Why Did your PR Get Rejected? Defining Guidelines for Avoiding PR Rejection in Open Source Projects

Nick Papadakis, Ayan Patel, Tanay Gottigundala, Alexandra Garro, Xavier Graham, Bruno da Silva
California Polytechnic State University
San Luis Obispo, CA, United States
{npapadak,apatel60,pgottigu,agarro,xgraham,bcdasilv}@calpoly.edu

ABSTRACT

Pull requests are a commonly used method of collaboration for software developers working on open source projects. In this paper, we analyze the most common reasons, sentiment polarity, and interaction length for pull request rejections, as well as the correlations between these factors in a large open-source project. We manually analyzed 231 rejected pull requests in Scapy, an open-source Python network tool and library hosted on GitHub, and systematically mapped sentiment and categorized rejection reasons. We found that the most frequent reasons for pull request rejection do not relate to internal code quality. Instead, the main reasons refer to source code management issues, incomplete comprehension of project functionalities, poor understanding of what reviewers expect, and misunderstanding the project guidelines (often due to a lack of complete/updated instructions and communication gaps). This work is an ongoing effort toward establishing practical guidelines for globally distributed contributors in open-source projects to minimize pull request rejection and maximize productivity leading to more fruitful remote collaboration. Future work involves expanding the analysis to more projects and incorporating quantitative methods.

CCS CONCEPTS

• **Software and its engineering → Programming teams**.

KEYWORDS

pull request, code review, open source software, developer communication

ACM Reference Format:
Nick Papadakis, Ayan Patel, Tanay Gottigundala, Alexandra Garro, Xavier Graham, Bruno da Silva. 2020. Why Did your PR Get Rejected? Defining Guidelines for Avoiding PR Rejection in Open Source Projects. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

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

1 INTRODUCTION

Software development is a collaborative effort that strongly relies on human aspects and people interaction, either remotely or in person. With recent advancements on source code management tools, many software projects today, and especially the open-source ones, use the pull request method to manage how remote contributors globally distributed collaborate to add or enhance features and fix bugs.

By building a better understanding of why pull requests are rejected, we could determine practical guidelines to help contributors avoid common pitfalls and better bridge the communication gap of globally distributed developers, which is a typical scenario in open-source projects. Open-source projects have become more commonplace in recent years, from small and impactful projects to large open-source ones, such as React Native, VSCode, and Tensorflow from big tech companies (e.g., Facebook, Microsoft, and Google). Even when open-source projects have development teams at large companies, they also allow for individual contributors to add to the project. Many of these projects establish contribution guidelines so that people have a better chance of getting their pull requests approved, but it is often still unclear what exactly causes a pull request to be rejected. Other researchers have analyzed pull request data but they either did not focus especially on pull request rejections [1] [7] or did not provide a qualitative analysis on rejection reasons [8] that could go beyond code quality [6] [2].

Therefore, in this paper, we focus on why pull requests in a large open-source project get rejected. We manually analyzed 231 rejected pull requests in Scapy, an open-source Python network tool and library hosted on GitHub, and systematically classified sentiment and categorized rejection reasons. We found that the most frequent reasons for pull request rejection do not relate to internal code quality. The main reasons refer to source code management issues, incomplete comprehension of project functionalities, poor understanding of what reviewers expect, and misunderstanding the project guidelines (often due to a lack of complete/updated instructions which leads to communication gaps).

By conducting a systematic and replicable qualitative analysis on pull request rejections, we intend to complement recent results already published in the literature. Also, this work is an ongoing effort toward establishing practical guidelines for globally distributed contributors in open-source projects to minimize pull request rejection and maximize productivity leading to more fruitful remote collaboration. Future work involves expanding the analysis to more projects and incorporating quantitative methods.

In the next section, we discuss related work. In Section 3, we describe the research questions that guide our study as well as other methodological aspects including data collection, the calibration

phase for rejection classification, final PR analysis and conflict resolution. Then, we present our results in Section 4. In Section 5, we discuss threats to validity, and in Section 6 we make final remarks and present future work.

2 RELATED WORK

In [3], Pletea et al. analyze sentiments of security related pull requests on GitHub. They found that security related pull requests had more negative sentiment than discussions on other topics. They concluded that developers need to be trained to address security concerns to reduce frustration and improve the overall atmosphere in the project. Security related pull requests accounted for only 10 percent of all the pull requests on GitHub. However, this idea can be applied to almost all pull requests on GitHub to help understand why certain requests contain a positive or negative sentiment. In our work, we analyze developer sentiment on conversations contained in all types of rejected pull requests, not only security.

In [4], the authors look at how developers can change the mood of other developers through comments and feedback and how this changes their behavior. The results show that the mood of developers was affected by feedback given to pull requests within a day of posting. In addition, there was a percentage of developers that did not contribute again after they received negative feedback on their pull request. This percentage increased when looking at contributions with more lines of code. This shows that constructive feedback should be valued more and should be included in the contribution guidelines for a project. In our work, besides analyzing pull request rejection reasons, we also analyze whether or not sentiment polarity has an association with the rejection.

In [2] and [6], researchers look at whether code quality plays a determinant role on the acceptance of a pull request. Unexpectedly, both studies did not find code quality as the most influential factor for pull request acceptance. In [2], they did find that the most important factor to whether a pull request was accepted or not was the reputation of the developer submitting the pull request. In our work, we look at other factors beyond code quality by manually analyzing developer conversation in the pull request comments.

In [10], Terrell et al. discuss gender bias in the acceptance of pull requests. Their results show that women's contributions tend to be accepted more often than men's. However, women's acceptance rates are higher only when they are not identifiable as women. The results suggest that although the average contributing woman on GitHub may be more competent than the average contributing man, there is still bias against her. This can be tied back to the previous paper in terms of the developer's reputation having a role in the acceptance of the pull request. The takeaway from this is that who the developer is should not matter as much as it does. Our work does not look specifically to gender bias. Hence, that is something we can incorporate in the future and then triangulate with their results.

In both [1] and [7], the authors focus on acceptance factors. While in [1] the attention is given to the reviewers' perspective, in [7] they applied quantitative techniques to extract association rules that lead to pull request acceptance. Instead, we take a qualitative approach on manually analyzing only rejected pull requests to build an understanding on rejection reasons that might be difficult to be captured in applying solely quantitative methods. In [**?**], the authors particularly focused on pull request rejection, but still their analysis concentrated only on quantitative aspects from applying data mining and their dataset included only pull requests from internal contributors.

Ultimately, we claim that our manual analysis and qualitative approach pull request conversations, although initially applied to only one project, is effective for building a broad and complete understanding on rejection reasons. Thus leading to practical guidelines for future contributions and fruitful developer collaboration.

3 RESEARCH METHODOLOGY

To guide our research, we decided to focus our effort on the following questions:

RQ1: What are the most frequent reasons why pull requests are rejected in open source projects?

RQ2: Is there a correlation between the reasons why a PR is rejected and the sentiment of the interaction between the reviewers and the author?

RQ3: Does the interaction length on a pull request (i.e. # of comments) correlate with the reason for rejection or the sentiment?

3.1 Target System: Scapy

For this study, we sought to select an active repository on GitHub from an impactful project that is vastly used and receive contributions from developers beyond the core team. We wanted to use a project that had more than a trivial number of rejected pull requests, but at the same time few enough that it would be feasible to review all rejections manually and at least twice. After establishing these goals, we performed an informal search, examining different GitHub repositories that most of us had used before. We decided on analyzing the rejected pull requests of Scapy, a Python-based packet manipulation program and library. It runs on Python versions 2.7, 3.4, and 3.7, and Linux, OSX, *BSD, and Windows platforms. At the time we write this paper, under the GitHub platform, the Scapy repository is used by other 2k repositories, forked 1.1k times, and has 4.8k stars. At the time of data collection, November 7th 2019, Scapy had 231 rejected pull requests, and the most recent rejected pull request had been closed on October 28th 2019 [5].

Scapy includes guidelines for open-source contributors to read before creating a pull request in "CONTRIBUTING.md". By the time we wrote this paper, their guidelines cover the following topics:

Coding Style and Conventions. The authors require all code to be PEP-8 compliant, recommend using Pylint to assist in writing good Python code, recommend reading the Google Python Style Guide, and ask that developers avoid creating large unnecessary list objects. They also note that much of the legacy code does not comply with these guidelines, but they require new contributions to.

Tests. The authors ask contributors to consider including tests for new features they develop or to trigger the bug they are trying to fix to help catch regressions.

New Protocols. The authors give instructions of where in the project support for new networking protocols should be added. They also provide a link to a detailed document on the design patterns to use when implementing support for a new protocol.

Features. The authors describe where to implement new features for existing protocols.

Core. The authors remind that contributions to Scapy's core must be very careful with its performance and memory footprint, as inefficient code added to core functionalities would "have a disastrous effect on Scapy's performances".

Python 2 and 3 Compatibility. The authors note that the project aims to run on both Python 2 and Python 3. They provide some rules to help contributors write code that is cross-compatible.

Code Review. The authors note that the project maintainers tend to be picky and that "you might be feel frustrated that your code (which is perfectly working in your use case) is not merged faster". They ask that contributors not be offended and keep in mind that the maintainers are trying to keep in mind code readability, maintainability, commit history, performance, integration, and API consistency with each commit.

3.2 Data Collection

In this research, we manually analyzed a total of 213 rejected pull requests. While analyzing the rejected pull requests, we examined and classified the sentiment of the rejected pull request and the reason the pull request was rejected. Sentiment was classified as positive, neutral or negative and was classified based on the comments in the pull request. A positive sentiment for a pull request could be due to words of appreciation, whereas a negative sentiment would include pull requests where foul or discouraging language is used. Neutral sentiment is where the comments did not express positive nor negative sentiment. A snapshot of a comment from a rejected pull request is seen in Figure 1. The reason this pull request was rejected was categorized as 'needs testing' and the sentiment of the pull request was labeled as positive. Figure 2 is an example of a negative sentiment and was rejected due to 'ego issues'. In this figure, the user under the name 'cclauss' mocks 'guedou', a core contributor to the project, for asking him to remove one change. As for the reason each pull request was rejected, most reasons were discovered during the calibration phase, but others were also added later in the process. For all reasons found, see Subsection 3.3.

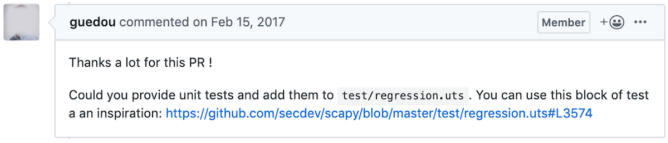

Figure 1: Snapshot of a rejected pull request that was categorized as 'needs testing' and of positive sentiment

In Figure 3, we show that the research began with a calibration phase. This was essential to ensure each researcher was analyzing and classifying each rejected pull request in the same manner. During this first step, each researcher reviewed the same 25 rejected pull requests (about 11% of our dataset). While reviewing the rejected pull requests, each researcher classified the sentiment of the pull request and the reason the pull request was rejected. For any conflict that occurred, a discussion took place amongst all five researchers until a final conclusion was unanimously agreed upon.

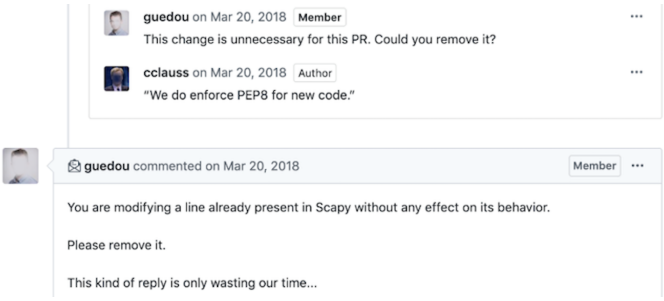

Figure 2: Snapshot of a rejected pull request that was categorized as 'ego issues' and of negative sentiment

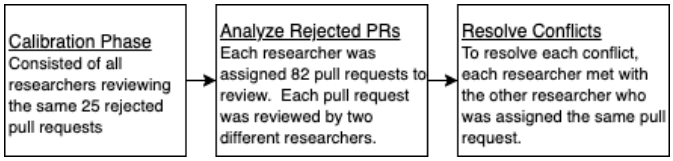

Figure 3: Process flowchart

After the calibration phase, we divided the remaining rejected pull requests in a way that each pull request was assigned to two researchers. Having each pull request reviewed twice helped reduce the likelihood of one researcher's bias impacting the final classifications excessively. This resulted in each researcher reviewing 82 rejected pull requests. Once each researcher finished classifying their assigned rejected pull requests, any conflicts were resolved during a discussion amongst the researchers.

3.3 Data Categorization

We had three main metrics that we used to measure pull request rejection data: reasons, sentiment, and interaction length. The reasons and the sentiment in rejected pull requests were both categorical data, while interaction length was numerical.

Sentiment. The pull request sentiment varied in three levels: positive, negative, or neutral. In the previous subsection we describe how we manually classified sentiment while also identifying rejection reasons.

Reasons for Rejection. The categories and subcategories for rejection reasons are listed in Table 1 together with the number of occurrences according to our classification. There are three rows in the table that we starred. Those are the reasons we considered "irrelevant", since we cannot draw any relevant conclusions. For example, a rejected pull request that was closed accidentally is not useful to draw conclusions that will help us create guidelines for open-source contributors. We identified several PRs that were closed by the author and they were not considered relevant either since there are no apparent reasons or comments documenting why the author closed the PR.

Interaction Length. We quantified the length of the interaction by counting the number of comments in each pull request thread. We chose to use this instead of word count since the individual comments often included code snippets, which artificially spiked

Table 1: Reasons of PR Rejections

Category	Subcategories	Occurrences
Version control issues	**Code rebase**	**34**
	PR conflict	**44**
	PR too large to rebase	2
Side effect issues	Fails tests	3
	Breaks compatibility	1
	Other unintended side effects	10
Unnecessary issues	Issues not reproducible	2
	Minor change	7
	Unnecessary functionality	**18**
	Unnecessary refactoring	3
Code issues	Wrong code location	3
	Needs testing	**13**
	Bad code formatting	3
	Wrong logic	1
Author issues	Closed-by-author(*)	68
	Author unable to fix issues	**11**
	Ego issues	3
	No description (*)	1
	Closed due to inactivity	**12**
	Closed accidentally (*)	2

the word count and led to a high occurrence of outliers. We also considered using the time it took from the beginning of the pull request to the time it was closed. However, we felt that this was not an effective way to measure pull requests in an open-source project since many contributors are implementing their changes in their own free time without necessarily receiving compensation.

4 RESULTS

RQ1: What are the most frequent reasons why pull requests are rejected in open source projects?

We highlight in Table 1 the most frequent PR rejection reasons in our dataset. Sixty eight PRs were labeled as "closed-by-author", meaning that those PRs were not actually rejected by reviewers. Since we cannot determine the exact reasons why those PRs were closed we take those PRs out of our analysis. Also, we eliminated the other two "irrelevant" reasons – "no description" with only one occurrence and "closed accidentally" with two occurrences. This leaves us with a total of 159 PRs for the following analysis[1].

Version control issues was the most frequent category within which "PR conflict" (27.6%) and "Code rebase" (21.3%) were the most common reason of rejected pull requests. A PR conflict was classified when there was another existing pull request that solved the same issue, but the author was not aware of its existence. Code rebase meant that if the code belonged somewhere else, or if only part of the PR was needed, a new PR was created and possibly merged, leaving the old PR closed and unmerged. The third most common rejection reason was "Unnecessary functionality" – 11.3% of the PRs were rejected because they were either implementing unnecessary new functionality or submitting functionality that is already implemented. Also, lack of test code was another common rejection reason (8.1%). Some author issues were also relatively common. 7.5% of the PRs were rejected due to inactivity and 7% rejected because the PR author was not able to fix issues raised by the maintainers while the PR was under review.

RQ2: Is there a correlation between the reasons why a PR is rejected and the sentiment of the interaction?

The vast majority of the analyzed PRs are neutral in sentiment (66.7%). 30.2% are positive and only 3.1% are negative. Besides, Figure 4 illustrates that neutral sentiment is associated with all the rejection categories in higher intensity compared to positive and negative sentiments. Then, we set out to analyze how would that look like if consider only the PRs with positive and negative sentiments. Figure 5 illustrates the correlation between sentiment and rejection categories without the neutral PRs. Although this subset represents a smaller number of PRs, we can see that there are more occurrences of negative sentiment in author issues than in version control issues, even though version control issues are more frequent than author issues as rejection reasons. After a deeper look into the dataset, we realize that all the three occurrences of "ego issues" were negative, out of a total of 5 negative PRs in the entire dataset after removing the "irrelevant" reasons. Therefore, we conclude that, in general, PRs tend to be neutral in sentiment. Out of a very small subset of PRs with positive and negative sentiments, we found a slight correlation between author issues and developer sentiment.

RQ3: Does the interaction length on a pull request (i.e. # of comments) correlate with the reason for rejection or the sentiment?

In Figure 6 we summarize the average number of comments per each rejection category. Longer interactions tended to be a result of author issues, followed by version control issues. Shorter interactions tended to be for those issues that involved unnecessary changes, more specifically when the functionality already existed or was not needed for the project specifications.

Now looking at a possible interplay between interaction length and sentiment, Figure 7 illustrates the average number of comments on each of the tree sentiment levels. As expected, negative and positive pull request threads tended to be the longest interactions while neutral ones were generally about half the length of the other ones. We hypothesize that this may have to do with the fact that people tend to act on things more when they are emotionally motivated, which seemed to be the case in a lot of these scenarios. Another possible explanation is that by interacting more through a higher number of comments people tend to express sentiment either side of the spectrum thus leaving the neutral zone. In addition, we observed that the positive sentiments were interactions that were geared towards helping the author to get the pull request approved (even though the PRs ended up rejected).

4.1 Suggested Guidelines for Pull Requests

From the results, we have created the following guidelines for open source project contributors.

- Know well the scope and requirements of your change before you create a pull request. Do not create a pull request if you are not able to fix issues raised by reviewers.

[1] A few PRs were rejected for more than one reason

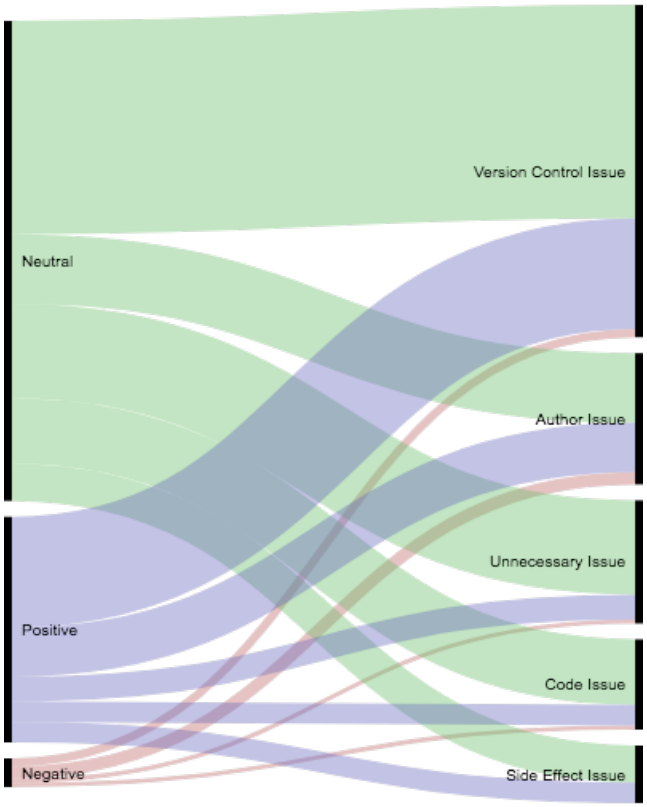

Figure 4: Correlation between sentiment and reason for rejection

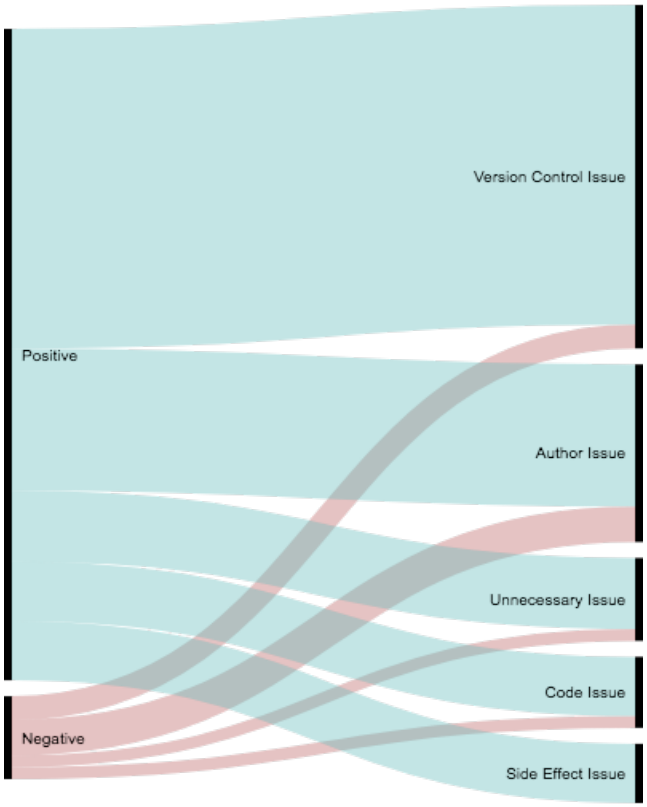

Figure 5: Correlation between sentiment and reason for rejection without neutral PRs

- Before creating a pull request, make sure that someone else has not already opened a pull request to address the same issue. Someone else may have done similar work and their PR might be either sitting on the reviewers' desk waiting for analysis or rejected for a reason (e.g., unnecessary functionality). Study the list of open and closed PRs in the project.
- Understand the current functionalities of the project before creating additional ones. If you are implementing a new functionality, make sure it is really necessary and that it is not already implemented.
- Always include tests to cover your changes. If there is a desired test coverage target for the project, make sure your code is in compliance, otherwise provide a compelling technical reason why your coverage is bellow the target.
- Make sure to follow up on your pull requests; some reviewers may request changes before it is accepted.

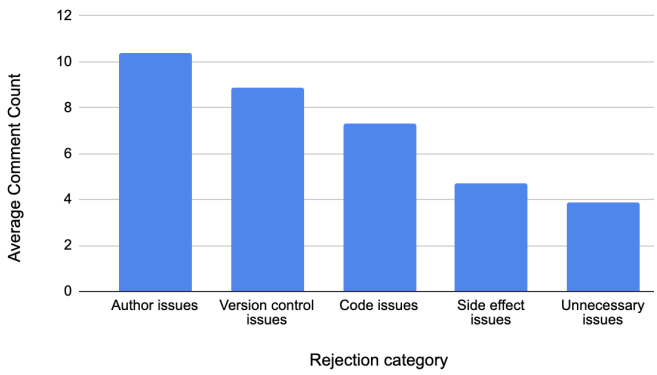

Figure 6: Number of comments vs reason for rejection

5 THREATS TO VALIDITY

The results of this research investigations could have been affected by a number of factors, one of them being time. As we did have time constraint, we were only able to analyse the Scapy library. Only looking at one repository creates a bias towards our categories, which might not be correct for other repositories. Due to the time constraint, we also were only able to calibrate with a total of 25 rejected pull requests for the categorization and sentiment analysis. Also, we only had 5 negative sentiment pull requests, or 2.2% of our 231 total requests. Conclusions drawn using this data could be invalid because our negative sentiment sample size is small. More time would permit more repositories to be analyzed and a longer calibration phase, which would also increase our pull requests' sample size.

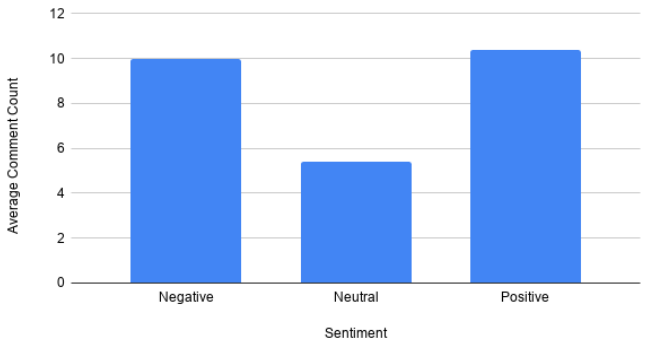

Figure 7: Number of comments vs sentiment

Another threat to validity is that the sentiment analysis is subjective. Analyzing sentiment is opinion-based and inherently subjective. People have different views of how they classify positive and negative and there is no strict rule to what classifies as each. Even though we used a set of 25 initial pull requests to set a sentiment standard, every person will still have their own personal opinions on what counts as positive or negative.

6 CONCLUSION

Clearly, there is reason to believe that a set of general guidelines will help open-source contributors get their pull requests accepted more often. Open-source software is only going to increase in its popularity, and we believe that adding this extra set of general rules will help other contributors and developers in the future.

There are three major takeaways from our research that helped us create these guidelines for future contributors. First, we found that the most common reason for a relevant pull request rejection seemed to be version control issues, or more precisely, the "pull request conflicts" issue. This issue encompasses pull requests that were started, as well as requests that were opened by accident. This is most likely due to the fact that most open source contributors that notice a way to contribute to a project will be more likely to just dive right into a pull request before checking to make sure that no one has already created a branch for their fix. Also, we noticed that neutral sentiment pull requests contained considerably fewer comments, on average, than negative or positive requests. This was somewhat unexpected, but not that surprising, as people have more reason to comment when there is sentiment involved. A person would be more likely to respond to an insult or compliment than a objective statement, whether it be with core contributors or just other developers. Lastly, we noticed that version control issues contained the most comments on average, while the unnecessary issues had the least. This is likely due to the fact that version control issues tend to encompass more complex topics, such as code rebasing, while unnecessary issues can be rejected simply by stating to the contributor that the pull request isn't needed. Using these takeaways, we created some simple, yet important guidelines for fellow developers to use when contributing to an open-source project, which can be found in Section 4.1.

In addition to these guidelines, open-source developers should understand that these guidelines are only general guidelines and that each situation has its own nuances and contributors. As with other anonymous forums on the web, GitHub can fall victim to the digital "truth serum" that is anonymous internet forums, since their pull request discussions are both anonymous and on the internet [9]. For example, in Figure 2, it is quite easy to fall victim to the feeling of truth serum. GitHub user 'cclauss' response to 'guedou', while witty to some people, would not be a helpful statement if this discussion was in a more standard office setting and in person. Open-source programmers like user 'cclauss' need to remember that the other contributors on these projects on GitHub are also real people, as that human connection can be easily lost when using the internet as the main form of collaboration.

We set out on this research investigation to see if there were any prevalent reasons for pull request rejection, so that we could develop some simple guidelines for open-source contributors to improve the acceptance of pull requests. Using data from the package management library Scapy, we were able to identify some specific correlations between sentiment and comment count, sentiment and reason for closure, and comment count and reason for closure. These correlations guided us to develop a list of guidelines (Section V) that can help future developers and contributors get their pull requests approved. While we did have some threats to validity, the guidelines that we created using the data in this paper should be quite relevant to GitHub projects with a structure such as Scapy, and fairly relevant to most GitHub projects in general. We hope that developers, not only on GitHub, but also other platforms like BitBucket, make note of these guidelines, and use them while contributing any open-source projects they encounter.

6.1 Future Work

Even though this study was successful for its scope, there are a few changes we would make if we were to conduct this research again. First, we would use a longer calibration phase. While we did start with 25 in the calibration phase, we found later in the study that 25 was likely not a big enough sample size to come up with all the reasons for rejection. Another potential area of improvement would be to use a larger sample of projects. While Scapy was adequate for the scope of this research project, our results are not significant enough to draw general conclusions from.

Expanding the scope to include more projects, specifically ones that have different purposes, would help give us a better perspective on potential guidelines to help open source contributors get their pull requests accepted. A few libraries we would consider adding in the future are Keras, VSCode, and NPM. These libraries are significantly larger than Scapy while also being used for vastly different purposes, which would help our research be more generalizable than it is now.

Finally, we would consider other factors that may have influenced the pull request rejection such as author background, code reviewer bias, and size of the pull request. The author background is important because it's likely pretty easy to determine which authors will make meaningful additions based on their history. We found code reviewer bias to be quite significant anecdotally. There were three main code reviewers for Scapy and they each varied slightly, with some more likely to accept small pull requests while others would reject it for not being big enough on its own. Finally,

the size of the pull requests likely has a significant effect on whether or not it is accepted.

REFERENCES

[1] Georgios Gousios, Andy Zaidman, Margaret-Anne Storey, and Arie van Deursen. 2015. Work Practices and Challenges in Pull-Based Development: The Integrator's Perspective. In *Proceedings of the 37th International Conference on Software Engineering - Volume 1* (Florence, Italy) *(ICSE '15)*. IEEE Press, 358–368.

[2] Valentina Lenarduzzi, Vili Nikkola, Nyyti Saarimaki, and Davide Taibi. 2019. Does Code Quality Affect Pull Request Acceptance? An empirical study. (08 2019).

[3] Daniel Pletea, Bogdan Vasilescu, and Alexander Serebrenik. 2014. Security and Emotion: Sentiment Analysis of Security Discussions on GitHub. In *Proceedings of the 11th Working Conference on Mining Software Repositories* (Hyderabad, India) *(MSR 2014)*. Association for Computing Machinery, New York, NY, USA, 348–351.

[4] Mateus F. Santos, Josemar A. Caetano, Johnatan A. Oliveira, and Humberto T. Marques-Neto. 2018. Analyzing The Impact Of Feedback In GitHub On The Software Developer's Mood. In *International Conference on Software Engineering and Knowledge Engineering (SEKE 2018)*. 1–6.

[5] Secdev. 2019. Scapy. https://github.com/secdev/scapy

[6] Marcelino C.O. Silva, Marco Tulio Valente, and Ricardo Terra. 2016. Does Technical Debt Lead to the Rejection of Pull Requests?. In *Proceedings of the XII Brazilian Symposium on Information Systems* (Florianopolis, Santa Catarina, Brazil) *(SBSI 2016)*. Brazilian Computer Society, Porto Alegre, BRA, 248–254.

[7] D. M. Soares, M. L. d. L. Júnior, L. Murta, and A. Plastino. 2015. Acceptance Factors of Pull Requests in Open-Source Projects. In *30th Annual ACM Symp. on Applied Comp.* (Salamanca, Spain) *(SAC '15)*. ACM, New York, NY, USA, 1541–1546.

[8] D. M. Soares, M. L. d. L. Júnior, L. Murta, and A. Plastino. 2015. Rejection Factors of Pull Requests Filed by Core Team Developers in Software Projects with High Acceptance Rates. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. 960–965.

[9] Seth Stephens-Davidowitz. 2017. Everybody Lies: How Google Search Reveals Our Darkest Secrets. (2017).

[10] Josh Terrell, Andrew Kofink, Justin D Middleton, Clarissa Rainear, Emerson R. Murphy-Hill, Chris Parnin, and Jon Stallings. 2017. Gender differences and bias in open source: pull request acceptance of women versus men. *PeerJ Computer Science* 3 (2017), e111.