Understanding & Mitigating Toxic Communication in Open Source Software Communities

Sahil Dadhwal (917598320) David Chu (916579612) Manami Nakagawa (924178393)
Jordan Penner (921536527) Haochen Dong (917051919)

1 Introduction & Motivation

Open-source software (OSS) communities are a critical part of the software community, with a diverse set of volunteer contributors of different backgrounds creating and improving tools used by developers worldwide [19]. These communities power much of today's digital infrastructure, from web frameworks to operating systems. However, just like any other social platform, users are subject to biases and emotional responses from other users [6, 7, 16, 18]. In some circumstances, communication within these platforms can become "toxic" in nature. Toxic communication can include, but is not limited to, passive-aggressive remarks, insults, and discriminatory language [10, 15]. Such toxic communication can create negative experiences for contributors or even maintainers, potentially leading to stress, burnout, and reduced participation in the project [12, 9].

While numerous studies have attempted to identify and categorize toxic communication in OSS communities, there is a noticeable gap on the measured impact of such communication on project outcomes [8, 10, 6]. This study aims to address this gap by investigating the effects of toxic behavior on contributor productivity and the longevity of OSS projects. By using open-source tools to classify toxic and non-toxic communications, we intend to reveal the quantifiable effects of toxic communication on a project's development trajectory. Our goal is to analyze different metrics to determine both project-wide activity and individual developer productivity trends across different OSS repositories.

Our research specifically examines: *(1) how toxic communication correlates with programmer productivity metrics*, *(2) whether patterns of toxicity increase around software release cycles*, and *(3) how contributor tenure levels relate to toxic communication.* Our motivation is that identifying these relationships will enable project maintainers to more effectively detect and address toxic communication, ultimately fostering healthier community dynamics and more sustainable open-source projects.

2 Background

2.1 Literature Review

Current literature on toxicity in OSS communities focuses on defining "toxicity" and quantifying it through language theory methods. As noted by Zhou et al. (2021), automating the detection of toxicity is difficult due to the complex and nuanced nature of human communication [20]. For example, passive-aggressive comments are difficult even for high-performance language models, as the surrounding context needs to be considered [11]. Miller et al. (2022) explore the causes of toxicity in OSS projects, classify them into different categories, and employ several well-defined strategies to sample "toxic" GitHub issues. These issues are then qualitatively analyzed to examine the relationship between the instigators and victims of toxic communication [10]. Raman et al. (2020) introduce a machine learning-based model, CMUSTRUDEL, that performs reasonably well in detecting toxic comments in GitHub repositories [12]. Zhou et al. (2021) introduce the challenges of debiasing existing toxicity classifier models and propose a data correction method to improve the fairness and accuracy of existing methods [20]. Tassabehji et al. (2024) propose using transactional analysis theory, a psychological framework focused on human communication, as a method of analyzing toxicity in a software development setting [17]. Sarker et al. (2024)

apply a large-scale stratified random sampling method on GitHub projects to analyze toxicity using a software engineering domain-specific toxicity detector, ToxiCR [15].

To further compare how toxicity in OSS projects differs from other online platforms such as X or Facebook, Miller et al. (2022) employ various strategies to ensure a diverse dataset of 100 toxic GitHub issues and qualitatively analyze them [10]. Miller et al. (2022) further examine the triggers for toxicity and the authors of the toxic issues. Most notably, Miller et al. (2022) categorize the toxic instances into different types of toxicity. Interestingly, Miller et al. (2022) found that comments that may have been classified as toxic by a machine learning model may simply be jokes or pranks between friends within a repository [10]. With the strong qualitative analysis presented in this paper, our project benefits from the findings that Miller et al. (2022) observed, as we can determine if different types of toxicity correlate with programmer productivity in OSS projects. However, Miller et al. (2022) note that language-based toxicity detectors trained on platforms other than GitHub may prove to be inaccurate for GitHub use, as they found that toxicity on GitHub is different from other social platforms [10].

Research on the effects of toxic communication has not been as strongly explored in the field of software development when compared to other fields. Interestingly, Raman et al. (2020) discovered that toxicity was decreasing steadily over time in the OSS community and that there was a correlation between the programming language choice for a code base and the level of toxic communication in the repository as they were training their model, CMUSTRUDEL [12]. Similar to Raman et al. (2020), our project not only proposes to use a classifier model to determine which comments are toxic, but to also quantify the toxicity in order to empirically determine if statistically significant relationships between toxicity and programmer productivity in OSS projects exist. Although the classifier employed by Raman et al. (2020) has yielded promising results, pre-trained sentiment analysis models are known to be poor at detecting passive-aggressive language [14] and there is no indication that Raman et al. (2020) address this shortcoming [13].

To mitigate the effects of biased associations on toxicity detectors and classifiers, Zhou et al. (2021) propose two debiasing methods [20]. First, Zhou et al. (2021) propose adversarial training with a "bias-only" model that contains predefined features that correspond to known biases for toxicity and a "full" model that contains all features. They posit that once training is complete, the "full" model will be fairer in classification tasks. Next, Zhou et al. (2021) propose data pre-processing and using a diverse dataset as another method of debiasing. Zhou et al. (2021) found that using a mix of "easy", "hard", and "ambiguous" samples (as measured through classification confidence) to train the model yielded better performance in classification tasks than only using "easy" samples for training. Although Zhou et al. (2021) give strong methods for debiasing toxicity classification tasks, it may be time consuming to pre-process and train a model if a large dataset is used.

Although toxic communication in OSS has not been extensively studied, research on its effects in software development work environments can provide valuable insights, as collaboration and communication are crucial for success in both instances. Tassabehji et al. (2024) used psychoanalytical transactional analysis (TA) theory to analyze how the misalignment of ego states and cross-trading leads to toxic behaviors that disproportionately affect women and other underrepresented groups [17]. Their TA-based OCTA-Pos model and STAR framework provide tools to diagnose and address toxic communication, highlighting the importance of understanding communication patterns to promote an inclusive environment. Although Tassabehji et al. (2024) mention power dynamics, they do not address how contributor experience affects toxic communication in the OSS community, highlighting a need to further investigate this factor.

To address the lack of large-scale studies on toxicity in OSS projects, Sarker et al. (2024) randomly selected 2,828 GitHub-based OSS projects and employed a software engineering domain-specific toxicity

detector, ToxiCR, to classify each comment [15]. Sarker et al. (2024) expand upon the works of Miller et al. (2022) and Raman et al. (2020), exploring toxic communication outside of specific contexts such as locked issues, and applying regression modeling to determine correlations between several variables that pertain to the project's characteristics, pull request (PR) context, and participant information [15]. Interestingly, Sarker et al. (2024) found that gaming projects are more likely to be toxic than non-gaming ones. With a focus on addressing the gaps in current studies on toxicity in OSS projects, Sarker et al. (2024) provide contemporary results and a novel approach to empirically analyze toxicity that this paper benefits from. However, Sarker et al. (2024) rely heavily on ToxiCR to classify if a comment is toxic or non-toxic, which means that classification may be unexplainable, if not difficult to explain, and that biases may be difficult to detect.

2.2 Research Questions

1. Does toxic communication in OSS communities negatively affect programmer productivity, which is measured through commits, issue resolutions, and discussion activity?

2. Is there a correlation between toxic communication and software releases?

3. How does the level of experience of the contributors (measured by the age of the account and previous contributions) correlate with their likelihood of engaging in toxic communication within OSS communities?

3 Methodology

3.1 Data collection

Our data collection methods have changed over the duration of our project. Initially, we fetched data from popular repositories (based on star counts) with the goal of getting a representative sample of data to analyze. This approach was impractical due to GitHub API rate limits restricting the amount of data we could extract a day. From the initial data collected, we learned that toxic communication is rare in randomly selected repositories, which made obtaining a representative sample size for a meaningful analysis difficult. Due to these constraints, we altered our approach to look at pre-collected repository data of repositories containing incivility. Additionally, to diversify our data, we used another source that contained pre-processed data. For this source, the data was sorted chronologically, so we manually selected dates that we assumed would contain toxicity. Although our data contain these limitations, we believe that they are sufficient for this report.

Our collected data originate from these two primary sources:

1. **Incivility Dataset:**
   We used the publicly available incivility dataset on GitHub, which identifies uncivil interactions in issue threads within GitHub OS projects [4]. This dataset was created by analyzing projects with at least 50 contributors and issues that were locked under the labels "too-heated", "off-topic", or "spam" between April 2013 and October 2023 [5]. The dataset contains 5,961 issue comments, with 1,365 specifically annotated as containing uncivil features. From this dataset, we extracted repository information to identify projects with documented instances of incivility.

2. **GHArchive Data**
   We supplemented the incivility data from GHArchive. GHArchive is a public archive of GitHub timeline events [3]. We selected specific dates within 2023-2024 for analysis, focusing on holidays, dates of major events (elections, Google I/O conference, etc.), and dates of important incidents

(CrowdStrike worldwide outage). The selection of these dates was under the assumption that time pressure around holidays, outcome of the presidential election, and problems from the CrowdStrike incident would result in a higher chance of having toxic communication than just choosing dates at random.

From the two sources, for each repository, we collected these metrics : Table 4

Table 1: Data Collection Metrics by Category

Data Category	**Collected Fields**
Comments on issues and pull requests	repo — comment id — user id — user login — created at — updated at — body — toxicity — type — issue number
Commit history	repo — sha — author id — author login — date — message
Issue creation and resolution data	repo — issue number — title — user id — user login — state — created at — closed at — comments count
Release information	repo — id — tag name — name — created at — published at — author id — author login
Contributor information	repo — user id — user login — contributions — account created at — public repos — followers

We used GitHub API to collect this data, utilizing a token rotation system to avoid hitting API rate limits. This approach allowed us to target repositories with known instances of toxic communication while also gathering the necessary metrics to evaluate our research questions regarding productivity, release patterns, and contributor experience. Table 2 summarizes the scale and scope of the dataset we collected.

Table 2: Summary of Data Collection

Metric	**Value**
Repositories Analyzed	501
- *From Incivility Dataset*	404
- *From GHArchive*	97
Total Comments Analyzed	63,668
Total Issues Collected	14,328
Total Commits Analyzed	37,126
Unique Contributors	6,780
Releases Tracked	1,597
Collection Period	2023-2024

Data collected from the Incivility Dataset and GHArchive. Toxicity detection performed using the unitary/toxic-bert model.

3.2 Toxicity Calculation

To detect and quantify toxicity, initially we utilized Google's Perspective API that allows any user to request a toxicity rating ranging from 0 to 1, where 1 is high toxicity, for a given string [1]. This tool is widely used in industry and is promising as a tool to determine on a nominal scale how toxic a communication can be. However, due to the small API rate limit, Perspective API was not a feasible option.

Instead, we implemented a pre-trained model. The new toxicity detection system, **unitary/toxic-bert**, assigns a toxicity score between 0 and 1, where 1 is high toxicity, to an input string [2]. We

conducted a few unit test comparisons with Perspective API scores on comments we generated as well as the unitary/toxic-bert scores and differences were insignificant. Given that, we will use this tool as our "source of truth" for how toxic a comment is determined to be.

From our current findings, the ratio of toxic to non-toxic comments is very small. With most scores having low toxicity, the toxicity distribution is skewed, which is why we used a percentile based approach to determine what is considered toxic, rather than a constant toxicity threshold that may be too high. This way, what is considered to be toxic is more dynamic with the amount toxicity present in the data. Specifically, we consider comments in the 90th percentile of toxicity scores within the dataset to be "toxic" for our analysis. This approach allows us to identify the relative most toxic comments within each community context.

3.3 Analysis of Data

As mentioned in Section 3.1, from all repositories that we consider analyzing, we extract the toxicity scores for each comment, as well as gather productivity metrics, such as comments on issues and pull requests, commit history, issue creation & resolution data, release information, and contributor information. After data collection, our analysis approach consisted of both *Exploratory Data Analysis (EDA)* and *Confirmatory Data Analysis (CDA)* techniques to address our research questions.

EDA: Our exploratory analysis was used to understand the distributions, patterns, and possible relationships in our collected data. Including creating visualizations for toxicity distributions in the dataset (as shown in Figure 1), examining patterns of toxicity with time relative to software releases, and considering potential correlations between contributor characteristics *(such as account age and follower count)* and toxicity level. The EDA phase helped us identify the appropriate statistical methods to control for in our models.

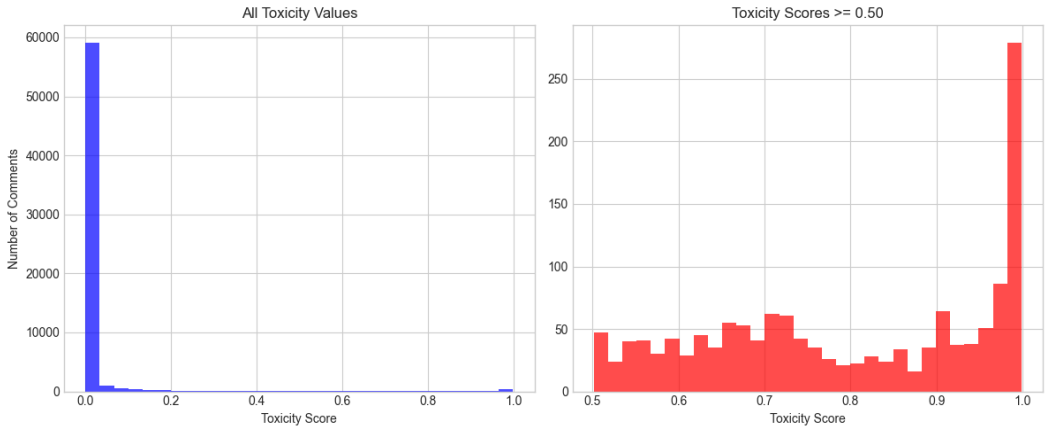


Figure 1: Distribution of toxicity scores

The insights we gathered for the EDA phase were able to connect to our hypothesis testing approach, as it helped us select suitable statistical methods and identify external factors that needed to be accounted for in our analysis.

CDA: Following the exploratory phase, we used analytical techniques to test the relationships were proposed in our research questions.

1. **RQ1: Toxicity & Productivity**
   We aggregated the toxicity scores and productivity measures (commits & issues created) per week. Then, we calculated both Spearman and Pearson correlation coefficients between the weekly proportions of toxicity and the productivity measures. We did time series plots to observe how patterns change over time and identify any notable relationships.

2. **RQ2: Toxicity & Releases**
   We analyzed toxicity levels before and after software releases, and we looked for any significant changes via paired t-tests. We also evaluated the correlation of between toxicity and time relative to release dates to determine if toxic communication spikes during release periods in projects.

3. **RQ3: Toxicity & Contributor Experience**
   We speparated contributors by account age into three experience groups:
   New (<1 year), Intermediate (1-3 years), and Experienced (3+ years)
   We then calculated correlation between toxicity and experience metrics (account age, contributions, and followers), creating visualizations to compare the toxicity patterns by experience levels.

For all correlation analyses, we implemented **Spearman's rank correlation coefficient** and **Pearson's correlation coefficient**. We consider p-values <0.05 to be statistically significant. The correlation method that gets used to determine the correlation will be the one that is statistically significant (which is dependent on the p-values). The approach to use EDA and CDA together helps to show unexpected trends in the data and verify our research questions, which ideally improves the validly of our finding.

4 Results

Following analysis from Section 3.3, these were some of our key findings & their relevance to the research questions. As shown in Table 3, all metrics we chose to analyze with toxicity had weak or no correlation.

Table 3: Correlation Coefficients Between Toxicity and Metrics

RQ	Metric	Spearman		Pearson		Finding
		rho	p-value	r	p-value	
1	Toxicity vs. Commits	-0.068	0.173	-0.126	0.011*	Weak negative
1	Toxicity vs. Issues Created	0.041	0.410	-0.106	0.032*	Weak negative
2	Toxicity vs. Days from Release	-0.000	0.993	0.006	0.743	No correlation
3	Toxicity vs. Account Age	-0.144	0.000*	-0.101	0.006*	Weak negative
3	Toxicity vs. Contributions	0.184	0.000*	0.029	0.465	Weak positive
3	Toxicity vs. Followers	-0.142	0.000*	-0.103	0.009*	Weak negative

* indicates statistically significant results (p <0.05)

4.1 RQ1: Toxicity & Productivity

Our analysis showed a **weak negative correlation** between toxicity and for both the productivity metrics (commits & issues created).

- In Figure 3a), for commits,
  the **Spearman correlation** was not significant (rho = -0.068 indicating a *weak negative correlation*, p = 0.173 indicating this relationship is *not statistically significant*).
  the **Pearson correlation** was significant (r = -0.126 indicating a *weak negative correlation*, p = 0.011 indicating this relationship is *statistically significant*).

- In Figure 3b), for issue creations,
  the **Spearman correlation** was not significant (rho = 0.041 indicating a *very weak positive correlation*, p = 0.410 indicating this relationship is *not statistically significant*).
  the **Pearson correlation** was significant (r = -0.106 indicating a *weak negative correlation*, p = 0.032 indicating this relationship is *statistically significant*).

4.2 RQ2: Toxicity & Releases

Looking into the relationship between toxicity levels and releases, we were able to observe changes in communication tone around the release events. Figure 2 displays toxicity levels before and after recent software releases across the latest 25 days in the dataset.

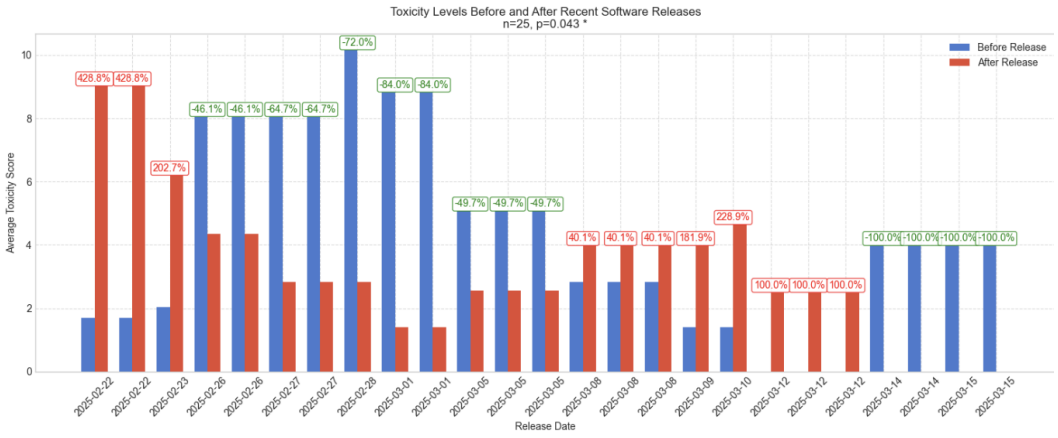
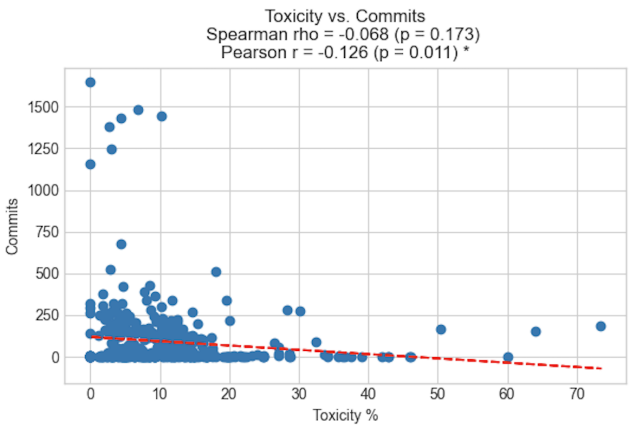
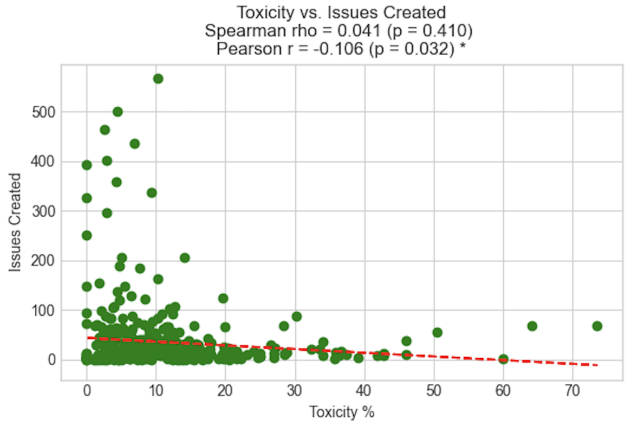


Figure 2: Toxicity before & after release (most recent 25 days)

Intuitively, from the data we can see a lot of variance, where some repositories show a significant growth in toxicity after a release, whereas others show a great decrease.
Statistically, our analysis showed **no correlation** between toxicity and days relative to a release.



(a) Correlation between toxicity and commits



(b) Correlation between toxicity and issues created.

Figure 3: Correlation between toxicity and project activity metrics.

- In Figure 4), we showed that,
  the **Spearman correlation** was not significant (rho = -0.000 indicating *no correlation*, p = 0.993

indicating this relationship is *not statistically significant*).
the **Pearson correlation** was not significant (r = 0.006 indicating a *weak correlation*, p = 0.743 indicating this relationship is *not statistically significant*).

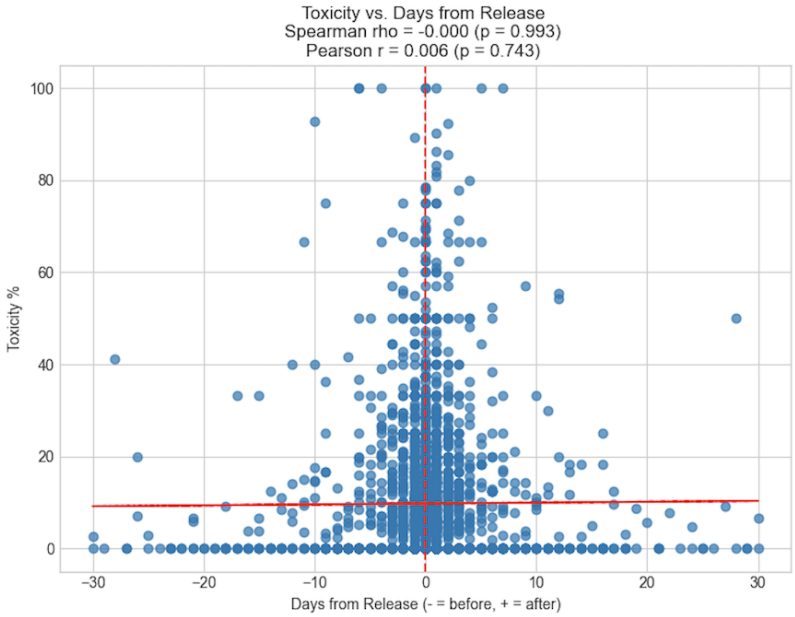
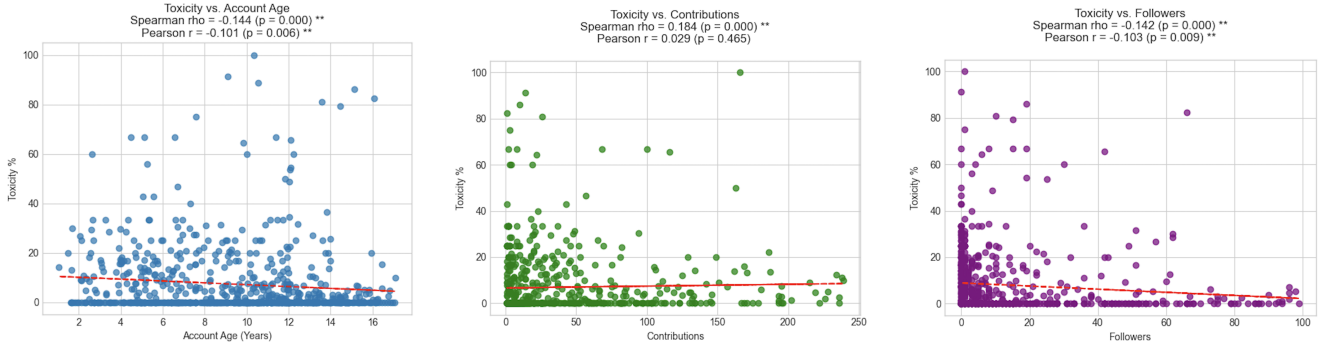


Figure 4: Correlation between toxicity and time (relative to release)

4.3 RQ3: Toxicity & Contributor Experience

After looking into the relationship between toxicity and experienced contributors (via account age, contribution count, and follower count) we found weak correlations.



(a) Correlation between toxicity and account age

(b) Correlation between toxicity and contributions count

(c) Correlation between toxicity and followers count

Figure 5: Correlation between toxicity and contributor experience

Our analysis showed a **different correlations** between toxicity and the contributor experience metrics (account age, contributions count, and followers count).

- In Figure 5a), for account age,
  the **Spearman correlation** was significant (rho = -0.144 indicating a *weak negative correlation*, p = 0.000 indicating this relationship is *statistically significant*).

the **Pearson correlation** was significant (r = -0.101 indicating a *weak negative correlation*, p = 0.006 indicating this relationship is *statistically significant*).

- In Figure 5b), for account contributions,
  the **Spearman correlation** was significant (rho = 0.184 indicating a *weak positive correlation*, p = 0.000 indicating this relationship is *statistically significant*).
  the **Pearson correlation** was not significant (r = 0.029 indicating a *weak positive correlation*, p = 0.465 indicating this relationship is *not statistically significant*).

- In Figure 5c), for follower count,
  the **Spearman correlation** was significant (rho = -0.142 indicating a *weak to moderate negative correlation*, p = 0.000 indicating this relationship is *statistically significant*).
  the **Pearson correlation** was significant (r = -0.103 indicating a *weak negative correlation*, p = 0.009 indicating this relationship is *statistically significant*).

5 Threats to Validity

Analyzing the toxicity of every comment from every open and closed issue from the near 300 million repositories on GitHub would be an infeasible task, as others have noted [15]. We use public datasets and use a subset of repositories to generate our datasets. Doing so will inevitably introduce sampling bias, which we attempted to mitigate by randomly selecting repositories where possible and maximizing the size of the sample size. Future research into this subject may benefit from further increasing the number of sampled repositories and issue threads. There is also an inherent bias in the determination of whether a comment is considered "toxic" or not. We use third-party tools to remove subjectivity in such a determination where possible, but we cannot guarantee the infallibility of the tools. As previously mentioned in the Literature Review, defining toxicity and categorizing a comment as toxic is a difficult task, as context can completely change the intended reception of a comment that may appear toxic at a glance or to an unsophisticated measurement tool, as shown in Table 4 where we had our unitary/toxic-bert model give the toxicity scores of comments that require sentiment context, and it returned incorrect values.

Table 4: Examples of Comments with Their Toxicity Scores

Comment	**Toxicity Score**
"Shut up! You're joking!"	0.919
"You're insane for pulling that off!"	0.947
"You're really smart for someone like you."	0.002
"Nobody cares about your opinion."	0.059

Toxicity scores range from 0 to 1, with higher values indicating more toxic content as classified by the unitary/toxic-bert model.

Our work may be threatened by external validity as well. Methodologies in project development, the project's contributors, and the nature of the project itself may significantly impact many of the metrics that were analyzed in this study. As with sampling bias, large randomized sample sets are our main tools to mitigate this inherent bias.

6 Discussion

6.1 Principal Outcomes

The analysis we conducted provides us with several conclusions related to our research questions:

RQ1: Does toxic communication in OSS communities negatively affect programmer productivity, measured through commits, issue resolutions, and discussion activity?
Our results showed weak negative correlations between toxicity and programmer productivity (measured by commits and issue creations) as shown in Table 3. Even though it is statistically significant, the weak correlations point towards toxic communication having a very limited impact on productivity. This contradicts our initial assumption that toxicity has a big negative impact on programmer productivity.

RQ2: Is there any correlation between toxic communication and software releases?
From our research, we found no significant correlation between toxicity and release dates as shown in Table 3. This shows that release pressure does not necessarily coincide with toxic communication. However, our visual analysis showed a lot of variance, with some repositories experiencing more toxicity after releases while others showed less toxicity, which tell us that factors within the projects themselves may influence how releases affect communication.

RQ3: How does the level of experience of the contributors (measured by the age of the account and previous contributions) correlate with their likelihood of engaging in toxic communication within OSS communities?
From our analysis, we noticed mixed results when talking about correlation of toxic communication with contributor experience. For account age specifically, it showed a weak negative correlation with toxicity, which tells us that newer accounts might be slightly more toxic. However, for contribution count, it shows a weak positive correlation with toxicity, and follower count showed a weak negative correlation as shown in Table 3. These results let us know that there are many factors to consider in the relationship between experience and toxic behavior, highlighting the need for further research.

Overall, the effects and correlations of toxic communication on the metrics we analyzed were very limited.

6.2 Comparison to Prior Studies

There has been limited attention on the effects of toxic communication on productivity in OSS projects, where collaboration and communication are key to success. Furthermore, there has been a lack in the exploration of the patterns of toxic communication, such as whether they increase during high-pressure periods, such as upcoming software releases. Understanding these patterns can reveal the stressors that trigger toxic behaviors and aid in creating mitigation strategies. Finally, the relationship between contributor experience and the likelihood of engaging in toxic communications has not been thoroughly studied. While some studies have examined productivity metrics in OSS, such as code commit frequency and issue resolution times, limited research links these metrics to toxic communication patterns. Our research fills in these gaps, by examining the impact of toxic communication on productivity, its patterns, and the role of contributor experience in shaping communication dynamics in OSS projects.

Our investigation into toxicity in OSS projects examined the relationship between toxicity, productivity, release dates, and contributor experience. We found that toxicity has a weak negative correlation with commits and issue creation, which contradicts the findings of Sarker et al. (2025) and Raman et al. (2020) [12, 15]. Raman et al. (2020) found that toxicity was correlated with poor-quality code changes, while Sarker et al. (2025) found that toxicity increases with larger code changes [15]. Although we do not measure the quality or length of code changes, we examine the quantified toxicity scores before and after release dates, which had varying levels of toxicity. With further exploration into this, we expect to be able to determine underlying causes for this variance in toxicity. We also found that contributor experience has no correlation with toxicity, as determined through account age, number of contributions, and follower

count, which aligns with the findings of Miller et al. (2022) [10]. In their findings, Miller et al. (2022) observed that the authors of toxic comments ranged from new users to even project maintainers, while Sarker et al. (2025) further expand upon this with their findings that authors of past toxic comments were more likely to repeat their behaviors [10, 15]. Because we want to determine the triggers for the toxicity in general, repeat toxic comments by the same author are not explicitly examined in detail.

6.3 Future Work

While little correlation was found between commit frequency and the presence of toxicity in issue discussions, the quality of commits (via lines of code, number of bugs introduced) may be impacted by toxicity as well, or some contents were labeled as toxicity but the actual situation was in a contrary way. The current toxicity scoring relies heavily on automated analysis of pre-trained model (unitary/toxic-ber [2]) which may not be able to fully capture contextual and emotional nuances. Further study could look into the correlation between toxicity and these metrics, such as allowing some comments to be labeled as highly toxic because of the language they used, but the actual content may be a heated discussion of technical issues rather than a personal attack.

Another unexplored metric that could be impacted by toxicity is the long-term sustainability of the project. We could attempt to observe a group of repositories with similar characteristics (such as size, number of contributors, etc), and study whether the overall toxicity in issue discussions impacts a project's ability to continue to evolve and fix bugs over longer periods of time. Future research could use mixed method approaches that combine, for example, quantitative analysis with qualitative interviews to get a deeper understanding of the motivations/effects of toxic communication, which show us common trends and correlations withing these repositories with toxic communication.

6.4 Conclusion

In this project, we investigated how toxic communication affects the activities and productivity of developers in open source communities. The analysis showed that toxic interactions do not directly affect productivity data, specifically the commit frequency or issue resolution times. It also shows that there may be more serious underlying conflicts. One of the main results was that senior contributors tended to be more involved in strong toxic interactions. This might be because they are more experienced and in turn more emotionally involved in the project, or they just do not have the patience to deal with a beginner's incompetence. Our results show us the importance of establishing appropriate conflict management mechanisms, such as developing a code of conduct or having a project maintainer to oversee and handle toxic communication conflicts. It is imperative for managers and contributors to be involved in more constructive/collaborative discussions in order to keep a productive open-source community. However, our study has limitations. First, the sample size of repositories we considered was limited and did not cover the entire open-source project. For future research, we need to expand the data collection method to cover more projects and improve the generalizability of our findings. There are also issues with the accuracy of the tool for evaluating toxicity. There is a possibility that some comments will be misclassified by the automatic scoring tool, so future research should combine manual annotation to improve the accuracy of the scores. In addition, other factors like project type and team size were not fully controlled, so in the future, the strength of the conclusions can be improved with multivariate analysis. In summary, this study has provided new perspectives for understanding the role of toxic communication in open-source communities and lays the foundation for future research and practice. By further exploring the motivations, impacts and coping strategies of toxic communication , it can provide theoretical support and practical guidance for efficient collaboration in open source communities.

7 Team Membership and Attestation

Sahil Dadhwal (Data Fetching, Data Stats/correlations/EDA/CDA, Data Analysis, Data Visualization, New ToxicityModel, RQ's,), Facilitate group, Revised: [Intro & Motivation + Lit Review + Threats to validity + Comparison to prior studies + Future work + Conclusion...Wrote:[RQ's, Methodology, Results, Discussion Principle Outcomes]
Jordan Penner (Data Fetching and Analysis, Revised Introduction and Motivation section, contributed to Results and Threats to Validity section),
Manami Nakagawa (Data Source finding, Data Fetching, Data Analysis, Data Visualization, Writing Revision),
David Chu (Literature Review, Comparison to Prior Studies, Proofreading, Reference Gathering)
Haochen Dong (Introduction, Literature Review, Result, Principal Outcomes, Future Work, Conclusion)

participated sufficiently and agree to the contents of this report.

Code & Data Availability Statement

All code used to fetch data and perform analysis can be found at https://github.com/pennerj6/260-project. Available data is located in the `data/` subdirectory but some large data files were excluded to avoid repository bloat.

References

[1] Perspective api. `https://www.perspectiveapi.com/`.

[2] unitary/toxic-bert. `https://huggingface.co/unitary/toxic-bert`.

[3] Github archive. `https://www.gharchive.org/`, 2023.

[4] Ramtin Ehsani, Mia Mohammad Imran, Robert Zita, Kostadin Damevski, and Preetha Chatterjee. Incivility dataset. `https://github.com/vcu-swim-lab/incivility-dataset/`, 2024.

[5] Ramtin Ehsani, Mia Mohammad Imran, Robert Zita, Kostadin Damevski, and Preetha Chatterjee. Incivility in open source projects: A comprehensive annotated dataset of locked github issue threads. In *Proceedings of the 21st International Conference on Mining Software Repositories*, MSR '24, page 515–519, New York, NY, USA, 2024. Association for Computing Machinery.

[6] Ramtin Ehsani, Rezvaneh Rezapour, and Preetha Chatterjee. Exploring moral principles exhibited in oss: A case study on github heated issues, 2023.

[7] Ramtin Ehsani, Shadi Rezapour, and Preetha Chatterjee. Analyzing toxicity in open source software communications using psycholinguistics and moral foundations theory, 2024.

[8] Mia Mohammad Imran and Jaydeb Sarker. "silent is not actually silent": An investigation of toxicity on bug report discussion, 2025.

[9] Johan Linåker, Georg J. P. Link, and Kevin Lumbard. Sustaining maintenance labor for healthy open source software projects through human infrastructure: A maintainer perspective, 2024.

[10] Courtney Miller, Sophie Cohen, Daniel Klug, Bogdan Vasilescu, and Christian Kästner. "did you miss my comment or what?" understanding toxicity in open source discussions. In *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*, pages 710–722, 2022.

[11] Shyamal Mishra and Preetha Chatterjee. Exploring chatgpt for toxicity detection in github, 2023.

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

[13] Jaydeb Sarker. Identification and mitigation of toxic communications among open source software developers. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, ASE '22, New York, NY, USA, 2023. Association for Computing Machinery.

[14] Jaydeb Sarker, Asif Kamal Turzo, and Amiangshu Bosu. A benchmark study of the contemporary toxicity detectors on software engineering interactions. *CoRR*, abs/2009.09331, 2020.

[15] Jaydeb Sarker, Asif Kamal Turzo, and Amiangshu Bosu. The landscape of toxicity: An empirical investigation of toxicity on github, 2025.

[16] Sayma Sultana, Gias Uddin, and Amiangshu Bosu. Assessing the influence of toxic and gender discriminatory communication on perceptible diversity in oss projects, 2024.

[17] Rana Tassabehji, Hugh Lee, and Nancy Harding. Problematic workplace behaviours in the software development profession: Using transactional analysis to diagnose toxicity and improve relationships at work. *Group amp; Organization Management*, 49(6):1454 – 1494, 2024.

[18] Bianca Trinkenreich, Igor Wiese, Anita Sarma, Marco Aurélio Gerosa, and Igor Steinmacher. Women's participation in open source software: A survey of the literature. *CoRR*, abs/2105.08777, 2021.

[19] Bogdan Vasilescu, Daryl Posnett, Baishakhi Ray, Mark G.J. van den Brand, Alexander Serebrenik, Premkumar Devanbu, and Vladimir Filkov. Gender and tenure diversity in github teams. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, page 3789–3798, New York, NY, USA, 2015. Association for Computing Machinery.

[20] Xuhui Zhou, Maarten Sap, Swabha Swayamdipta, Noah A. Smith, and Yejin Choi. Challenges in automated debiasing for toxic language detection, 2021.

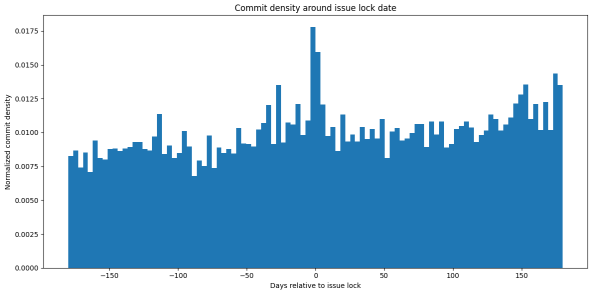
A Appendix:

A.1 Other Analysis Attempts

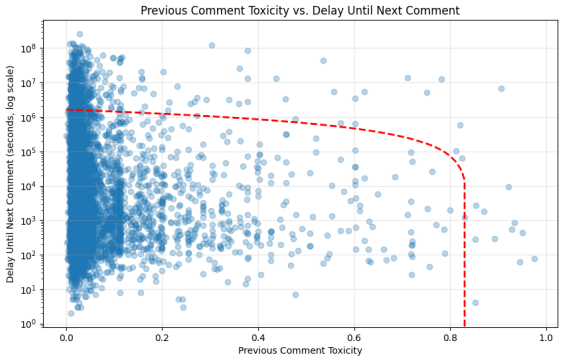
Although we tried other analyses, the graphs did not show the strong results to answer or discuss the research question.

A.1.1 Attempt 1

We also attempted to analyze using Incivility Dataset [5], as shown in Figure 6. Although nothing could be inferred from Figure 6(a), we observed from Figure 6(b) that there is a tendency for more toxic comments to receive quicker responses.



(a) Commit density around issue lock date

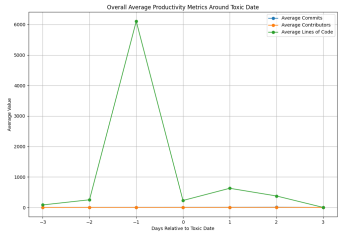


(b) Comment toxicity vs delay until next comment

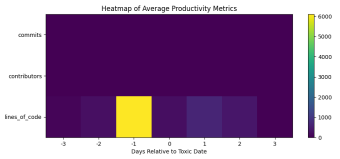
Figure 6: Attempt 1

A.1.2 Attempt 2

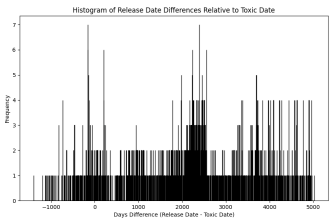
We attempted to draw the graph using the data set released in this paper[12], as shown in Figure 7. However, we were unable to obtain results that would be worth discussing. From Figure 7 (a) and (b), it can be seen that "Lines of Code" is related to the day on which toxicity occurred; however, it was found that there were only large outliers.



(a) Average Productivity Metrics Around Toxic Date



(b) Heatmap of Average Productivity Metrics



(c) Release - Toxic Date Difference

Figure 7: Attempt 2

B Disclaimer on the Use of AI

Large Language Models are a very useful tool, which we utilized in our report. ChatGPT was consulted to perform the following tasks, or tasks in a similar regard, for our project:

1. GitHub Code.
   (a) Preliminary code generation and scaffolding. Example: LLMs were used to generate boilerplate URL fetching code, which was then tuned to ensure the desired parameters were correct.
   (b) Used to adjust/change code implementation to include more robust error handling/debugging (with logger statements, try except exception blocks, clear code comments for non-programmers to follow, etc)
   (c) Used to implement GitHub API token rotation so that 1 user can utilize multiple API tokens to avoid rate limit.
   (d) Used to implement rate limit handling (example: with max retry blocks/attempts)

(e) Used to consult any bugs, prompted as such: "Why does this line "datetime.datetime.now()"give this error?" solution -¿ import datetime

(f) Used to create helper functions, such as save data to CSV or get data from CSV

2. Ideas for analysis and presentation of acquired data.

   (a) Example: for Figure 3a, an LLM was used to brainstorm ideas to properly weight repositories when comparing those with fewer commits to those with significantly more.

3. Proofreading.

   (a) LLMs were used as a tool to aid in proofreading drafts, correcting grammar, recommended phrase changes.

   (b) Also used to help condense paragraphs that were too long. Prompted - "How can I make this paragraph shorter without leaving any information out"

4. Latex Code.

   (a) LLMs were used to develop latex code for this report. (Example, an LLM was prompted to create a table in latex given the parameters we fed it, and from there we tuned it to match our needs.) The blueprint for latex code to make a table structure was generated by AI, and used/modified for all tables in this report.

5. AI Check

   (a) As suggested by the TAs, AI was used to check for instances of AI. We found it to be very inaccurate, but still tried to work with it.