Roundtree. 2022. Facial recognition technology codes of ethics: Content analysis and review. In *2022 IEEE International Professional Communication Conference (ProComm).* IEEE, 211–220.

[54] Jaydeb Sarker. 2022. Identification and mitigation of toxic communications among open source software developers. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering.* 1–5.

[55] Jaydeb Sarker, Asif Kamal Turzo, and Amiangshu Bosu. 2025. The Landscape of Toxicity: An Empirical Investigation of Toxicity on GitHub. *FSE* (2025).

[56] Jaydeb Sarker, Asif Kamal Turzo, Ming Dong, and Amiangshu Bosu. 2023. Automated Identification of Toxic Code Reviews Using ToxiCR. *ACM Trans. Softw. Eng. Methodol.* 32, 5 (July 2023), 118:1–118:32. https://doi.org/10.1145/3583562

[57] Martin Saveski, Brandon Roy, and Deb Roy. 2021. The Structure of Toxic Conversations on Twitter. In *Proceedings of the Web Conference 2021 (WWW '21).* Association for Computing Machinery, New York, NY, USA, 1086–1097. https://doi.org/10.1145/3442381.3449861

[58] Charlotte Schluger, Jonathan P. Chang, Cristian Danescu-Niculescu-Mizil, and Karen Levy. 2022. Proactive Moderation of Online Discussions: Existing Practices and the Potential for Algorithmic Support. *Proc. ACM Hum.-Comput. Interact.* 6, CSCW2 (Nov. 2022), 370:1–370:27. https://doi.org/10.1145/3555095

[59] Margaret-Anne Storey, Alexey Zagalsky, Fernando Figueira Filho, Leif Singer, and Daniel M German. 2016. How social and communication channels shape and challenge a participatory culture in software development. *IEEE Transactions on Software Engineering* 43, 2 (2016), 185–204.

[60] Damian A Tamburri, Philippe Kruchten, Patricia Lago, and Hans van Vliet. 2015. Social debt in software engineering: insights from industry. *Journal of Internet Services and Applications* 6, 1 (2015), 10.

[61] Damian A Tamburri, Fabio Palomba, and Rick Kazman. 2019. Exploring community smells in open-source: An automated approach. *IEEE Transactions on Software Engineering* 47, 3 (2019), 630–652.

[62] Forgotten Server Development Team. 2018. Issue Comment on GitHub: Forgotten Server #2494. https://github.com/otland/forgottenserver/issues/2494/#issuecomment-417775273

[63] Bianca Trinkenreich, Klaas-Jan Stol, Anita Sarma, Daniel M German, Marco A Gerosa, and Igor Steinmacher. 2023. Do i belong? modeling sense of virtual community among linux kernel contributors. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE).* IEEE, 319–331.

[64] Xinru Wang, Hannah Kim, Sajjadur Rahman, Kushan Mitra, and Zhengjie Miao. 2024. Human-LLM collaborative annotation through effective verification of LLM labels. In *Proceedings of the CHI Conference on Human Factors in Computing Systems.* 1–21.

[65] Zijie J Wang, Dongjin Choi, Shenyu Xu, and Diyi Yang. 2021. Putting Humans in the Natural Language Processing Loop: A Survey. In *Proceedings of the First Workshop on Bridging Human–Computer Interaction and Natural Language Processing.* 47–52.

[66] Xingjiao Wu, Luwei Xiao, Yixuan Sun, Junhang Zhang, Tianlong Ma, and Liang He. 2022. A survey of human-in-the-loop for machine learning. *Future Generation Computer Systems* 135 (2022), 364–381.

[67] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388* (2025).

[68] Ruoyu Zhang, Yanzeng Li, Yongliang Ma, Ming Zhou, and Lei Zou. 2023. LLMaAA: Making Large Language Models as Active Annotators. In *Findings of the Association for Computational Linguistics: EMNLP 2023.* 13088–13103.

[69] Wenxuan Zhang, Yue Deng, Bing Liu, Sinno Pan, and Lidong Bing. 2024. Sentiment Analysis in the Era of Large Language Models: A Reality Check. In *Findings of the Association for Computational Linguistics: NAACL 2024*, Kevin Duh, Helena Gomez, and Steven Bethard (Eds.). Association for Computational Linguistics, Mexico City, Mexico, 3881–3906. https://aclanthology.org/2024.findings-naacl.246/

[70] Yunfeng Zhang, Q Vera Liao, and Rachel KE Bellamy. 2020. Effect of confidence and explanation on accuracy and trust calibration in AI-assisted decision making. In *Proceedings of the 2020 conference on fairness, accountability, and transparency.* 295–305.

[71] Yiming Zhu, Peixian Zhang, Ehsan-Ul Haq, Pan Hui, and Gareth Tyson. 2024. Exploring the capability of chatgpt to reproduce human labels for social computing tasks. In *International Conference on Advances in Social Networks Analysis and Mining.* Springer, 13–22.

[72] Frances Zlotnick. 2017. GitHub Open Source Survey 2017. http://opensourcesurvey.org/2017/. https://doi.org/10.5281/zenodo.806811

Received 2026-03-26; accepted 2026-06-18