Toxicity Ahead: Forecasting Conversational Derailment on GitHub

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

Robert Zita
Elmhurst University
Elmhurst, IL, USA
rzita8729@365.elmhurst.edu

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

Preetha Chatterjee
Drexel University
Philadelphia, PA, USA
preetha.chatterjee@drexel.edu

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

ABSTRACT

Toxic interactions in Open Source Software (OSS) communities reduce contributor engagement and threaten project sustainability. Preventing such toxicity before it emerges requires a clear understanding of how harmful conversations unfold. However, most proactive moderation strategies are manual, requiring significant time and effort from community maintainers. To support more scalable approaches, we curate a dataset of 159 derailed toxic threads and 207 non-toxic threads from GitHub discussions. Our analysis reveals that toxicity can be forecast by tension triggers, sentiment shifts, and specific conversational patterns.

We present a novel Large Language Model (LLM)-based framework for predicting conversational derailment on GitHub using a two-step prompting pipeline. First, we generate *Summaries of Conversation Dynamics* (SCDs) via Least-to-Most (LtM) prompting; then we use these summaries to estimate the *likelihood of derailment*. Evaluated on Qwen and Llama models, our LtM strategy achieves F1-scores of 0.901 and 0.852, respectively, at a decision threshold of 0.3, outperforming established NLP baselines on conversation derailment. External validation on a dataset of 308 GitHub issue threads (65 toxic, 243 non-toxic) yields an F1-score up to 0.797. Our findings demonstrate the effectiveness of structured LLM prompting for early detection of conversational derailment in OSS, enabling proactive and explainable moderation.

CCS CONCEPTS

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

KEYWORDS

Toxicity, Bug Report, Empirical Study, Open Source Software

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACM Reference Format:
Mia Mohammad Imran, Robert Zita, Rahat Rizvi Rahman, Preetha Chatterjee, and Kostadin Damevski. 2026. Toxicity Ahead: Forecasting Conversational Derailment on GitHub. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Toxic language undermines the health of online communities, including those centered around software projects. A 2024 GitHub survey reported that 64.23% of developers experienced or witnessed negative interactions [23], a slight increase from the 60.0% recorded in 2017 [72]; notably, 21.4% reported that such interactions led them to stop contributing. Despite the increasing recognition of the negative impact of toxic interactions, to our best knowledge, all existing toxicity detection methods are post-hoc [44, 51, 54, 56], identifying toxic content only after it appears. While post-hoc detection can mitigate some of the damage caused by toxic interactions, it fails to prevent the initial harm and may allow negative behaviors to persist unchecked for extended periods. A reactive approach not only delays intervention but also burdens community moderators and risks alienating contributors who might have otherwise remained engaged. Consequently, there is a pressing need for proactive solutions that can anticipate and preemptively address potential toxicity [39].

The primary strategy of proactive moderation involves human moderators actively engaging with ongoing conversations to prevent them from devolving into toxic behavior or, at the very least, to swiftly address any negativity should it arise and before it escalates any further [9, 58]. While effective, manual proactive moderation in OSS is impractical since moderators need to continuously monitor ongoing conversations across several communication channels (e.g., issues, chats, discussion boards). On the other hand, automated moderation offers scalability but demands a deep understanding of community norms and context. However, unlike platforms such as X (formerly Twitter) or Reddit, GitHub exhibits more subtle toxic behaviors, such as entitlement, miscommunication, or resistance to new practices, rather than overt aggression [25]. Moreover, domain-specific terms in software engineering (e.g., 'kill,' 'dead,' and 'dump') can pose challenges to generic automated toxicity detection [27, 56].

In this paper, we investigate how GitHub conversations derail into toxicity and present an automated approach for predicting such

derailment. Leveraging recent advances in LLMs, we introduce a novel framework that integrates advanced prompting techniques to support proactive moderation in OSS communication channels. More specifically, we address these three research questions:

RQ1: *What are the characteristics and patterns of conversational derailment in GitHub discussions?*

Effective prediction requires deep understanding of how technical conversations deteriorate. We curated a dataset of 159 derailed toxic and 207 non-toxic GitHub conversations, annotated with derailment and toxicity points. Then, we empirically analyze temporal dynamics, linguistic patterns, and contextual triggers that precede toxicity. Our findings show that derailment on GitHub often precedes toxicity by a narrow margin; the median distance between the first derailment point and the first toxic comment is just 3 comments, and 64% of toxic comments occurred within 24 hours. We identify a set of consistent early warning signals at derailment points, including elevated use of reasoning terms (e.g., "because", "since"), WH-questions (e.g., "why", "how"), and second person pronouns (e.g., "you", "your"), as well as tones like *Frustration* or *Impatience*.

RQ2: *Can Large Language Models effectively predict conversational derailment on GitHub?*

We develop and evaluate a novel LLM-based framework that generates interpretable conversation summaries to predict derailment. More specifically, we leverage *Least-to-Most* prompting to generate high-level *Summaries of Conversation Dynamics* (SCDs). These summaries abstract away technical details to highlight interaction patterns, emotional tone, and rhetorical shifts. Based on these SCDs, we predict the likelihood of a conversation derailing into toxicity. To enable forecasting rather than detection, our approach for SCD generation only considers comments that occur before the first toxic remark, ensuring that predictions are based solely on pre-toxicity context. This design aligns with prior derailment forecasting formulations [9].

Our framework achieves F1-scores of 0.901 (Qwen) and 0.852 (Llama), significantly outperforming established baselines including CRAFT [9] and Hua et al.'s few-shot SCD [26] approaches while maintaining interpretability for human moderators. Through ablation studies, we find that sentiment evolution and tension triggers are the most critical components for prediction accuracy.

RQ3: *To what extent does the proposed LLM-based derailment prediction approach generalize to independent GitHub datasets?*

We validate our approach on independent GitHub data to establish confidence in its broader applicability across different communities and time periods. External validation on the Raman et al.'s [51] dataset (308 threads, 65 toxic and 243 non-toxic) shows reasonable generalizability, with our LtM strategy achieving F1-scores of 0.797 (Qwen) and 0.776 (Llama). The approach outperforms baselines on external data despite different collection methodologies, time periods, and class distributions, suggesting that our method captures broader patterns of GitHub conversational dynamics.

Paper Contributions. Our investigation yields several key contributions. We provide the empirical characterization of GitHub conversational derailment patterns, revealing predictable deterioration signals including temporal proximity (median 3 comments before toxicity), distinctive linguistic markers, and common triggers.

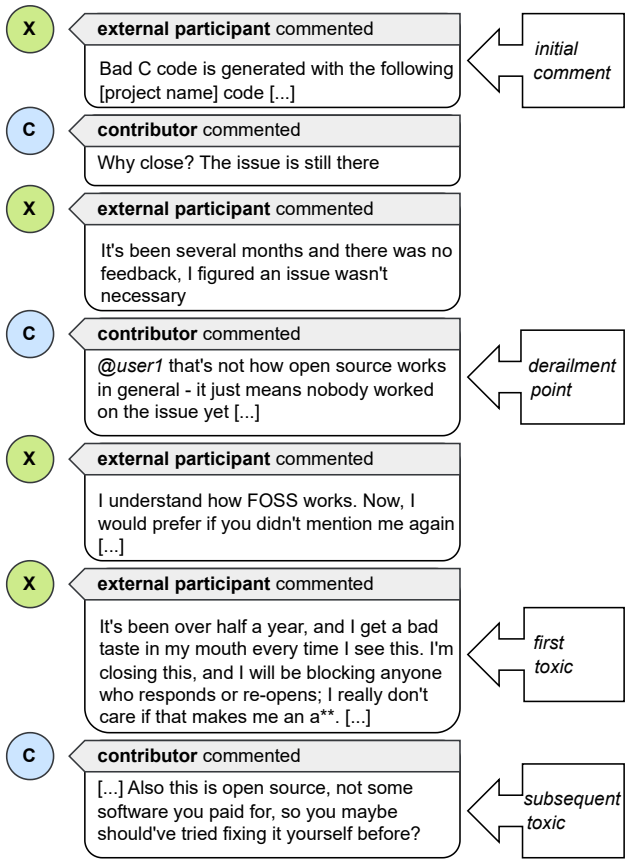

Figure 1: Example of a toxic conversation on GitHub.

We introduce a novel LLM-based framework using structured Least-to-Most prompting to generate explainable Summaries of Conversation Dynamics, achieving F1-scores of 0.901 while maintaining interpretability. Finally, we provide actionable insights for OSS communities seeking to implement proactive moderation strategies.

Our study's datasets, scripts, and output logs are publicly available online to facilitate reproducible research [2].

2 EXAMPLE OF CONVERSATIONAL DERAILMENT ON GITHUB

In online forums, toxicity often occurs after identifiable signs in the previous comments by the discussion participants [9]. In this research, we focus on understanding the early signs that a conversation will turn toxic on GitHub issues and pull request discussions. These preceding comments, where it becomes clear that the conversation has moved away from being productive and taken a turn towards negativity, are called *derailment points* [68].

Overall, toxic conversations often contain the following identifiable elements: 1) a conversation-initiating comment, 2) a derailment point comment, 3) a first toxic comment, and 4) (zero or more) subsequent toxic (or non-toxic) comments. Figure 1 shows an example of a toxic conversation, highlighting these different structures. In this conversation between an OSS project contributor and an external participant (i.e., someone who has never made a commit to the repository), the contributor derails the conversation by making a

Table 1: Definitions and examples of uncivil tone-bearing discussion features (TBDF).

TBDF	Definition	Example
Bitter Frustration	Expressing strong frustration, displeasure, or annoyance	*No answer, no reaction, what kind of support is that.*
Impatience	Expressing dissatisfaction due to delays	*Issue not fixed in 30 days? Must be gone!*
Mocking	Ridiculing or making fun of someone in a disrespectful way	*Legend says this issue will still exist even on the end of mankind.*
Irony	Using language to imply a meaning that is opposite to the literal meaning, often sarcastically	*Maybe you should actually write that down somewhere. You know, like in the documentation.*
Vulgarity	Using offensive or inappropriate language	*Who cares, same sh*t.*
Threat	Issuing a warning that implies a negative consequence	*Any further responses will result in you being blocked from the repo entirely.*
Entitlement	Expecting special treatment or privileges	*that's how good we are. I don't want your contribution. [...]*
Insulting	Making derogatory remarks towards another person or project	*This looks like it was done by a 5 year old.*
Identity attacks/ Name-calling	Making derogatory comments based on race, religion, gender, sexual orientation, or nationality	*I would not be surprised if this database is maintained by the [nationality].*

mocking comment. The external participant responds with frustration and then makes a toxic, insulting remark. This is followed by another toxic comment, this time made by the contributor.

3 DATASET

Understanding the characteristics of derailed conversations is crucial for developing effective intervention strategies. To facilitate that, we curate two datasets comprising conversations from GitHub issues and pull requests: one containing derailed toxic threads and the other consisting of non-toxic threads. We describe our annotation process in detail, with an emphasis on ensuring label reliability and curating examples that are both representative of OSS discourse and suitable for downstream analysis.

3.1 Toxic Conversations Dataset

We start with a dataset recently released by Ehsani et al. [16], which focuses on incivility in GitHub conversations. Toxicity is a subset of incivility, focusing on harmful language, while incivility more broadly includes behaviors that undermine constructive discussion [19, 52]. More specifically, incivility is defined as *"features of discussion that convey an unnecessarily disrespectful tone toward the discussion forum, its participants, or its topics"* [12], while toxicity is defined as *"rude, disrespectful, or unreasonable language that is likely to make someone leave a discussion"* [33]. Therefore, leveraging this incivility dataset provides an appropriate starting point for identifying toxic interactions in GitHub threads.

We use an LLM-aided model-in-the-loop annotation approach to identify the uncivil comments that are also toxic [6]. Recent research shows that such a model-in-the-loop annotation methodology works well for this type of data, including hate and violent speech detection tasks [21, 31, 46, 53, 64, 67, 71]. This methodology has been leveraged in software engineering toxicity detection as well [30, 44]. In our annotation, we leveraged the prompt released by Imran et al. for toxicity detection in GitHub bug reports [30]. For each uncivil comment identified in Ehsani et al.'s dataset, we used GPT-4o to assess whether the comment was toxic, providing the full preceding conversation context up to that point. GPT-4o labeled 832 comments across 273 threads as toxic. To validate these predictions, two authors independently reviewed each flagged comment. The initial inter-annotator agreement was 0.78 (Cohen's Kappa). The annotators resolved all disagreements through in-person discussion in order to finalize the toxicity annotations.

Ehsani et al.'s dataset is based on 404 locked conversations (issues and PRs) on GitHub where the reason they were locked is listed as 'too heated,' 'spam,' or 'off-topic.' These 404 conversation threads contain 5961 comments annotated with various categories of uncivil TBDFs (Tone Bearing Discussion Features). The definitions and examples of the incivility-related TBDFs are shown in Table 1. Since the focus of our study is to identify conversations that derail into toxicity, we excluded conversations in which toxicity occurred in the initial post (i.e., we cannot predict derailment without processing the initial post). This step resulted in a dataset consisting of 175 toxic GitHub threads.

3.2 Non-Toxic Conversations Dataset

To compare toxic conversations with ordinary, non-toxic GitHub issues and pull requests, we collected a sample of non-toxic dataset using the toxic dataset as a reference point. Specifically, we gathered 15 threads that were posted immediately before and 15 threads immediately after each toxic thread within the same repository, maintaining temporal and project-local continuity. We applied several exclusion criteria, removing: 1) non-English conversations, 2) threads with no comments after the initial post, and 3) locked threads marked as "resolved" (i.e., locked for reasons unrelated to toxic discourse).

From the collected posts, two annotators (authors of this paper) randomly selected posts one by one and verified their non-toxic nature, using the definition in section 3.1. The annotators continued this process until the number of *non-toxic* conversations reached a total approximately matching the number of *toxic* conversations. During this procedure, they identified 23 threads as toxic and 207 as non-toxic. The final set of 207 non-toxic threads constitutes our *Non-Toxic Conversations Dataset*. The newly identified 23 toxic threads are combined with Ehsani et al's dataset of 175 toxic conversations.

3.3 Derailed Toxic Dataset

Since we investigate on identifying derailed toxic threads, we further refined our dataset by identifying the points at which conversations begin to derail. Specifically, we observed that instances of incivility preceding the toxic comments often indicate derailment,

as they signal a shift away from the original conversational intent [68]. Given that Ehsani et al. previously annotated incivility in conversations containing TBDFs (see Table 1), we hypothesize that any uncivil TBDF occurring before a toxic comment marks a potential derailment point. However, our investigation revealed that derailment can also arise outside of these TBDF categories, particularly in the presence of pronounced negative politeness strategies (e.g., *'I'm terribly sorry, but'*, *'Would you mind'*) [14]. Therefore, we annotate the derailment points following the definitions provided in Section 2 and use TBDFs as a guide whenever they were available.

Two of the paper's authors reviewed the 198 toxic threads (175 from Ehsani et al.'s heated conversations dataset and 23 obtained while creating the Non-Toxic Conversations Dataset) to identify possible derailment points preceding toxic comments. Their inter-annotator agreement reached 0.914 (Cohen's Kappa). The annotators determined that 34/198 threads exhibited "sudden toxicity", i.e., toxic threads do not exhibit derailment; instead, in these conversations toxicity occurs suddenly and unexpectedly. We excluded them from our analysis. The annotators discussed where they disagreed and resolved the disagreements. In 88 cases, they found multiple derailment points before the toxic comment. In 71 cases, annotators found exactly 1 derailment point. We also excluded 5 cases where the annotators could not agree whether there existed a derailment point or not. Finally, we retained 159 toxic threads (142 from Ehsani et al.'s and 17 newly identified threads from Section 3.2) with 382 derailment points, which form our *Derailed Toxic Dataset*.

3.4 Raman et al.'s Toxicity Dataset

Since we propose a new approach in conversational derailment detection in SE domain, we experiment on evaluating the generalization of our proposed methodology. In order to evaluate that, we use the manually annotated toxicity dataset released by Raman et al. [51]. It spans GitHub issue threads from 2012–2018, containing 167 toxic and 444 non-toxic threads, identified from locked or flagged discussions. However, there were only 314 threads available with comment-level annotations; since our evaluation setup (See more: Section 5.3) requires identifying the exact location of the first toxic comment, we exclude those without such detail. We also filtered out six threads that began with toxic comments, 308 threads remain (65 toxic, 243 non-toxic), making the dataset notably more imbalanced than our own. There is no overlap between this dataset and ours.

4 RQ1: WHAT ARE THE CHARACTERISTICS AND PATTERNS OF CONVERSATIONAL DERAILMENT IN GITHUB DISCUSSIONS?

Understanding how conversations derail on GitHub is fundamental to developing effective prediction systems. Relative to conversations on other online platforms such as Reddit and Wikipedia, GitHub's technical conversations are distinct in the way that they exhibit toxicity [43]. Research also noted that diffient communities exhibit different characteristics in toxic communication patterns [60]. Therefore, it stands to reason that conversational derailment on this platform may also be distinct. In this RQ, we empirically examine how conversational derailment manifests on GitHub. Specifically, we investigate: 1) the dynamics of conversational derailment by examining its timing and distance from the thread's start, 2) the linguistic signals preceding derailment, 3) the presence of uncivil tones, and 4) common triggers that lead discussions off track.

We base our analysis on the Derailed Toxic Dataset. While limited in size and predominantly sourced from *locked as heated* GitHub issues, this dataset is sufficient to observe derailment patterns. Locked issues are not a concern, as the locking occurs after the derailment and toxicity have occurred, and the locking mechanism does not affect the communication pattern. In addition, for the empirical analysis in this section, as our focus is on the interaction patterns leading up to toxicity, for any discussion thread, we exclude the comments that occurred after the first toxic comment.

4.1 Timing and Distance to Derailment Points

We calculate the median number of comments from the conversational first derailment point to the first toxic comment for each conversation thread in Derailed Toxic Dataset. In our toxic threads dataset, in terms of total comments, the median comment count 11, and mean comment count 17.6. However, the median first toxic comment occurrence position is 8 and mean is 12.03. And median first derailed comment occurrence position 4 and mean is 5.92. While, the median *distance* is 3 comments and a mean *distance* is 6.10. The close proximity between derailment and toxic comments suggests that once a thread derails, it is likely to directly devolve into toxicity. This aligns with Cheng et al.'s findings, which indicate that negative context and mood increase the likelihood of trolling behavior [11].

The timing of the first toxic comment relative to the derailment point provides additional insights. Considering a 8-hour workday, we observe about 46% (73/159) of the of toxic comments occur within 8 hours of the first derailment comment [9] and about 64% (102/159) occurs within 24 hours. This shows the importance of timely intervention. However, more than 25% cases (40/159), the difference is more than 7 days, which indicates toxicity can also occur after a long period of derailment. This characteristic contrasts with platforms like Wikipedia, where a discussion is likely inactive if the last comment was added 2–3 days ago [58].

Note that in the few cases where there are multiple derailment point comments preceding the toxic comment, for this analysis, we considered the first derailment point in the conversation.

4.2 Linguistic Features

In Chang et al.'s study, participants noted that the easiest way to forecast conversational derailment is by analyzing user phrasing [10]. Indicators include the use of direct address such as 'you' instead of generic terms like 'all' or 'always', as well as certain rhetorical postures or argumentative patterns can signal a conversation may be turning toxic [10].

We analyze potential language indicative of derailment in GitHub discussions. In the 382 derailment point comments in the Derailed Toxic Dataset, we sampled the 200 most frequent unigrams, excluding articles, particles, and common prepositions. We intentionally retained negation and question terms because they are strong markers of argumentative or confrontational language in

Table 2: Lexical cues in derailment point comments. Statistical significance (Chi-square test) between derailment and regular comments is indicated by * ($\alpha = 0.05$).

Linguistic Features	Comment Type (%)			p-value	Cramer's V
	Derail (n = 382)	Toxic (n = 159)	Regular (n = 1,371)		
Second Person Pronouns	60.7%	75.5%	43.9%	$< 0.0001^*$	0.127
WH Question Words	57.1%	59.7%	43.9%	$< 0.0001^*$	0.104
Negation terms	70.2%	71.1%	55.3%	$< 0.0001^*$	0.132
Reasoning terms	70.4%	70.4%	61.4%	$< 0.0001^*$	0.055
Emphasis terms	53.4%	59.7%	42.5%	$< 0.0001^*$	0.123
Communication Verbs	33.5%	36.5%	24.9%	$< 0.0001^*$	0.133

this context [32, 36, 61, 70]. Two annotators collaboratively categorized (see Table 2) the unigrams into linguistic groups using the card sorting method [59]. They met in person, discussed, and resolved differences, consulting a dictionary as needed. Based on these categories, we counted the frequency of each unigram in the derailment point comments after applying basic preprocessing steps (e.g., tokenization and lemmatization).

Table 2 shows the percentages of occurrence in derailment points (382 count) along with the first toxic comments (159 count), and regular comments (non toxic and non derailment point comments); the total regular comments count were 1,371. We observed that in derailment points the elevated use of second person pronouns ('you', 'your', etc) [38], negation terms ('not', 'no', etc.), "WH" questions ('what', 'why', 'how', 'where', etc.), reasoning terms ('because', 'since', etc.), communication verbs ('say', 'comment', 'tell', etc.), and emphasis terms ('actually', 'really', etc.) than general comments but lower than toxic comments.

As Table 2 shows, all lexical differences between derailment and regular comments were statistically significant under a Chi-square test of independence [41] (χ^2, $p < 0.05$) after applying the Benjamini–Hochberg (BH) correction [8]. Effect sizes measured by Cramer's V (0.05–0.13) indicate small to moderate associations, confirming consistent but modest linguistic distinctions.

Derailment point comments frequently combine structured reasoning with mild confrontation. Reasoning terms dominate (70.4%), often paired with negation (70.2%) and direct questioning (57.1%), reflecting logical yet oppositional exchanges. Compared with regular comments, derailment points show +16.8% more second-person targeting, +14.9% more negation, and +13.2% more questioning. Although they share similar reasoning intensity with toxic comments (70.4% vs. 70.4%), they contain less personal confrontation (60.7% vs. 75.5%) and reduced emphasis (53.4% vs. 59.7%). These findings suggest that derailment often emerges through argumentative yet non-abusive phrasing, where discussions shift from logical disagreement toward personal conflict.

4.3 Incivility TBDFs in Derailment Points

The tone of the comments is useful feature for proactive moderation. For example, moderators on Wikipedia assess tone to predict potential derailment [58]. Since the majority of the toxic conversations (142/159) in the Derailed Toxic Dataset came from Ehsani et al.'s dataset, the comments in these conversations already have incivility-related tones annotated. The Tone Bearing Discussion Features show the type of incivility at derailment point comments;

Table 3: Top TBDF categories in derailment point comments (142 toxic threads from Ehsani et al.).

TBDF Category	Derail. Pt. Cmts. (362)	Toxic Cmts. (142)
Bitter Frustration	155 (42.82%)	35 (24.65%)
Impatience	82 (22.65%)	13 (9.15%)
Mocking	36 (9.94%)	17 (11.97%)
Insulting	21 (5.80%)	36 (25.35%)

we found 362 such annotated comments. For this analysis, we considered only those 362 derailed comments. Table 3 show the percentages of TBDFs in derailment comments and toxic comments.

The major uncivil TBDFs are: Bitter Frustration: 42.82% (162/362), Impatience: 22.65% (84/362), and Mocking: 9.94% (39/362). For comparison, these same TBDFs occurred in toxic comments at the rates of 24.65% (35/142), 9.15% (13/142), and 11.97% (17/142), respectively. Notably, while Insulting and Vulgarity are more prominent in toxic comments– 25.35% and 9.86% respectively, they are less frequent in derailment points, occurring at 5.80% and 2.49%. This contrast indicates that certain forms of incivility such as *Bitter Frustration* and *Impatience* are more predictive of conversational derailment than direct toxicity. It also highlights a progression from subtle incivility to overt toxicity, and reinforces the importance of early signals such as frustration and impatience in anticipating derailment.

4.4 Derailment Triggers

Understanding what causes a conversation to derail can inform the development of effective early intervention strategies using automated, algorithmic approaches [57, 58]. Prior research in software engineering has identified potential triggers of toxicity in OSS discussions [17, 18, 43]. Ehsani et al. [16] proposed a guideline for annotating incivility triggers.

Building on their methodology, two authors independently annotated derailment triggers in our dataset, focusing on specific conversational or contextual elements that precipitated the initial shift. Although much of our toxic data overlaps with that of Ehsani et al., we chose to annotate derailment triggers with particular attention to the first derailment point in each conversation. This distinction was necessary because it was unclear from Ehsani et al.'s description whether their annotations considered the conversation holistically or targeted only the first uncivil comment when conversations started to go off-track.

The annotation achieved a Cohen's Kappa score of 0.84, indicating strong inter-annotator agreement. Disagreements were resolved through discussion to ensure full consensus. The most prevalent trigger was 'Failed Use of Tool/Code or Error Messages' followed at

23.27% (37/159), where tool difficulties or bug troubleshooting led to derailment. For example: *"[CODE SNIPPET] ... What more proof do you need? That is everything."* The tension was caused here due to code error, which the user expressed in Frustrated tones. The conversation later evolved into toxicity. The second most prevalent derailment trigger was 'Technical Disagreement' followed at 20.12% (32/159), where tool difficulties or bug troubleshooting led to derailment. For instance: *"[CODE SNIPPET] Ask yourself what **intention** it expresses. This is some kind of esoteric gibberish without reference to the subject area. [...]"*. In this case, the disagreements about method naming derails the conversation and later the conversation escalated to toxicity. Another major category was 'Communication Breakdown', which accounted for 16.98% (27/159) of cases. This included misunderstandings, misinterpretations, typos, or language barriers causing perceived hostility. For example, *"It is impolite to assume that each user opening an issue is stupid and lazy. Of course, I search the issue tracker. [...]"* Here, a misunderstanding between the commenters triggered conversation derailment.

5 RQ2: CAN LARGE LANGUAGE MODELS EFFECTIVELY PREDICT CONVERSATIONAL DERAILMENT ON GITHUB?

Building on RQ1's identification of derailment patterns, we investigate whether LLMs can effectively leverage conversation dynamics to predict derailment before toxicity occurs. We draw inspiration from Hua et al., who demonstrated an automated derailment forecasting system by leveraging *Summaries of Conversation Dynamics* (SCD) for predictive modeling [26]. SCDs offer a concise representation of a conversation's progression, characterizing the types of interactions that shaped its trajectory and forecasting future developments. Starting from Hua et al.'s original formulation, we incorporate insights from Section 4 to reflect the conversational patterns observed to GitHub discussions.

5.1 Baseline Models

We compare our SCD-based technique to two baselines:

CRAFT: CRAFT is one of the earliest and best-known models for predicting conversational derailment [9]. The CRAFT tool is based on a neural network model. Since its inception in 2019, various other strategies for predicting conversational derailment have been explored, and CRAFT has been used as a baseline in these subsequent studies to evaluate the performance of newer approaches [3, 34, 40, 66].

Hua et al.'s approach: Hua et al. [26] proposed that generating SCD can show the trajectory of the conversation and can be used to predict downstream conversational derailment task. They developed a few-shot procedural prompt where the LLM was provided with manually written SCD examples. Based on the prompt, they generated the SCD and predicted derailment as a downstream task.

5.2 LLM Prompt Design

We design a two step LLM prompting procedure for GitHub derailment prediction:

Step 1 – GitHub-Specific SCD Generation: We convert the raw GitHub conversation into a high-level summary that captures interaction dynamics, emotional tone, and discourse strategies, excluding technical specifics.

Step 2 – Derailment Prediction: We estimate the probability of derailment based solely on the summary, using a simple scalar prediction prompt, i.e., a prompt that instructs the LLM to produce a single numerical value, specifically, a probability between 0 and 1, representing the likelihood that a conversation will derail into toxicity. This prompt design separates the reasoning and classification stages, enabling us to build explainability into the pipeline and reducing the reasoning demands on the LLM.

5.2.1 GitHub-Specific SCD Prompt. We adapt Hua et al. [26]'s few-shot SCD prompt to GitHub's conversational style and dynamics. Specifically, we replace general conversation examples with domain-specific interactions, such as discussions around pull request rejections or issue closures. Additionally, we instruct the model to ignore technical details like code snippets or file paths, and instead focus on conversational dynamics, i.e., how participants respond to one another, where misunderstandings arise, and how tone shifts over time. A typical summary generated using this GitHub-adapted SCD prompt is as follows:

> *Multiple users debate reverting a recent PR. Speaker1 expresses strong opposition, referencing a previous incident involving similar code. Speaker2 challenges Speaker1's framing and accuses them of misrepresenting past decisions. Speaker3 supports Speaker2 and notes that Speaker1 had already raised this concern in another thread. Speaker1 becomes confrontational, citing a perceived pattern of dismissal. The conversation becomes increasingly heated as past interactions are used to question motives. The tone escalates, with limited signs of resolution.*

While this adaptation yielded clearer and more relevant summaries, we hypothesized that it may not yet be optimal. As Hua et al. developed SCD prompts targeting general-purpose conversations, it may not be most effective for the highly technical discussions found on GitHub. We explored whether decomposing the problem [35] and integrating the properties of GitHub derailed conversations, uncovered in Section 4, could yield better SCDs for predicting derailment on GitHub. Previous research shows that decomposing the prompts into incremental steps enhances the LLM's accuracy [35, 69]. Inspired by those studies, we adopted *Least-to-Most* (LtM) prompting strategy [69], and designed a step-wise summarization prompt that guides the model from high-level observations to detect smaller breakdowns in conversation patterns. This allows us to integrate our insights from Section 4. We incorporated the following key components into the LtM prompt (as Steps 3–6):

(1) **Individual Intentions (II)**, a feature we directly integrated from Hua et al.'s framework to analyze participants' motivations and goals. Morrill et al. similarly found that dialogue is shaped by intentions such as agreement, disagreement, confrontation [45]. Additionally, studies have shown that communication styles play a role in shaping interactions within GitHub discussions [7, 63].
(2) **Conversational Features (CF)** (e.g., questioning, rhetoric), where we adopted the categories established by Hua et al. [26], which includes rhetorical questions, hedging, questioning logic,

and other linguistic patterns. This approach aligns with our findings described in Section 4.2;

(3) **Sentiment and Tonal Features (STF)**, which capture emotional dynamics and shifts throughout the discussion, enabling us to track how sentiment evolves before derailment occurs, as discussed in Section 4.3; and

(4) **Tension Triggers (TT)**, which identify potential catalysts for escalating conflict that serve as early indicators of possible derailment, based on our findings described in Section 4.4;

We explicitly instruct the model to exclude technical content and focus on interactional dynamics. The summary is synthesized at the final step.

In developing the LtM prompt, we initially experimented with prompts that included explicit definitions for each reasoning step, for instance, describing categories of tension triggers, conversation strategies or specifying tonal cues like sarcasm or frustration. However, we observed that such definitions often introduced unnecessary verbosity and led to inconsistent outputs in SCD generation. In contrast, we found that a more concise, open-ended prompt, omitting explicit definitions, led to more consistent and focused outputs. Rather than prescribing rigid categories, this design allowed the model to apply its learned understanding of conversational structure. The final LtM prompt is as follows:

Least-to-Most (LtM) SCD Generator Prompt

You are a skilled Conversation Analyst specializing in GitHub discussions. Your objective is to capture the conversation dynamics without getting caught in the technical details.

Your Analysis Method: Follow these steps in order, building your understanding from basic patterns to complex dynamics:

Step 1: Identify Main Elements: Identify the main elements by quickly scanning the conversation and pinpointing the key components or topics being discussed.

Step 2: Enforce Exclusion Criteria
Do NOT include:
- Any technical claims, arguments, or explanations
- Any code names, module names, or PR details
- Any direct or indirect quotations
- Any mention of what was being implemented or reviewed
Example 1 (Excluded): "The discussion addressed discrepancies in $evaluate\_admissions()$ outputs [...]."

Step 3: Note Individual Intentions
Infer what each participant is aiming to achieve. Example: [...]

Step 4: Identify Conversation Strategies
Identify rhetorical or structural tactics used by each speaker. Example: [...]

Step 5: Track Sentiment Evolution
Describe how the tone shifts across the conversation. Use descriptors such as neutral, guarded, dismissive, confrontational, collaborative, defensive, escalating. Avoid inference. [...]

Step 6: Find Tension Triggers
Detect explicit shifts in tone, rhythm, or interaction style that signal rising tension—only when clearly supported by observable evidence. Avoid inference. [...]

Step 7: Synthesize Trajectory Summary
Combine insights from steps 1-6 to create a concise trajectory summary.
Remember: Your value lies in revealing the human dynamics beneath technical discussions. Focus on HOW people communicate, not WHAT technical points they make.
Example: "Multiple users debate reverting a recent PR.[...]"

Your Task: Using the above guideline, write the final trajectory summary for the following GitHub discussion transcript:

$< insert\ conversation\ transcript >$

Write only the final summary within double quotation.

Here is an example SCD summary generated using this prompt:

> *USER_01 initiates a discussion about addressing deprecation warnings in a project, suggesting bumping the version number as a solution. USER_02 and USER_03 express reluctance to create a new major version, leading USER_01 to interpret this as a lack of future planning for the project. The conversation shifts to a more confrontational tone when USER_02 suggests an alternative fix and sarcastically requests an apology from USER_01, which escalates USER_01's response into a threat of ceasing contributions and blocking participants. USER_02 attempts to de-escalate by reiterating the availability of an alternative solution but maintains a guarded stance.*

5.2.2 Derailment Prediction Prompt. We employ a follow-up prompt that asks the model to predict derailment probability (0-1 scale) based on the generated SCD summary:

Derailment Predictor Prompt

You will be provided with a summary of a GitHub conversation trajectory. A Conversation Trajectory Summary captures the evolution of a discussion by identifying key elements, triggers of tension, conversational strategies, interaction patterns, sentiments, and tonal shifts among participants. It provides a concise overview of significant dynamics and tone changes without detailing the technical aspects or specific arguments.

Now based on the given GitHub conversation trajectory summary, what is the probability that the conversation may derail and become toxic on a scale of 0 to 1 (Toxicity is defined by "any rude, disrespectful, or unreasonable comment that is likely to make one leave a discussion.")?

Conversation trajectory summary: $< insert\ summary >$

Write exactly one word: the probability rounded to two decimal places.

This two-step prompting process allows us to first generate a comprehensive summary of the conversation and then use that summary to make a more informed prediction about the potential for toxicity.

5.3 Experiment Setup

5.3.1 Large Language Models. We conduct all of our experiments using two publicly available LLMs: Llama (*llama-3.3-70B* version) and Qwen (*qwen2.5:32b-instruct* version) model, as they are among the top-performing open-weight state-of-the-art LLMs at the time of conducting this study. We specifically chose freely available LLMs and avoided proprietary models (e.g, GPT-4 or Claude) to improve reproducibility and because the cost of paid models may make them impractical for real-world deployment.

We set the model temperature to 0 to minimize output variance and set a uniform a context window size of 32k (maximum length supported by Qwen 2.5 version). For each toxic conversation in our dataset, we provide all the comments up to, but excluding, the first toxic comment. Formally, let $C = \{c_1, c_2, \ldots, c_n\}$ be the ordered set of comments in a GitHub conversation, and let c_t be the first toxic comment such that: $t = \min\{\, i \mid c_i \text{ is labeled toxic} \,\}$, then, the model input is:

$$C' = \{c_1, c_2, \ldots, c_{t-1}\}.$$

Thus, the model only sees conversation context before toxicity emerges. It does not have access to the toxic comments themselves (and any comments afterwards). This formulation follows the forecasting prediction setup used by Chang et al. [9].

Whenever a conversation exceeded the context window limit, we truncated it by removing utterances from the beginning until the total length fell within the allowable range.

5.3.2 Metrics. We compare three popular metrics:

- **Precision** refers to the proportion of true positive observations among all the predicted positive observations: $\text{Precision} = \frac{\text{True Positive}}{\text{True Positive + False Positive}}$.
- **Recall** represents the proportion of true positive observations out of all actual positive observations: $\text{Recall} = \frac{\text{True Positive}}{\text{True Positive + False Negative}}$.
- The **F1-score** is the harmonic mean of Precision and Recall: $\text{F1-score} = 2 * \frac{\text{Precision * Recall}}{\text{Precision + Recall}}$.

5.4 Results and Discussion

We conduct experiments using our dataset of 159 derailed toxic threads and 207 non-toxic threads, a total of 367 data points. Of these, none of the threads exceeded the context window limit. Chang et al. showed that the prediction decision threshold (i.e., probability cutoffs) can vary widely for different datasets in the conversational derailment task [9]. Since we asked the LLMs to provide a prediction score of derailment between 0 to 1, we include results from different thresholds, $\theta \in \{0.1,\ 0.3,\ 0.5,\ 0.7\}$. The results of this experiment are shown in Table 4.

Our Least-to-Most SCD prompting strategy demonstrates strong performance across both models, achieving F1-scores of 0.901 (Qwen) and 0.852 (Llama) at the 0.3 threshold. The approach maintains balanced performance with high precision (0.945 for Qwen, 0.890 for Llama) while preserving good recall (0.862 for Qwen, 0.818 for Llama). Compared to the two baselines, our method significantly outperforms CRAFT, which consistently underperformed across all thresholds with best F1-score at 0.580. Against Hua et al.'s SCD prompting strategy, our approach shows superior results at most thresholds, particularly excelling at the practical 0.3-0.7 range where balanced precision-recall trade-offs are crucial [10].

Table 4: Derailment prediction results for different models on Derailed Dataset and Non-Toxic Dataset. (SCD = Summaries of Conversation Dynamics, θ = *Threshold*).

Model	Strategy	θ (≥)	Precision	Recall	F1
CRAFT [9]	-	0.1	0.419	0.937	0.579
	-	0.3	0.425	0.912	0.580
	-	0.5	0.585	0.522	0.551
	-	0.7	0.764	0.346	0.476
Qwen 2.5:32B Instruct	Hua et al. FewShot SCD [26]	0.1	0.440	1.000	0.612
		0.3	0.980	0.604	0.747
		0.5	1.000	0.258	0.410
		0.7	1.000	0.164	0.281
	Least-to-Most SCD	0.1	0.443	1.000	0.614
		0.3	0.945	0.862	**0.901**
		0.5	0.987	0.478	0.644
		0.7	0.981	0.327	0.491
Llama 3.3:70B	Hua et al. FewShot SCD [26]	0.1	0.750	0.981	0.850
		0.3	0.913	0.723	0.807
		0.5	0.929	0.572	0.708
		0.7	0.983	0.358	0.525
	Least-to-Most SCD	0.1	0.702	0.962	0.812
		0.3	0.890	0.818	**0.852**
		0.5	0.932	0.686	0.790
		0.7	0.975	0.491	0.653

For the Qwen model, the Least-to-Most strategy achieved the highest F1-scores across all threshold values. A similar trend was observed for the Llama model, with the exception of the 0.1 threshold where Hua et al.'s strategy performed better due to very high recall (156/159 correct predictions) but at the cost of much lower precision. This trade-off highlights an important consideration: while high recall captures most toxic instances, low precision leads to excessive false positives. In GitHub repositories where non-toxic conversations vastly outnumber toxic ones, maintaining high precision is critical to avoid unnecessary false alarms [51]. However, the high precision should not come at the cost of low precision as Chang et al. noted that balancing precision and recall is crucial, as too many false positives reduce a tool's helpfulness while too many false negatives reduce effectiveness [10]. At higher thresholds (≥ 0.3), our Least-to-Most strategy ensures good F1-scores with high precision, making it practically viable for deployment.

Based on these results, we envision a threshold-based intervention strategy to mitigate toxicity: higher thresholds (e.g., $\theta > 0.7$) could alert moderators to review flagged content, while lower thresholds (e.g., $\theta = 0.3$ *to* 0.7) could trigger automated bots to issue reminders promoting civil discourse.

5.5 Error Analysis

To better understand the performance and limitations of the model and datasets, we conduct an error analysis. We limited the error analysis to the best-performing configuration, i.e., the LtM SCD prompting strategy on the Qwen model at 0.3 threshold. We

Table 5: Ablation Study on our Least-to-Most (LtM) Derailment Prediction Strategy using Qwen and $\theta = 0.3$.

Ablation	Precision (Δ)	Recall (Δ)	F1 (Δ)	$p - value$ (Significant)
No II	0.962 (+1.8%)	0.792 (-8.1%)	0.869 (-3.6%)	0.2153 (×)
No CF	0.941 (-0.4%)	0.805 (-6.6%)	0.868 (-3.7%)	0.1628 (×)
No STF	0.898 (-5.0%)	0.774 (-10.2%)	0.831 (-7.8%)	0.0032 (✓)
No TT	0.906 (-4.1%)	0.786 (-8.8%)	0.842 (-6.5%)	0.0190 (✓)
Full LtM	0.945	0.861	0.901	–

conducted an open coding process to analyze the errors [65]. Two authors independently reviewed all misclassified cases and assigned preliminary labels to them. They then met to compare their labels, resolve disagreements, and refine the categories through iterative discussion. This process was repeated until complete consensus was achieved.

There were two types of error categories: 1) 8 cases where the model predicted non-toxic conversations as derailing; and 2) 22 cases where the model predicted derailed toxic conversations as non-derailing. Two authors reviewed the conversations, examined the generated SCD, and determined the most likely reason for the error. They finalized the error categories using open coding, and further improved them thorough discussion and applying axial coding [1], with some cases belonging to more than one category.

In the 8 false positives, the main issues were the *model overestimating tension* in otherwise civil exchanges (3 cases), and situations where the *SCD was accurate but the predictor misjudged* the tone's seriousness with respect to toxicity (3 cases). Among the 22 false negatives, most errors were due to *missing or underestimating subtle toxic signals* like frustration (10 cases), *failure to detect sarcasm or nuanced tones* (3 cases), *accurate SCDs with flawed predictor judgment* (3 cases), and cases where *toxicity occurred long after derailment* (3 cases), reducing the perceived severity. To illustrate, consider the following SCD where the model overestimated the seriousness of the tone as confrontational when the commentators were sharing their perspective, "*Participants discuss changes to game mechanics involving Repair Facility and Heavy Repair Turrets (HRT). [...] The conversation shifts from neutral to confrontational as participants assert their viewpoints and express dissatisfaction with proposed changes.*". In reality, participants were simply exchanging perspectives without hostility. In another case, the model missed sarcasm about image quality: "*Two users discuss an image and a JSON file issue. [...] reiterating concerns about image quality, the tone remains critical but not confrontational, focusing on clarity and quality standards.*" Here, subtle humor and sarcasm were misinterpreted as neutral critique, highlighting the model's difficulty with nuanced tones. These findings highlight specific weaknesses in both tone interpretation and the alignment between SCD generation and downstream prediction.

5.6 Ablation Study

To better understand the contribution of individual semantic components in our summarization prompt, we conducted an ablation study focused on our Least-to-Most (LtM) derailment prediction.

As introduced in Section 5.2, the LtM prompt integrates multiple high-level conversational features: Sentiment and Tonal Features (STF), Individual Intentions (II), Conversation Features (CF), and Tention Triggers (TT). Through the ablation study, we aim to quantify the effect of removing specific components in shaping the predictive utility of the summaries. We evaluate ablations exclusively on the Qwen model at $\theta = 0.3$ threshold because it achieved the best performance in our experiments.

5.6.1 Prompt Modifications Per Component. Each ablation removes one semantic component from the original LtM prompt:

- Removed **II** (LtM Prompt Step 3: *Note Individual Intentions*); renumbered subsequent steps.
- Removed **CF** (LtM Prompt Step 4: *Identify Conversation Strategies*); renumbered subsequent steps.
- Removed **STF** (LtM Prompt Step 5: *Track Sentiment Evolution*); renumbered subsequent steps.
- Removed **TT** (LtM Prompt Step 6: *Find Tension Triggers*); renumbered subsequent steps.

All other steps and examples were preserved to maintain structural consistency. The goal was to generate SCDs comparable to those from the baseline strategy. The full ablation prompts are included in the replication package. In addition to precision, recall and F1-score, we also perform statistical significance test by employing McNemar's test [42]. To control for multiple comparisons, we applied the Benjamini-Hochberg (BH) correction [8].

5.6.2 Results. Table 5 presents the precision, recall, F1-score, and BH corrected p-values from McNemar's test for each feature ablation. McNemar's test evaluates whether differences in model predictions are statistically significant under paired comparisons. A p-value below 0.05 denotes a statistically significant change in predictions.

Ablating the **STF** (sentiment and tone features) component results in the largest F1-score reduction (-7.8%), primarily due to a -10.2% drop in recall and a -5.0% decrease in precision. This change is statistically significant ($p = 0.0032$), indicating that STF features play a critical role in recall-oriented classification performance.

Removing **TT** (tension triggers) leads to a -6.5% F1 decline, driven by an -8.8% reduction in recall and a -4.1% drop in precision. The prediction shift is statistically significant ($p = 0.0190$), confirming the contribution of trigger features to predictive performance.

Ablating **II** results in a smaller F1 decrease (-3.6%), with a minor gain in precision (+1.8%) and a larger drop in recall (-8.1%). The BH-corrected p-value ($p = 0.2153$) does not indicate statistical significance. Likewise, removal of the **CF** feature yields a −3.7% F1 decrease, with minimal change in precision (-0.4%) and a -6.6% drop in recall; this change is also not statistically significant.

Overall, the results show that STF and TT features have the strongest and statistically significant impacts on out LtM prompted model performance, particularly in preserving coverage (recall).

6 RQ3: TO WHAT EXTENT DOES THE PROPOSED LLM-BASED DERAILMENT PREDICTION APPROACH GENERALIZE TO INDEPENDENT GITHUB DATASETS?

In order to validate our approach's generalizability, we evaluate it on Raman et al.'s [51] publicly available dataset [37], which is

Table 6: Derailment prediction results for different models on Raman et al.'s dataset.

Model	Strategy	θ	Precision	Recall	F1
Qwen	Hua et al. FewShot SCD	0.1	0.280	1.000	0.438
		0.3	0.807	0.708	0.754
		0.5	0.875	0.431	0.577
		0.7	0.955	0.323	0.483
	Least-to-Most Strategy	0.1	0.236	1.000	0.382
		0.3	0.753	0.846	**0.797**
		0.5	0.816	0.615	0.702
		0.7	0.857	0.369	0.516
Llama	Hua et al. FewShot SCD	0.1	0.594	0.877	0.708
		0.3	0.746	0.723	0.734
		0.5	0.804	0.631	0.707
		0.7	0.853	0.446	0.586
	Least-to-Most Strategy	0.1	0.513	0.908	0.656
		0.3	0.659	0.831	0.735
		0.5	0.754	0.800	**0.776**
		0.7	0.804	0.692	0.744

independent of our curated data and follows a different annotation procedure.

6.1 Dataset and Experimental Setup

As introduced in Section 3.4, the dataset comprises 168 toxic and 444 non-toxic threads. However, we found 314 threads have comment-level annotations available in their replication package. Since, evaluating the conversational derailment prediction requires comment-level annotation to find the exact toxic comment location, we filtered those 314 conversations. We further filter 6 conversations where toxicity observed at first comment. Therefore, we end up with 308 GitHub issue threads (65 toxic, 243 non-toxic). We apply the same preprocessing and modeling pipeline described in Section 5.3 to ensure comparability across datasets. Note that the dataset may possibly include sudden toxic conversations. We have not verified them manually.

As before, we evaluated the two prompt based techniques on this dataset using the Llama (*Llama-3.3:70B-3.3:70B* version) and Qwen (*Qwen-2.5:32B-Instruct* version) model, setting the thresholds: $\theta \in \{0.1, 0.3, 0.5, 0.7\}$. We report Precision, Recall, and F1-score as before for each setting.

6.2 Results and Discussion

The results are presented in Table 6. The Qwen model outperformed Llama on this benchmark, consistent with trends observed in our curated dataset. The LtM SCD strategy achieved the highest F1-score of 0.797 at a threshold of 0.3. For the Llama model, the same strategy yielded the best F1-score of 0.776, at a threshold of 0.5.

These findings reaffirm the effectiveness of the LtM prompting strategy over the baseline Hua et al. SCD few-shot prompt in detecting early conversational derailment on GitHub. While Qwen produced stronger results overall, Llama exhibited more stable performance across thresholds. Additionally, lower thresholds increased recall at the expense of precision, highlighting the importance of threshold calibration in real-world moderation settings.

7 RECOMMENDATIONS

The results indicate several practical directions for improving proactive moderation in open-source communities, particularly on GitHub. Based on empirical observations and model behavior across datasets, we outline recommendations for two groups: (1) researchers studying conversational derailment, and (2) GitHub maintainers involved in community moderation.

Recommendations for GitHub Maintainers. GitHub repository maintainers can integrate LLM-based early warning systems into existing moderation pipelines. Summarization-driven derailment detection provides a lightweight mechanism to flag discussions that may require attention, enabling timely intervention without exhaustive manual review.

Intervention thresholds should be calibrated according to moderation goals. Higher thresholds (e.g., > 0.7) are suited for passive alerts aimed at de-escalation [48], while intermediate thresholds (e.g., 0.3–0.7) can trigger automated reminders that act as conversational mediators [58]. This can be integrating within existing GitHub's infrastructure. Building on GitHub's current tagging options (e.g., 'abuse', 'off-topic', 'resolved') for [22], a dedicated derailment' tag could streamline moderator review and response.

Maintainers could incorporate the model's SCDs into issue and pull request dashboards to identify threads showing early signs of tension. Integration frequency should align with repository activity and cost constraints. Since 64% of toxic exhanges occur within 24 hours of derailment, higher-frequency runs (e.g., hourly) are most useful for new or rapidly evolving threads, whereas running the model after each new comment suffices for slower discussions. The SCDs provide interpretable summaries that can help maintainers make informed moderation decisions.

Recommendations for Researchers. Future research should build upon the current study by refining prompt designs to capture conversational signals of derailment more consistently across contexts. While this work has demonstrated the value of modeling tonal shifts, tension triggers, and sentiment trajectories, further efforts are needed to generalize these features across diverse OSS communities, languages, and moderation norms. Refinement should also aim to reduce prompt sensitivity and improve reproducibility of LLM-based summarization across datasets.

Developing standardized benchmarks across multiple platforms, such as GitHub and BugZilla, would enable consistent evaluation and cross-comparison of models. These benchmarks should include annotated derailment points, incivility categories, and moderation outcomes.

Efficiency and scalability remain key challenges. Incrementally updating summaries with new comments, rather than reprocessing entire threads, could allow near–real-time assessment with reduced computational cost. Beyond accuracy, researchers should evaluate latency, resource trade-offs, and feedback dynamics between predictive models and human moderators.

Transparency and explainability should remain research priorities. Future work should further refine how SCD convey rationales for tension and escalation, improving trust and accountability [58].

8 RELATED WORK

Our work builds upon and extends previous research in two main areas: toxicity analysis in software engineering and conversational derailment prediction.

Toxicity analysis in SE artifacts. A large body of work has explored negative communication in developer-facing platforms such as GitHub, Stack Overflow, and code review tools. Studies have examined incivility [16, 19, 20], emotional tone [28, 29, 47, 49], toxicity [13, 51, 54, 55] and their effects on contributor engagement [30, 43, 50, 62].

Several investigations have examined the consequences of toxicity. For example, toxic interactions have been linked to developer stress and increased dropout rates [51]. Other work has explored moderation strategies, the role of bots, and the benefits of early detection tools in mitigating toxic dynamics [25, 43]. Additionally, analysis of locked discussions has provided insights into recurring patterns of incivility and contributed key datasets [16, 18].

Researchers have developed tools to automate toxicity detection in software engineering [44, 51, 56]. However, all those tools are **post-hoc**, addressing toxicity only after it has occurred. Our work builds on this line by proposing a *proactive detection strategy*, shifting from retrospective classification to early forecasting of derailment events that precede toxicity.

Conversation derailment. Predictive studies of derailment have largely centered around Wikipedia and Reddit [5, 9, 26, 38, 40, 68]. The CRAFT model [9] pioneered early detection of online toxicity via linguistic and structural features. Hua et al. [26] introduced the concept of *Summaries of Conversation Dynamics (SCD)* for forecasting conversational trajectory, a foundation we extend using GitHub-specific dynamics.

Recent work also investigates hierarchical transformers, neural network, user behavior modeling, and contextual features such as reply structure and edit markers [3, 4, 15, 24, 34, 40, 66]. However, these approaches often lack domain adaptation for technical forums like GitHub, where toxicity emerges through more subtle signals like entitlement or miscommunication [25, 43].

Our contribution lies in integrating LLM-driven strategy with structured prompting tailored to a technical platform like GitHub's, providing both performance and explainability in derailment forecasting.

9 THREATS TO VALIDITY

We note potential threats to the validity of our study in the following categories: construct validity, internal validity, and external validity.

Construct validity. Construct validity concerns whether our methodology accurately captures the concept of derailment as a precursor to toxicity. A key limitation is that not all toxic behavior emerges gradually; some instances arise abruptly without prior conversational signals, making them undetectable by our approach. Additionally, derailment is annotated based on human judgment, which may vary across annotators. LLM-generated SCDs can also introduce hallucinations or overlook subtle shifts in tone. To mitigate these risks, we used detailed annotation guidelines, ensured high inter-annotator agreement, and conducted error analyses to identify common misclassifications.

Internal validity. Internal validity relates to whether the results are attributable to our method rather than uncontrolled factors. Our use of LLMs introduces variability due to stochastic generation and prompt sensitivity; small changes in input can affect model outputs. Moreover, although we followed a structured annotation process, human error or bias may still influence labeling consistency. We addressed these issues by employing a model-in-the-loop framework, cross-checking annotations, and designing prompts systematically to minimize ambiguity.

External validity. External validity reflects the generalization of our findings beyond the specific dataset and setting. Our dataset is limited to GitHub issue threads and may not generalize to other platforms such as JIRA, GitLab, or non-OSS communities, which have different conversational norms. Additionally, the curated dataset is relatively small, potentially limiting applicability across all GitHub communities. Furthermore, because many public GitHub discussions are part of the web-scale data used in LLM pretraining, it is possible that some threads in our dataset overlap with or resemble the model's pretraining data. This potential exposure may slightly inflate performance estimates, underscoring the need to validate the framework on data from unseen platforms or domains. While our prompting framework is domain-agnostic and we have evaluated on a different dataset, broader validation is needed to confirm its wider applicability.

10 CONCLUSION

We present a proactive approach to forecast toxicity detection in Open Source Software communities through early identification of conversational derailment on GitHub. We annotated and analyzed a dataset of 159 derailed toxic conversations and 207 non-toxic conversations. We developed a novel LLM-based prompt using Least-to-Most strategy to generate *Summaries of Conversation Dynamics* and predict conversation derailment, achieving F1-scores of 0.901 with Qwen and 0.852 with Llama, significantly outperforming established baselines. External validation on an independent imbalanced dataset yielded F1-scores up to 0.797, demonstrating generalizability.

Our findings show that early derailment prediction is a feasible and effective strategy for moderating technical discussions. The explainable nature of our SCD-based approach enables transparent moderation decisions and flexible threshold-based interventions, allowing communities to shift from reactive to proactive moderation strategies. While our work has limitations regarding platform specificity and detection of sudden toxicity without warning signs, it provides the empirical foundation and practical tools necessary for implementing early warning systems in real-world OSS projects. Future work should extend to other platforms, incorporate behavioral signals, and evaluate live deployment and intervention outcomes to support healthier, more inclusive OSS communities. Another key direction can be generating human-written SCD in GitHub conversations and do instruct tune LLMs to mitigate the errors we observe at Section 5.5 so that the LLMs can do better SCD generation.

REFERENCES

[1] 2006. *Constructing grounded theory: A practical guide through qualitative analysis.* sage.
[2] 2025. Replication Data. https://doi.org/10.5281/zenodo.15723482

[3] Enas Altarawneh, Ameeta Agrawal, Michael Jenkin, and Manos Papagelis. 2023. Conversation Derailment Forecasting with Graph Convolutional Networks. In *The 7th Workshop on Online Abuse and Harms*.

[4] Atijit Anuchitanukul, Julia Ive, and Lucia Specia. 2022. Revisiting contextual toxicity detection in conversations. *ACM Journal of Data and Information Quality* (2022).

[5] Jiajun Bao, Junjie Wu, Yiming Zhang, Eshwar Chandrasekharan, and David Jurgens. 2021. Conversations gone alright: Quantifying and predicting prosocial outcomes in online conversations. In *Proceedings of the Web Conference 2021*.

[6] Max Bartolo, Alastair Roberts, Johannes Welbl, Sebastian Riedel, and Pontus Stenetorp. 2020. Beat the AI: Investigating adversarial human annotation for reading comprehension. *Transactions of the Association for Computational Linguistics* (2020).

[7] Mohamed Amine Batoun, Ka Lai Yung, Yuan Tian, and Mohammed Sayagh. 2023. An empirical study on GitHub pull requests' reactions. *ACM Transactions on Software Engineering and Methodology* 32, 6 (2023), 1–35.

[8] Yoav Benjamini and Yosef Hochberg. 1995. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)* 57, 1 (1995), 289–300.

[9] Jonathan P Chang and Cristian Danescu-Niculescu-Mizil. 2019. Trouble on the Horizon: Forecasting the Derailment of Online Conversations as they Develop. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*.

[10] Jonathan P Chang, Charlotte Schluger, and Cristian Danescu-Niculescu-Mizil. 2022. Thread with caution: Proactively helping users assess and deescalate tension in their online discussions. *Proceedings of the ACM on human-computer interaction* 6, CSCW2 (2022), 1–37.

[11] Justin Cheng, Michael Bernstein, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. 2017. Anyone can become a troll: Causes of trolling behavior in online discussions. In *Proceedings of the 2017 ACM conference on computer supported cooperative work and social computing*.

[12] Kevin Coe, Kate Kenski, and Stephen A Rains. 2014. Online and uncivil? Patterns and determinants of incivility in newspaper website comments. *Journal of communication* 64, 4 (2014), 658–679.

[13] Sophie Cohen. 2021. Contextualizing toxicity in open source: a qualitative study. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*.

[14] Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 250–259.

[15] Christine De Kock and Andreas Vlachos. 2021. I Beg to Differ: A study of constructive disagreement in online conversations. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*.

[16] 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*. IEEE.

[17] Ramtin Ehsani, Rezvaneh Rezapour, and Preetha Chatterjee. 2023. Exploring Moral Principles Exhibited in OSS: A Case Study on GitHub Heated Issues. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*.

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

[19] 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).

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

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

[22] GitHub. 2025. GraphQL Interface: Minimizable. https://docs.github.com/en/graphql/reference/interfaces#minimizable Accessed: 2025-06-27.

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

[24] Jack Hessel and Lillian Lee. 2019. Something's Brewing! Early Prediction of Controversy-causing Posts from Discussion Features. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. https://doi.org/10.18653/v1/N19-1166

[25] 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* CSCW2 (2023).

[26] Yilun Hua, Nicholas Chernogor, Yuzhe Gu, Seoyeon Jeong, Miranda Luo, and Cristian Danescu-Niculescu-Mizil. 2024. How did we get here? Summarizing conversation dynamics. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

[27] Mia Mohammad Imran, Preetha Chatterjee, and Kostadin Damevski. 2024. Shedding Light on Software Engineering-specific Metaphors and Idioms. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*.

[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 (ICSE '24)*. Association for Computing Machinery, New York, NY, USA, Article 182.

[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*.

[30] Mia Mohammad Imran and Jaydeb Sarker. 2025. "Silent Is Not Actually Silent": An Investigation of Toxicity on Bug Report Discussion. *International Conference on the Foundations of Software Engineering* (2025).

[31] Maliha Jahan, Helin Wang, Thomas Thebaud, Yinglun Sun, Giang Ha Le, Zsuzsanna Fagyal, Odette Scharenborg, Mark Hasegawa-Johnson, Laureano Moro Velazquez, and Najim Dehak. 2024. Finding Spoken Identifications: Using GPT-4 Annotation for an Efficient and Fast Dataset Creation Pipeline. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*. 7296–7306.

[32] Nan-Jiang Jiang and Marie-Catherine de Marneffe. 2022. Investigating reasons for disagreement in natural language inference. *Transactions of the Association for Computational Linguistics* 10 (2022), 1357–1374.

[33] Jigsaw. n.d.. Perspective API Documentation. https://developers.perspectiveapi.com Accessed: 2025-05-30.

[34] Yova Kementchedjhieva and Anders Søgaard. 2021. Dynamic Forecasting of Conversation Derailment. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

[35] Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. [n. d.]. Decomposed Prompting: A Modular Approach for Solving Complex Tasks. In *The Eleventh International Conference on Learning Representations*.

[36] Irene Koshik. 2003. Wh-questions used as challenges. *Discourse Studies* 5, 1 (2003), 51–77.

[37] CMU STRUDEL Lab. [n. d.]. toxicity-detector. https://github.com/CMUSTRUDEL/toxicity-detector.

[38] Sharon Levy, Robert E Kraut, Jane A Yu, Kristen M Altenburger, and Yi-Chia Wang. 2022. Understanding conflicts in online conversations. In *Proceedings of the ACM Web Conference 2022*.

[39] 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* (2021).

[40] Zhenhao Li, Marek Rei, and Lucia Specia. 2022. Multimodal conversation modelling for topic derailment detection. In *Findings of the Association for Computational Linguistics: EMNLP 2022*.

[41] Mary L McHugh. 2013. The chi-square test of independence. *Biochemia medica* 23, 2 (2013), 143–149.

[42] Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika* 12, 2 (1947), 153–157.

[43] Courtney Miller, Sophie Cohen, Daniel Klug, Bogdan Vasilescu, and Christian Kastner. 2022. "Did You Miss my Comment or What?" Understanding Toxicity in Open Source Discussions. In *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*.

[44] 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*.

[45] Todd Morrill, Zhaoyuan Deng, Yanda Chen, Amith Ananthram, Colin Wayne Leach, and Kathleen Mckeown. 2024. Social Orientation: A New Feature for Dialogue Analysis. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*.

[46] Thi Huyen Nguyen and Koustav Rudra. 2024. Human vs ChatGPT: Effect of Data Annotation in Interpretable Crisis-Related Microblog Classification. In *Proceedings of the ACM on Web Conference 2024*. 4534–4543.

[47] Nicole Novielli, Fabio Calefato, Davide Dongiovanni, Daniela Girardi, and Filippo Lanubile. 2020. Can we use se-specific sentiment analysis tools in a cross-platform setting?. In *Proceedings of the 17th International Conference on Mining Software Repositories*.

[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*.

[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*.

[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] 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*.

[52] Farig Sadeque, Stephen Rains, Yotam Shmargad, Kate Kenski, Kevin Coe, and Steven Bethard. 2019. Incivility detection in online comments. In *Proceedings of the eighth joint conference on lexical and computational semantics*.

[53] Soumya Sanyal, Tianyi Xiao, Jiacheng Liu, Wenya Wang, and Xiang Ren. 2024. Are machines better at complex reasoning? Unveiling human-machine inference gaps in entailment verification. In *Findings of the Association for Computational Linguistics ACL 2024*. 10361–10386.

[54] Jaydeb Sarker, Asif Kamal Turzo, and Amiangshu Bosu. 2020. A benchmark study of the contemporary toxicity detectors on software engineering interactions. In *2020 27th Asia-Pacific Software Engineering Conference*. IEEE.

[55] Jaydeb Sarker, Asif Kamal Turzo, and Amiangshu Bosu. 2025. The Landscape of Toxicity: An Empirical Investigation of Toxicity on GitHub. *International Conference on the Foundations of Software Engineering* (2025).

[56] Jaydeb Sarker, Asif Kamal Turzo, Ming Dong, and Amiangshu Bosu. 2023. Automated identification of toxic code reviews using toxicr. *ACM Transactions on Software Engineering and Methodology* (2023).

[57] 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 2024 CHI Conference on Human Factors in Computing Systems*. 1–16.

[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. *Proceedings of the ACM on Human-Computer Interaction* 6, CSCW2 (2022), 1–27.

[59] Margrit Schreier. 2012. Qualitative content analysis in practice. (2012).

[60] Xiaoying Song, Sharon Lisseth Perez, Xinchen Yu, Eduardo Blanco, and Lingzi Hong. 2025. Echoes of Discord: Forecasting Hater Reactions to Counterspeech. In *Findings of the Association for Computational Linguistics: NAACL 2025*, Luis Chiruzzo, Alan Ritter, and Lu Wang (Eds.). Association for Computational Linguistics, Albuquerque, New Mexico.

[61] Florian Strohm and Roman Klinger. 2018. An empirical analysis of the role of amplifiers, downtoners, and negations in emotion classification in microblogs. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 137–146.

[62] Junyi Tian, Lingfeng Bao, Shengyi Pan, and Xing Hu. 2024. Analyzing and Detecting Toxicities in Developer Online Chatrooms: A Fine-Grained Taxonomy and Automated Detection Approach. In *Proceedings of the 31st Asia-Pacific Software Engineering Conference (APSEC 2024)*.

[63] Dong Wang, Masanari Kondo, Yasutaka Kamei, Raula Gaikovina Kula, and Naoyasu Ubayashi. 2023. When conversations turn into work: a taxonomy of converted discussions and issues in GitHub. *Empirical Software Engineering* 28, 6 (2023), 138.

[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] Michael Williams and Tami Moser. 2019. The art of coding and thematic exploration in qualitative research. *International management review* 15, 1 (2019), 45–55.

[66] Jiaqing Yuan and Munindar P Singh. 2023. Conversation Modeling to Predict Derailment. In *Proceedings of the International AAAI Conference on Web and Social Media*.

[67] Oleg Zendel, J Shane Culpepper, Falk Scholer, and Paul Thomas. 2024. Enhancing Human Annotation: Leveraging Large Language Models and Efficient Batch Processing. In *Proceedings of the 2024 Conference on Human Information Interaction and Retrieval*. 340–345.

[68] Justine Zhang, Jonathan Chang, Cristian Danescu-Niculescu-Mizil, Lucas Dixon, Yiqing Hua, Dario Taraborelli, and Nithum Thain. 2018. Conversations Gone Awry: Detecting Early Signs of Conversational Failure. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

[69] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, et al. [n. d.]. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. In *The Eleventh International Conference on Learning Representations*.

[70] Xiaodan Zhu, Svetlana Kiritchenko, Saif Mohammad, Xiaodan Zhu, and Colin Cherry. 2014. An empirical study on the effect of negation words on sentiment. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 304–313.

[71] Yiming Zhu, Zhizhuo Yin, Gareth Tyson, Ehsan-Ul Haq, Lik-Hang Lee, and Pan Hui. 2024. APT-Pipe: A Prompt-Tuning Tool for Social Data Annotation using ChatGPT. In *Proceedings of the ACM on Web Conference 2024*. 245–255.

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