Enhancing Diversity and Inclusion in Open-Source Software Communities

Submitted in partial fulfillment of the requirements for
the degree of
Doctor of Philosophy
in
Institute for Software Research
School of Computer Science

Huilian Sophie Qiu

Doctor of Philosophy

Thesis Committee:
Dr. Bogdan Vasilescu, Chair
Dr. Laura Dabbish
Dr. James Herbsleb
Dr. Emerson Murphy-Hill

Carnegie Mellon University

Pittsburgh, PA

Sep 2022

Acknowledgements

Thanks, Monkey King, Jade Emperor Lord, Heaven Empress, and Fairy Sister!

Abstract

Open-source software (OSS) is now ubiquitous and indispensable, supporting applications in virtually every domain. Therefore, sustaining this digital infrastructure is of utmost societal importance. One of the significant challenges in OSS sustainability is its low gender diversity. It is a well-known fact that the open-source software community is heavily skewed towards men. A low gender diversity environment is non-inclusive to non-male people. Women are one of the under-represented groups, taking up at most 10% of the OSS population. Several studies have demonstrated that women face more discrimination; for example, in some ecosystems, women have lower code acceptance rates, longer code review delays, and doubts about their skills and abilities. The low diversity and non-inclusive culture can lead to three major challenges. First, it limits the contributor pool, which harms OSS sustainability because OSS projects need a constant supply of effort for development and maintenance. Second, it impedes project success because evidence shows that a higher gender diverse team is more productive and performs better. Third, it affects gender representation and equity, thus preventing all contributors from enjoying the benefits of OSS, such as finding a job.

With much evidence showing the presence of gender discrimination, this dissertation studies why this happens and what might be an effective intervention. The first three studies in this dissertation are mixed-methods empirical studies that aim to explain the low representation of women among other marginalized groups. Because OSS development is a socio-technical activity, I use theories from social sciences and humanities, such as sociology, economics, and linguistics, to derive hypotheses and explain and contextualize results. The first three studies are arranged by the phases of an OSS contributor, with one chapter on each of the phases: newcomer, contributor and long-term contributor, and disengaged.

To conclude the dissertation, I take one step further to develop an intervention to improve the overall diversity and inclusion in OSS. As the curb-cutting phenomenon describes, designs that cater to marginalized groups also benefit a wider range of people. I use insights from the first three studies to inform the design of a dashboard for maintainers to monitor the health of their project community. I tested the dashboard through two rounds of think-aloud studies and one round of longer-term diary studies with OSS maintainers for usability and effectiveness. Overall, maintainers are excited about our dashboard's information and agree that our health indicators are informative and helpful.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Open-source software (OSS) today is ubiquitous and indispensable. As Eghbal's well-known "roads and bridges" analogy [2] suggests, OSS forms our digital infrastructure; just like their physical counterparts, roads, and bridges, OSS is supporting applications in virtually every domain. For example, more than 40% of websites use Apache HTTP Server[1] and its economic value was estimated to be more than 7 billion dollars in the US alone [3]. Therefore, sustaining this digital infrastructure is of utmost societal importance.

One of the biggest challenges in OSS sustainability is its low gender diversity. It is a well-known fact that the open-source software community is heavily skewed towards men [4] (see Section 1.2 for an overview). This low gender diversity environment is found to be non-inclusive to non-male people. As Nafus [5] pointed out that sexist behaviors in OSS are "as constant as it is extreme." Women are one of the under-represented groups, taking up at most 10% of the tech population [6]. Several studies have demonstrated that women are facing more discrimination. Women who are outsiders to a project have a lower pull request acceptance rate when their gender can be inferred from their profile [7]. In addition, Bosu *et al.* [6] found that, in some ecosystems, such as Android, Chromium OS and LibreOffice, women face a lower code acceptance rate and delayed code review feedback. Some female mentors in OSS reported that their skills and abilities are underestimated; for example, some newcomers do not take their advice/feedback as seriously as those from male mentors [8]. Some female contributors reported that they face "a harsh onboarding experience or OSS environment" [8], including acrimonious talk [5]. Some stated that they have to "prove themselves by working extra hard" [8]. The low diversity and non-inclusive culture can lead to three major challenges: limits pool, harms project success, and negatively affects representation and equity.

First, a non-inclusive environment limits the available contributor pool. A constant supply of effort is essential to OSS projects' sustainability because OSS projects need contributors for fixing bugs, adding new features, and adapting to evolving technical and non-technical environments and requirements. When projects lack appropriate levels of contributor effort, they are at risk of being undermaintained [2, 9, 10], which can cause serious problems. For example, both OpenSSL and Bash are widely used OSS but were maintained by a single developer for a long time. These two libraries had security bugs, *e.g.*, the "Heartbleed" bug[2]

[1] https://w3techs.com/technologies/history_overview/web_server

[2] https://www.digitaltrends.com/computing/heres-a-list-of-websites-allegedly-affected-by-the-heartbleed-bug/

in OpenSSL could allow hackers to capture secure information being passed to vulnerable web servers, and the "Shellshock" bug in Bash could allow unauthorized access to a computer system that went unnoticed or unfixed for many years.

Second, low gender diversity can harm project success. High gender diversity is found to be associated with better performance. A software team consisting of mostly white male programmers is generally not a good representation of their intended users because software is rarely designed for a single demographic subgroup. However, this is often neglected because OSS developers often have less concern over who their users are [5]. In software engineering, a recent study [11] confirmed that mixed-gender software engineering teams are associated with better performance because men and women tend to display different personalities, and more successful teams can leverage positive personality traits that are associated with better team performance [12].

Third, it negatively affects representation and equity. Prior studies have shown that a male-dominated environment is associated with discrimination against minority groups [5]. Discrimination towards women in male-dominated fields can cause the "imposter syndrome" effect: women tend to consider themselves disqualified or frauds despite being knowledgeable. Such effects can lead to anxiety, depression, lowered self-esteem, and self-handicapping behaviors [13].

Furthermore, a non-inclusive culture hinders marginalized groups' personal development. More than half of the respondents to a GitHub survey noted that their OSS experience helped them get their current job and build their professional reputation [14]. A non-inclusive culture that discourages marginalized groups obstructs them from gaining these opportunities.

However, this dissertation does not limit the scope to solving problems specific to only marginalized groups. The "curb-cutting" effect [15, 16] describes a phenomenon that designs that benefit marginalized groups, such as curb-cuts for people with disabilities, also allow people to push baby carriages, shopping carts, luggage on wheels, bicycles, *etc.*. This dissertation uses the problem of low gender diversity as a starting point to find methods to include overall diversity and inclusion in OSS. With higher diversity and inclusion, it will be easier for OSS projects to attract a wide variety of contributors and retain them, thus improving projects' sustainability.

While there has been a long string of scholars on OSS participation, relatively little is known about why there is a lack of diversity and inclusion, what attributes to marginalized groups' long-term engagement or premature disengagement, and, most importantly, what can be some effective interventions. In this dissertation, I divide an open-source contributor's career trajectory into roughly three phases (illustrated in Figure 1.1): newcomer, contributor and long-term contributor, and disengaged. These phases roughly follow the onion model [17, 18], which describes OSS teams as a core-peripheral structure.

In the next two sections, I present prior studies on different phases of open-source contributors and statistics of gender distributions in OSS.

1.1 Literature review on OSS contributors

There is a rich body of literature on OSS participation. In this section, I group related studies into different phases of a typical OSS contributor's career trajectory.

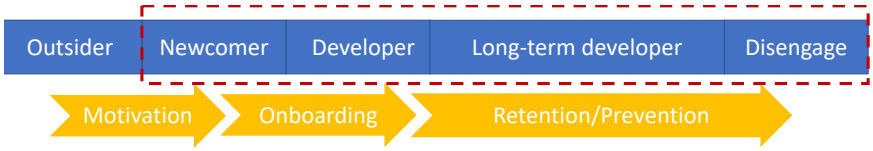


Figure 1.1: An open-source contributor's different phases. Phases bounded by the red rectangle are studied in this dissertation.

Many prior studies use the *onion model* [19, 17, 20] to describe an OSS project's structure - core contributors who contribute most of the code and manage the projects and peripheral contributors who make a smaller portion of contributions. In this dissertation, I do not distinguish between core and peripheral contributors. My primary concerns are attracting and retaining contributors. It is very likely, however, that a long-term contributor becomes a core contributor to one of the OSS projects.

1.1.1 From an outsider to a newcomer

Studies revealed that there are intrinsic, *e.g.*, having fun, and extrinsic motivations, *e.g.*, better jobs, and career advancement, to join OSS [21, 22, 23]. A recent study showed that more contributors are driven by intrinsic motivations and newcomers use their OSS experience as their portfolio in job hunting [24]. The literature found that women's motivations to use technology relate to accomplishments while men's motivations are more related to their enjoyment of technology [25]. Balali *et al.* [8] argued that the difference in motivations might explain why some women's disengagement.

1.1.2 From a newcomer to a contributor

There are studies on what makes an OSS project attractive to contributors. On social coding platform, *e.g.*,GitHub, users can make inferences of a project's characteristics based on signals, *i.e.*, visible features, and cues on the user interface [26]. For example, Trockman *et al.* [27] showed that certain signals on social coding platforms could allow users to infer the quality of a project. Santos *et al.* [28] found that license restrictiveness and their available resources influence a project's attractiveness. Knowing that different genders have different problem-solving styles [29], one important yet unanswered question is how contributors of different genders value different aspects of a project.

There is a large body of work on identifying the barriers contributors face when making their initial contributions. For example, Steinmacher *et al.* [30, 31] identified 58 barriers that may hinder OSS newcomer's onboarding experience. Some of the barriers are finding a task to start with, lack of domain expertise, not receiving an answer, code comments not being clear.

Some of the barriers are related to differences between genders. Using the GenderMag kit, Padala *et al.* [32] found that the tool for OSS is gender-biased, and women in general face more barriers than men. Moreover, in Balali *et al.*'s work [8], they listed out additional challenges that female newcomers need to face, including their low self-efficacy.

A natural follow-up study is to explore what type of projects is more friendly to marginalized groups. Foundjem *et al.* [33] found a significant correlation between high gender diversity (65% for both females and non-binary contributors) and increased patch acceptance rates (13.5%).

However, before a newcomer onboards an OSS project, one needs to identify a project to make a contribution to. Relatively little is known about how newcomers can find a friendly project. There exist many websites that try to help first-time OSS contributors find a suitable project. Some of these websites curate a list of tutorials for newcomers, [3] [4] some have a checklist to evaluate a project's fitness, [5] and some collect projects that want help.[6] Nevertheless, with such resources available, Tan *et al.* [34] found that many contributors still fail to make a contribution. Our work contributes to this gap of knowledge and explores signals that can help newcomers find a suitable OSS project.

1.1.3 From a contributor to a long-term contributor

Some studies found evidence against some bias allegations. Although women are believed to be stuck with non-code tasks, a 2013 study on FOSS survey showed that 76% of the female contributors contribute code [35]. El Asri *et al.* [36] found that female contributors, in fact, are as productive as their male counterparts. Their career trajectory follows relatively the same pattern as male contributors and remains more involved in projects.

Nevertheless, subtle bias and discrimination are still present. For example, Terrell *et al.* [7] found pull requests (PRs) from women who are not part of the project are less likely to be accepted than their male counterparts, but when gender is not visible, women have a higher PR acceptance rate. Wang *et al.* [37] pointed out that a bigger confidence-competence gap, *i.e.*, low self-efficacy despite technical brilliancy, is an additional threat women are facing in their OSS journey. Vedres *et al.* [38] found that it is not the female gender category, but rather the female behavioral pattern, *e.g.*, having more women contributors as collaborators, that put women in disadvantages; men following a similar pattern also face disadvantages.

A 2019 survey for FLOSS contributors [39] found that people's attitude towards female contributors has improved, but there are people who have strong opinions against the study of gender in OSS. More importantly, this survey [39] reported that more than one-third of the female survey participants faced sexism, such as offensive comments or insinuations on women's incompetence, and one-fifth of them felt that their code is harder to get accepted. However, Imtiaz *et al.* [40] conducted a quantitative study on GitHub using Williams and Dempsey's gender bias framework but found that most of the gender bias effects, such as tight-rope and prove-it-again, are invisible.

While these studies identified the presence of bias and discrimination, still more work has to be done to study how to improve marginalized groups' sustained participation. The studies mentioned above focused on individual behaviors. This dissertation analyzes sustained participation from the perspective of contributors' social connections on GitHub.

[3] https://www.firsttimersonly.com/

[4] https://github.com/freeCodeCamp/how-to-contribute-to-open-source

[5] https://opensource.guide/how-to-contribute/

[6] https://up-for-grabs.net/

1.1.4 Disengage from OSS

A recent study by Iaffaldano *et al.* [41] provides an overview of why OSS contributors take a break or eventually withdraw from the community. Some of the personal reasons include other professional or life event priorities, which are also found by Miller *et al.* [42] and loss of interest in the project. Project related reasons include changes in projects or the lack of communication. In addition, some contributors mentioned that the social behavior of the community can also drive people away. Being reactive may help newcomers feel welcomed whereas ignoring contributions may drive contributors away.

Literature also found that stress and burnout can be a reason for disengagement from OSS, as evident in many blog posts, talks, or podcasts [43, 44, 45] In addition to a high volume of requests [44], unfriendly or even aggressive tones are also a source of burnout [46], making projects hard to attract and retain contributors.

Negative interaction, such as pushback in code review [47] and toxic language [46, 5] can demotivate and burn out developers. Egelman *et al.* [47] found that, in a corporate setting, reviewers clocking code changes during code reviews can be a source of negative experience. Prior works have explored how to automatically detect negative experiences, such as pushback behaviors with logs-based metrics [47] and toxicity with linguistic features [46]. This dissertation further investigates how to automatically detect negative interactions among OSS contributors.

More specifically, plenty of studies focus on reasons behind disengagement of marginalized groups, such as women or newcomers. Research shows that women developers are generally more likely to leave the project than men [48]. Women face more barriers in OSS [8, 49], such as unwelcoming language [50], unsolicited sexual advances [50], gender bias in tool design [32], distrust in their competence as a mentor [8], lower code acceptance rate [7, 6], or the lack of inclusion for a female leader [6]. Scholars also pointed out that solutions to support women's sustained participation may be different from that of men [51, 52].

1.2 Literature review on OSS gender distribution

As the problem of low gender diversity is gaining more attention, many studies have tried to estimate the gender composition in the OSS community. Although all reports on a low percentage of women contributors, these numbers range from 1% to 12%. Many reasons can cause dissimilarity among the data, such as the data collection methods, time, sampled populations, and sample size. In this section, we group prior works that reported gender distribution in data collection methods: survey *vs.* data mining. Each method has its merits and shortcomings. For each method, we order the studies by the time they collected the data. Note that since these data were collected in different sub-populations and with varying sample sizes, the chronological ordering does not imply any longitudinal trend. Finally, we list studies that reported gender ratios in specific projects or ecosystems.

1.2.1 Surveys

Table 1.1 lists the studies that rely on survey data to calculate gender distribution. Surveys can capture people's self-identified gender and arguably increase the precision of gender

Table 1.1: Women ratios reported from survey data.

Year	Source	Sample size	Ratio	Citation
2001	Online survey	5,478	0%	Robles *et al.* [53]
2002	Online survey	2,784	1.1%	Ghosh [54]
2001∼2002	Email	684	2.5%	Lakhani *et al.* [22]
2002	Email	79	5%	Hars and Ou [55]
2003	Online Survey	1,588	1.6%	David *et al.* [56]
2013	Online survey	2,183	10.35%	Robles *et al.* [57]
2015	Online survey	816	24%	Vasilescu *et al.* [58]
2017	Online survey	6,000	5%	GitHub [50]
2017	Online survey	64,000	7.6%	StackOverflow [59]
2019	Online survey	119	10.9%	Lee *et al.* [39]
2021	Online survey	242	7.6%	Gerosa *et al.* [24]

Table 1.2: Women ratios reported from mining data.

Year	Source	Sample size	Ratio	Citation
2012	Email subs + US Census	1,931	8.27%	Kuechler *et al.* [62]
2012	StackOverflow	2,588	11.24%	Vasilescu *et al.* [63]
2015	GitHub + genderComputer	1,049,345	8.71%	Kofink [64]
2015	GitHub + genderComputer	873,392	9%	Vasilescu *et al.* [65]
2017	GitHub + social media	328,988	6.36%	Terrell *et al.* [7]
2017	OpenStack + genderize.io	-	10.4%	Izquierdo *et al.* [4]
2019	GitHub + Namsor	300,000	9.7%	Qiu *et al.* [48]
2019	Gerrit + genderComputer+social media	4,543	8.8%	Bosu and Sultana [6]
2020	GitHub + genderComputer+Namsor	1,954 core	5.35%	Canedo [66]
2021	GitHub + genderComputer+Simple Gender[67]	1,634,373	5.49%	Vasarhelyi *et al.* [68]
2021	GitHub + genderize.io	65,132	10%	Prana *et al.* [69]
2022	Software Heritage + Gender Guesser	21.4M	10%	Rossi *et al.* [70]

identification [60]. Surveys targeting a specific population can provide in-depth and more accurate insights. However, survey data, albeit highly reliable and accurate, are prone to selection bias. People who responded to the survey may be qualitatively different from those who did not respond because of differences in survey accessibility and individual motivation [61]. Moreover, survey datasets are usually small, making it hard to obtain generalizable results.

1.2.2 Mining trace data

Table 1.2 shows the studies that rely on mining data to report gender distribution. Gender inference based on mined user information provides a more representative, large-scaled sample,

Table 1.3: Women ratios in different ecosystems or open-source projects.

Year	Source	Ecosystem(s)	Sample size	Ratio	Citation
2014	Mailing list	Drupal	3,342	9.81%	Vasilescu *et al.* [73]
2014	Mailing list	Wordpress	3,611	7.81%	Vasilescu *et al.* [73]
2016	Online survey	Apache	765	5.2%	Sharan [74]
2005-2016	GitHub	Linux	14,905	8%	Cortázar [75]
2016	Online survey	Debian	1,479	2%	Raissi *et al.* [76]
2019	GitHub + Namsor	Angular.js	1,601	3.4%	Asri and Kerzazi [77]
2019	GitHub + Namsor	Moby	1,824	3.5%	Asri and Kerzazi [77]
2019	GitHub + Namsor	Rails	3,723	4.2%	Asri and Kerzazi [77]
2019	GitHub + Namsor	Django	1,672	5.3%	Asri and Kerzazi [77]
2019	GitHub + Namsor	Elasticsearch	1,127	4.2%	Asri and Kerzazi [77]
2019	GitHub + Namsor	TensorFlow	1,735	5.8%	Asri and Kerzazi [77]
2019	Gerrit + genderComputer	Android	258 core	3.87%	Bosu and Sultana [6]
2019	Gerrit + genderComputer	Chromium OS	151 core	3.97%	Bosu and Sultana [6]
2019	Gerrit + genderComputer	Couchbase	24 core	4.17%	Bosu and Sultana [6]
2019	Gerrit + genderComputer	Go	90 core	7.77%	Bosu and Sultana [6]
2019	Gerrit + genderComputer	LibreOffice	68 core	1.47%	Bosu and Sultana [6]
2019	Gerrit + genderComputer	OmapZoom	60 core	10%	Bosu and Sultana [6]
2019	Gerrit + genderComputer	oVirt	34 core	2.94%	Bosu and Sultana [6]
2019	Gerrit + genderComputer	Qt	159 core	3.12%	Bosu and Sultana [6]
2019	Gerrit + genderComputer	Typo3	73 core	4.1%	Bosu and Sultana [6]
2019	Gerrit + genderComputer	Whamcloud	19 core	0%	Bosu and Sultana [6]
2021	Online survey	Linux	2,350	14%	Carter *et al.* [78]

and it also avoids the burden of the survey respondents and the efforts taken to collect survey results.

However, researchers often need to infer gender because not all platforms collect users' gender, and not all users disclose their genders online. From the studies, we summarize two primary information sources for gender inference: names and information from other social media platforms. Commonly used name-based gender inference tools include Namsor [71], genderComputer [63], GENDER GUESSER,[7] and genderize.io.[8] See Sebo [72] on a comprehensive comparison among these tools. The most significant shortcomings of these computational tools are non-perfect accuracy and the assumption of binary gender. Some studies cross-link a user's account to other social media platforms, such as LinkedIn, Facebook, Google search, and the now deprecated Google plus. This method can capture users' self-reported gender.

1.2.3 Ecosystems

Table 1.3 lists studies that report gender ratios in specific software ecosystems. Many of these studies focus on specific projects rather than the entire ecosystem. In addition to these

[7] https://pypi.org/project/gender-guesser/
[8] http://www.genderize.io

quantitative figures, some studies also provide comparisons across ecosystems. For example, women are more represented in COBOL legacy systems than in new systems using Java or C++ [79]. Women are also more likely to be found in Ruby but not in pure backend or PHP-focused frontend communities [68]. However, to the best of our knowledge, there is not a study that covers all major ecosystems.

1.3 Thesis

In this dissertation, I conduct a series of empirical studies using a mixed-methods approach to gain a better understanding of factors that influence diversity and inclusion in OSS. Because software development is a collaborative, human-centric activity, I use social science theories to inform study design, derive hypotheses, and explain and contextualize results. The abundant trace data from online coding platforms, *e.g.*,GitHub, allows us to perform large-scale data analyses to answer research questions empirically. I employ quantitative and qualitative methods in my research because they complement each other and allow me to get different perspectives on the research question. I use quantitative methods, such as sophisticated statistical analyses and advanced machine learning models, to discover patterns from large-scale, longitudinal data. I use qualitative methods, such as surveys and interviews, to validate our computational operationalizations and gain insight from real contributors.

The transparency design of the social coding platforms and their rich trace data also allow us to build a dashboard that reflects the status of a project to help maintainers monitor their project community's health. According to the signaling theory, people may use visible cues, *i.e.*, signals, to infer a person or an item's hidden properties. In this dissertation, while studying the reasons behind the low gender diversity, I also search for signals that can reflect diversity and inclusion in a community and flag potential problems to raise maintainers' awareness. Our dashboard displays the signals that are found by prior work or this dissertation to be relevant to diversity and inclusion but are otherwise hard to observe on social coding platforms.

1.3.1 Thesis statement

Here is my thesis statement:

The open-source software (OSS) community has low gender diversity and is non-inclusive to women and other minority groups. Using gender diversity as a starting point, I conduct a series of empirical studies to gain a better understanding of the reasons behind low diversity and to seek possible solutions to improve inclusion in the OSS community. I use results from these empirical studies to build a dashboard as an intervention to help maintainers enhance their projects' diversity and inclusion. Making signals that can reflect a project's level of diversity and inclusion more visible improve OSS projects' diversity and inclusion.

Note: In this dissertation, I use "we" when describing works that I collaborated with other researchers.

1.3.2 GitHub as the research context

This dissertation uses GitHub, one of the most widely used social coding platforms, as the research context. I chose GitHub for several reasons. First, GitHub is the most widely used online social coding platform, with more than 56M users as of September 2020.[9] Second, GitHub has a rich set of features, *i.e.*, visible cues, to reflect contributors' social dynamics. For example, there is the number of daily commits [80] as a signal of a contributor's commitment and competence, or the number of stars [81] for a repository to reflect its popularity. Users can put up signals, such as badges in the README [27] or a CODE OF CONDUCT [82], to demonstrate their project's level of maintenance. These features can serve as signals that can help contributors make informed decisions [26].

The abundance of signals leads to many interesting research questions. For example, the presence of a daily activity streak may affect how contributors behave [80]. Social signals, such as users' names or profile pictures, may influence how their PRs are treated [7, 83].

Finally, all actions on GitHub leave traces and are available for everyone. Therefore, we are able to use this information to conduct analyses on their social networks, contribution patterns, code quality, *etc.* The trace data and signals are available and easy to manipulate with MySQL and MongoDB [84]. This makes many empirical analyses possible.

1.3.3 A note on the use of binary gender

Although gender, a socially constructed concept, is non-binary, it is sometimes not impractical to include non-binary gender in a computational model. Throughout this dissertation, I use GitHub trace data, which does not record participants' gender. Therefore, we often need to infer gender based on information we can observe, usually names, photos, and information from other social media platforms, *e.g.*, Google+ and LinkedIn.

However, all these methods have their limitations. Commonly used gender inference tools using names or photos often assume binary gender, and the accuracy is not perfect [72]. Even though we can sometimes find users' self-reported gender from their profiles on other websites, the number of users with whom we can link their accounts is relatively small. As a result, to obtain a large dataset of contributors with their gender, the best method we can rely on is the name-based inference tools.

Despite these limitations, I argue that conducting gender analysis with imperfect gender inference is necessary and imperative. While the severe underrepresentation of women and other marginalized groups is widely recognized in OSS, much more work is needed to understand the reasons behind the low diversity. Moreover, we have little empirical evidence on how the situation has changed over time and in different sub-communities and ecosystems.

Though imperfect, a large dataset of OSS contributors with inferred gender allows us to observe the approximate gender distribution at different times and in different ecosystems [70]. These observations can provide us feedback on the effectiveness of our efforts to improve gender diversity in OSS. They can also point us to the most and the least diverse communities to conduct future studies. With access to large datasets with inferred gender, we can also build statistic models to test hypotheses developed from social sciences theories on gender differences (Chapter 3). Moreover, since women contributors are rare in open-source, using

[9] https://octoverse.github.com/

inferred gender, we can preselect a small group of people whose likely to be women and then manually verify their genders.

Therefore, because obtaining a large dataset of accurate self-reported gender, including non-binary options, is impractical and almost infeasible, in some places, I use binary gender as a simplification to make the analyses tractable. When possible, however, such as when inviting participants for interviews or surveys, I only use computationally inferred gender as a rough approximation to help me find non-men contributors. I use their self-reported gender in the analyses and reports. Overall, I only use automatic name-based gender inference to capture gender distribution and perform analyses at the population level. As a result, my studies' results should be considered approximations of the actual gender distributions in the OSS community.

1.3.4 Dissertation outline

This dissertation presents a series of empirical studies that focus on different phases of an OSS contributor.

Chapter 2: Help Contributors Choose Projects

While prior works have extensively studied contributors' onboarding experience, this chapter focuses on the earlier and relatively less studied stage in the onboarding process: how *newcomers* choose which projects to contribute to. This work is based on signaling theory, a framework borrowed from economics [85, 86] and biology [87]. The signaling theory states how one may use visible cues to infer a person or an item's hidden properties. This is relevant to OSS social coding platforms as Dabbish *et al.* [26] showed that contributors make inferences on projects based on signals. To better guide new contributors to find a suitable project, we interviewed contributors with various degrees of experience for their insight on how to use signals on GitHub to infer how inclusive and newcomer-friendly a project is. From the interviews, we identified helpful signals and built a model to test if they are significantly associated with bringing in newcomers. This chapter consists of the following conference paper:

[88] H. S. Qiu, Y. L. Li, S. Padala, A. Sarma, and B. Vasilescu, "The signals that potential contributors look for when choosing open-source projects," Proceedings of the ACM on Human-Computer Interaction, vol. 3, no. CSCW, pp. 1–29, 2019.

Chapter 3: Sustained Participation

This chapter concerns contributors and long-term contributors. Although contributors' sustained participation has attracted much attention, little is known about the gender difference. Applying survival analysis, we found that women contributors leave GitHub earlier than their male counterparts. In other words, women contributors have shorter career spans on GitHub. Software development is collaborative; therefore, this chapter studies contributors' sustained participation using social network theories. Social capital theory, a theory explaining resources one can gain from their network connections, provides insight into how contributors' network connections can affect their sustained participation and why there is a difference

between genders. We used the social capital theory to identify network structures associated with contributors' prolonged participation. We used a survival analysis model and surveys to triangulate our results. The results reveal possible signals on social coding platforms to support women in developing social capital. This chapter consists of the following conference paper:

[48] H. S. Qiu, A. Nolte, A. Brown, A. Serebrenik, and B. Vasilescu, "Going farther together: The impact of social capital on sustained participation in open source," in 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE). IEEE, 2019, pp. 688–699.

Chapter 4: Detecting Interpersonal Conflicts

This chapter presents our study on disengagement prevention: we study how to detect interpersonal conflicts in issue discussions and code reviews automatically. Our work builds on two prior studies. Egelman *et al.* [47] introduced the concept of *pushback* to refer to the perception of unnecessary interpersonal conflict in code review and presented a classifier using meta-information in Google's code review, *e.g.*, number of comments, reviewing time. Around the same time, Raman *et al.* [46] proposed to use linguistic features to detect *toxicity*, *i.e.*, rude, disrespectful comments in GitHub issue conversations. The two concepts, *i.e.*, pushback and toxicity, are distinct yet similar. We conducted a systematic evaluation of the two complementary methods, *i.e.*, meta-information and linguistic features, on detecting the two concepts, in both corporate (Google) and open-source (GitHub) settings and both types of conversations (issue and code review). The evaluation also allowed us to identify signals that can flag potential interpersonal conflicts in open-source development. This chapter consists of the following conference paper:

[89] H. S. Qiu, B. Vasilescu, C. Kästner, C. Egelman, C. Jaspan, and E. Murphy-Hill, "Detecting interpersonal conflict in issues and code review: Cross pollinating open-and closed-source approaches," in 2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS). IEEE, 2022, pp. 41–55.

Chapter 5: Intervention: A Dashboard for Maintainers

This chapter presents an intervention we built: a dashboard for open-source maintainers to monitor health indicators that are found to impact diversity and inclusion by prior research. We identified health indicators from the literature, studies presented in previous chapters, and interviews with maintainers. Our dashboard focuses on the indicators that are important but not currently readily visible on social coding platforms such as GitHub. Among others, our dashboard included indicators of pushback in code review, tone of issues and pull request discussions, and social capital measures. In addition to summaries of these indicators, we provided coaching on what possible management actions maintainers can take to improve the health of their project. We also included a gamification function that compares the focal project with similar projects to give maintainers a reference on how well they are doing.

We iterated and refined our design through two rounds of think-aloud studies with open-source maintainers. We then tested the usability and effectiveness of this intervention via a two-week dairy study with open-source maintainers. Through our user studies, we found that

maintainers were generally excited about the information that our dashboard provides and agreed that our health indicators are informative and helpful.

Chapter 6: Conclusion

I conclude this dissertation with a reflection on its contribution and discussions. I also list some potential future work that I plan to explore after graduation.

Chapter 2

Help Contributors Choose Projects

While prior work has extensively studied the motivations of open-source contributors in general, relatively little is known about how people choose which project to contribute to, beyond personal interest. This question is especially relevant in transparent, social coding environments like GitHub, where visible cues on personal profile and repository pages, known as signals, are known to impact impression formation and decision making. In this chapter, we report on a mixed-methods empirical study of the signals that influence contributors' decision of joining in a GitHub project. We first interviewed 15 GitHub contributors about their project evaluation process and identified important signals they used, including the structure of README and the amount of recent activities. Then, we proceeded quantitatively to test out the impact of each signal based on the data of 9,977 GitHub projects. We reveal that many important pieces of information lack easily observable signals, and that some signals may be both attractive and unattractive. Our findings have direct implications for open-source maintainers and the design of social coding environments, e.g.*, features to be added to facilitate better project searching experience.*

2.1 Introduction

Open-source software infrastructure is ubiquitous, powering applications in virtually every domain [2]. Yet, despite their importance, many open-source projects lack appropriate levels of contributor effort and are thus at risk of being undermaintained [2, 9, 10]. In projects with only one or two core contributors, of which there are many [90], lack of time or interest of the main contributors poses serious sustainability risks [9, 91, 41]. Recruiting new contributors can, therefore, help ensure the sustainability of open-source projects.

Many researchers have studied why skilled workers contribute to open-source. Prior work found that starting to contribute to, and remaining engaged with open-source is influenced by a mixture of intrinsic and extrinsic factors [92], among which identifying with the community, feeling obligated to contribute back, learning opportunities, personal needs, and signaling one's skills to potential employers are all important [93, 22, 94, 95].

What is less known, however, is how people decide to contribute to particular projects *based on partial information about the projects*. This question is especially relevant today because, compared to their predecessors, social coding platforms like GitHub, Bitbucket,

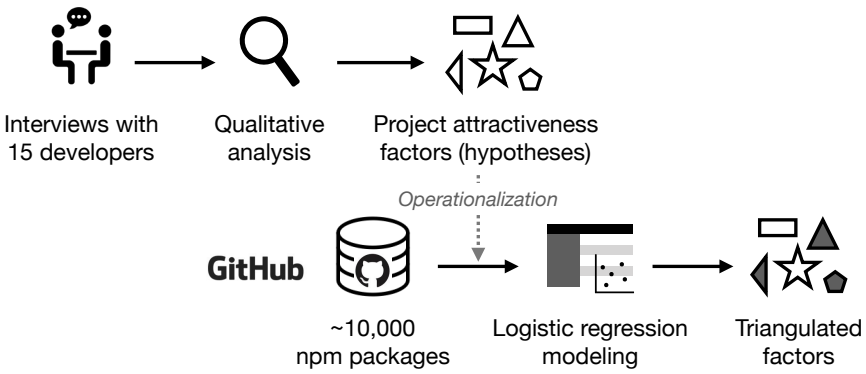


Figure 2.1: Overview of our study design.

and GitLab offer a high level of *transparency*, achieved by displaying a multitude of visible cues (or *signals* [96]) on individual and project public profile pages [26, 97]. For example, on GitHub—the most popular open-source hosting platform—there are signals of individual popularity, such as a user's number of followers, and signals of project activity, *e.g.*,, the number of contributors and issues, among many others. As prior studies show, this high level of transparency enables people to make rich inferences about each other's technical expertise and level of commitment [26, 97]. Similarly, to inform their decision whether to join a project, in many cases potential contributors must rely on partial information derived from signals available online. It is therefore important to study how people infer the characteristics and qualities of an open-source project based on the cues they can observe, and how these signals influence their decision to contribute to the project.

In this paper, we build on the literature on transparency in social coding environments to empirically explore a new question:

RQ$_1$. *How do people use signals, if at all, when choosing an open-source* GitHub *project to contribute to?*

Our study uses a mixed-methods design (Figure 2.1). We start qualitatively by interviewing 15 GitHub users, sampled to represent a diversity of experience contributing to open-source, gender, and geographic, cultural, and technical background. From these interviews, we identify which signals are perceived as most influential when evaluating open-source GitHub projects for potential contribution. Then, we proceed quantitatively by mining trace data from 9,977 open-source GitHub projects (stratified by number of stars) and testing hypotheses, using multiple regression modeling, about the impact of the different signals on attracting new project contributors.

Our results reveal several key signals used to inform the decision whether or not to contribute to a GitHub project: i) a README file with thorough contents and clear structure, describing what the project does, how to get started using it, what a new contributor could work on, and what guidelines they should follow; ii) the availability of scaffolding, such as issue and pull request templates, or issue labels; iii) how actively maintained the project is, along multiple dimensions, such as the number of contributors and the recency of commits; iv) the friendliness of the maintainers in issue and pull request discussions; and v) project popularity.

Moreover, we find that some signals can be considered both attractive and unattractive by different users. For example, from the interviews, we found that, while typically positive, the presence of detailed contributing guidelines is also seen by some contributors as "off-putting", as it can set a higher bar to participation and impose too much process overhead. Also, some signals are important in the decision process but may be unclear to first-time GitHub contributors. For example, our model shows that politeness is an important signal for arbitrary new contributors but not for first-time GitHub contributors.

Our results have direct implications for multiple stakeholders. First, we provide open-source project maintainers with actionable insights that can help make their projects more attractive to external contributors. Second, we uncover several cues that potential contributors look for in a project, such as the responsiveness of the project maintainers and the friendliness of the community discussions, that are currently not readily observable in the GitHub UI; our participants browsed through multiple pull request and issue threads to make qualitative inferences about these properties. These insights can help tool builders and designers of collaboration platforms like GitHub develop new signals, *e.g.*, in the form of badges [27], to make these properties more salient.

In the next sections, we frame our discussion in the context of signaling theory, consider related research, describe our methodology, present the results of our interviews and data modeling, and finally discuss implications of our findings.

2.2 Related Work

The process of attracting and onboarding contributors to open-source projects has a long history of scholarship; for an overview see, *e.g.*, Crowston *et al.* [98]. The process consists of multiple stages. Starting from an intention to contribute to open-source, one should ① *discover a relevant project*, ② *find an opportunity to contribute*, then ③ *make a first contribution* (*e.g.*, submit an issue report or a pull request). Then, by continuing to make contributions and ④ *demonstrate commitment* to the project over time, one can ⑤ *be recognized as a core contributor* or maintainer. As turnover is natural in open-source, eventually some contributors will ⑥ *disengage*.

2.2.1 Knowledge gap: How people choose which projects to contribute to

There is a rich body of literature (*e.g.*, [99, 100, 101, 102, 103]) on what happens to open-source contributors *after they identify a project* they intend to contribute to (stages ②—⑥), in terms of their onboarding into the project core team and their long-term participation and turnover. In particular, Steinmacher *et al.* [31, 104, 105] reported, in a series of studies, on how the onboarding process can be long and demotivating for newcomers, who face various social and technical challenges when trying to *find a first task* they can complete and adapt to the project's contribution standards, culture, and norms. The authors identified 19 reasons that a new contributor's pull request was rejected, both social and technical, including receiving impolite answers from maintainers, the pull requests being duplicated, not needed, or mismatched with the maintainers' vision, lack of tests, not following guidelines,

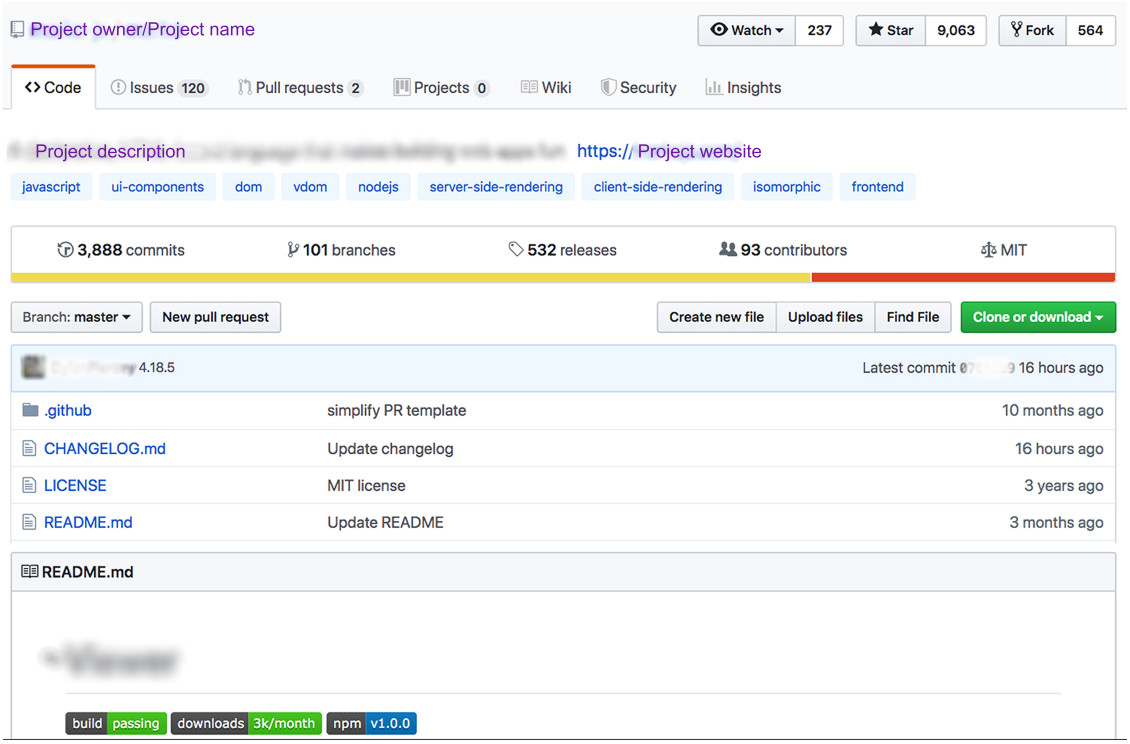


Figure 2.2: A snapshot of a GitHub project page (anonymized).

and not receiving an answer at all; these latter barriers have also been reported in other online collaboration contexts outside open-source, especially Wikipedia [106].

In contrast, we focus on the earlier and relatively less studied stage in the onboarding process: *how people choose which projects to contribute to* (stage ①). Two forces can influence this decision [107]: individual motivation and project attractiveness. Individual motivations are generally well understood, and can be both intrinsic, *e.g.*, personal need for that software or feeling obligated to contribute back, and extrinsic, *e.g.*, career advancement [22]. However, what project actions and characteristics influence project attractiveness to outsiders is still an open question [108].

Studying what makes projects attractive is especially important because, as opposed to individual motivation which is typically inherent to the potential contributors, project attractiveness can be to a larger extent controlled by the project maintainers, as we will argue in the remainder of this paper. Therefore, increasing project attractiveness has the potential not only to reduce some onboarding barriers, but also to improve the sustainability of open-source projects.

2.2.2 Signaling and transparency in online coding environments

On transparent, social coding environments like GitHub, the question of how people choose projects is especially relevant, as a wealth of signals (visible cues indicating otherwise less readily observable qualities [109]) about an open-source project's history of activity and contributors is available on the project's homepage, *e.g.*, the number of commits, contributors,

forks, issues, pull requests, star gazers, and watchers. In addition, GitHub renders a project's README.md file as part of the project's homepage. This file gives maintainers a chance to further customize their project's signals, either through free text, *e.g.*, contributing guidelines and documentation on how to install the software, or through badges [27] embedded into a project's README; badges such as build passing and PRs welcome are customizable images that typically reflect the status of different online services the project is using, *e.g.*, continuous integration testing, or expressions of intent, *e.g.*, soliciting pull request contributions. An example of a typical GitHub project page is shown in Figure 2.2. Finally, the transparency provided by individual "profile pages" on GitHub, which aggregates personal information and information about one's history of contributions to open-source projects on GitHub, enables inferences about the contributors' expertise and level of commitment [97, 110], and even makes salient their demographics [111, 58].

Signaling theory, going back almost half a century in economics [85, 86] and biology [87] (see Kirmani & Rao [112] for an overview), provides a framework for reasoning about how these visible cues might impact project attractiveness in open-source. Signaling theory has also been widely applied to social computing systems, to understand how people make inferences using online profile data in contexts as diverse as social networking sites [96, 113, 114], fashion [115], peer-to-peer lending markets [116] and rentals [117], and peer production [97].

In general, signaling theory is applied in scenarios where selections are made under information asymmetry. These are decision making situations typically involving two parties, a signaler, with access to all the information, and a receiver, who is less informed, in which the former would be selected by the latter based on the information carried by the signal. Across all such selection scenarios, an important attribute of signals is their *visibility*: receivers tend to prefer signals that are easier to observe and to interpret over those that are costlier to assess, even when the former are less reliable [118]. Another important attribute of signals is their *production cost*: signals that are costlier to produce, therefore harder to fake, are considered more reliable [119]. For example, in biology, the peacock's heavy tail feathers are both visible and costly to maintain, as they are a highly observable ornament which makes the animal more vulnerable to predators. Therefore, the peacock's tail feathers signal the bird's quality [87]: having survived despite this handicap, the peacock is perceived by potential mates as more attractive and more fit [120]. In economics, a similar signal is holding a degree from a reputable institution: the job seeker's ability, which is otherwise less visible, is being communicated to potential employers by the high-status degree, which required substantial effort to obtain [85].

Many similar selection scenarios occur in open-source development: for example, choosing which repositories to watch [121], which pull requests to accept [97], which developers to follow and receive updates from [122, 123], and which ones to recruit [110, 124]. In all these scenarios, the signals available on social coding platforms like GitHub have been shown to play a role. Our work contributes to the literature on signaling and transparency in online collaboration environments by studying another important selection scenario: *how do people use signals in transparent environments like GitHub when deciding which open-source project to contribute to.* Such signals could be found, for example, on a project's README file: READMEs already contain many highly visible cues, since GitHub renders the file by default on a project's profile page (Figure 2.2). Some of these cues could be reliable signals. For example, comparing to a short or uninformative README, a well-structured and detailed

README on the usage and contributing process could show that the project owners are aware of their audience and have spent time on maintaining the project. As a result, one could expect that the owners are more willing to provide support.

2.2.3 Prior empirical evidence on how people choose projects

While prior research on this particular question is scarce, there is some empirical evidence suggesting how the different signals visible on GitHub might influence people's decision to contribute to a project. We note four studies in particular.

Dabbish *et al.* [26] reported on an interview study with 24 GitHub users of the types of inferences that people made based on the visible signals on GitHub. While the authors did not systematically pursue the question of project attractiveness to potential contributors, their findings are relevant to our research question, as some of the signals and corresponding project qualities their study uncovered could impact people's decisions to contribute to a project. Specifically, Dabbish *et al.* found that: (i) the recency of activity in a project signals project liveness and maintenance; (ii) the amount of attention a project receives, as indicated by the number of stars and watchers, signals artifact importance, project quality, and community support; (iii) a high number of open pull requests signals low conscientiousness in dealing with external contributors; and (iv) the number of forks and watchers of a project signals audience size and potential impact of contributing—this inference was the only one explicitly cited as a motivation to contribute.

More recently, and concurrently with our work, Fronchetti *et al.* [125] reported on an archival analysis of data from 450 open-source GitHub projects, studying which project characteristics are related to the growth pattern in the number of new committers per project, computed over a period of 72 weeks. The authors sampled, in decreasing order of popularity as indicated by the number of stars, 30 projects each across the 15 most popular programming languages on GitHub. Then, using a Random Forest classifier to model the growth pattern in new committers, they found that the number of stars has the highest explanatory power among all predictors considered, followed by the time to merge pull requests, project age, and the number of programming languages used in the project. On the other end of the spectrum, the presence of CONTRIBUTING, LICENSE, and CODE OF CONDUCT files, as well as the presence of issue and pull request templates, all of which are often recommended as community best practices, were among the worst ranked factors in their model. While these results offer valuable insights into which signals might be used by potential open-source contributors when choosing projects, given the choice of Random Forest classifier the directionality of the reported associations remains unknown. Moreover, it remains unknown how the results would generalize beyond the relatively small sample of most popular projects per language (the median number of stars in their dataset is 10,470); for example, the lack of explanatory power for the different community best practices such as CONTRIBUTING files or issue and pull request templates could simply be due to the sampling strategy, as the absolute most popular projects are likely to all already implement these best practices. Finally, it is unclear how the different factors extracted from repositories have been selected. In contrast, we use a mixed-methods design to first qualitatively uncover which signals our interviewees use and how they make inferences using these signals, then quantitatively model, using

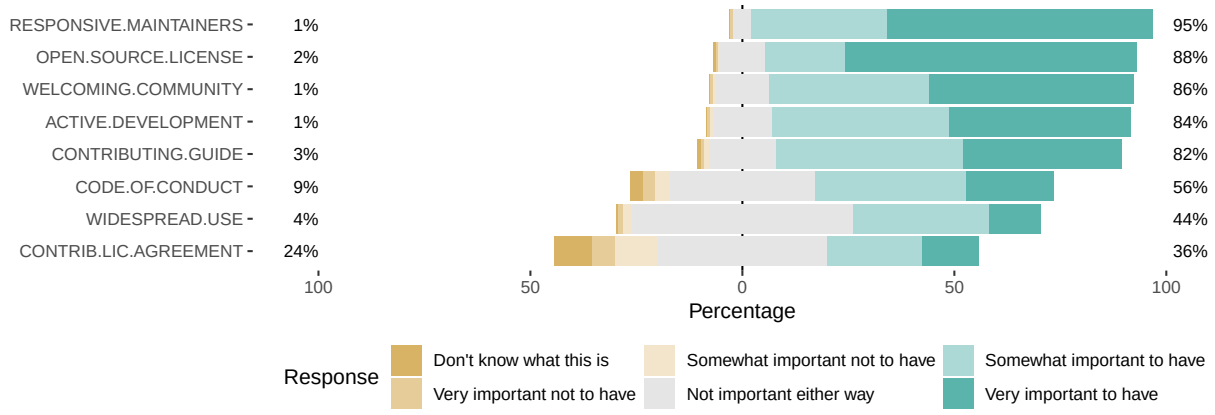


Figure 2.3: Breakdown of responses ($N = 3127$) to the question "When thinking about whether to contribute to an open source project, how important are the following things?" from GitHub's 2017 Open Source Survey [1].

multivariate regression, how the project attributes made visible by these signals associate with the likelihood of attracting new project contributors in a large sample of 9,977 projects.

We also note a study by Borges and Valente [81], who surveyed 791 developers on the meaning of GitHub stars, finding that three out of four respondents consider the number of stars before using or contributing to a GitHub project. However, in their study design the authors do not distinguish usage and contribution to GitHub repositories, so it remains unclear which signals affect which.

Finally, as part of GitHub's 2017 Open Source Survey [1], the authors asked respondents to rank several factors based on importance when thinking about whether to contribute to an open-source project: an open source license, a code of conduct, a contributing guide, a contributor's license agreement (CLA), active development, responsive maintainers, a welcoming community, and widespread use. Figure 2.3 summarizes the survey results, which are publicly available [1]: all factors are considered somewhat important or very important to have by at least 36% of respondents; maintainer responsiveness ranks as topmost important (95% of respondents).

2.2.4 Summary

In summary, potential contributors have access to a wealth of information about open-source projects on GitHub, which could act as signals for qualities that are important when deciding which project to contribute to. Some of this information is highly visible on the platform by default through built-in visible cues (*e.g.*, a project's number of stars). Project maintainers can choose to make visible other pieces of information through a project's README file (*e.g.*, a code quality badge). Finally, since for open-source projects the entire history of activity is accessible publicly (*e.g.*, all commits, issue discussions, and pull requests, together with all the actors involved), users on the platform are free to use many additional, less readily observable pieces of information when making decisions and forming impressions.

2.3 Qualitative Analysis Methods

To explore what signals people use when deciding which open-source projects to contribute to on GitHub and how the signals impacted their decisions, we first conducted semi-structured interviews with 15 GitHub users, then, based on the interview results, we mined and analyzed GitHub trace data to test the significance of each signal. Our mixed-methods strategy is *sequential exploratory* [126], as we use the quantitative results generated in a second step to assist in the interpretation of the qualitative interview findings. Here we describe the qualitative methods.

2.3.1 Interview Protocol

We developed a semi-structured interview protocol that could enable participants to evaluate a project's "attractiveness" for external contributors based on the information available on GitHub. In short, participants were asked to evaluate five given open-source projects and talk aloud about what information they were using and how that influenced their evaluations.

The main challenge in developing the interview protocol was separating the two forces that can influence the decision to contribute to an open-source project [107]: individual motivation and project attractiveness. We describe the iterative process through which we addressed this challenge.

Iterative design of the interview protocol. We started with two main design options and ran a series of pilot interviews to finalize the interview protocol: 1) asking participants about their actual *past* experience contributing to different projects, or about their intentions to contribute to new projects in the near *future*; 2) asking participants to evaluate the open-source projects for their *own* intended contribution, or for *someone else*.

In a first pilot round, we interviewed three colleagues and friends who are active on GitHub, asking participants to recollect their past experience of finding a new project to contribute to and describe their choice. The interviews confirmed the two expected shortcomings of this design: people's memory of the selection process was too vague and incomplete to be reliable; and people commonly reported choosing projects because they were using them and wanted to fix bugs or develop new features, *i.e.*, personal motivation.

Next, to help delineate individual motivation from the effects of different GitHub signals on project attractiveness, in a second pilot round with six other friends and colleagues active on GitHub we introduced two changes. First, we employed a think-aloud technique [127], asking participants to look for a new project to contribute to while talking aloud about what signals they were considering. This allowed us to follow the participants' moment-by-moment cognitive process more precisely. Second, we changed the focus from recommendations for oneself ("would you contribute to this project?") to recommendations for a third party ("would you recommend Jane to contribute to this project?"). In addition, each participant was given a pre-determined set of the same five JavaScript front-end projects, chosen purposefully (Section 2.3.2). Specifically, we constructed a scenario where participants were asked to make recommendations for a recent graduate with a bachelor's degree in computer science, Jane, now working for a startup as a junior front-end engineer. No information about Jane's interests, beyond JavaScript front-end, was given. Through piloting, we found that the use of

Table 2.1: GitHub metrics for the five open-source projects presented to the interviewees

Project	Issues	Pull requests	Releases	Contributors	Watchers	Stars	Forks	Branches	Badges
1	2	37	216	18	18	7	3	290	7
2	334	104	46	1003	7305	124,012	59,243	29	11
3	38	0	7	6	17	214	11	2	0
4	9	0	8	1	1	1	0	2	1
5	33	4	399	7	10	8	2	9	1

the Jane persona helped alleviate the effect of participants' personal preference when choosing projects, allowing them to focus on the GitHub signals.

Final version of the interview protocol. Our final protocol maintained the semi-structured think-aloud format with the scenario of recommending projects for Jane. In addition, we also asked the participants to summarize their criteria when selecting projects and to offer suggestions for project maintainers to improve the attractiveness of their respective projects. Finally, at the end of the interview we collected basic demographics (gender, occupation, and open-source experience).

Limitations. We note that because of the scenario used in our interview protocol (making recommendations for a relatively novice developer interested in JavaScript front-end), our results may not generalize to other developers, *e.g.*, experts. We also acknowledge that (1) recommendations made for someone else can differ from choices one would make for themselves, and (2) the profile of the person onto which our interviewees made projections may itself be a source of potential bias (*e.g.*, the gendered profile of the recommendee in our protocol, Jane, may trigger biases among male interviewees). As discussed above, this study design element—recommending projects for another developer—was necessary to delineate decisions influenced by individual motivations from those influenced by project attractiveness signals. A comprehensive set of interviews, where all variables relevant to the recommendee's profile (*e.g.*, gender, level of experience, interests) are crossed, goes beyond the scope of this study, but could be a worthwhile direction for future research.

2.3.2 Project Selection

We selected five projects that collectively reflect a variety of signals possible on a GitHub page. At the time of our interviews, the values of the different project metrics were the ones shown in Table 2.1 (as of April 28, 2018). Our project selection was based on the following specific criteria:

Domain. Since our persona Jane was designed as a front-end engineer, we only chose front-end-related JavaScript projects so that the participants' decisions would not be confounded by Jane's personal interest. To control for potential differences in practices and culture in different open-source ecosystems, we further required that all selected projects be part of *npm*,[1] the most popular package manager for the JavaScript programming language.

[1] https://www.npmjs.com

Table 2.2: Participants' demographic information

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15
Experience	6m	12y	1m	9y	>8y	2y	4y	<2y	1y	5y	1y	1.5y	8y	3y	10y
Gender	M	F	F	M	M	M	M	F	M	M	M	F	M	F	F
Full-time dev.	Yes	No	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	No

GitHub *metrics.* Activity and popularity metrics, such as the number of contributors, stars, forks, and watchers, are among the most visible cues on a GitHub repository page (high visibility signals cf. Section 2.2), since they are part of the standard UI. We chose projects to ensure high variance in these numerical metrics across our set of five: one project has a very large number of contributors (over 1,000), stars, forks, and watchers; one is a one-person project with only one watcher, one star, and no fork, and the three other projects are in between. In addition, during pilot interviews we observed that participants also paid attention to a project's pull requests, issues, and releases. In our final selection, we stratified to ensure variance along all of these as well.

Finally, we sampled such that we could include one project that had last been updated more than one month before, and thus might be considered inactive, since during pilot interviews project dormancy status seemed important. The other four projects were still active at the time.

Quality of README. During pilot interviews participants paid close attention to a project's README. To ensure variance in the README "quality" across our projects, we stratified our sample by the amount of information in (length of) the READMEs. Moreover, since prior work has shown that badges have high signaling value [27], we also sampled for variance across repository badges; our five projects range from no badge to over 10 different badges.

Limitations. Note that we tried to stratify across more dimensions than there are projects in our final sample (five total), meaning that some dimensions are confounded. This design decision was necessary to keep the interviews short.

2.3.3 Interview Participants

We sampled candidate participants from among GitHub users who had recently made pull requests to collaborative open-source JavaScript projects on GitHub, which we define as those projects involving at least three contributors, as per the public GitHub data mined from Google's BigQuery;[2] this helps exclude many "toy" projects [128] and increases the likelihood that that our interviewees are experienced open-source practitioners. Information about the programming language (JavaScript) was extracted from the label that GitHub assigns automatically to each repository. As an additional data cleaning and filtering step [128], we also excluded projects we could manually label as "educational" based on keywords present in their description, *e.g.*, course number (COS496).

Then, we sent out several rounds of email invitations (123 emails total) and carried out 15 interviews via Google Hangouts or Skype, at which point we considered that we had

[2] https://cloud.google.com/bigquery/public-data/github

reached theoretical saturation [129] after an informal analysis. The interviews were conducted individually and each of them took between 20 to 45 minutes. The participants were not compensated.

Among the 15 participants, the length of open-source experience ranged from one month to more than 10 years. Nine participants were full time software engineers. The occupations of the other six participants ranged from technical writer to researcher. Five were located on the US West Coast, three on the East Coast, three in Asia, one in Africa, two in Europe, and one in Oceania. Table 2.2 summarizes the participants' demographic information.

2.3.4 Data Analysis

The interviews were audio-recorded, transcribed verbatim, and coded independently by two authors. Then the coded transcripts were analyzed based on the grounded theory methodology [129]. We first identified signals mentioned by participants and how they were using these signals to make decisions. We then grouped these signals and participants' comments into categories and extracted relationships between the categories. We repeatedly discussed the categories and refined them iteratively as more interviews were conducted; this is also when we resolved a few disagreements, through discussion, between the two coders. We continued this process until new interviews did not reveal new signals that were not captured by our codes (theoretical saturation).

2.4 Interview Results - Recognizing the Signals

Our qualitative analysis identified a rich set of signals that the participants rely on when evaluating whether a GitHub project is worth contributing to by the Jane persona.

2.4.1 Website

The website link in the project description is usually the first thing the participants saw. Six participants (P1, P2, P7, P8, P10, P13) mentioned that "*the first thing I typically do is see if they have a website at all*" (P2). A website is even more important for UI libraries, to "*show a demo of what the components look like. It would be helpful to make people more interested in the project I think.*" (P10). Maintaining a good website is also recommended by many open-source practitioners.[3]

2.4.2 README

The README.md file is one signal that every participant commented on, *e.g.*, "*the README is a project's welcome mat*" (P14). Several aspects of the README seem important:

Structure. Prior work [130, 131] found that projects with good READMEs tend to be more sustainable and more popular. Our participants confirmed that a well structured README can give a nice first impression. P12, a technical writer, summarized that an ideal README

[3] https://opensource.guide/finding-users/

should have a table of contents, contributing guidelines, and information on how to get in touch with the community, "*which is very very important for a newcomer*" (P12). P7 mentioned that there is an "*unofficially agreed template of a project,*" and maintainers should "*follow what everyone else is doing*" (P7).

Project description. Participants were looking for clear descriptions of the project in the README. Not being able to understand the project's goals induced negative impressions, even rejections, among some participants (P2, P7, P8, P12). For example, P7 noted that a good README *"allows [one] to understand what this project is about, how to install it, and how to use it. It also gives examples of code snippets for its API and their effects."* (P7)

Contact information. Being able to communicate with project maintainers was seen as important to contributors, especially newcomers. P2, P11, and P12 mentioned that mentioning the project's Slack channel in the README is a welcoming signal. P14 mentioned that it is nice to be able to follow the maintainer on Twitter. Having a Twitter handle is in fact suggested by some open-source practitioners. Such practices may alleviate the barrier of communication difficulties, which was identified by Steinmacher *et al.* [105], to some degree.

Code quality badges. Five participants (P3, P7, P10, P11, P14) mentioned badges but had diverging opinions about them. Some noted that the presence of badges, especially code coverage, suggests that the maintainers care about code quality (P7, P11) and that contributing to this type of project can improve one's skills (P11). In contrast, others explained that they ignore badges because "*a lot of projects have build passing badges but actually the project is broken or really out of date*" (P10).

Logo. Four participants (P5, P8, P10, P14) mentioned the Logo in the README, *e.g.*, "*They've even got a logo. That's quite promising. Because that means someone cares enough about the project.*" (P10).

2.4.3 Contributing Guidelines

Contributing guidelines, either in the README or the CONTRIBUTING.md file, were important in all participants' decision processes. Some noted that the contributing document is a decisive signal in the sense that lacking one would induce an immediate negative impression (P3, P15). In contrast, the existence of contributing files "*suggests they have some experience with handling new contributors*" (P4). Participants expect that contributing guidelines have several characteristics:

Prominent. The first thing participants mentioned about contributing guidelines is how easily they can be found (P3, P4, P5, P12, P14, P15). As per signaling theory, since potential contributors tend to prefer signals that are easier to observe and to interpret over those that are costlier to assess, it is desired to have a link to the CONTRIBUTING.md or a contributing section in the README, *e.g.*, "*the README is most important. It should describe without having to navigate away from that page the key information people need*" (P14).

Thorough. Many participants (P1, P2, P3, P10, P12, P14, P15) remarked on the contents of contributing guidelines, expecting code style guidelines and project conventions, as well as how to submit a pull request. In particular, maintainers should set the expectation by listing out things that need help and things that are allowed or disallowed. P14 also pointed out that it is nice that "*It says 'please ask first' because otherwise people might feel that the pull*

requests always have to be merged in" (P14). Thorough contributing guidelines may lower the barrier of lacking knowledge about procedures and conventions, identified by Balali *et al.* [8]. Contributing guidelines should also explain the GitHub jargon, *e.g.*, "*a lot of new people who don't know GitHub don't necessarily know what the issue tracker was*" (P14). Moreover, some terms are project-specific. During the interviews, some people were confused by some terms they had not seen before, *e.g.*, "pre-commit" (P14).

However, having too detailed contributing guidelines may be perceived as too much process, especially by newcomers, who may find the instructions difficult to follow (P2, P4, P5, P10, P15). P15 summarized that "*if you are a new developer and you are just learning, you might not get this sort of hands-on response if you didn't properly submit an issue or pull request; your issue / pull request might just sit there and get closed without much explanation*" (P15). In addition, potential contributors may interpret language such as "*talk to [the maintainers] before any significant pull request*" (P2) as unwelcoming. Pull requests that do not follow project guidelines or that are considered not needed or interesting by maintainers are common barriers faced by newcomers [105].

Open to non-code contributions. Six contributors (P2, P5, P10, P11, P12, P14) stressed the importance of explicitly mentioning other acceptable types of contributions besides code, such as writing documentation. At the same time, invitations to submit issue reports without also soliciting code contributions can be seen as uninviting for someone interested in contributing more. As P12 put it: "*They only ask for filing an issue if something breaks. So I think they are more looking for people to test all these components for them, rather than asking for code contributions.*" (P12).

2.4.4 Scaffolding

Most participants commented on the guidelines for submitting issues and visited the issue trackers during the interviews. There are several signals they look for there:

Labels. Two types of labels, which we classify as technical and social, emerged as important signals. The social labels, pointing people to issues that are suitable for beginners,[4] are especially useful for newcomers. As P1 summarized, "*good open source projects would have labels like 'help wanted', 'good first issue'*" (P1). These can help contributors find their way around a new project.

In contrast, other labels can give contributors some technical information about the issue, *e.g.*, the programming language, or whether it's a bug or a feature request. Having issues clearly labeled with technical attributes can help contributors find the issues they aren't just able to resolve, but are also interested in working on. As one of the participants said: "*you want to work on X, and you come in and see the things that need to be done on X*" (P14), such as front-end.

Templates for issues and pull requests. Seven contributors (P1, P2, P3, P10, P12, P14, P15) noticed the templates for submitting issues or pull requests. Having a template can prevent newcomers to submit issues or pull requests that are "*stupid*" or lack information, because the "*template will take them through a bunch of different pieces of information that*

[4] E.g., "Good First Issue" proposed by Kent Dodds in 2015 https://blog.kentcdodds.com/first-timers-only-78281ea47455

they need to submit" (P14). It is a sign that shows "*there's a good structure for contributing to [this project]*" (P10).

2.4.5 Activity

Participants also look for a multitude of signals indicating the project is being actively maintained.

Number of contributors. While this was a prominent signal during our interviews, participants disagreed on what is a good team size, referring especially to newcomers. Recall that our sample comprises one large project (over 1,000 contributors), one single-person project, and the rest are small-medium sized (6, 7, and 18 contributors). A priori, we could have expected that larger projects are more likely to attract developers [132]. Indeed, among 11 participants who talked about team size, five mentioned reasons why a big project may be a better choice for newcomers. One reason is that with more contributors in the team, the project can be more sustainable. If there are only 1 or 2 people in the team, once these members leave, either the contributors' efforts are wasted or they need to take on the onerous maintenance job themselves (P6, P10, P15).

Another reason is about the mentorship opportunities one can access in big projects. Maintainers tend to be busy and they might be slow to respond to newcomers. If there is a large community, there is a higher chance that someone will be available to assist newcomers (P1, P2, P6). P2 also suggested that newcomers should avoid single-person projects because it is possible that these projects are unfriendly to external contributors (otherwise they would have more).

On the other hand, P4 and P15 listed out reasons against choosing big projects, referring mostly to the process overhead in submitting a pull request, which may intimidate newcomers. Pull request "bureaucracy" is a known barrier for newcomers [105]. However, P2 acknowledged that "*[while] the barrier to entry may be higher because the standard is higher, there are more people to help you*" (P2).

Six participants (P2, P3, P4, P5, P10, P15) suggested they prefer to start with smaller projects. One advantage of a small project is that the maintainers may be more responsive. Unlike big projects, which are "*so widely used and huge that it might take a while for maintainers to respond*" (P3), "*there's a chance that the author would be willing to reply to any pull requests you make*" (P10).

Another advantage of a small project is that contributors can get more feedback from the maintainers, which can help them improve their pull requests. Otherwise, P10 noted that "*if you have too many people, the developers don't have time to look at your individual pull requests. I bet that if I put a pull request, it will build fail or something and no one would care, they would just ignore it.*"

Furthermore, four participants (P2, P3, P4, P15) pointed out that smaller projects are preferable for newcomers to learn the GitHub workflow because in bigger projects "*it would be hard to figure out where to start even though things are relatively well labeled*" (P15).

Recent commits and contributors. Many participants (P1, P2, P5, P6, P7, P8, P10, P12, P13, P14, P15) suggest looking at the number of *recent* commits and contributors, rather than the total number. Otherwise, people will assume that the project is *"not under*

active development, because nothing has happened [for some time]" (P14). Recency of activity signals that *"the project is not dead"* (P5).

Contributions are evenly distributed. Some participants (P2, P6, P10, P14, P15) suggest that contributors should also pay attention to whether the contributions are evenly distributed among existing team members. P6 has summarized the rationale: "*For projects of middle or small size, if contributions are evenly distributed among contributors, it is acceptable. But if only one or two people are the core contributors, then it would be dangerous; [the project may be left unmaintained]*" (P6).

This practice is also recommended by Karl Fogel. In his book *Producing Open Source Software*, he recommends to "measure commit diversity, not commit rate."[5]

Average time for responses to issues or pull requests. Another important signal is how long it takes maintainers to respond to issues or pull requests (P1, P3, P10, P11, P12, P14, P15). To make this inference, participants browsed through multiple issues or pull requests.

Numbers of open issues or unmerged pull requests and their reasons. When looking at the list of issues / pull requests, participants noted that it was important to look at the number of open issues / unmerged pull requests and why they are not resolved (P10, P11, P13, P14, P15). The reason is well summarized by P11: "*I want to know why these PRs are not merged. If I send a PR, I don't know whether or not this project is being maintained. I wouldn't want to waste the effort put in to understand their code base or write code. I don't want to write something and be treated like that*" (P11).

While the number of these unresolved issues or pull requests can be easily observed, the reasons are difficult to infer. P15 pointed out that an active project should make sure that "*either pull requests are getting merged, [or] having some kind of labeling system, so people understand why so [and it] doesn't just feel like it's lack of progress*" (P15).

Percentage of issues or pull requests by external contributors. Two participants (P10 and P14) have looked at how many issues or pull requests had been made by outsiders. Looking at only the number of open or merged pull requests can be deceiving in some cases. As P10 discovered, "*[This project has] a lot of closed PRs, which is interesting. But all are from the same person. I would say they are just using PRs as branching. They are just branches being merged.*" P14 described this type of projects as "*technically open without actually being meaningfully open*" (P14).

Responsiveness in issues and pull requests. Many participants (P1, P3, P4, P5, P10, P11, P12, P13, P14, P15) examined how fast do maintainers respond to issues and pull requests. Their expectations are summarized by P14: "*[An] active project [should have] some conversation happening, and generally it has been positive and ideally with reasonably quick responses. It doesn't have to be lightning quick. But more than three days between responses is not a great place to start*" (P14).

Another signal that active discussions give is the mentorship and learning opportunity offered by code review. As one participant puts it, code review is "*pretty good because you will need to follow their appropriate code style. That's an important code style. Being able to integrate [code] into their own system is a useful skill to have*" (P10).

[5] https://producingoss.com/en/evaluating-oss-projects.html

2.4.6 Code quality

Eight participants (P4, P7, P9, P10, P11, P12, P13, P14) examined the code quality during their evaluation. One signal they look for is the presence of tests. As P7 put it, "*I wouldn't use component libraries without unit tests*" (P7). Another signal they paid attention to is the use of continuous integration (CI), especially in big projects. As P14 noted, "*If the developers can't reply immediately, it's helpful to have a CI that tells you if your code works, and if the code style is ok or not*" (P14). Two participants (P10, P13) also looked at the structure of the code itself, commenting on the importance of modularity, which makes it easier for people to understand. P9 mentioned the size of the code, which may affect whether people would use the library.

2.4.7 Popularity

The number of stars and the number of downloads of a project reflect the project's popularity. Although nine participants (P1, P3, P4, P5, P7, P8, P10, P11, P13) commented on the number of stars, only three (P1, P5, P10) mentioned that the popularity may influence their decisions. Both P1 and P10 mentioned the potential impact of contributing as an important motivation, *e.g.*,: "*Everyone uses [project X]. If you contribute to it your change is gonna have a huge impact.*" (P1) Moreover, P5 mentioned that "*if this [project] has tons and tons of stars, and there weren't that many contributors, I would think they weren't super friendly to new people*". However, P7 acknowledged that the number of stars can be faked, therefore it is not an entirely reliable signal. P3 also explained that she would not worry about popularity, because a less popular project "*gives you more self-efficacy that forces you really to look at things, google things, try everything out, and then ask for help*" (P3).

2.4.8 Community Openness

Five participants (P1, P2, P3, P5, P15) remarked on the openness of the community, as it transpires through the language used, *e.g.*, in the project documentation and issue discussions.

Language in contributing docs. Three participants noted the gender exclusiveness of the language in documentation, referring to one project which talks about "*nice guys*" that will review and merge pull requests when describing how to contribute. Two participants voiced concerns about the gender inclusiveness of this phrase. As one of the participants suggested, projects should "*avoid language that uses 'guys' or assumes that people are [all] one gender or one demographic*" (P15).

Participants also mentioned the language exclusiveness towards newcomers. Although no one identified any instance of aggressive expressions towards newcomers, some did mention that they would "*look at the language throughout to feel whether it's inclusive or it feels maybe a bit of a boy's club or sort of aggressive, or intimidating for beginners; these would make me stay away*" (P15).

Two participants also noted that "don't" may sound intimidating. Phrasings like "*'please do this', 'you are welcome to do that', by turning the language around*" are recommended instead (P15).

Conversations in issues or pull requests. The openness of the community can also be inferred from these conversations. According to P3, a good conversation should be "*commenting back and forth, [...] pretty thorough. I think it's helpful. No one is mean necessarily*" (P3). Sometimes, not following the process can "*get people mad at you*" (P5).

Code of conduct. The presence of a code of conduct signals a welcoming community. One participant told us that "*This project has a code of conduct, and they've adopted the standard contributor covenant.*[6] *So my belief is that this would be a welcoming community because people are conscious of having a code of conduct*" (P2). Being kind to contributors has been encouraged by many people and organizations. For example, Scott Henselman posted a blog in 2015 that pledged people to treat newcomers nicely, including writing a contributing guideline, tagging issues that are good for newcomers, and having a code of conduct.[7] Prior research by Tourani *et al.* [133] has also discussed the importance of having a code of conduct; however, only relatively few projects have them, though they are becoming increasingly common [133].

Gender representation. Two female participants pointed out that the gender balance among the existing contributors, as inferred from their GitHub profile information, might be a potential barrier to female newcomers. More specifically, they both pointed out that a medium size group (in our case, the project has 5 contributors) of male contributors may form a clique that a female contributor could have difficultly breaking into. However, if the project's only contributor is a man, then it is "*not as difficult a community to break into as a group of men.*" (P14). In addition, for large projects with hundreds of contributors, "*because there are so many people contributing, it doesn't matter so much whether it's all male*" (P14). As the other participant summarized: "*If I saw a project where it seems like a mix of genders, I would definitely feel more excited about the project*" (P15).

2.5 Quantitative Analysis Methods

To triangulate our interview findings, we set out to quantitatively test the overall hypothesis that the signals we identified from the interviews are indeed associated with attracting more new contributors. We collected a large dataset of open-source GitHub projects, operationalized the signals uncovered during our interviews, and used multiple regression analysis to model the association between the different signals and the likelihood of attracting new project contributors (binomial logistic regression), as a way to validate the perceived importance of each signal.

The multivariate regression analysis seeks to uncover whether any (and which) project characteristics and signals, observed over a fixed period of time (details below) help explain the average differences between projects in likelihood of attracting new contributors, as observed over a subsequent fixed period of time. The multivariate nature of the regression modeling enables us to quantify the strength of the association between each explanatory variable and the binomial outcome while *adjusting for other covariates*, *i.e.*, removing confounding effects.

Specific hypotheses. Based on the interview results, we hypothesize that other variables held fixed, open-source projects are more likely to attract new contributors when: they list a

[6] https://www.contributor-covenant.org/

[7] "Bring kindness back to open source" https://www.hanselman.com/blog/BringKindnessBackToOpenSource.aspx

project website (H_1); are more popular (H_2); are active (H_3); have a comprehensive README (H_4); list the owners' contact information or support channels, *e.g.*, Twitter, Slack (H_5); include badges reflecting code quality (H_6); include CONTRIBUTING instructions (H_7); label their issues to help steer contributors (H_8); provide issue or pull request templates (H_9); have fast response times to pull requests (H_{10}); and are welcoming towards newcomers (H_{11}).

Data. We collected a sample of 9,977 open-source JavaScript libraries published on the npm package registry[9] and available publicly on GitHub as follows. We started from a pre-existing list of the 50,000 npm packages with the most GitHub stars (min 6, median 69, max 70,266) and further randomly sampled another 2,000 npm packages with at most 6 stars as of June 1st 2018 (when the other data ended), to better stratify the data. Next, we used GHTorrent [84] to identify which of these projects: (1) were not forks of another repository; (2) had at least one commit between January 1st 2018 and June 1st 2018, to filter out completely inactive projects; (3) had non-empty README files; and (4) had at least one issue / pull request on GitHub, with at least one comment, to ensure that our measures of maintainer responsiveness and politeness (see discussion in Sections 2.4.5 and 2.4.8, respectively) are not undefined.

Measures. For each project, we used the GitHub API and GHTorrent to measure the set of variables in Table 2.3 (summary statistics in Table 2.5). The response variable in the regression models is a boolean flag *has new contributors*; see table for definition. The table also describes the main explanatory variables used, corresponding to the specific hypotheses above. In addition, we tested the presence of three interaction effects between project size / level of activity and having contributing guidelines, badges, and a link to a project website, respectively; see table for rationale.

Modeling considerations. We built two multivariate binomial logistic regression models corresponding to the two versions of our binary response variable *has new contributors*: one for any new pull request submitters and one for new pull request submitters that are also new to GitHub, not just the given project.

In each case, we log-transformed variables, as needed, to reduce heteroscedasticity [135] (Table 2.4 lists which variables were log-transformed). We also tested for multicollinearity using the variance inflation factor (VIF), comparing to the recommended maximum of 5 [136] (Table 2.6); no variable exceeded the threshold. We assess the goodness of fit of the regression models using McFadden's pseudo R^2 measure [137] (Table 2.4). Finally, we report the regression coefficients together with their p-values and estimates of their effect sizes (units of variance explained) from ANOVA analyses (Table 2.4); odds ratios for the different factors can be obtained by taking the exponential of the regression coefficients.

2.5.1 Replication Package

Our data collection and data analysis scripts, and the input data for the regression models in Table 2.4, are part of a replication package available online.[10]

[8] The first page of closed issues on a project's GitHub profile shows 30 entries.

[9] https://www.npmjs.com

[10] https://doi.org/10.5281/zenodo.3371186

Table 2.3: Overview of the different variables we computed and modeled.

Variable	Signal	Definition / Rationale
		RESPONSE VARIABLE
Has new contributors	§2.4.5	Boolean flag measuring presence of new pull request submitters between June 1st 2018 and September 1st 2018. To test the sensitivity of our analysis to this operationalization, we distinguish between new contributors *with* (model "Any new contributors" and hypotheses $H_{1...11}$) and *without* (model "GH first-timers only" and hypotheses $H'_{1...11}$) GitHub experience in other projects prior to the current one.
		CONTROL VARIABLES
Has external committers	§2.4.5	Boolean flag indicating if there were commits made by non-core contributors; core is defined as people each authoring at least 5% of the commits from January 1st 2018 to June 1st 2018. Controls for general openness of the project to newcomers.
Project age		The age of the project on June 1st 2018, in days. Controls for software evolution: a project in a developing stage may have more issues for new contributors to work on than a mature one.
Num issues	§2.4.5	Total number of issues (not including pull requests) on June 1st 2018. This number is a highly visible signal at the top of a GitHub project's main page. Projects with more issues are likely to have more work available for contributors, as well as larger potential contributor pools.
		MAIN EXPLANATORY VARIABLES
Has website (H_1, H'_1)	§2.4.1	Boolean flag indicating if the project contains a homepage URL.
Num stars (H_2, H'_2)	§2.4.7	The number of stars on June 1st 2018 as per GHTorrent, as a proxy. for project popularity.
Num recent commits (H_3, H'_3)	§2.4.5	Total number of commits from January 1st 2018 to June 1st 2018, as a proxy for project activeness.
Num headers (H_4, H'_4)	§2.4.2	The number of markdown headers (H1–H3) in the README, as a proxy for comprehensiveness.
Has contact info (H_5, H'_5)	§2.4.2	Boolean flag indicating if the README contained references to a Twitter handle or Slack channel.
Has badges (H_6, H'_6)	§2.4.6	Boolean flag indicating if the README contained code coverage or continuous integration badges.
Has contrib (H_7, H'_7)	§2.4.3	Boolean flag indicating if the repository contained a CONTRIBUTING.md file or if the README contained a section on how to contribute.
Has labels (H_8, H'_8)	§2.4.4	Boolean flag indicating if the project has labels applied on issues or pull requests.
Has template (H_9, H'_9)	§2.4.4	Boolean flag indicating if templates were used for submitting issues or pull requests.
Is fast (H_{10}, H'_{10})	§2.4.5	Boolean flag indicating if the median response time to the the 30^8 most recently opened issues which were closed as of June 1st 2018 is below the first quartile of projects. Responses count if a non-obviously-bot user nor the issue author comments or performs an action on the issue.
Is impolite (H_{11}, H'_{11})	§2.4.8	Boolean flag indicating if the project's median impoliteness score ranks in the top quartile across our sample. We collected impoliteness scores for the comments in the first page of closed issues as seen on June 1st 2018 using the Stanford Politeness API [134], after removing markdown formatting and replacing each code block with the token "`CODE.`".
		INTERACTIONS
Has contrib × Num recent commits (H_7, H'_7)		Contributing guidelines may impact larger, more active projects differently, as there could be more need for help navigating project norms and processes.
Has badges × Num recent commits (H_6, H'_6)		Badges displaying negative project qualities, *e.g.*, broken build, may create more negative impressions the less active the project is, making it appear abandoned.
Has website × Num recent commits (H_1, H'_1)		A potentially broken link, more likely to occur in a less actively maintained project, may increase the appearance of abandonment.

Table 2.4: Summary of logistic regression results showing which signals associate with new contributors.

	Any new contributors Response: *has new contributors* Pseudo $R^2 = 20\%$		**GitHub first-timers only** Response: *has new contributors* Pseudo $R^2 = 21\%$	
	Coeffs (Err.)	Deviance	Coeffs (Err.)	Deviance
(Intercept)	0.79 (0.47)		-1.93 (0.75)**	
has external committers	0.60 (0.06)***	88.54***	0.36 (0.10)***	11.95***
project age (log)	-0.60 (0.07)***	80.25***	-0.50 (0.11)***	22.18***
num issues (log)	0.43 (0.03)***	236.81***	0.56 (0.05)***	140.38***
has website	-0.43 (0.10)***	19.80***	-0.17 (0.17)	0.92
num headers (log)	0.10 (0.03)**	9.59**	0.08 (0.05)	2.06
has contact info	-0.12 (0.07)	2.86	-0.03 (0.10)	0.10
has contrib	-0.31 (0.11)**	0.76	-0.46 (0.20)*	10.14**
has badges	0.14 (0.09)	1.51	-0.49 (0.16)**	6.79**
has labels	-0.13 (0.05)*	6.19*	-0.08 (0.09)	0.84
has template	0.48 (0.16)**	9.11**	0.25 (0.16)	2.52
num recent commits (log)	0.12 (0.03)***	62.53***	0.07 (0.05)	38.66***
is fast	-0.04 (0.06)	0.52	-0.10 (0.09)	1.04
num stars (log)	0.21 (0.02)***	97.53***	0.14 (0.03)***	17.02***
is impolite	-0.32 (0.07)***	20.99***	-0.08 (0.12)	0.43
has contrib : num recent commits (log)	0.11 (0.04)**	7.18**	0.05 (0.05)	0.87
has badges : num recent commits (log)	-0.04 (0.04)	1.11	0.10 (0.05)*	4.05*
has website : num recent commits (log)	0.09 (0.04)*	6.24*	0.09 (0.05)	3.01
AIC	10442.37		4694.43	
Num. obs.	9977		9977	

$^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$

2.6 Regression Modeling Results - Triangulating the Signals

In this section we discuss the quantitative analysis results for our main model ("Any new contributors" in Table 2.4). We further test the robustness of our results to the operationalization of *new contributors* by modeling "GitHub first-timers only" as the dependent variable (Table 2.4). Both models have acceptable goodness of fit (20%–21%). We will contrast the qualitative and quantitative results and discuss implications of our results later, in Section 2.7.

2.6.1 Attracting any new contributors

From Table 2.4 (model "Any new contributors"), we first observe that the control variables expectedly account for around 60% of the variance explained by the model (sum of the cell values corresponding to the control variables in the Deviance column in the table, divided by the total amount of Deviance explained by the model, *i.e.*, sum over all rows), with predictable effects: projects with more open issues (signaling contribution opportunities [26]) or that are younger or historically more open to newcomers are more likely to attract additional new contributors, on average.

Moving on to the main explanatory variables, we observe that projects with more GitHub **stars** (supporting H_2 ✓), more **recent commits** (signaling the project's activity level [26]; H_3 ✓), more **comprehensive README files** (more headers; H_4 ✓), and having **issue or pull request templates** (H_9 ✓) are statistically significantly more likely to attract new contributors, supporting our hypotheses and the qualitative data collected during the interviews. Taken together, the four variables account for approximately 27% of the total variance explained by the model.

Among these variables, the number of GitHub stars, a signal of project popularity, explains the largest amount ($\simeq$ 15%) of the total variance explained by the model. We illustrate the interpretation of the regression coefficient: for every factor e increase in the number of stars (note the log-transform), and after controling for the amount of project activity and other covariates, the odds of attracting new contributors for the average project in our sample increase $\exp(0.21) \simeq 1.23$ times. Fronchetti *et al.* [125] found, similarly, that project popularity is the most important factor that explains newcomers' growth pattern.

The first model also shows that the number of recent commits explained a large amount ($\simeq$ 10%) of total variance explained by the model. As some of the participants pointed out and also discussed by Dabbish *et al.* in [26], the number of recent commits signals the projects' activity level and contributors' commitment to a project. More recent commits in a project signals that there are active contributors who could provide help or feedback if needed. Therefore, programmers may be more willing to join the project.

Arguably, all four of these signals (stars, recent commits, comprehensive READMEs, and templates) have relatively high production costs, as they require deliberate and in some cases sustained efforts (*e.g.*, sustained commit activity over time) from project core developers to maintain. Given this production cost, signaling theory predicts that the signals are reliable. Our quantitative results are consistent with this prediction.

Table 2.4 also shows that posting a **website URL** (H_1 ✗), having **contributing guidelines** (H_7 ✗), using **issue labels** (H_8 ✗), and being **impolite** (H_{11} ✓) have, on average, statistically significant negative effects on attracting new contributors. The effect sizes are, however, relatively small: taken together, the four variables explain $\simeq$ 9% of the total variance explained by the model.

It is not surprising that being impolite correlates with lower likelihood of attracting new contributors: Balali *et al.* [8] found that a "harsh project atmosphere" is one of the main barriers that a newcomer faces. However, it is surprising that having a website URL, contributing guidelines, and issue labels correlates with lower likelihood of attracting new contributors.

The interaction effects (Figure 2.4) with the number of recent commits for two of the variables, *has website* and *has contrib*uting guidelines, suggest a more nuanced interpretation. For the *has contrib*uting guidelines dummy (Figure 2.4 right), the estimated coefficient is negative for low values of *num recent commits* but positive for high values. That is, for the less active projects, having contributing guidelines correlates with lower likelihood of attracting new contributors, holding the other variables fixed; but for the more active projects the relationship flips, and having contributing guidelines correlates with higher likelihood of attracting new contributors, as hypothesized.

For the *has website* dummy (Figure 2.4 left), the estimated coefficient is only negative for low values of project activity (*num recent commits*), and is otherwise indistinguishable from

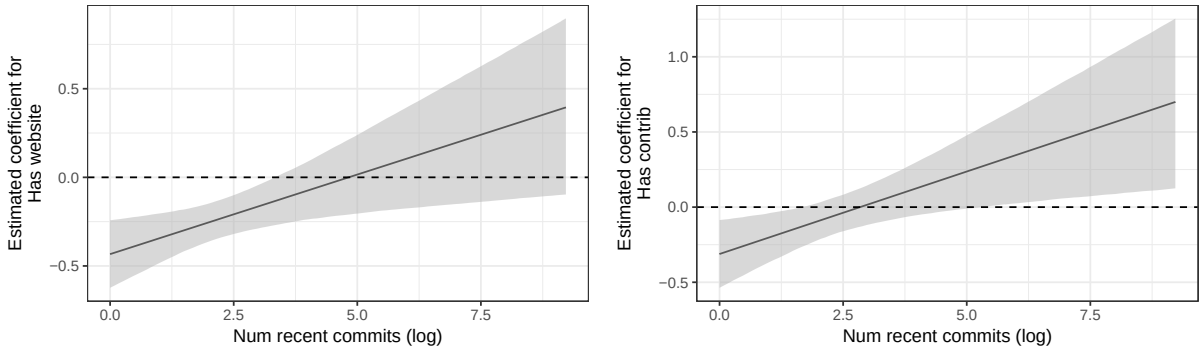


Figure 2.4: Visualization of the interaction effects Has website (left) / Has contrib (right) × Num recent commits.

zero. That is, only for the less active projects having a website URL correlates with lower likelihood of attracting new contributors, holding the other variables fixed, whereas for more active projects having a website URL has no effect. One explanation could be that the websites of smaller, less active projects may be more often out of date or unmaintained, accentuating potential negative first impressions. Another explanation could be that unobserved third variables are confounding the association. More research is needed to better understand this relationship.

For the *has labels* dummy, we did not theoretically expect an interaction effect, therefore we did not test for one. Still, the negative effect of having labels might be explained by a limitation of our operationalization: due to lack of uniformity in how "good first issue" labels are named across projects, we only recorded a binary flag of whether a project has any labels at all, as a proxy, while it could be that only labels similar to "good first issue" have the hypothesized positive effect on attracting new contributors. Future work could refine our operationalization.

Finally, we note from Table 2.4 that having **contact information** (H_5 ✗), **code quality badges** (H_6 ✗), and **fast reponses** (H_{10} ✗) to issues or pull requests do not have statistically significant effects, contrary to our hypotheses.

2.6.2 Attracting first-time GitHub contributors

The "GitHub first-timers only" model in Table 2.4 uses as dependent variable the presence of first-time contributors, who have never made any GitHub contribution before. This alternative operationalization enables us to assess the robustness of our results and study whether contributors without any public traces of open-source GitHub experience might look for different signals when evaluating projects. Such differences could be the signals that are more reflective but unknown to first-time contributors either due to the contributors' own lack of experience or the signals' low visibility. They could also be signals that are only relevant or important to first-time contributors. Since the Jane persona we provided to participants was designed to be a first-time contributor and participants were projecting their own experience

onto her, comparisons between the two models can also help to differentiate participants' projection and first-time contributors' own decision.

A comparison between the two models shows that the **number of stars** remains a strong positive predictor of attracting newcomers (H_2 ✓): the more popular a project, the more likely it is on average to attract newcomers. In addition, having **contributing guidelines** has significantly negative effect when attracting first-time GitHub contributors (H_7 ✗). However, most of the effects of the main explanatory variables have changed. Having **badges** has a statistically significant but negative effect (H_6 ✗). The interaction effect with recent project activity is also statistically significant and behaves similarly to the *has website* (left) interaction in Figure 2.4: the negative correlation between having code quality badges and likelihood of attracting new contributors is only visible in less active projects. We speculate that using CI and showing code quality badges may increase the process overhead and barrier to entry, and could be discouraging to first-time contributors, who may not have sufficient CI experience.

Other signals have statistically insignificant effects in the second model: the number of **recent commits** (H_3 ✗), having a **website URL** (H_1 ✗), **contact information** (H_5 ✗), the **number of headers in the README** (H_4 ✗), **labels** (H_8 ✗), **templates** (H_9 ✗), **fast responses** (H_{10} ✗), and **politeness** (H_{11} ✗). Signaling theory offers one possible explanation: these signals are not visible enough, therefore receivers, in this case, first-time GitHub contributors, might prefer signals that are easier to observe and to interpret. The number of recent commits is not a directly visible signal, rather it requires combining the number of commits and the last commit date. Similarly, labels and templates do not typically appear on the main page. For example, templates usually show up only when users begin to compose an issue or a pull request. Finally, evaluating the politeness and responsiveness of a project also requires contributors to look into documentation and conversations. It is also possible that new contributors lack a benchmark of politeness as a reference and may consider potentially impolite interactions as the norm; however, as they meet more people, they gradually become aware of the culture of a project and try to avoid impolite teams.

2.6.3 Commonalities and discrepancies between interviews and models

The project's popularity, signaled by the number of GitHub **stars**, and having a **contributing guideline** are the only explanatory variables that had consistent effects between our two models, which aligned with the interview results.

In the other cases, we found interesting discrepancies between the interviews and regressions. Particularly notable is the **responsiveness** which was expected to be important signal both according to interview participants as well as GitHub's 2017 Open Source Survey [1] results (Figure 2.3), but shows no results in the regression.

The negative correlation between having contributing guidelines and likelihood to attract new contributors in less active projects (recall the interaction effect above) warrants further investigation. One possible explanation is a limitation of our experimental design: contributing guidelines may have had stronger, positive effects closer in time to when they were introduced in each project, but our fixed window of observation (June 1st to September 1st 2018) hides

this. Further investigations go beyond the scope of this paper, but could be a worthwhile direction for future research.

Beyond threats to construct validity (our operationalization of responsiveness of the core team could be imperfect), signaling theory offers one possible explanation for the lack of noticeable effect for the responsiveness variable. Even if potentially reliable and hard to fake (*i.e.*, an assessment signal), the signal is not plainly visible on a project's main page. While in our interviews some participants did click through individual issues or pull requests pages to estimate the response time, in a less artificial setting people may not spend as much time on evaluating projects and may rely on more visible signals instead. More research is needed to understand whether the lack of hypothesized effects is due to limitations in our operationalizations or other causes.

2.6.4 Limitations

We now note some important limitations of our quantitative study. We discussed limitations of our qualitative analysis previously, in Section 2.3.

First, we computed a set of proxies (Table 2.3) to operationalize the different theoretical constructs emerging from the qualitative analysis. While our variables are arguably reasonable measures for the theoretical constructs they are meant to capture, and even though we manually inspected and iteratively corrected data collection errors, as needed, until we were confident that our data is correct, it is important to note that other operationalizations for the same concepts are possible. For example, for the response variable one could also consider other contributions besides pull requests. Different operationalizations may lead to different statistical modeling results and therefore different conclusions. We described clearly our assumptions and operationalizations and we provide a replication package[10] to facilitate future extensions to our work. Exhaustively computing and testing multiple operationalizations for each construct goes beyond the scope of this paper.

Second, the GitHub data we mined and analyzed are observational in nature, hence the different signals we considered are not true experimental treatments. This could create endogeneity problems [138, 139], which could lead to biased estimates of the treatment effects in our regressions.[11] Endogeneity could manifest in several ways. For example, even though prior work and our qualitative analysis both suggest that higher number of stars may drive higher numbers of contributors to a project, it is also possible that an unobserved variable may jointly determine both high number of stars and high number of contributors, or that both might be true. The association between the number of stars and the likelihood of attracting new contributors, surfaced by our models, may not allow readers to conclude this correlation is causal, because observational data is not randomly assigned. Moreover, endogeneity can be caused not only by omitted variables, but also by some of the regression variables used. In our study, the number of stars a project has is endogenous when examining the quality of projects or the intent to contribute to it. When a project has a higher number of stars it may attract more contributors, but it is also likely that a project which has a high number of contributors, may attract more stars.

[11] We kindly thank one of the reviewers for pointing this out and suggesting mitigation strategies. This paragraph incorporates the reviewer's comment almost verbatim.

Endogeneity has received much attention in the econometrics literature and many statistical approaches have been proposed to assess or control its impact. Perhaps the most popular approach we considered is to instrument for the possibly endogenous predictor variable [140], in our case *number of stars*. Given such instrumental variables, one then typically pursues an estimation method such as two-stage least squares (2SLS) [141]. The basic idea is to extract variation in the possibly endogenous predictor that is independent of the unmeasured confounders and use this variation to estimate the treatment effect and "control" for the unmeasured confounders. Many extensions to non-linear models such as logistic regression, which we use in our study, have been proposed [142, 143, 144, 145, 146]. However, we decided against two-stage methods for several reasons: i) these methods are only as good as the exogenous instrumental variables selected [147, 148, 149] and we could not identify appropriate, theoretically motivated instruments for number of stars; and ii) with large sample sizes, as in our case, the estimated coefficient for the residuals is more likely to reach statistical significance, *i.e.*, it becomes more likely to falsely detect endogeneity [150, 151].

Instead, we limit ourselves to checking for correlation between the possibly endogenous number of stars variable and the logistic regression residuals. Neither model had statistically significant Pearson's product-moment correlation: $p = 0.86$ for GitHub first-timers only and $p = 0.94$ for any new contributors. Although our models explain only around 20% of the variance in the data, suggesting there may be omitted variables, we did include in the regressions variables corresponding to *all* of the theoretical constructs emerging from the interviews, in addition to controls for the obvious covariates. Therefore, based on the lack of correlation between the possibly endogenous number of stars variable and the logistic regression residuals we believe that the relatively low explanatory power of our models is more likely due to natural noise in the data, common at this scale and in this domain [128], rather than omitted important variables that could cause endogeneity.

Alternative analysis techniques such as propensity score matching, which can help reduce the risk of endogeneity [152], or recent heuristics [153] for evaluating the robustness of results to omitted variable bias, based on coefficient movements after inclusion of controls and movements in R-squared values, go beyond the scope of this paper but could be worthwhile future directions.

2.7 Implications

Our study has implications for open-source maintainers, platform designers, and researchers.

2.7.1 New Signals

Among the information our participants needed to inform their evaluation of contribution worthiness for each open-source project in our sample, only some is readily observable from prominent signals displayed on a project's landing page or README file on GitHub. For example, the *number of stars*, a proxy for project popularity, and the *number of contributors*, measuring team size, are already part of the GitHub UI. However, other pieces of needed information can be much less salient. Some, like the *number of downloads*, which indicates not only popularity but also the size of the user base, have direct signals, but these are

not typically visible on GitHub directly. For example, in the case of projects with releases published on npm, the number of downloads is displayed on a package's npm page, but not on its GitHub repository page by default. We learned from the interviews and our models that popular projects tend to attract more new contributors. Badges such as 

could be used to augment a project's existing GitHub popularity signals (the number of stars), making project popularity information more salient.[12] Trockman *et al.* [27] found that badges can impact perceptions of open-source projects.

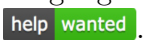
Some other pieces of information used by our interview participants and having statistically significant effects in our models currently have no direct signals at all and, instead, need to be inferred from indirect cues. The *tone of the community*, for example, is an important factor in our interviews: "*it's most important that the people seem nice*" (P5). From the first regression model, we can see that (im)politeness also has a statistically significant effect. In our interviews some participants had to browse through multiple issues and pull requests, reading the discussions therein. If these conversations were positive (P14) and people were not mean (P3), participants concluded that the community is probably friendly and welcoming. As discussed in Section 2.2.2, signaling theory explains that assessment signals, which are costly to produce / fake, tend to be reliable. A signal of the tone of a community would arguably be an assessment signal and therefore be reliable, as maintaining a welcoming tone would require sustained effort from project maintainers over time. However, signaling theory also explains that receivers (the potential contributors evaluating projects) tend to prefer signals that are easy to observe and to interpret over those that are costlier to assess [118]. This suggests that automated techniques could be used to develop new signals of the tone of a community, *e.g.*, in the form of badges, to further increase transparency and make these important underlying qualities salient. Recent advances in detection of emotions [154], politeness [155], and sentiment [156, 157, 158] suggest that this approach is feasible.

Similarly, we envision assessment signals of the responsiveness of the project maintainers, *e.g.*, displaying the average response times to issues and pull requests submitted by external contributors. Even though our models did not validate the importance of this signal, maintainer responsiveness showed up prominently in our interviews and is also well-supported as a desirable project quality by prior work (see Section 2.2.3).

We also uncovered a range of best practices and associated signals that our interview participants noted help create good first impressions when evaluating a project for potential contribution: listing an external project website, having a detailed README file, including information on how to contribute, listing contact information for the maintainers, and using labels and issue / pull request templates to help newcomers learn the project processes and norms. Two of these signals, denoting the comprehensiveness of the README file and the presence of templates, we were also able to validate quantitatively. In terms of production cost, a well thought-out README file is arguably the most expensive, as it requires a high initial investment to develop and subsequent sustained maintenance to keep it up-to-date. Signaling theory predicts that this investment is worthwhile though: our study finds using mixed methods that projects with more detailed READMEs are more likely to attract new contributors.

[12] GitHub's recent "Used by" button https://twitter.com/github/status/1131468413983961088 is similar.

Finally, we identified some conventional signals that project owners could consider adopting, as they are perceived to attract new contributors. Our interviews suggest that potential contributors are receptive to explicit requests for help, yet typically there is no associated highly visible signal at the project level. One of the recommendations for maintainers that our participants repeatedly mentioned is explicitly expressing that they want help and welcome contributions. There are multiple ways in which this intent can be expressed more visibly, including explicit language in the README such as "Accepting PRs" or the equivalent badges 

and 

. While such conventional signals are expectedly less reliable as per signaling theory since they are less costly to fake, they are still cheap to produce and may contribute to creating the impression of a welcoming community.

However, it is also possible for there to be too many signals on a GitHub project's page. Prior work by Trockman *et al.* [27] found a non-linear association between the number of repository badges displayed and the number of downloads, after controling for covariates, *i.e.*, projects with "too many" badges tend to be less popular. More research is needed to understand the situations with too many signals beyond badges, and whether some existing signals could be removed.

2.7.2 Personalized Design

During our interviews, we anecdotally observed that different contributors may interpret the same signals differently. For example, P15 explained: "*a lot of [project selections] depend on your confidence. So when it's a bigger project, are you are someone that feels comfortable jumping into the middle of things or you need a little bit more hand-holding or welcoming into the project, then it feels like this is probably the one that is easy to wander around but doesn't have the capacity to personally welcome you and help you figure out where to start*" (P15). In contrast, P3 noted that "*I don't worry about the popularity of the project because I feel like if you find things less saturated, you actually benefit more from it. There's less hand holding and you get to really dive in; it gives you more self-efficacy that forces you really to look at things, google things, try everything out, and then ask for help*" (P3).

The GenderMag literature [29] shows that groups of people that tend to differ along four problem-solving facets also tend to experience different barriers to technology and tend to use software differently [159, 8]. The facets are motivation (intrinsic vs extrinsic), computer self-efficacy (high vs low), information processing style & tinkering (reading documentation upfront vs tinkering), and attitude towards risk (high vs low risk aversion). It is possible that GitHub contributors who tend to differ along the four problem-solving facets (often gender is an attribute that people who differ along these dimensions cluster on) would also interpret the different signals differently. For example, the two quotes above suggest potential differences in interpretation of signals depending on one's self-efficacy level. This suggests that future work could take individual differences in problem-solving style into account when developing new signals, to better account for how contributors might interpret the same signals differently, *e.g.*, using the GenderMag [29] process.

2.8 Conclusions

In this chapter we used mixed methods to explore how open-source contributors decide whether or not to recommend submitting pull requests to different open-source projects based on the signals available on the project's GitHub webpage. Qualitatively, we interviewed 15 GitHub contributors about their project selection process and the signals used to inform this decision. Quantitatively, we estimated two logistic regression models using trace data from 9,977 GitHub projects, to validate each identified signal from the interviews.

Among our main findings, we highlight that contributors make inferences based on a multitude of signals, including how actively maintained and popular the project currently is, the friendliness and responsiveness of the maintainers in issue and pull request discussions, the availability of issue and pull request templates and issue labels, and a well-structured and thorough README which includes contributing guidelines. However, not all these signals are currently easily observable, *e.g.*, inferring the welcomeness and responsiveness of project maintainers involves multiple steps.

This work has direct implications for open-source maintainers and the design of social coding environments: both sets of stakeholders could focus on developing reliable new signals for the less readily observable project qualities we identified as important. Ultimately, these signals could help direct contributor effort to open-source projects where this effort is most needed, contributing to the sustainability of open-source ecosystems as a whole.

A notable limitation of our study as a whole is that controlling for topic (all projects used in the interviews are front-end-related JavaScript projects) makes it impossible to determine how important topic was compared to the identified signals. Future work should explore alternative research designs. Future work should also consider refining our operationalizations and replicating these findings on other projects that are not part of npm.

Appendix

Table 2.5: Summary statistics for the variables in Table 2.3.

Statistic	Mean	St. Dev.	Min	Median	Max
Has any new contributors	0.36	0.48	0	0	1
Has first-time-GH contributors	0.09	0.28	0	0	1
Has external committers	0.28	0.45	0	0	1
Project age	1,334.66	538.95	571	1,214	3,830
Num issues	105.45	404.52	0	22	13,198
Has website	0.35	0.48	0	0	1
Num headers	11.09	10.78	1	8	262
Has contact info	0.14	0.35	0	0	1
Has contrib	0.22	0.41	0	0	1
Has badges	0.57	0.50	0	1	1
Has labels	0.53	0.50	0	1	1
Has template	0.03	0.17	0	0	1
Num recent commits	32.86	173.44	1	6	10,087
Is fast	0.25	0.43	0	0	1
Num stars	593.95	2,464.59	0	72	70,266
Is impolite	0.16	0.37	0	0	1

Table 2.6: VIF multicollinearity test values for the variables in Table 2.3.

	Any new contributors	GH first-timers only
Has external committers	1.48	1.67
Project age (log)	1.22	1.28
Num issues (log)	2.50	3.33
Has website	1.14	1.18
Num headers (log)	1.06	1.06
Has contact info	1.06	1.09
Has contrib	1.13	1.22
Has badges	1.05	1.07
Has labels	1.13	1.25
Has template	1.04	1.09
Num recent commits (log)	1.48	1.77
Is fast	1.01	1.01
Num stars (log)	2.08	2.45
Is impolite	1.03	1.03

Chapter 3

Sustained Participation

*Sustained participation by contributors in open-source software is critical to the survival of open-source projects and can provide career advancement benefits to individual contributors. However, not all contributors reap the benefits of open-source participation fully, with prior work showing that women are particularly underrepresented and at higher risk of disengagement. While many barriers to participation in open-source have been documented in the literature, relatively little is known about how the social networks that open-source contributors form impact their chances of long-term engagement. In this paper we report on a mixed-methods empirical study of the role of social capital (*i.e.*, the resources people can gain from their social connections) for sustained participation by women and men in open-source GitHub projects. After combining survival analysis on a large, longitudinal data set with insights derived from a user survey, we confirm that while social capital is beneficial for prolonged engagement for both genders, women are at disadvantage in teams lacking diversity in expertise.*

3.1 Introduction

Sustained participation by contributors in open source software (OSS) is critical to the survival of OSS projects [160, 9], and it can provide many benefits to individual contributors [110]. For example, a recent survey [14] found that OSS work helped more than half of the respondents obtain their current positions, and that OSS work in general helps people build their professional reputation. Given the advantage that open source experience can bring to an individual and the benefit that sustained participation can provide to OSS projects, it is essential to study what retains or repels contributors.

Not surprisingly, sustained participation in OSS has attracted considerable attention among researchers, with prior work focusing on developers' motivation [160, 161, 94], the kind of tasks they perform [162, 163] , and rejection experiences [164, 165, 166, 167, 168].

However, the benefits that contributors can gain from their OSS social relations and structures have not been studied. Such benefits are known in the social sciences as *social capital* [169, 170]. Social capital can be built through individuals' social networks and has been shown to affect various kinds of human endeavors, from knowledge sharing [171] to labor force participation [172] and from philanthropy [173] to financial development [174].

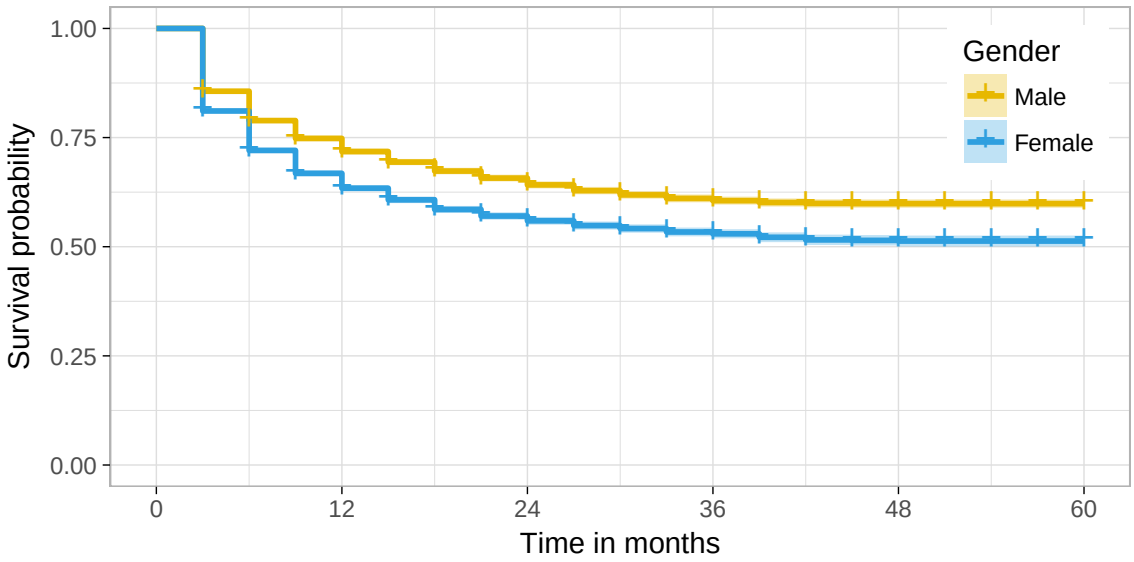


Figure 3.1: Kaplan-Meier estimators: women disengage significantly earlier. (chi-sq= 645, p$< 2e^{-16}$ per a log-rank test)

In OSS, studies have shown that prior social ties can influence forming or joining a new team [175, 176]. However, they did not explore whether and how social ties can prolong contributors' participation.

While social capital can be built and leveraged by everyone, it can impact women differently in male-dominated environments. For example, prior work in the film industry [177] found that while men benefit from strongly connected networks, women do not; moreover, women benefit from diversity in teams and tasks. In OSS, women are severely underrepresented and, as we show, likely to disengage from GitHub participation earlier than men (Figure 3.1).

To better understand contributors' disengagement, we perform a longitudinal, quantitative analysis of the structure of OSS contributors' social networks on GitHub and the impact of this structure on prolonged engagement, through the lens of social capital theory. Moreover we report on a user survey to better understand what constitutes social capital for GitHub open source contributors and how it is associated with their participation sustainability. Our findings highlight that:

- Contributing to projects where team members are more familiar with each other (from prior collaborations) is in general associated with decreased risk of disengagement;
- Women are at higher risk of disengagement than men.
- Higher team diversity along dimensions of programming language expertise is associated with a decreased risk of both short and long term disengagement. *Moreover*, gender and language diversity interact: when team members have more diverse programming language backgrounds, women are less likely than men to disengage early.

Our results have implications for project choosing, team formation, and project management in OSS. Based on our results, we especially recommend that women take project social capital and expertise diversity into consideration when choosing a project to join, and that project managers consider these aspects when allocating developers to tasks, in more centrally managed contexts. We also argue that social coding platforms like GitHub could benefit

from recommendation engines for newcomers looking for projects to join; these should take social capital into account when making a recommendation (cf. [178]); furthermore, GitHub could facilitate project maintainers tracking trends in factors negatively associated with the development of social capital, particularly among women.

3.2 Development of Hypotheses

We build on *social capital theory*, a popular social sciences theory used to explain individual and group success and performance (for an overview see Adler and Kwon [179]). *Social capital* is the set of benefits individuals can gain from their social connections and social structures, such as access to information and emotional support [179]; it is a complement to human capital, which refers to an individual's ability [170].

OSS is a social environment that can be modeled as collaborative social networks [180], where social capital can form: projects are community-based in nature; contributors have ample opportunities to connect with each other by interacting and collaborating over time; they agree on common norms; and they share collective goals—the development and maintenance of OSS. Once present, social capital can "make individuals' experiences of working on open source projects both satisfying and rewarding" [181]. In this paper we argue that *social capital also impacts the overall open source tenure of contributors*, and that female and male contributors benefit from social capital differently, on average.

There are two main network structures conducive of social capital: strong, dense, and cohesive ties generate *bonding social capital* [182], while weakly connected ties, acting as brokers between subgroups, generate *bridging social capital* [170].

The first, *bonding social capital*, emerges from *network closure*, *i.e.*, strongly connected ties [182]. Tie strength increases with the amount of interaction between individuals, emotional density, intimacy, or reciprocal service [183]. In a closed network, information is passed more accurately through direct communication [184], and trust develops more easily since it is more expensive for people to break norms when actions are more easily noticed [182]. At the same time, network closure increases group cohesiveness and solidarity among group members, who become more likely to remain engaged.

In OSS, contributors are motivated by both intrinsic and extrinsic factors, among which aspects related to bonding social capital, such as identifying with the community and feeling obligated to contribute back, are highly important [94]. Prior work showed how identification, obligation, emotional attachment, trust relationships, and shared goals and norms (all of which are more likely to develop in cohesive teams [185]) positively impact individual and team outcomes. It follows that bonding social capital should positively impact the contributors' willingness to sustain their OSS activity. In OSS participants are often free to disengage at any time, therefore the extent to which they have a sense of social identity, or perceive themselves to be part of the community, may substantially increase their intention to continue [186, 187].

In contrast to bonding social capital, *bridging social capital* focuses on how network individuals who maintain weak ties can benefit from a brokerage position [170]. In closed networks people who are strongly connected may have the same information or the same source of information. Bridging otherwise disconnected groups, what Burt calls structural holes [170], can enable access to broader sources of information and improve the information's

quality, relevance, and timeliness [179]. While bridging social capital is especially beneficial in competitive scenarios, when timely and non-redundant information about job opportunities can be an advantage, it can also be an asset in OSS. Weak ties can expose contributors to, *e.g.*, new technologies and new projects, providing opportunities to continue their engagement. Already, evidence suggests that past collaborative ties impact contributors' choice of OSS projects to participate in [176]. Network brokers can also decrease the centralization of OSS communities and increase communication between experts and peripheral users [188].

To summarize, network closure and structural holes, representing both types of social capital, seem important for sustained participation in open source. We expect that:

H$_1$. *During their open source tenure, the more often people participate in projects with high potential for building social capital, the higher their chance of prolonged engagement.*

However, network closure may not always be beneficial. As Lutter [177] notes "cohesive networks might foster discrimination and exclusion, as network closure is likely to divide [individuals] into insiders and outsiders". Outsiders, *i.e.*, those who are not part of the "core" group, can have a harder time accessing information, leading them to miss out on some chances [189, 190, 169]. Furthermore, people within a social group tend to develop their own habitus, often unconsciously. Such habitus embodies membership but also restricts outsiders from accessing and identifying with the group [191, 192, 193].

In OSS in general and GitHub in particular, socio-demographic diversity is lower than anywhere else in tech [194]. Women are particularly underrepresented, with recent surveys placing them at less than 5% [195]; women are also more likely than men to encounter stereotyping or unwelcoming language [5, 58, 7]. However, as prior results from the film industry, a similarly male-dominated field, show, women can overcome the negative effects of network closure: being more often attached to open teams with regard to diversity of ties, information flow, and genre background increases chances of career survival [177]. That is, since women tend to be outsiders to the strongly connected groups of (mostly male) decision-makers, diversifying their ties makes them less dependent on the in-group for acceptance [196]. Therefore, given women's minority (and likely outsider) status in OSS in aggregate, we expect:

H$_2$. *During their open source tenure, the more often women participate in open teams wrt diversity of ties and information, the higher their chance of prolonged engagement.*

3.3 Related work

Discrimination exists in online software engineering communities and women are known to face greater barriers than men [197]. Terrell *et al.* show that women whose gender identities are revealed have lower pull request acceptance rate [7]. Mendez *et al.* have observed biases against women in GitHub tools and infrastructure [178], while Ford *et al.* identified barriers for female participation on Stack Overflow [198]. Social network analysis has also been applied to OSS [175, 176, 180, 199, 200, 201, 202, 203], although these studies did not consider gender.

Sustained participation, turnover and disengagement have attracted significant attention as well, *e.g.*, using qualitative methods, Fang *et al.* reveal that situated learning and identity construction are associated with sustained participation [160], while Lin *et al.* show that contributors who join the project earlier, write code instead of documents, or are responsible

for modifying code have higher chances of remaining in the team [162]. The relation between turnover and project quality has been studied by Foucault *et al.* [101]. A complementary perspective has been taken by Zhou and Mockus that identified metrics such as number of comments and the size of the peers' groups as characteristics of new contributors that will become long-term contributors [204]. These conclusions, however, focused on individual behaviors and project qualities. In this paper, we analyze sustained participation from the perspective of contributors' social connections on GitHub.

3.4 Methods

We designed a mixed-methods study characterized by a concurrent triangulation strategy [126] to help triangulate our findings. Quantitatively, we collected a multivariate longitudinal data of 58,091 GitHub contributors, and performed survival analysis to model the effects of social capital on disengagement. Qualitatively, we surveyed a sample of 88 contributors to gain additional insights into the role of social capital on GitHub.

3.4.1 Data

Our main data source is the February 2017 version of GHTorrent [205], a publicly available historical database of GitHub public activity traces, containing data for approximately 16M users. Gender is not recorded in GitHub profiles and, consequently, is also not available in GHTorrent. Therefore, we inferred it from people's names, as described in Section 3.4.2, and augmented the GHTorrent data. However, since social network analysis on a data set of GitHub's size would be computationally unfeasible, we first compiled a smaller sample of 58,091 users, as follows.

Preprocessing and Filtering Starting from the $\sim$ 16M users in GHTorrent, we filtered out organizational users (*i.e.*, metausers, not usually corresponding to a single person), users with deleted accounts, users who never authored any commits and users with names not containing any space (gender inference techniques rely on a person's first and last names; *e.g.*, Alice would be excluded, but Alice Smith and Alice Marie Smith would not). We acknowledge that some cultures do not split names into parts, or some people are known mononymously. We chose this conservative heuristic, which excludes some valid names, since we noticed during manual exploration of the data that many single-part names are English words or nicknames from which we cannot extract gender information. Approximately 1.8M GitHub users in our data had non-organizational, non-deleted accounts, authored at least one commit, and had names consisting of at least two parts.

Identity Merging Since git version control settings are set locally by each client, there are some cases where git commits are not attributed to the correct GitHub account, which introduces noise in the data. Moreover, the same contributor may have used different git "aliases" (*i.e.*, names and emails) in different projects or over time [206]. To have a more accurate representation of one's activity and contributions, we performed identity merging on the different (name, email) tuples in our data using a series of heuristics (cf. [206, 207, 208]).

Sampling After initial filtering and identity merging, we randomly sampled 300,000 users and applied our gender inference technique (Section 3.4.2) to label each account as Female

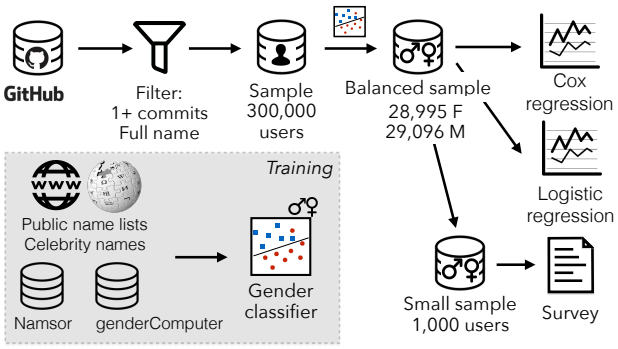


Figure 3.2: Overview of our methodology.

(9.7%), Male (84.85%), or Unknown (5.45%). Some of our social network analysis measures (Section 3.4.3) require, for every person, to collect all the repositories they contributed to, and for every repository, to collect all other contributors and all *their* repositories. To reduce computational effort and to address the Female–Male imbalance in our sample, we randomly down-sampled the group of male contributors to the same size as the female group. After removing users who have only contributed to educational projects, our final dataset contains 28,995 users labeled Female and 29,096 users labeled Male. Figure 3.2 gives an overview of our data collection process.

3.4.2 Gender Inference

Various approaches and tools for name-based gender inference have been proposed [209, 210]. All operate with the simplifying assumption that gender is binary; we also assume binary gender here to simplify data collection and analysis. We tried many of these tools and found that each has strengths and blind spots. In particular, most tools are based on databases of English names and as such fail, *e.g.*, on Asian names.

We have considered approaches that use social network data, specifically *Google+* [7], but the gender API has been deprecated; tools that can infer gender from photos, *e.g.*, *Face++*, but discarded these since GitHub profile photos are scarcely available; and tools that can infer gender from text [211], but discarded these since we have a very limited amount of text for each user – mostly commit messages, which are usually too short to provide enough information.

Instead, we identified two main contenders among tools that rely on broader datasets of names in different languages, and integrate them in a classifier (*i.e.*, a voting system). Our first contender is *genderComputer*[1] [212]. As opposed to other tools it uses location information to disambiguate; *e.g.*, it is able to distinguish between Italian Andrea (predominantly male) and

[1] https://github.com/tue-mdse/genderComputer

Table 3.1: Accuracy of the different gender inference methods (bolded are the highest accuracy for that language).

Language	genderComputer (%)	NamSor (%)	Our classifier (%)
Chinese	17.58	6.70	**60.00**
Japanese	76.76	26.88	**79.71**
Korean	18.82	13.51	**68.07**
All	79.41	74.07	**83.62**

German Andrea (predominantly female). Our second contender is *NamSor*[2] which classifies personal names by gender, country of origin, and ethnicity, with good coverage of different languages, countries, and regions. We trained and tested a Naive Bayes classifier that takes as input the gender predictions output by *genderComputer* and *NamSor* for a given name as well as features of the name itself, and produces a gender label as output, *i.e.*, one of Female, Male, or Unknown.

As training (80%) and test (20%) data, we compiled a list of 11,706 names from two sources. First, we randomly sampled 8,706 names from *genderComputer*'s open source dataset, which covers 28 countries. Second, since both input gender tools often have difficulty with East Asian names, we further collected a total of 3,000 romanized Chinese, Japanese, and Korean names from celebrity name lists on Wikipedia, websites for baby names, or name lists found in online public datasets, *e.g.*, lists of recent school graduates or of enrolment.

For each name, we obtained the gender inferences from *NamSor* and *genderComputer*. We also extracted features from the name itself, including the last character (*e.g.*, in Spanish, names ending in 'a' tend to be female), the last two characters (*e.g.*, in Japan, names ending in 'ko' tend to be female), and tri-grams and 4-grams to capture romanized Chinese, Japanese, and Korean names. We also included *NamSor*'s inference on the contributors' countries of origin from their last names as a feature. Using the country of national origin inferred from last names, instead of the country of residence declared on the GitHub profile, is an improvement on prior work, because it can increase the gender inference accuracy for people residing outside their (or their ancestors') country of origin, *e.g.*, Italian Andrea's living in the US. We note, however, that this approach can still fail in some cases, *e.g.*, for a person with a Chinese last name and a non-Chinese first name such as Andrea Zhang.

Table 3.1 reports the accuracy of the gender inference tools and our classifier overall as well as on names in East Asian languages, which are typically the hardest to make inferences on [210]. Overall, our combination classifier has higher accuracy on all categories of names than either *genderComputer* or *NamSor*. Our classifier fails mostly on gender neutral names, such as Robin and a Chinese name Yan that can be both male and female, depending on what Chinese character it is associated with. We also do not have enough training samples to make accurate inference from languages such as Burmese.

3.4.3 Operationalizations of Concepts

To model the effects of different dimensions of social capital on sustained participation on GitHub, our statistical modeling technique (survival analysis, Section 3.4.4) involves

[2] http://www.namsor.com

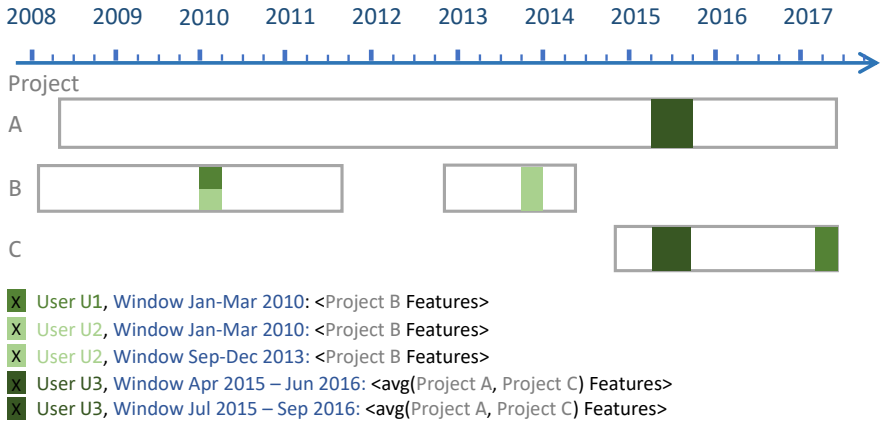


Figure 3.3: Illustration of data points we collect.

operationalizations of the different theoretical concepts discussed in Section 3.2. We introduce the following operationalizations.

Panel Data An implicit assumption for social capital effects to manifest is that project members had a chance to interact with each other. Since GitHub projects can be long-lived and since open-source projects in general face high turnover [162, 101], we assemble a longitudinal panel data set with measures computed over shorter time intervals; specifically, we aggregate all data from 2008 to 2016 into consecutive three-month windows, *i.e.*, we compute quarterly values for all measures.

Note that this involves two levels of aggregation. First, for every person and every project they contributed to, we compute quarterly values for different project-level measures (details below). Second, whenever someone contributed to more than one project in the same three-month window, thus having different sets of values for different projects in that window, we average out their project-level measures across their different projects that window; our results are qualitatively similar (significance and directionality of regression coefficients) if we compute the maximum instead of the average across projects. Figure 3.3 illustrates the structure of our data.

GitHub Disengagement—Outcome Variable The dependent variable in our model is the occurrence of the disengagement event: *i.e.*, if every commit a person authors is an indication of repeated engagement, we consider a person's last recorded commit as an indication of disengagement if "long enough" time has elapsed for potential subsequent commits to be observable. Naturally, programmers may take a break from GitHub and return later for more contributions. Moreover, one's last recorded commit may be very close to the end of the observation period, so it is not clear whether they will return to contribute more; this common phenomenon in longitudinal data is known as right censorship (the disengagement event did not happen during the course of study) [213].

We considered 12 months of inactivity as "long enough" to confidently detect disengagement, and used this operationalization in our survival models. Specifically, we consider that

a GitHub contributor has disengaged at time t if they have not committed anything to any open-source project for 12 months after t; *i.e.*, the *has_disengaged* value is 1 in the three-month window containing t, and 0 in all previous windows. Consequently, we also consider that people whose last recorded commit is less than 12 months prior to the end of our data are still active. Our models are robust to this operationalization and the results are qualitatively similar (significance and directionality of regression coefficients) with 6 months instead of 12. Note that we excluded 9,269 people with 12 months or more of inactivity that returned to make new contributions. Among them, 4,932 were male, 4,337 female.

Team Cohesion Measures $\mathbf{H_1}$ assumes that during their open-source tenure, the more often GitHub contributors participate in projects with high potential for building social capital, the higher their chance of prolonged engagement, *i.e.*, strongly connected networks and presence of ties between subgroups increase the likelihood of sustained participation in open-source. While subgroup or community detection has been extensively studied in the social network analysis literature [203], as argued by de Vaan *et al.* [214] these techniques are not suited for the operationalization of social capital constructs. Indeed, community detection techniques interpret ties as a static construct, while interpersonal relations, trust, and the implied social capital develop in time. Hence, to argue presence of a tie between two developers, the relationship between them should be durable, and this durability should be reflected in the operationalization. Therefore, as operationalizations for ties in team structures, we follow Lutter [177] and de Vaan *et al.* [214] and compute two distinct but related measures of social capital: interpersonal team familiarity and team recurring cohesion.

Team Familiarity We adapt Newman's [215] measure of average interpersonal familiarity within a team, which captures the intensity of prior collaborations between each pair of current team members; the measure of strength of a developer's social connection to a project by Casalnuovo *et al.* [176] is conceptually similar. Team familiarity is aggregated over pairs of contributors (dyads), and as such it is capable of capturing both ties within subgroups and between subgroups, corresponding to bonding and bridging social capital.

To calculate dyadic interpersonal familiarity for project p in time window t, we iterate over all time windows prior to t. Let i and j be two contributors to project p and let and r_{is} and r_{js} be the sets of projects they worked on in time window s, respectively. The familiarity between i and j at time t is defined as the number of projects they worked on together in past windows $s < t$, adjusted by the team size of each project at that time, assuming that people who work in a smaller team are more familiar with each other. Only collaborative projects ($|r_s| > 1$) are considered. Then, the values of each window s are summed to result in the interpersonal familiarity measure w_{ijt} defined as

$$\sum_{s=1}^{t-1} \sum_{r_s \in (r_{is} \cap r_{js}), |r_s|>1} \frac{1}{|r_s| - 1}$$

.

To measure team familiarity for project p in time window t, we define Team familiarity$_{pt}$ as the sum of w_{ijt} for all pairs of contributors i and j normalized by the number of pairs of contributors to p in time window t: The values range from 0 to 299.0.

Recurring Cohesion To capture tendencies for possible network closure from team cohesion, we again follow Lutter [177] and de Vaan et al [214] in calculating a measure of recurring

cohesion, which captures cliques of at least three people who have previously worked together. If three programmers have worked on some project before, and they later worked together again, the network containing this three-person clique can be considered more cohesive than that where any three people only share dyadic ties. A clique is defined as a group of people who at some time prior to current window t worked on a common project within a three-month window; to reduce the complexity of enumerating and checking all possible cliques of large teams, we only consider cliques of up to five members.

After identifying all q_p cliques for a project p at time t, we construct a $q_p \times q_p$ matrix $\mathbf{M}^p$, where each entry (v, w) contains the number of people shared by cliques v and w. Then we use all the off-diagonal, lower triangular values of $\mathbf{M}^p_{v,w}$ to calculate the recurring cohesion as:

$$\text{Recurring cohesion}_{pt} = \frac{1}{2(q_{pt}-1)} \sum_{v<w \wedge v,w \in p_t} \frac{|v| + |v \cap w|}{|p_t|}$$

If there are no cliques, this measure is assigned 0; if there is exactly one clique, say v, the measure is calculated as $\frac{|v|}{|p_t|}$. The values range between 0 and 1547.5.

Team Diversity Measures $\mathbf{H}_2$ tests whether attachment of women to open teams with regard to diversity of ties and information increases their chance of prolonged engagement relative to men's. To operationalize diversity of information we compute the share of newcomers and heterogeneity of programming language expertise. Indeed, the more newcomers are in a team, and the more diverse expertise team members have, the more diverse is information exchanged in the team.

Share of Newcomers Following Lutter [177] and Perretti and Negro [216], we calculated each team's share of newcomers, *i.e.*, the fraction of newcomers in a project in time window t relative to the size of the project team at time t. The more newcomers there are in a team, the more new ideas can be brought in, and the more new combinations of relationships can be formed. We operationalize newcomers at project level, *i.e.*, people who never contributed to a given project prior to time t.

Heterogeneity of Programming Language Expertise Prior work has shown that diverse knowledge is important to innovation and sustainable competitive advantage in many domains [217]. A similar effect may be visible in OSS teams, where assembling a diverse team with expertise in different programming languages or technologies may provide a competitive advantage, and may help create social connections between members that bridge communities and create opportunities.

Following Lutter's measure of genre diversity in the film industry, based on the distance measure of de Vaan *et al.* [214], we calculate a measure of programming language background heterogeneity at project team level, that considers each team member's prior experience with different programming languages from prior open-source GitHub projects. We begin with a list of the most popular 33 languages on GitHub [218]; all other languages in our data are labeled 'Other', generating a set of $K = 34$ languages. On GitHub each project is labeled with the predominant programming language used therein. Given a project p labeled with the predominant language k, we consider that all developers who contributed to p have experience with k: while individuals may vary in their experience with k, given the size of the dataset we expect a reduction to the mean in terms of individual knowledge; *i.e.*, we

expect that, on average, project contributors would have had experience in the predominant language.

For each contributor i in project p in the current time window t, we calculate the vector $f_i = (f_{i1}, ..., f_{iK})$ for each language k, where f_{ik} is 1 if i has worked in projects labeled with the predominant language k. Then, the programming language background distance d_{ijt} between two contributors i and j in the time window t is defined as the cosine of their respective experience vectors. Possible values for this measure range from 1, indicating complete similarity in the language histories of i and j, to 0, indicating complete dissimilarity. Future refinements to this measure, beyond the scope of the current paper, could also consider how similar different programming languages are with each other [219]. We then aggregate these similarity measures at project level, over all pairs of contributors i and j, $i > j$, adjusted for team size, and subtract the result from 1 to obtain a degree of dissimilarity:

$$\text{Language heterogeneity}_{pt} = 1 - \frac{1}{\binom{|p_t|}{2}} \sum_{i>j \wedge i,j \in p_t} d_{ijt},$$

Control Variables As control variables we consider:

Is Project Owner and *Is Project Major Contributor* both control for the contributor's position in the project. We define major contributors as those authored at least 5% of the project commits during a given window [220]. Being a repository owner or major contributor indicates higher levels of commitment, hence, we expect differences in disengagement rates.

Number of Followers and *Number of Repository Stars* both control for visibility of the contributors and projects, respectively [121]. Popular developers, or developers contributing to popular projects, tend to have a different experience on GitHub and may be less likely to disengage [26, 97].

Niche width, *i.e.*, the number of programming languages of the developer's past GitHub commits are spread across. We expect individuals knowing multiple languages to be more versatile and less likely to disengage.

3.4.4 Survival Analysis (Quantitative)

To test our hypotheses quantitatively, we use survival analysis, a statistical modeling technique that specializes in time to event data [213]. Survival analysis is particularly suitable for modeling right-censored data like ours.

Estimation We model jointly the effects of the different social capital factors in Section 3.4.3 on the time to the GitHub disengagement event, while controling for covariates. For each GitHub developer in our sample, we have a *survival time* T on record (number of quarters until *has_disengaged* becomes 1). The probability of reaching a given survival time t is given by the *survival function* $S(t) = P(T > t)$, and the probability of leaving the state at time t is given by the *hazard rate* $h(t) = \frac{P(T<t+\Delta t|T \geq t)}{\Delta t}$. The Cox model is a non-parametric regression which can estimate, using partial likelihood, the effect of some independent variables X on the hazard rate, $h(t, X) = \theta(t)f(X)$; *i.e.*, it can estimate the coefficients β of the regression $h(t, X) = \theta(t)\exp(\beta' X)$, where β' denotes the vector transpose of β [213]. The coefficients β can be directly interpreted, *e.g.*, if $\beta_i = 2$, then a unit increase in X_i decreases the probability of survival by $\exp(2) = 7.4$ times.

Many developers disengage early, in their first quarter. In open-source, occasional contributions [221] are common. To model how the different factors contribute to explaining the variability in disengagement rates differently early compared to later on, we split the data set into two parts: developers who disengage in the first quarter and the rest. Since the former only contribute one observation each (one quarter), we model this group using logistic regression (`glm` in R). For the remaining developers, the data set contains repeated quarterly observations. To model these, we estimate a Cox proportional-hazards model.

Diagnostics Whenever variables had highly skewed distributions, we removed the top 1% of values as potential high-leverage outliers, to increase model robustness [222]; we also log-transformed variables, as needed, to reduce heteroscedasticity [135]. We then tested for multicollinearity (and removed predictors, as needed) using the variance inflation factor (VIF), comparing to the recommended maximum of 5 [136]. Next we inspected the Schoenfeld residual plots [223] (graphical diagnostics) to test the assumption of constant hazard ratios over time. Finally, we report p-values for model coefficients as well as estimates of their effect sizes (fraction of variance explained) from ANOVA analyses.

3.4.5 Developer Survey (Qualitative)

To better understand how social capital might impact women and men on GitHub differently, we conducted a user survey.

Survey design The aim of the survey was to gain additional context information about how open source contributors perceive their respective projects and the way they collaborate in those project. The survey instrument thus focuses on contributors to collaborative open source GitHub projects (with at least three contributors, to exclude "toy" projects [128]). Respondents were instructed to choose such a project and base their answers on their experience therein.

We asked open ended questions focusing on their perceived responsibilities and (if applicable) reasons for them to stop contributing. Furthermore, we asked Likert scale questions covering individual satisfaction of contributors being part of this particular project [224], perceived work engagement [225], perceived social capital [226] (the principal construct of our study) and the frequency of communication using different means of communication. We opt to measure individual satisfaction since it has been repeatedly related to loyalty [227], and therefore more satisfied developers can be expected to be less likely to disengage; while work engagement has been shown to be related to turnover intentions [228]. We also aim to assess communication as additional context information about how open source contributors collaborate. For the first three scales we rely on existing instruments that we adapted for our context. In order to assess the frequency of communication we developed a scale that covers different potential means of communication such as *reading each other's code*, *text messaging*, *email* and others. This scale is divided into four levels ranging from *"never or hardly ever"* to *"every day or almost every day"*. The provided means of communication cover typical technologies, *e.g.*, text, audio/video messaging, and typical means of communication in OSS projects, *e.g.*, reading each other's code, commenting on existing code. We also included in person communication for co-located teams.

We also included multiple questions that focus on individual programming skills. The purpose of these questions is not only to assess the potential bandwidth of different skill levels. It can also be expected that differences related to skill level can have an impact on the social structure within a project. Similarly to the niche width in the repository data analysis, we asked participants to identify programming languages that they feel comfortable using. The list we used was based on the most commonly used programming languages in GitHub. We also asked contributors for how many years they have been active in OSS projects in general and how they rate their skills in comparison to their fellow project contributors. This question has been found to be mostly related to actual programming experience by Siegmund *et al.* [229]. The latter question is related to the tenure diversity shown to be a predictor for turnover in GitHub teams [58]. Finally we included typical demographic questions: the age and gender of the participants and their education level. Wang and Fesenmaier have shown that when keeping age and educational level constant, men have been members of an online community for a longer period of time [230]. The educational level was based on the Educational Attainment scale by the United States Census Bureau.

Procedure The population of interest for our study includes female and male contributors to open source GitHub projects with at least 3 members. We piloted the survey internally with 3 individuals and externally by contacting a total of 800 individuals (400 identified as female and 400 as male by the gender prediction algorithm). Based on the 43 responses we received (5.38% response rate), we revised the survey instrument. For the final survey, we sent 500 invitations to contributors identified by the gender prediction algorithm as women and 500 invitations to those identified as men. The delivery of 6 invitations failed. The survey was available for 2 weeks. We received 107 responses, for a response rate of 10.7%. Responses were anonymous and participation was voluntary. Out of the 107 survey responses received, 93 were complete. Out of the complete responses, 32 respondents identified as female, 56 as male, and 5 did not disclose their gender, which leaves 88 usable responses for the following analysis.

The average reported GitHub tenure of our survey respondents was 2.50 years, slightly less than what other studies found (*e.g.*, [197] found an average of 3.07 years). This difference could be explained by the larger share of female participants in our survey (36% as opposed to 25% in the survey by Vasilescu et al. [197]) and the fact that female participants in general report shorter tenures than male participants. The tenure of our survey participants is thus generally comparable to that of others in a similar setting. For open ended questions, we conducted an open coding procedure (one author, expert qualitative researcher). For perceived responsibilities we referred to the contributor types that can be found in the GitHub open source survey [195]. For potential reasons to discontinue contributing to an OSS project we reversed the motivations to contribute to open source [22]. The categories were iteratively refined.

Accuracy of gender prediction We found a strong correlation between the computed and reported gender. Out of the 107 responses we received, a total of 53 were responses to the survey that we sent to contributors that were identified by the algorithm as female and 54 were responses that were identified by the algorithm as male. Out of the 54 participants our algorithm identified as male, 52 identified themselves as male in the survey and 2 elected not to disclose their gender. Out of the 53 participants our algorithm identified as female,

37 identified themselves as female, 13 identified as male and 3 elected not to disclose their gender.

The algorithm was thus nearly perfect in terms of predicting whether or not a contributor indeed is of male gender (96.30%), as expected given that males are the majority group. The accuracy for predicting whether or not a contributor is of female gender was lower (69.81%) but still above chance. Our algorithm also did not classify female as male contributors: indeed, all participants that were classified as male either reported to be male or did not disclose their gender. This also suggests that the probability of the algorithm missing the contributions of women should be low, since it is capable of detecting male contributors with high accuracy (cf. [209] for discussion of the importance of not misclassifying women).

3.4.6 Replication Package

Our data collection and data analysis scripts, the survey instrument, and the input data for the regression models in Table 3.3, are part of a replication package.[3]

3.5 Results

3.5.1 Survey results

What responsibilities do survey respondents have? We asked participants about what they perceive to be their overall responsibilities in the project they selected. To analyze the answers we conducted an open coding procedure based on on the different contributor types in the GitHub open source survey.[4] While applying the contributor types to the survey responses we discovered additional codes ending up with nine distinct but not mutually exclusive responsibility categories.

While participants reported anything between no responsibilities at all and five different responsibilities, most participants reported either one or two. For both genders contributing code is by far the most common perceived responsibility (76.14%), with project management (30.68%) and project lead (22.73%) following at a distance. Male contributors mainly perceive themselves as leaders or managers (37.50% of males report those as their perceived responsibilities) while females appear to take over more non-code related activities such as documentation and proposing ideas (62.50% of females report those as their perceived responsibilities). While this observation concurs with the higher participation of males in the mailing lists related to designing technology [212], the difference is not statistically significant ($p = .869$ for non-code related activities).

How do survey respondents communicate? We analyzed whether and how respondents interact with each other based on different means of communication. We found that 10 out of 88 respondents never communicated with their fellow project members; Eight of those identified as male (9.09%) and two as female (2.27%). Most of our survey participants thus communicated via any of the provided means of communication.

[3] https://doi.org/10.5281/zenodo.2550931

[4] https://github.com/github/opensource-survey/blob/master/survey-instrument.md

Participants most commonly communicated via text messages, comments on code and reading each others code in general (almost half of respondents communicate in this way at least once or twice a week). Mail and in person communication are less popular (35.23% and 28.41%, respectively) followed by social networks (11.36%), video messaging (15.91%) and audio messaging (20.45%). Although there are no statistically significant differences between female and male contributors in terms of their communication behavior ($p = .979$), a closer look into the respective frequencies reveals that female contributors are slightly more active communicating with their fellow project members. This observation concurs with the results of Razavian and Lago: their study has shown that communication is seen by software architects as feminine expertise [231]. In particular, women use text and audio messages as well as social networks more frequently. Males on the other hand appear to use comments on code more frequently than females.

How experienced are the survey respondents? We also asked survey participants about their age, educational background and experience related to both programming in general and contributing to open source projects in particular.

The respondents were mostly between 18 and 34 years old (56.8%) and have a bachelor's or master's degree (67.0%). They reported feeling comfortable using between two and six of the proposed programming languages (77.3%; niche width). Comparing female and male contributors we found that male contributors reported a significantly higher number of programming languages they feel comfortable using ($F = 6.646$, $p < .05$, $\eta^2 = 0.072$). We also found males to report a significantly higher level of expertise ($F = 5.643$, $p < .05$, $\eta^2 = 0.062$). Both are medium effects as demonstrated by η^2 values [232]. There were however no significant differences between female and male contributors in terms of reported age, level of education and years of experience in open source projects. One explanation could be that female contributors are less confident about their programming expertise than male contributors, while neither their education level nor their experience in contributing to open source suggest a valid reason for this perceived difference. This would concur with Wang *et al.*'s finding on women's confidence-competence gap [233].

Why do people stop contributing to GitHub projects? Most of our survey participants are still active in open source (73.9%). Out of the 32 respondents who identified as female, 6 reported that they stopped contributing to open source, while 26 reported that they are still active. Among males, out of the 56 respondents, 17 reported that they stopped contributing while 39 reported that they are still active.

We then conducted a logistic regression analysis on the survey data, using data from the different scales, to model the factors that explain and predict disengagement (binary variable). The multi-item scales we used (individual satisfaction, perceived work engagement, and perceived social capital) are all reliable (Cronbach's α between 0.84 and 0.92). We built an explanatory model, including data from the three scales above, as well as programming experience and reported gender as independent variables. Results from this regression analysis (Table 3.2) showed that **perceived bridging social capital** and **years of programming experience** are significant predictors of individual disengagement. Both bridging social capital and years of experience are comparably strong predictors for individual disengagement (cf. deviance explained in Table 3.2). Gender had no significant direct influence on disengagement.

Table 3.2: Regression model for the user survey data ($N = 88$).

	GitHub disengagement response: *has_disengaged = 1*	
	exp(Coeffs) (Err.)	LR Chisq
(Intercept)	14.41 (2.55)	
Individual satisfaction (Avg)	2.23 (0.52)	2.95
Work engagement (Avg)	2.00 (0.38)	3.97*
Bridging social capital (Avg)	0.22 (0.60)*	8.37**
Bonding social capital (Avg)	0.61 (0.34)	2.18
Experience relative to team	0.74 (0.31)	0.91
Years of experience	0.72 (0.14)*	6.87**
Education	0.77 (0.24)	1.27
Self-reported gender	2.83 (0.69)	2.44
Niche width	0.96 (0.17)	0.06

$^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$

When looking into self-reported reasons for discontinuing to participate in a GitHub open source project, we found two main reasons: (1) not having enough time to contribute anymore; and (2) no immediate personal need for the respective project. Lack of time was reported to be caused by work related (*"changes in job"*, *"work became over bearing"*) as well as personal reasons (*"diversifying hobbies"*, *"personal life"*). Lack of time was also identified by Lee *et al.* as the most common barrier to participation faced by one-time-contributors to FLOSS projects [234]. Other reported reasons were *"the end of funding of our project"*, frustration (*"failure of our team of backend and front-end"*) or the perception that *"the project [...] is finished"*. When comparing reasons to disengage we found female contributors to report personal reasons significantly more often ($F = 4.87$, $p < .05$, $\eta^2 = 0.188$). This is a large effect, concurring with the higher likelihood of women leaving and reentering the labor force for personal reasons [235].

3.5.2 Survival analysis results

Who are the GitHub data developers? Out of 58,091 programmers, 39,643 have taken a break longer than half a year, and 25,196 programmers have taken a break longer than 1 year.

The average age of an account (number of months since the first commit) is 15.01 months; women are statistically younger than men ($p < 2.2^{-16}$, Cliff's $\delta = 0.23$) these results concur with our survey and earlier observations [236, 197]. On average, a programmer contributes to 9.55 projects (median = 4); statistically, women contribute to fewer projects than men ($p < 2.2^{-16}$, Cliff's $\delta = 0.16$). The effect size is in both cases are small (< 0.33) [237].

How does social capital associate with disengagement? Figure 3.1 plots the Kaplan-Meier estimates revealing that contributors are most likely to drop out in the first two years,

Table 3.3: Regression models for early-stage disengagement ($N = 29,235$ users; 140,441 data rows) and later-stage disengagement ($N = 26,299$ users; 143,984 data rows).

	Early-stage (GLM) response: *Disengaged = 1*		**Later-stage (Cox)** response: *Disengaged = 1*	
	Coeffs (Err.)	LR Chisq	Coeffs (Err.)	LR Chisq
(Intercept)	1.61 (0.07)***			
Followers	0.61 (0.02)***	990.53***	0.70 (0.02)***	394.39***
Stars	0.89 (0.02)***	45.18***	0.86 (0.02)***	103.26***
Commits to date	0.63 (0.01)***	1635.38***	0.64 (0.02)***	718.15***
Is major contrib.	0.77 (0.05)***	29.05***	0.63 (0.06)***	62.96***
Is repo owner	0.56 (0.03)***	363.80***	0.51 (0.04)***	310.35***
Niche width	0.47 (0.05)***	244.20***	0.54 (0.05)***	132.70***
Is female	1.27 (0.03)***	68.79***	1.32 (0.04)***	59.96***
Team familiarity	0.84 (0.08)*	4.83*	0.79 (0.09)**	13.22***
Rec. cohesion	0.85 (0.04)***	30.77***	0.86 (0.04)***	28.46***
Share newcomers	1.07 (0.04)	3.37	0.78 (0.04)***	35.70***
Lang. heterogen.	0.70 (0.11)**	44.44***	0.63 (0.14)***	44.43***
Lang. heter.:Female	0.73 (0.15)*	4.36*	0.69 (0.18)*	4.30*
Female:Team fam.	1.09 (0.11)		1.05 (0.17)	
Female:Cohesion	1.02 (0.05)		1.01 (0.04)	

$^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$

and women are more likely to drop out than men in general. Table 3.3 presents summaries of our regression models: a logistic regression for contributors who disengage within their first three months of activity (left), and a Cox regression for contributors who disengage later (right).

In both models the control variables behave as expected. More popular (*i.e.*, followers), active (*i.e.*, commits to date) and versatile (*i.e.*, niche width) developers are less likely to disengage. Similarly, project owners, major contributors and contributors to highly starred projects are less likely to disengage. Moreover, as expected, female contributors are at higher risk of disengagement than males: in the short term, being female increases the odds of disengagement from GitHub by 27%; in the long term, by 32%.

The two variables related to team cohesion have statistically significant effects, and these effects are consistent between the two models. Contributing to projects where team members are more familiar pairwise with each other from prior collaborations (Team familiarity), or projects where cliques of three or more developers recur from prior projects (Recurring cohesion), is associated with decreased risk of disengagement.

The variables related to team diversity also have statistically significant effects. Heterogeneity in the programming language backgrounds of project team members is associated with decreased risk of disengagement both short and long term. Moreover, language heterogeneity has a statistically significant interaction with gender: women are more likely to disengage when language heterogeneity is low. Contributing to projects with high turnover (Share of newcomers) is associated with higher risk of disengagement after the first three months.

3.6 Discussion

3.6.1 Hypotheses

$\mathbf{H}_1$ linked social capital to the duration of engagement of OSS developers. Both aspects related to bonding social capital, such as the need to reciprocate, and those related to bridging social capital, such as exposure to new technologies and ideas can be related to developers' motivation. Therefore, $\mathbf{H}_1$ stated that the more often people participate in projects with high potential for building social capital, the higher their chance of prolonged engagement. Our study **strongly supports** this hypothesis. Both regression models (Tables 3.2 and 3.3) indicate that social capital, measured by an established survey measurement instrument [226] and by team familiarity and recurring cohesion metrics respectively, is a statistically significant predictor for disengagement. The regression coefficients are lower than one, meaning that *the increase in social capital decreases the chance of disengagement*, other variables held fixed.

$\mathbf{H}_2$ stated that attachment of women to open teams with regard to diversity of ties and information increases their chance of prolonged engagement relative to men's. Table 3.3 shows that $\mathbf{H}_2$ is **partially supported**. On the one hand, we found evidence that attachment of women to open teams with regard to diversity of information (language heterogeneity) increases their chance of prolonged engagement: language heterogeneity interacts with gender. On the other hand, no such interaction could be found for diversity of ties (recurring cohesion and team familiarity), therefore we conclude the support is only partial.

3.6.2 Implications

Our results provide empirical evidence that social capital impacts the prolonged engagement of contributors to open-source. Hence, researchers can consider social capital as a lens to investigate social phenomena in OSS.

Given the importance of and concerns about the sustainability of OSS [238, 10], our results suggest that social coding environments like GitHub should be redesigned to support women in developing social capital, on the one hand, and project maintainers in tracking and being able to react to factors that negatively impact the formation of social capital, on the other hand. We envision: 1) better search functionality and recommendation engines for newcomers looking for projects to join, that take the target project team cohesion and expertise diversity explicitly into account when making a recommendation, to facilitate the formation of social capital, in particular for women (cf. [178]); 2) stemming from the previous point, better mentorship support for newcomers in general and women in particular, whereby mentors can be automatically recommended to potential mentees to facilitate the formation of social capital (cf. [8]); and 3) UI elements besides the ones currently available on GitHub repository pages, such as badges [239], that allow project maintainers to track worrisome trends in factors negatively associated with the development of social capital (*e.g.*, team expertise diversity and turnover).

3.6.3 Threats to Validity

Like any empirical study, our work is subject to threats to validity. First, our results depend on the data collected by GHTorrent, which may not be a full replica of GitHub data [128]. We carefully cleaned and filtered our data to avoid the GitHub mining "perils" [128]. The project-level metrics are calculated based both on the contributors' own forks and their base repositories (the repository to which they make pull requests). We also focus on commits instead of pull requests because only a fraction of projects use pull requests [128]. We repeatedly manually checked data outliers *e.g.*, large repositories that are not software projects, but tutorials. We excluded projects with large number of zero-commit forks and repositories with huge numbers of forks and commits (top 1%).

A second threat to validity may come from our gender classifier. The accuracy of the classifier is limited by the information users display on GitHub. Many users do not use their real names so we cannot extract their gender information reliably [195]. Some users display names in a language for which our gender classifier does not have data. Moreover, there are many top female developers from East Asia [233]. It is difficult to verify their gender identity because their names are gender neutral and their profile pictures are not necessarily their own photos. Furthermore, our gender classifier, as any automatic classifier we are aware of, is based on the assumption of binary gender, and as such our work cannot explicitly take into account contributions by non-binary software developers.

Third, we used a single coder for the open ended survey questions which might result in a subjective interpretation of the responses. We attempted to mitigate this threat by building on established categories.

Finally, statistical modeling required many operational decisions (*e.g.*, time windows, length of inactivity): ours follow best practices and prior work. Again following best practices, we tested sensitivity of our operational decisions. Given space restrictions, we prioritized replicability and validity, reporting all decisions made, but in cases of insensitive parameters did not always discuss the rationale for a specific value.

3.7 Conclusions

In this paper we have studied the impact of social capital on sustained participation of open source contributors and, in particular, on gender differences in this impact. We have performed a mixed-methods empirical study combining survival analysis on a longitudinal data set of 58,091 open source contributors and their GitHub contributions, with a survey of 98 developers. Our studies show that in general social capital positively affects sustained participation in open source on GitHub. For women, diversity of the project members' expertise becomes crucial to sustain their participation: we found that higher team diversity along dimensions of programming language expertise is associated with decreased risk of disengagement both short and long term.

Our secondary contribution is the very first gender inference tool explicitly targeting Chinese, Japanese, and Korean names, achieving 83.62% accuracy overall, and at least 60.00% on (South) East Asian names. This opens multiple directions of further research from

replication of earlier gender studies [62, 212, 58, 7] for East Asian contributors to exploration of new datasets such as Stack Overflow in Japanese.[5]

In the same way as we have studied the impact of language heterogeneity on the disengagement of women, future work should also consider the impact of gender diversity and gender homophily, i.e., preference of people to interact more with people of the same gender, of the teams on the disengagement of women [58, 67]. Furthermore, our study can be replicated to investigate the relation between social capital and sustained participation on other platforms, *e.g.*, Stack Overflow, and the impact of different demographic aspects.Finally, understanding the relation between social capital and sustained participation on GitHub is the key to designing appropriate interventions aiming at ensuring engagement of women in open source software projects more broadly.

[5] https://ja.stackoverflow.com/

Chapter 4

Detecting Interpersonal Conflicts

Interpersonal conflict in code review, such as toxic language or an unnecessary pushback, is associated with negative outcomes such as contributors' disengagement from OSS. Automatic detection is one approach to prevent and mitigate interpersonal conflict. Two recent automatic detection approaches were developed in different settings: a toxicity detector using text analytics for open source issue discussions and a pushback detector using logs-based metrics for corporate code reviews. This chapter tests how the toxicity detector and the pushback detector can be generalized beyond their respective contexts and discussion types, and how the combination of the two can help improve interpersonal conflict detection. The results reveal connections between the two concepts and signals that can reflect potential interpersonal conflict.

4.1 Introduction

In online communities and offline workplaces alike, interpersonal conflicts, understood broadly as including hostility, hate, aggression, toxic language, bullying, etc, has been a major concern and topic of research [240, 241, 242]. The consensus is not only that such forms of interaction are antisocial, but also that they are all associated with negative outcomes in the communities or groups where they take place, including decreased well-being, job satisfaction, stress, and turnover [44, 243, 46]. In addition, these outcomes tend to disproportionately affect members of underrepresented groups [244, 245, 246].

In software engineering, the problem of interpersonal conflicts is also well recognized. For example, in software development, some communities and maintainers have a reputation for being toxic [247, 248, 249]. Although relatively milder, impolite language is seen as a barrier to newcomers [31, 88]. There are repeated anecdotes of sexist behavior, harassment, or contributors concealing their identity to avoid abuse [5, 12, 195, 133, 51]. More broadly, evidence is also starting to emerge about anger [154], negative emotions [250], impoliteness [251, 252], pushback [47], or directly toxicity in issue discussions [253, 46, 254, 255], code reviews [256], and Gitter developer communication [257]. The programming-related Q&A platform Stack Overflow is also notorious for being 'toxic' [258].

However, despite comparable agreement about the importance of the problem, there is relatively less progress in software engineering compared to other domains in terms of automatic detection for prevention or mitigation [259, 260]. Several factors contribute to this

lag, including inherent difficulty of the problem, but also domain specificity of some toxic interactions and scarcity of labeled data.

Prior research on automatic detection of toxicity and related constructs in software engineering has room for improvement. In particular, we note that approaches published previously in the software engineering literature have generally all been based on textual analytics [46, 261]. For example, Ramen *et al.* [46] experimented with different sets of features, all text-based, to train a classifier to detect open-source software (OSS) toxic issue discussions, which is defined as "rude, disrespectful, or unreasonable comment[s] that [are] likely to make someone leave a discussion" – a definition of toxicity used also in other public discussion forums such as Wikipedia or the New York Times, originating from Google's project Jigsaw [262]. However, follow-up work by Sarker *et al.* [257] showed that Raman *et al.* 's approach has limited generalizability.

Meanwhile, researchers have long been arguing that meta-infor-mation can be very useful to refine inconclusive classification [263]. For example, people with a history of hate speech are more likely to engage in such behavior again than people without any history [264]. In software engineering, Egelman *et al.* [47] showed that using only meta-information can detect pushback, defined as "the perception of unnecessary interpersonal conflict in code review while a reviewer is blocking a change request."

Notably, the two concepts – 'toxicity' as operationalized by Ramen *et al.* [46] and 'pushback' as operationalized by Egelman *et al.* [47] – are similar, but distinct. For instance, while some types of Egelman *et al.*'s pushback could be considered toxic (*e.g.*, personal attacks), others would not (*e.g.*, persistent nitpicking). Moreover, the types of software discussions analyzed and the study settings in the two studies are arguably very different — Egelman *et al.*'s classifier was applied only on code reviews internally at Google and Raman *et al.*'s classifier was applied only on public GitHub issues (not code reviews). Despite these difference, it seems possible that these two approaches could inform one another as a way to improve detection of interpersonal conflict.

In this paper, we contribute: (1) a comparison of how toxicity and pushback manifest in open source and in a company, and (2) a systematic evaluation of our ability to predict toxicity and pushback in different settings and using different approaches. To this end, we use existing and new labeled datasets that capture both concepts in open-source and corporate code reviews. We use 10-fold cross-validation to evaluate and compare the two previous classifiers and also develop a new combined classifier using features from both. Our results provide insights on how these classifiers work in different contexts. The comparisons and discussion also shed light on the relationship between the two concepts, toxicity and pushback, and the two settings, open source and corporate.

By improving the accuracy of automated approaches to detect toxicity, pushback, and possibly other forms of interpersonal conflict in software discussions, this research paves the way for designing tools to prevent, mitigate, and further study these phenomena, including designing interventions to offer just-in-time guidance to developers in such situations. A detector can also be a powerful tool for researchers studying the effectiveness of tool design and other interventions. More generally, this research offers an opportunity to apply a technique to both open and closed source software, possibly benefiting from synergies, a rarity in software engineering research, in our experience.

4.2 Related Work

This paper builds directly on two recent approaches to detecting interpersonal conflict in software engineering artifacts, by Egelman *et al.* [47] and Raman *et al.* [46]. In Egelman *et al.*'s study at Google, the authors conducted interviews to develop the concept of *pushback* and designed logs-based metrics to detect pushback in code reviews. These metrics were rounds of a review, active reviewing time, and active shepherding time. Their logistic regression model obtained high recall (93%–100%) and low precision (6%–11%).

The other approach that this paper builds directly on is that of Ramen *et al.* [46]. The authors manually annotated toxic issue threads from projects on the GitHub platform, and experimented with outputs from different sets of generic text-based classifiers to train a new classifier to detect toxic issue discussions specific for open source. They reported the highest 10-fold cross-validation accuracy when combining Stanford's Politeness Detector [155] with Google's Perspective API.[1] The present paper expands on Raman *et al.* 's text-based features, compares them with Egelman *et al.*'s classifier [47], and experiments with combining the two classifiers.

In addition to the pretrained general-purpose linguistic tools used by Raman *et al.*, we also explore other linguistic techniques to detect interpersonal conflict. Vocabulary-based approaches have been used for text classification. Open-vocabulary analysis extracts features from the text being analyzed using statistical methods [265]. For example, Sood *et al.* [266] showed that an SVM classifier using binary presence and frequency of n-grams as features can be used to predict personal insults on social news sites. Monroe *et al.* [267] showed that the log odds-ratio of an n-gram (the frequency of being in one group of text divided by 1 minus the frequency) in two different groups can be used to identify n-grams that are over-represented in one group relative to the other. We build on Monroe *et al.*'s work in Section 4.5 by attempting to find out if there is a set of vocabulary that can distinguish between the positive labels (toxic or pushback) and the negative labels (non-toxic or non-pushback).

Closed-vocabulary analysis relies on predefined lists of words as features. Building on the classic linguistic theory of politeness by Brown and Levinson [268], Danescu-Niculescu-Mizil *et al.* [155] developed a computational parser for politeness strategies. Politeness theory divides politeness strategies into *positive politeness* and *negative politeness*. Positive politeness strategies encourage social connection and rapport, such as gratitude, optimistic sentiment, solidarity, etc. Negative politeness strategies try to minimize the imposition on the hearer, for example, by being indirect or apologizing for the imposition [269, 270, 268]. On the other hand, impolite behaviors can be direct questions (*e.g.*, "why?") or sentences that start with second-person pronouns, which may sound forceful. Prior studies showed that the politeness strategy parser [271] is able to predict if a conversation may turn awry [272] and can generalize well to various contexts. We build on this work by using politeness strategy features in our classifiers.

Finally, in the software engineering community, sentiment analysis [273] is a popular technique for analyzing issue discussions [154], pull request comments [274], and forum discussions [275]. Prior work has shown that sentiment analysis classifiers need to be trained using software engineering data because many traditionally negative phrases may have

[1] https://perspectiveapi.com/

neutral sentiment in the context of software engineering [157], for example, "execute" (for a survey see Zhang *et al.* [276]). Popular software engineering sentiment analysis tools include Senti4SD [275] and SentiCR [277]. Senti4SD, developed by Calefato *et al.* [275], is trained on 4,000 posts extracted from Stack Overflow. This dataset is part of the Collab Emotion Mining Toolkit [278]. SentiCR [277] is trained on 1,600 manually labeled code review comments. In our study, we build on this work by using sentiment analysis developed for code reviews as a feature in our classifiers.

4.3 Research Questions

Our overarching goal is to bridge the gap between the existing literature on toxicity [46] and pushback [47] in software development. Besides the two concepts themselves, there are three fundamental differences between the prior work studies in this area, which we systematically explore in this paper: (1) the context (open- *vs.* closed-source), (2) the type of discussion (issues *vs.* code review), and (3) the approach to classify (text-based *vs.* logs-based). Overall, we answer the following research questions and sub-questions:

First, we explore how well the two classifiers generalize beyond the respective settings in which they have been developed, while maintaining their specific target concepts (toxicity and pushback) and fundamental approaches to classification (text- and logs-based):

RQ$_2$. *How well do existing classifiers generalize across context and type of discussion?*

To answer this question, for each classifier we explore one additional setting beyond the one in which they have been developed. For the toxicity classifier [46], we experiment with open source code reviews in addition to the original issue discussions. Similarly, for the pushback classifier [47], we experiment with comments on open-source pull requests, the approximate equivalent of the original Google code reviews:

RQ$_{1.1}$. *How well does a text-based toxicity classifier designed for open-source issues perform when classifying toxicity in open-source pull requests?*

RQ$_{1.2}$. *How well does a logs-based pushback classifier designed for corporate code reviews perform when detecting pushback in open-source code reviews?*

Second, given the theoretical overlap between the concepts of pushback and toxicity, we explore how well the two fundamental approaches to classification, text-based (toxicity) and logs-based (pushback) generalize to detecting the other concept if appropriately trained on relevant data for that other concept:

RQ$_3$. *How well do existing classifiers generalize for both toxicity and pushback?*

RQ$_{2.1}$. *How well does a text-based (toxicity) classifier perform when classifying pushback, in both open and closed-source code reviews?*

RQ$_{2.2}$. *How well does a logs-based (pushback) classifier perform when classifying toxicity in open-source code reviews and issue discussions?*

Table 4.1: The relationship between our four datasets and their corresponding RQs.

	Classifiers			Num
	Text-based	Logs-based	Combined	
D1 Toxic Open-Source Issue Comments	Raman et al. [46]	RQ2.2	RQ3.1	80 to
D2 Toxic Open-Source Code Review Comments	RQ1.1	RQ2.2	RQ3.1	102 t
D3 Pushback in Corporate Code Review	RQ2.1	Egelman et al. [47]	RQ3.2	493 p
D4 Pushback in Open-Source Code Review	RQ2.1	RQ1.2	RQ3.2	201 p

Finally, we explore to what extent using design insights from one classification approach can be used to improve on the other:

RQ$_4$. *To what degree can combining existing approaches improve detection of toxicity and pushback?*

RQ$_{3.1}$. *How well can a combined text- and logs-based classifier classify toxicity?*

RQ$_{3.2}$. *How well can a combined text- and logs-based classifier classify pushback?*

For completeness, in addition to answering these questions, we also replicate the original experiments on toxicity in open source issues [46] and pushback in Google code reviews [47].

4.4 Datasets

To answer our research questions, we used a mix of existing (whenever possible) and new datasets on toxicity and pushback. First, we used the two existing data sets from prior work on issue toxicity in open source [46] and code review pushback at Google [47]. Additionally, we created two new datasets on code review toxicity in open source and code review pushback in open source. Table 4.1 displays each of these four datasets as a row, labeled D1-D4, summarizes how each of our research questions and the prior work relates to each data set, and describes the size of the datasets.

4.4.1 Design Decisions and Tradeoffs

Before describing each dataset in detail, we note several important high-level design decisions, assumptions, and tradeoffs we had to make when creating the two new datasets, and in order to meaningfully compare results across all four datasets.

Unit of labeling In the original toxic issue comments dataset by Raman *et al.* [46], ground truth labels are available for individual comments and the issue thread-level toxicity labels are an aggregation of comment-level labels, *i.e.*, if there is at least one comment labeled as toxic, the entire discussion is labeled as toxic. In contrast, the pushback code review dataset by Egelman *et al.* [47] contains only thread-level labels. Since we are reusing these datasets without relabeling, we maintain the same unit of labeling also in the two newly created datasets of the same concept.

Unit of classification Our experiments focus on classifying toxic or pushback entities at the thread level, because the logs-based metrics, such as the rounds of review, used by Egelman *et al.* are not applicable for individual comments. However, because the text-based classifier works at the comment level, for pushback datasets where we only have thread-level labels, we had to assign each comment the same label as the thread-level label. We will discuss the limitation when we present the results.

The notion of code review Our two new code review toxicity and pushback datasets are extracted from open-source projects on the GitHub platform whereas Egelman *et al.* 's dataset [47] was extracted from internal Google code reviews. In addition to the differences between the corporate and open-source contexts in terms of culture, process, and their observed consequences, the mechanics of code reviewing also differ. Google uses a proprietary dedicated code review management system [279] where all review comments are associated with specific code changes. On GitHub, projects typically manage code reviews as part of pull request threads. However, even though canonically code review comments on GitHub are expected to be attached to specific lines of code and can therefore be distinguished from more general discussion comments part of the same pull request thread, practices vary widely across projects [280]. For reasons of uniformity across projects when sampling candidates for manual labeling, and since we expect that indicators of pushback may occur across pull requests as a whole, not just review comments attached to specific changed lines, we consider the conceptual equivalent of a Google code review to be an entire GitHub pull request thread, including all its general and line-specific comments, *i.e.*, an "open-source code review thread" hereafter.

Representativeness When sampling toxicity and pushback pull request candidates for manual labeling, we use several heuristics to narrow down the search space (details below) instead of random sampling. While this compromises the statistical representativeness of our datasets, it is necessary to do this since the two phenomena we study are relatively rare; random sampling is unlikely to discover many, if any, instances of these phenomena. We note that this is not only a limitation of the two prior work studies we build on, but also of all similar work on hate speech detection *etc.* [281]. Alternative approaches to building labeled datasets for hate speech detection are, as of 2021, still actively being researched [281].

Open source vs corporate metrics While we try to replicate Egelman *et al.* 's pushback detection method, some measures are unfortunately not observable on GitHub. For example, we cannot replicate "shepherding time," which in Egelman *et al.* 's study is the total amount of time an author spent actively viewing, responding to reviewer comments, or working on the selected code change, including looking up APIs or documentation. The public GitHub trace data about pull request threads captures only wall clock times, which is an overapproximation of the active shepherding time. We are particularly interested in evaluating how well such approximation metrics, that are less precise but more widely available outside of a corporate setting, can capture the same phenomena.

4.4.2 Toxic OSS Issues (D1; pre-existing)

This dataset, originally created by Raman *et al.* [46], consists of 80 GitHub issue discussions labeled as toxic by the authors. Starting from the GHTorrent database [84], Raman *et al.* [46]

identified potentially toxic issue comments using the keyword "attitude" (the authors of the toxic comments are often criticized in the same thread by others, typically the project maintainers, about their attitude), and from issue threads "locked as too heated"—one of the mitigation strategies afforded by the GitHub platform. Raman *et al.* then manually reviewed a sample of candidate issue threads from this initial list and assigned ground truth toxicity labels.

We decided to replace the control group in Raman *et al.*'s dataset [46] because we noticed that those non-toxic comments' total number of characters is significantly shorter than for the toxic comments. Since a priori we have no reason to expect that toxic issues are generally longer than non-toxic issues, and we want to capture other aspects of toxic comments, we compiled a new set of non-toxic issues. Inspired by Egelman *et al.* [47], we constructed stratified samples by propensity score matching on the length of all comments within an issue thread (which is not used in any of our prediction models), after excluding code segments and comments from obvious bots [252], *e.g.*, a continuous integration tool. Our new set of non-toxic issues contains two non-toxic issues for every toxic issue.

4.4.3 Toxic OSS Code Review (D2; novel)

We compiled a dataset of 102 toxic open-source code review threads (*i.e.*, pull request threads with all their associated comments) and a separate corresponding control group of non-toxic open-source code review threads, using a similar approach to the one originally taken by Raman *et al.* [46] for issues. Specifically, we use three heuristics to narrow down the search space for candidates in the GHTorrent [84] database, followed by manual review and labeling. Egelman *et al.* [47] showed in their study of pushback that inter-rater agreement is very high when using multiple annotators, implying that a single annotator is sufficient. One author of the paper carried out the labeling independently, assigning "toxic" and "non-toxic" labels to the threads as a whole if at least one of the comments was considered to be toxic; when in doubt, we discussed the respective examples as a group and assigned labels collectively.

The three heuristics were:

- Locked as "too heated"—this built-in GitHub mitigation mechanism is available for both issues and pull requests; or
- Containing the keyword "attitude"; or
- Containing "code of conduct", a novel addition relative to Raman *et al.* 's heuristics [46]. We anecdotally observed that a project's code of conduct, when present, is invoked by maintainers when responding to a toxic comment.

Then, as in dataset D1, we performed propensity score matching on the total length of comments to assemble a control group containing two non-toxic open source code reviews for every toxic one.

4.4.4 Pushback in Corporate Code Review (D3; pre-existing)

We used the collection of code reviews gathered by Egelman *et al.* [47] from Google's internal corporate repository. The authors collected these using two methods:

First, Egelman *et al.* [47] pulled a stratified random sample of code reviews, then surveyed authors, reviewers, and other engineers about whether they perceived each code review as

having elements of "pushback." The authors then labeled a code review as containing pushback if at least one respondent noted that the review contained at least one element of pushback. Code reviews are labeled as not containing pushback if (a) at least one person responded to a survey about it, and (b) all survey responses about that code review indicated that no elements of pushback were present.

Second, those same respondents could report a code review that they thought contained pushback. They labeled these reported code reviews as "containing pushback", except that we discarded those that participants indicated were problematic only because of excessive review delays, which are not part of Egelman *et al.* ' definition of pushback [47].

4.4.5 Pushback in OSS Code Review (D4; novel)

To construct an open source counterpart to the corporate code review pushback dataset, we replicated the survey instrument used by Egelman *et al.* [47], with only surface-level modifications to adapt to pull requests and their specifics on the GitHub platform instead of Google-specific terminology.

We then compiled a sample of GitHub code reviews that each:

- had at least 10 comments, to ensure that at least some amount of interpersonal interaction was present, and
- had no more than 50 comments, to limit the reading effort expected from survey respondents.

Additionally, to ensure some diversity in code review outcomes, half of the sampled code reviews were merged pull requests and half were closed without being merged. We emailed survey invitations to the authors and reviewers who display their emails publicly.

As with Egelman *et al.* 's survey [47], we also asked participants to report other code reviews that they thought contained pushback; 63 were reported this way. The reasons that these code reviews were reported as pushback are shown in Figure 4.7 in Appendix. We then labeled these discussions using the survey data in the same way as in Dataset D3. As a result, this dataset contains only conversation-level labels.

Since one can maximize the recall of a classifier by predicting all data points as positive, the minimum precision score is the percentage of positive data points. Therefore, to make D3 and D4 more fairly comparable, we downsampled D4's negative data points to match the positive-negative ration in D3. In the end, D4 contains 201 pushback threads and 323 non-pushback threads.

4.5 Exploratory Analysis

As a first step, before applying machine learning, we explored how well a more basic word-frequency approach could distinguish discussions with one label compared to the other (*e.g.*, toxic *vs.* non-toxic) in each of the four datasets. To this end, we used an open-vocabulary analysis [267] to automatically identify words and phrases that are used distinctly more often in one label than the other, and then manually reviewed these looking for themes. This analysis serves two purposes. First, it helps to triangulate that the manually assigned labels are meaningful, if "obvious" differences between the two classes are detectable using

this independent approach. Second, it informs the design of more sophisticated automated classification, by identifying promising features.

Concretely, for the automated part we used log odds-ratio with Dirichlet prior [267] to identify n-grams that are significantly overrepresented in positive labels, *i.e.*, those labeled as toxic or pushback, compared to the negative labels, *i.e.*, those labeled as non-toxic or non-pushback. Since our data sets do not have an equal volume of text, we measured frequency using the log of an n-gram's odds-ratio. Because some n-grams may appear only in one label and not the other, we added a smoothing Dirichlet to the vocabulary. We pre-processed the text by removing URLs and numbers. We did not remove stopwords before performing the analysis because removing them can interrupt sentences and potentially eliminate some meaningful n-grams. We then ranked n-grams by z-scores and kept those with absolute z-scores above 2.326, which corresponds to the statistical significance cutoff of $p < 0.01$. Finally, we kept the 10 unigrams, bigrams, and n-grams with the highest positive z-scores (from positive labels) and 10 with the lowest negative z-scores (from negative labels).

We then manually examined the usage of these n-grams in our data sets by sampling comments containing them. We looked for patterns in these comments that could help us distinguish toxic or pushback comments from non-toxic or non-pushback ones, respectively. That is, we applied this process to all four of our datasets

To illustrate the results of this exploratory analysis, consider the results for dataset D2 Toxic Open-Source Code Review Comments in Table 4.2. In the table, empty cells indicate that no more n-grams were above or below the z-score cutoff. Due to space constraints, the tables (Tables 4.3, 4.4, and 4.5) for the remaining three data sets are shown in Appendix. Below, we describe several patterns that we observed from this analysis.

Second Person Pronouns One clear pattern we can observe from the word frequencies is the use of the second person pronoun "you" in toxic text, including phrases like "you are", "if you want". "You" is the unigram with the highest z-score in both D1 (Table 4.3) and D2. In Table 4.2, n-grams with second-person pronouns are in bold.

To investigate further, from D1 and D2 we randomly sample 10 toxic comments and 10 non-toxic comments that include "you." Some of these comments involve direct attacks on the second person recipient, such as "[y]ou don't care to be a part of the project," "[y]ou are expected to comply," "[y]ou decided to insult [...]." This echoes what Danescu-Niculescu-Mizil *et al.* [155] found: the use of second-person pronouns at the beginning of a sentence is more likely to be impolite. The same pattern is observed In D3 (Table 4.4).

In non-toxic comments in D2, the only n-gram that contains "you" is "could you", which is a negative politeness strategy that tries to minimize the imposition on the hearer by being indirect. The counterfactual form "could" is more polite than the future-oriented variant "can" [155]. This is also true in D3, where we again see some hedge words and other politeness strategies, such as "could you", "should be", and "seems" among non-pushback code reviews.

Gratitude Gratitude is another common theme in non-pushback text, both in open and closed source code reviews (D3 and D4 (Table 4.5)). These n-grams included "thanks" and "thanks for" that appear among non-pushback code review comments.

Technical Discussion In D1 and D2, we see many software engi-neering-related n-grams, *e.g.*, "tests" and "the pull", among non-toxic comments but almost none among toxic comments.

Table 4.2: N-grams that are over-represented in either class in *D2 Toxic OSS Code Review Comments*. N-grams with second-person pronouns are in bold. N-grams with software engineering terms are underlined.

	unigram	z-score	bigram	z-score	ngram	z-score
Toxic	**you**	12.172	it is	5.555	**if you want**	3.397
	people	7.292	**you want**	4.81	it is not	2.712
	even	7.097	that is	4.272	**do you think**	2.576
	do	6.71	going to	4.256	**you need to**	2.397
	what	6.644	**you are**	4.187		
	is	6.373	trying to	4.053		
	want	6.078	**if you**	3.682		
	your	5.796	to do	3.668		
	because	5.657	do not	3.556		
	why	5.547	**you think**	3.539		
Non-toxic	tests	-4.773	**could you**	-2.815		
	unit	-4.858	the pull	-2.889		
	vs	-4.982	as the	-3.137		
	file	-5.15	and the	-3.143		
	files	-5.165	of files	-3.296		
	for	-5.574	we can	-3.48		
	test	-5.76	pull request	-3.668		
	from	-5.872	code to	-3.856		
	at	-6.732	to the	-4.031		
	line	-6.782	instead of	-5.004	the pull request	-2.276

In D3 and D4, we likewise see more technical terms among non-pushback comments. In Table 4.2, n-grams with software engineering terms are underlined.

Code of Conduct We occasionally see "code of" and "the code of" appear in the top-10 lists. Typically, these two n-grams appear when referring to "the code of conduct", often as a reminder that someone violated the code of conduct. For example, one contributor wants the maintainer "to enforce the code of conduct [...]." Interestingly, we observe this pattern in D4 (pushback in open source code reviews), which was sampled without using this as a search term.

No Pattern and Overfitting Finally, among all four datasets, we see some n-grams with no discernible rationale for why they might be indicators or contraindicators of toxicity or pushback. For instance, in Table 4.2, the bigrams consisting of only stop words, *e.g.*, "as the", "and the", and "to the", appear to just be noise, rather than true indicators of non-toxic open-source code review. As an example of overfitting, the top unigram in D3 ("<tech1>") indicates a widely-used, Google-specific piece of technology.

Overall, this exploration confirms that discussions in the positive labels, tend to shift focus away from the technical aspects themselves and onto interpersonal issues. The ground truth labels on all four datasets appear meaningful, since there are noticeable differences in the relative frequency of words and phrases between discussions with presence and absence of toxicity and pushback. Moreover, the analysis implies that there is substantial overlap

between the two concepts of pushback and toxicity, suggesting that incorporating text-based features into classifiers for both concepts is worthwhile. However, the absence of a clear pattern for many n-grams suggests that a purely frequency-based approach would be insufficiently discriminatory for an accurate classifier. In what follows, we introduce more sophisticated classification approaches.

4.6 Methods for Classification

4.6.1 Building classifiers for toxic comments and pushback in code reviews

Text-based features In this paper, we reuse and improve the classifier developed by Raman *et al.* [46], which takes outputs from several text-based pretrained classifiers as features. We first preprocessed the text by removing URLs, quotes, numbers, *etc.* Then we feed the text into the following three pre-trained NLP classifiers, and use the outputs as features.

Following Raman *et al.* [46], we collect (1) the toxicity score and identity attack score from the Perspective API ([0, 1] range, with 1 being the most toxic/aggressive) and (2) count the occurrences of different politeness strategies using the politeness parser [155, 271] (normalized to [0, 1]). In addition, we used (3) a sentiment analysis tool developed for software engineering code review comments, SentiCR [277], with reportedly better performance on GitHub data than other sentiment analysis tools [276]. The output from SentiCR is either positive sentiment (1) or negative sentiment (-1).

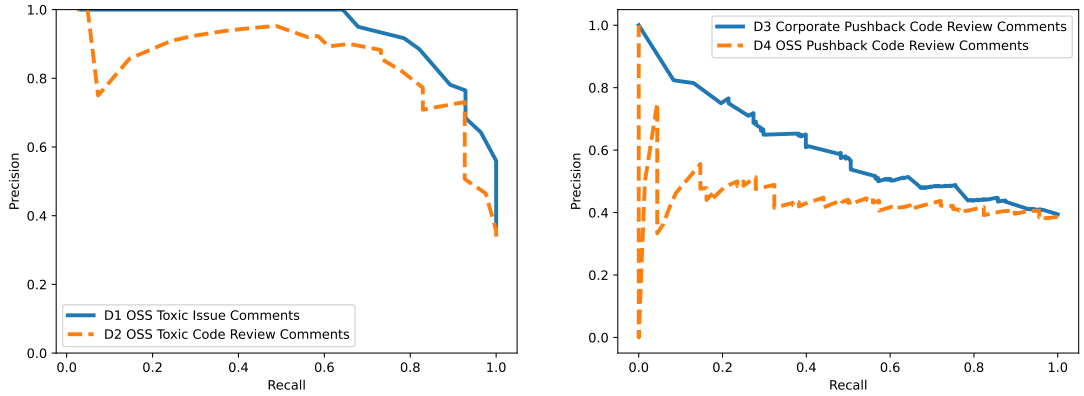
Logs-based features Because we are interested in answering whether the pushback classifier by Egelman *et al.* [47], which uses logs-based features, can be applied to open-source code review comments (RQ1), we calculated logs-based metrics for D2 and D4, the two novel datasets. Egelman *et al.*'s work on code review in the company used rounds of review, active reviewing time, and active shepherding time to build a classifier for pushback. They defined:

- *Rounds of review* as the number of batches of contiguously authored comments, as it "captures the extent to which there was back-and-forth between the author and reviewer."
- *Active reviewing time* is "the time invested by the reviewer in providing feedback," which includes actively viewing, commenting, or working on code review.
- *Active shepherding time* is the time "the author spent actively viewing, responding to reviewer comments, or working on the selected CR, between requesting the code review and merging the change into the code base."

The above "active" times may include time outside of code review, *e.g.*, editing files, but does not account for in-person conversations.

As discussed in Section 4.4, for GitHub data we could not exactly replicate all three logs-based metrics used by Google, because of differences between Google's code review tool and the GitHub pull request workflow. Therefore, by necessity we operationalized these metrics for open-source code review comments (D2 and D4) differently:

- We approximated *Rounds of review* as the number of comments on a pull request, since GitHub code review comments are not always grouped into batches the way Google's are.



(a) Text-based classifier P-R curves on D1 and D2.

(b) Logs-based classifier P-R curves on D3 and D4.

Figure 4.1: Text-based classifier P-R curves

- We approximated *Active shepherding time* as the time difference between the initial PR post and the last comment. Note that the difference between our shepherding time and the one by the company is that the company uses the actual amount of time an author spent on a code change, whereas ours is the wall-clock time of the entire review process, which may result in longer shepherding time overall.
- We did not attempt to approximate *Active reviewing time*, because we could not distinguish how much of the time between the submission of code and the last comment was taken by reviewers or by the author.

Training We trained a random forest [282] classifier for each classification task because of its accuracy and robustness against overfitting [283, 284].

Following Raman *et al.* [46], we performed 10-fold nested cross validation to find the best model and reduce bias from random data splits. We first randomly split our labeled data into a training set (67%) and a test set (33%). We used stratified sampling to preserve the ratio between labels and ensure that each set contains both positive and negative labels.

We then fit and cross validate a random forest model using the training set for 10 trials. In each trial, the training set is further split into 10 folds randomly. Each fold is used once as a cross validation set, while the remaining 9 folds are used for training. The random forest model learns the best combination of hyperparameters, such as `n_estimators` and `max_depth`, optimizing for F1 score, the harmonic mean of precision and recall.

After each trial, we tested the random forest model with the combination of hyperparameters that produced the highest F1 score during training (67% of the entire labeled dataset) on the held-out test data (33% of the entire labeled dataset).

For the text-based classifier, the classification is performed at comment level. Then we aggregate the classifications to form thread-based labels. For pushback datasets (D3 and D4) where we only have thread-level labels, we assign all comments the same label as the thread-level label. For the logs-based classifier, the classification is performed at thread-level.

4.6.2 Classifier Performance Analysis

To evaluate the performance of our classifiers, we computed and compared the Areas Under the Precision-Recall (P-R) Curves, *i.e.*, the P-R AUC scores. Precision tells us how many comments labeled by our classifier as toxic/pushback are in fact toxic/pushback, and recall tells how many toxic/pushback comments in our test dataset are classified as toxic/pushback. P-R curves explore the classic precision/recall tradeoff in applications where the data is imbalanced [285], as is ours — toxicity and pushback are both relatively rare. P-R curves are also commonly used to evaluate classifiers when researchers care more about positive (toxic or pushback) than negative labels. This is also the case in our work — for downstream prevention, mitigation, and future research on toxicity and pushback, we believe that it is more important to identify true instances of toxicity and pushback than it is to identify that some comment or conversation is not toxic or pushback. A P-R AUC score summarizes the performance of a classifier into one value and can be interpreted as the average of precision scores calculated for different recall thresholds, with higher values (closer to 1) being preferable.

To compute the P-R curves, we uniformly vary the classifier's probability threshold for predicting the positive class, which corresponds to exploring the precision-recall tradeoff. To compare classifiers, we performed pairwise t-tests on their P-R AUC scores computed after the 10 cross-validation trials. At each trial, we applied the random forest classifier with the best hyperparameter combination on the held-out test data and computed an AUC score. As a result, from our 10-fold nested cross validation training process, we obtained 10 AUC scores (one per trial). For each t-test, we also report Cliff's delta measure of effect size.

In addition, we estimate the importance of each feature [284] in our random forest classifiers during the training phase, using a standard approach based on the mean overall improvement in a tree's impurity. The impurity, in classification tasks, is measured by the Gini index, interpreted as the probability of an item being incorrectly classified if it was randomly labeled according to the distribution of a specific feature [283].[2]

4.7 Results

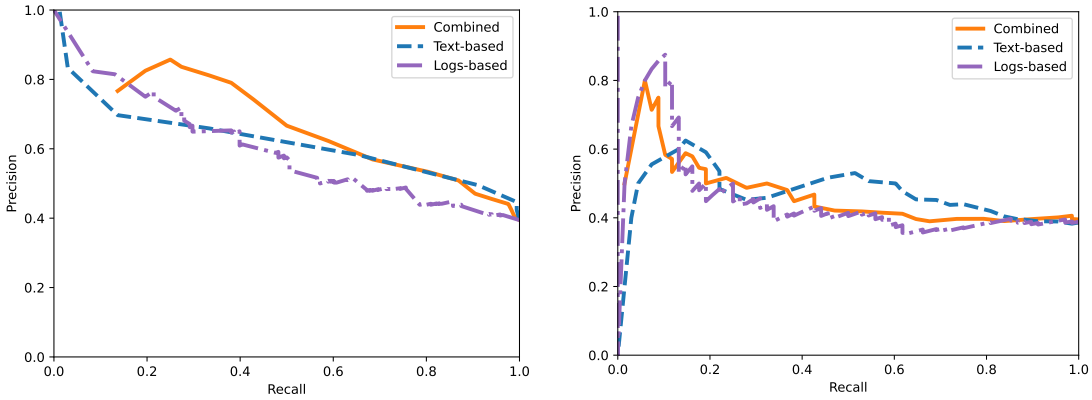
RQ 2*: How well do existing classifiers generalize across context and type of discussion?*

To answer this question, we plot the P-R curves by the classifiers using the same features on different datasets and compare the average AUC scores. Figure 4.1 shows one of the curves from the 10 trials.

We start by comparing the P-R AUC scores for the text-based toxicity classifier on D2 (open-source code reviews) relative to the benchmark D1 (open-source toxic issues), answering $\mathbf{RQ}_{1.1}$. The P-R curves are shown in Figure 4.1a. We find that at the thread level, the text-based classifier performs better on D1 than on D2 ($t = 5.640$, p-value $= 0.0001$; Cliff's $\delta = 1$ / large effect; the AUCs are 0.907 and 0.844 respectively).

We manually checked some randomly sampled toxic comments from D1 and D2 that our text-based classifier failed to identify. We found that some of them are responding to toxic behavior. For example, phrases like "you spent a long time insulting people" are responses

[2]Our code is available at https://doi.org/10.5281/zenodo.6051070



(a) P-R curves of all three classifiers on D3

(b) P-R curves of all three classifiers on D4

Figure 4.2: P-R curves on pushback classification

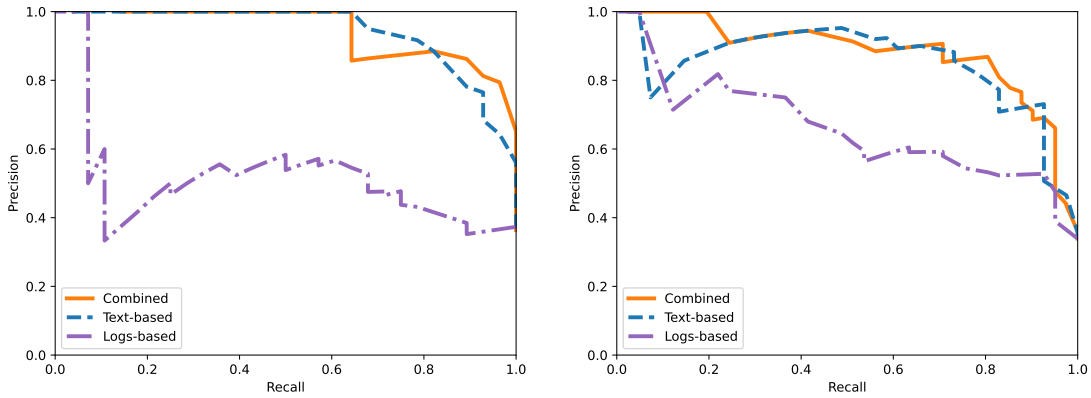
to someone else's insult and are clearly a signal of the presence of toxicity. Some other ones contain *covert toxicity*, such as sarcasm, entitlement, or the use of "?!" or emojis. Covert toxicity is difficult for language models to detect in general [286]. These comments also have a low predicted toxicity score by our classifier; some even use the word "please" as in "Please consider that this thread [...] is so problematic. [...] get this PR closed ASAP."

The impurity-based feature importance analysis (Figure 4.4a in Appendix) provides some explanations on what features are important in both datasets. The x-axis is the importance score of the features. The sum of importance scores of all features is 1. The two most important features during the training phase are from the Perspective API. They are followed by three politeness features: second person pronouns, the presence of negative words, (*e.g.*, "begging for complete code review" and "many bugs documented and unresolved"), and the use of first person pronouns. The use of second person pronouns echoes our findings of the word frequency analysis, where we see the use of "you" overrepresented in toxic text.

Reflecting on differences between the issue conversations and code review conversations that could cause the performance degradation when detecting toxicity in the latter case, we speculate two reasons based on exploring the two labeled datasets. One is that many code-specific comments are much shorter than discussion comments, yielding less linguistic information. The other possibility is that the code review conversations in our dataset more often include code chunks and removing inline codes may reduce information for the text-based classifier.

Next we compare the P-R AUC scores for the logs-based classifier on D3 (pushback in corporate code review) and D4 (pushback in open-source code review), answering $\mathbf{RQ}_{1.2}$. The P-R curves are shown in Figure 4.1b. Our results show that the logs-based classifier has a lower performance when transferred to the open-source context, despite being retrained ($t = 40.008$, p-value $< 2.2e-16$; Cliff's $\delta = 1$; the average AUC scores are 0.693 and 0.445 for D3 and D4).

We speculate that there are two main reasons for the lower performance. First, limited by the information publicly available on GitHub, we could only compute measures for two of the three logs-based features used originally inside Google. Therefore, we have less information.



(a) P-R curves of all three classifiers on D1

(b) P-R curves of all three classifiers on D2

Figure 4.3: P-R curves on toxicity classification

Indeed, in D3, reviewing time, the feature missing in D4, ranks as the most important (Figure 4.5 in Appendix). Second, our measure of shepherding time computed for open-source code reviews is only an approximation, using wall-clock time rather than the amount of time spent actively working on code in review. Therefore, the logs-based features we computed for open-source data are not as accurate as those on corporate data.

Summary: *Both the text-based classifier and the logs-based classifier have performance degradation when generalizing to other contexts.*

RQ_2: *How well do existing classifiers generalize for both toxicity and pushback?*

To answer this research question, we compare the performance of the classification approach originally designed for one construct (toxicity or pushback) to the classification approach originally designed for the other construct.

We start by evaluating the performance of the text-based classifier on datasets D3 and D4, compared to the performance of the logs-based classifier as a benchmark, answering $\mathbf{RQ}_{2.1}$. Figure 4.2 shows one of the P-R curves from the 10 trails.

On *D3 Pushback in Corporate Code Review*, the text-based classifier outperforms the logs-based classifier on average ($t = 9.766$, p-value $= 1.304e-08$; Cliff's $\delta = 1$ / large effect; the mean AUC scores are 0.757 and 0.693 for the text-based and logs-based classifiers respectively); note, this logs-based classifier is the one using all three measures of pushback, available inside the company Google. This suggests that pushback as a construct shares many linguistic similarities with toxicity. In addition, the better performance of the text-based classifier suggests that, in a corporate setting, interpersonal conflicts can be more subtle than delay of reviews or excessive comments.

For *D4 Pushback in Open-Source Code Review*, comparing the P-R AUC scores shows that, unlike previously on D3, the text-based classifier and the logs-based have a similar performance ($t = 0.246$, p-value $= 0.810$; the average P-R AUC scores are 0.447 for the

text-based classifier and 0.445 for the logs-based one); note, the logs-based classifier in this case contains only the measures available publicly on the GitHub platform.

One possible explanation is that in contrast to the previous corporate dataset, there are relatively fewer examples of open-source pushback code reviews in our sample that could be traced back to reasons with linguistic markers. Therefore, there is less for the text-based classifier to discern. To test this hypothesis, we compiled a subset of D4, D4-1, containing as positive examples all the reported pushback open-source code review threads that are linked to linguistic markers (Figure 4.7 in Appendix), such as "harsh comments" (39 out of 63 self-reported pull requests), and as negative examples the remaining self-reported pushback open-source code review threads with only likely non-linguistic markers, such as "excessive review delays." Comparing the performance of the logs-based and text-based classifiers on D4-1 does not support our hypothesis: the text-based classifier underperforms the logs-based one ($t = -5.072$, p-value $= 0.0002$; Cliff's $\delta = -0.86$ / large effect; the average AUCs are 0.566 and 0.679 respectively). The reason could be that pushback threads labeled with linguistic-related reasons are often labeled with non-linguistic ones too, *e.g.*, "requesting a change without justification."

Another possible explanation is that since pushback classification is done at the thread level, within a thread the actual comments indicative of pushback are too rare for the whole text of the threads to be significantly different on average between the pushback and non-pushback classes. To test this, we created dataset D4-2, in which we assigned pushback labels at the comment level. Specifically, we used the responses to our survey asking participants to copy-paste the text fragments indicating pushback, in addition to offering pushback pull requests as a whole, to identify which comments in the thread contained those exact fragments. We then labeled those comments as pushback and all other comments in the same threads as non-pushback. Then we performed the classification and thread-level aggregation as usual. Comparing the performance of thread-level text- vs logs-based classification on D4-2, we observe that the text-based classifier now outperforms the logs-based one ($t = 2.591$, p-value $= 0.026$; Cliff's $\delta = 0.54$ / large effect; the average AUCs are 0.534 and 0.471 respectively), supporting our hypothesis.

The feature importance analysis (Figure 4.4b in Appendix) for the text-based classifier on both pushback datasets D3 and D4 present some insights into what linguistic features are associated with pushback comments. On both datasets, the toxicity score and identity attack score from the Perspective API have the highest importance. They are followed by several politeness strategies. The third most important feature in D3 is the presence of positive lexicons whereas in D3 is the number of hedge words, such as "likely", "maybe", "seems". Having second person pronouns is also an important feature to classifying D3 Pushback in Corporate Code Review but less so to D4 Pushback in Open-Source Code Review.

Summary: *When detecting pushback, the text-based classifier performs better than the logs-based classifier for corporate code review comments, but they have similar performance for open-source code review comments.*

Next we compare the performance of the logs-based classifier against the performance of the text-based classifier on detecting toxicity, answering $\mathbf{RQ}_{2.2}$. We plotted the P-R curves for the text-based and the logs-based classifiers on D1 and D2, shown in Figure 4.3. We find

that the text-based classifier performs better than the logs-based one on both D1 ($t = 45.515$, p-value $< 2.2e-16$; Cliff's $\delta = 1$; P-R AUC scores are 0.907 and 0.516 respectively) and D2 ($t = 13.591$, p-value $= 2.22e-10$; Cliff's $\delta = 1$; P-R AUC scores are 0.844 and 0.665, respectively). The good performance of the text-based classifier implies that toxicity is more of a linguistic phenomenon. Meta-data, such as the logs-based features we computed, could not capture enough information to distinguish toxic language.

Summary: *The logs-based classifier does not perform as well as the text-based one when detecting toxic open-source issues and code review comments.*

RQ_3: *To what degree can combining existing approaches improve detection of toxicity and pushback?*

We start by comparing P-R AUC scores of the text-based and the logs-based classifiers against that of the combined classifier when detecting toxicity, on both D1 and D2, which answers **$RQ_{3.1}$**. The P-R curves are shown in Figure 4.3. Overall, we find that the combined classifier has better performance than the logs-based classifiers but is similar to the text-based classifier. On D1, the combined classifier outperforms the logs-based one (a t-test between the logs-based classifier and the combined classifier: $t = -51.975$, p-value $< 2.2e-16$; Cliff's $\delta = -1$; the AUC scores are 0.516 and 0.895 respectively) but is indistinguishable from the text-based classifier (a t-test between the text-based classifier and the combined classifier: $t = 0.376$, p-value $= 0.712$, the text-based classifier's AUC is 0.907).

Similarly, on D2, the combined classifier outperforms the logs-based classifier (a t-test between the logs-based classifier and the combined classifier: $t = -24.226$, p-value $= 9.001e-12$; Cliff's $\delta : -1$; AUCs are 0.665 and 0.871 respectively). However, the combined classifier outperforms the text-based classifier (a t-test between the text-based classifier and the combined classifier: $t = -2.3884$, p-value $= 0.0363$; Cliff's $\delta : -0.58$ / large effect; the AUC of the text-based classifier is 0.844).

The feature importance analysis (Figure 4.6a in Appendix) shows that text-based features are more important in detecting toxicity than logs-based features. This suggests that, again, toxicity is more about the language than the logs-based metrics. The toxicity score and identity attack by the Perspective API have the highest importance. They are followed by the two logs-based features, which are followed by several politeness strategies. The use of second-person pronouns is also among the top 5 most important features, which echoes our findings in the word frequency analysis.

Summary: *For toxicity, the combined classifier has a similar performance to the text-based one on toxic issue comments but a better performance on toxic code review comments. The combined classifier performs better than the logs-based one in detecting toxicity.*

Finally, we compare the AUC scores between the text-based and the combined classifier and between the logs-based and the combined classifier when detecting pushback (D3 and D4), which answers **$RQ_{3.2}$**. The P-R curves are shown in Figure 4.2.

On *D3 Corporate Pushback Code Review Comments*, the combined classifier performs better than the logs-based (a t-test between the logs-based and the combined classifier: $t = -12.511$, p-value $= 2.108e-08$; Cliff's $\delta = -1$ / large effect; AUC are 0.693 and 0.755)

but about the same as the text-based one (a t-test between the text-based and the combined classifier: $t = 0.363$, p-value $= 0.723$; the text-based classifier's AUC is 0.757).

On the contrary, on *D4 Open-Source Pushback Code Review Comments*, the performance of the logs-based classifier is similar to the the combined classifier ($t = -2.1171$, p-value $= 0.052$; the average AUC scores are 0.445 and 0.455 respectively). Similarly, the combined classifier's performance is indistinguishable from that of the text-based classifier (a t-test between the text-based and the combined classifier: $t = -0.929$, p-value $= 0.373$, the text-based classifier's AUC is 0.447).

From the feature importance analysis on the combined classifier on our two pushback datasets D3 and D4 (Figure 4.6b in Appendix) shows that the logs-based features have higher importance than the text-based ones. Among the text-based ones, toxicity score and identity attack have the highest importance, followed by several politeness strategies.

Summary: *For classifying pushback in code reviews, the combined classifier performs better than the logs-based classifier but about equivalently to the text-based classifier in a corporate setting; and performs about equivalently to the text-based classifier and the logs-based classifier in an open-source setting.*

4.8 Discussion

Classifiers' cross-domain application For **RQ**$_1$, we found that prior classifiers' performance [46, 47] degrades when applied to new datasets. For open-source code review comments, one reason may be that, compared to issues, discussions in PRs are generally more technical, and hence, less personal. One reason the logs-based classifier performed relatively poorly in open-source code review may be that we were not able to accurately reproduce one of the corporate pushback features, active shepherding time.

Relationship between toxicity and pushback By answering **RQ**$_2$, how well can the classifiers generalize across domains and datasets, we can conclude some relationship exists between the two concepts. Pushback is initially centered around delays in code review, which is associated with lower productivity [47], whereas toxicity is centered more around the negative interactions among contributors during code review [46]. However, Egelman *et al.* [47] reported that, in addition to lengthy reviews, pushback is also characterized by interpersonal conflict. This is supported by our finding that the text-based classifier has a better performance than the logs-based one on pushback detection in a corporate setting, suggesting that pushback in a corporate setting is more subtle than lengthy discussions or delayed reviews. Similarly, in open-source, toxic language is also a significant part of pushback. Among the pushback code review comments users reported, more than half of them have reasons related to communication (Figure 4.7 in Appendix). However, we found that the logs-based features did not improve toxicity detection. This suggests that toxicity is mostly about language, and meta-data cannot capture the nuance.

Corporate vs. open-source settings When answering **RQ**$_2$, we were also able to compare the two contexts, corporate and open source. We found that the text-based classifier works better than the logs-based one when classifying corporate pushback. However, it was surprising

that the logs-based classifier and the text-based one have similar performance when classifying open-source pushback. This differs from the impression we had from the survey responses. From the survey responses, we observed many complaints about maintainers delaying the review process. When looking at some of the PRs, we saw that many of the maintainers mentioned having a holiday or being busy with day jobs as reasons for the delay. One comment from the open-source pushback survey reflected that "It's not PR and not about code review, but it's about open source world."

Moreover, both the text-based and the logs-based classifiers have better performance on corporate pushback code review comments than on open-source ones. This suggests some differences between the two datasets. Perhaps these differences arise from uniformity in Google's code review practices [279] compared to the multitude of practices used on GitHub [280].

This also raises the issue of transferring our results to other settings. When answering $\mathbf{RQ}_1$, we found that using the same set of features on data from a different context resulted in lower performance. However, the multiple levels of comparisons we conducted in this study can act as a guideline while developing a system for toxicity and pushback detection in other contexts.

Prediction vs. classification In this paper, we performed classification on conversations after they had concluded, largely because logs-based features are not applicable to individual comments. As a result, our current models cannot yet be applied to all scenarios where automated detection of toxicity or pushback are of interest, *e.g.*, comment-level classification for just-in-time intervention. Instead, we target primarily scenarios where thread-level classification is needed, *e.g.*, to reflect on when discussions have gone awry (of interest to practitioners) or to detect and study when, how, and why toxicity and pushback occur (of interest to researchers).

Future work can explore how to use text-based features to do real-time detection and offer editing suggestions. Cherjyan *et al.* [261] proposed a Conflict Reduction System that can rephrase offensive sentences. However, their datasets are heavily focused on swearing and profanity. Our findings can greatly enrich the set of text features that can be used to detect and prevent potential toxic comments.

Text analytics improvements Our text classifier combined three different NLP techniques, but other NLP techniques on larger datasets is a future research direction. Some paths that can be explored include using text embedding [287] or conversational structure [272]. One could also use Snorkel [288], a weak supervision model, to help augment our labeled dataset.

Prior studies have shown that general NLP models may not be directly applicable to software engineering corpora [289, 157]. For example, "error" and "test" are mostly neutral in the software engineering context but have negative connotations in general English. Han *et al.* [290] report that Perspective API can misclassify toxic inputs due to a domain mismatch or novel lexicon of toxicity. Therefore, some fine-tuning is needed on top of the Perspective API to attain better performance. Raman *et al.* [46] suggested fine-tuning a classifier using a domain-specific lexicon. However, this is a difficult task that needs careful design and evaluation. Thresholds and datasets are all variables that can be explored. Moreover, when evaluating the effectiveness of the domain-specific lexicon tuning, how do we decide what

words should be in the list and what should not? These questions are worth exploring in the future.

4.9 Threats to validity

Internal validity The data we used for training and testing our classifiers is small in two respects. The first is from a machine learning perspective, where more data often yields more reliable conclusions. The second is from an ecosystem perspective; the data we studied represents a small subset of all the discussions going on within GitHub and Google, likely limiting the generalizablity of our results.

Another limitation is that our data, both existing and newly collected, rely on human raters to judge interpersonal conflict. While Egelman and colleagues' showed some degree of reliability across different raters, nonetheless perceptions of interpersonal conflict invariably differ from person to person. Such differences threaten the true accuracy of our ground truth data.

External validity A major threat to generalizability is the context in which we collected our data. For corporate code reviews, we used data from Google; classifying code reviews in other companies would likely yield different results. Likewise, our other datasets are from GitHub; data obtained from other platforms may also yield different results.

Construct validity The lack of comment-level labels in pushback datasets D3 and D4 likely confused the classifiers using text-based features. Because all comments within a pushback conversation share the same label, some neutral or positive comments are also labeled as pushback. Since our text-based classifier works on the comment level, it can get confused when seeing comments associated with polite strategies (*e.g.*, indirect start) and impolite strategies (direct questions) that are both labeled as pushback.

In our analysis, we bridged concepts and contexts in prior work [46, 47], between open and closed source; and issues and code reviews. However, we did not exhaustively explore this space. For instance, we did not collect data for toxic corporate code reviews or issues. Given the results that the text-based classifier works well on Google's pull requests, using it to detect or understand toxic comments may be worthwhile future work.

4.10 Conclusion

In this paper, we cross-pollinated with two techniques designed to detect interpersonal conflict. In applying these text- and logs-based techniques to broader contexts than those for which they were originally designed, we uncovered several novel insights. For instance, we found that prior work that detected code review pushback using logs data [47] can be improved substantially by analyzing the text contained in those code reviews. While the opposite was not true – logs data did not improve issue toxicity detection – we nonetheless found that logs can be a useful feature in toxicity classifiers. Building on these techniques, we envision a future where tools can help software developers learn from or avoid interpersonal conflict, enabling projects to be more inclusive of a wider variety of contributors.

4.11 Appendix

Table 4.3: Over and underrepresented words in *D1 Toxicity in Open-Source Issues Comments*. N-grams with second-person pronouns are in bold. N-grams with software engineering terms are underlined.

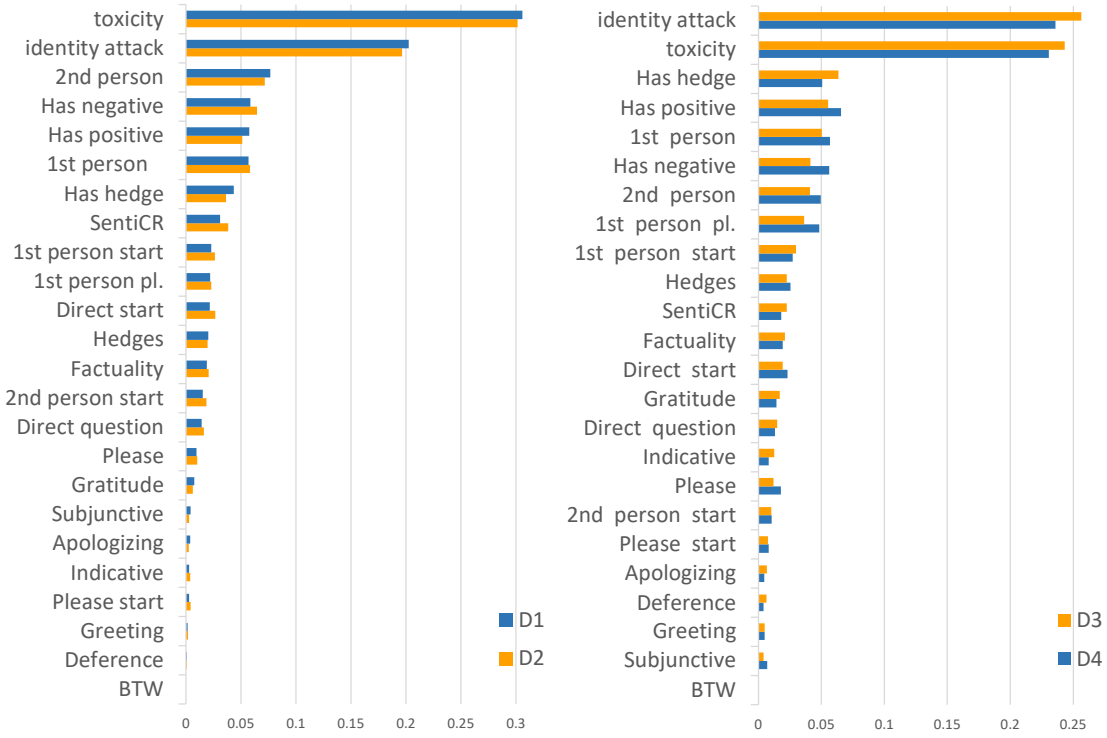
	unigram	z-score	bigram	z-score	ngram	z-score
Toxic	**you**	30.77	this is	12.756	this is not	6.013
	it	23.724	in the	11.822	**you want to**	5.217
	that	22.437	**you are**	11.651	**you need to**	4.869
	of	22.051	it is	10.608	there is no	4.303
	and	21.318	**you have**	9.389	**if you want**	4.272
	is	18.917	to be	9.371	**you have to**	4.036
	this	18.524	**that you**	9.145	to do with	4.036
	your	18.121	**if you**	8.727	**if you want to**	3.971
	have	16.647	to do	7.535	part of the	3.94
	what	15.62	have to	7.514	the problem is	3.799
Non-toxic	via	-3.526	team and	-2.825		
	unit	-3.82	plenty of	-2.838		
	team	-3.871	of experience	-2.954		
	assigned	-3.979	with our	-2.972		
	returns	-4.32	and provide	-2.972		
	function	-4.452	to remove	-3.037		
	item	-5.104	with an	-3.042		
	ticket	-5.121	issue was	-3.263		
	duplicate	-5.528	assigned to	-3.44		
	click	-5.62	looking for	-3.573		

Table 4.4: Over and underrepresented words in *D3 Pushback in Corporate Code Review*. N-grams with second-person pronouns and gratitude are in bold. N-grams with software engineering terms are underlined.

label	unigram	z-score	bigram	z-score	ngram	z-score
Pushback	<tech1>	5.352	**you want**	3.04	**you want to**	2.792
	tests	4.452	want to	2.849	on nov at pm	2.637
	<tech2>	3.683	of these	2.849	nov at pm	2.577
	our	3.599	of our	2.626		
	build	3.564	is to	2.575		
	libraries	3.362	if we	2.525		
	break	3.245	depend on	2.464		
	thing	3.197	we use	2.464		
	see	3.177	the cl	2.441		
	rollback	3.152	this case	2.311		
Non-pushback	submit	-5.338	to represent	-3.831	make sure the	-2.566
	groups	-5.485	to me	-3.834	to do the	-2.64
	feature	-5.514	how about	-3.882	not sure if	-2.69
	<tech3>	-5.64	to submit	-3.96	seems to be	-2.805
	map	-5.664	this function	-4.106	in this cl	-2.813
	rate	-6.042	the new	-4.286	which is not	-2.919
	thanks	-6.303	**could you**	-4.363	**do you have**	-3.189
	section	-6.336	for the	-4.432	how do we	-3.604
	the	-6.492	change the	-4.439	to change the	-4.009
	for	-6.9	**thanks for**	-5.291	**thanks for the**	-4.891

Table 4.5: Over and underrepresented words in *D4 Pushback in Open-Source Code Review*. N-grams with second-person pronouns, gratitude, and "code of conduct" are in bold. N-grams with software engineering terms are underlined.

label	unigram	z-score	bigram	z-score	ngram	z-score
Pushback	runtime	17.511	is of	6.622	**the code of**	3.957
	suggestion	9.676	the project	6.452	the new format	3.721
	argument	9.32	**code of**	6.171	for the new	3.457
	us	8.762	of type	6.006	the commit message	3.185
	people	8.35	the linter	4.638	to the project	3.096
	timer	8.218	read the	4.583	as long as	3.003
	non	7.254	it is	4.313	the number of	3.003
	high	7.068	social media	4.186	to read the	2.957
	requirements	6.29	the old	4.13	just wanted to	2.874
	de	6.276	commit message	4.026	we dont want	2.874
Non-pushback	access	-5.923	the following	-3.402		
	struct	-5.992	an error	-3.412		
	config	-6.197	it seems	-3.536	is going to	-2.311
	tests	-6.282	the same	-3.715	it would be	-2.5
	server	-6.351	**thank you**	-3.786	all of the	-2.5
	line	-6.431	the server	-4.021	this should be	-2.802
	field	-6.632	did not	-4.047	it seems that	-2.872
	build	-7.262	the tests	-4.12	**thank you for**	-2.972
	info	-7.309	file line	-4.287	let me know	-3.111
	error	-7.319	line in	-5.301	file line in	-4.287



(a) D1 vs D2 (b) D3 vs D4

Figure 4.4: Text-based classifier feature importance scores.

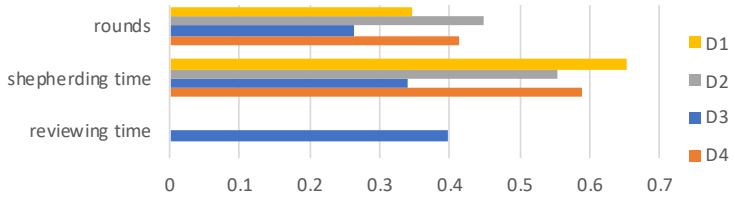
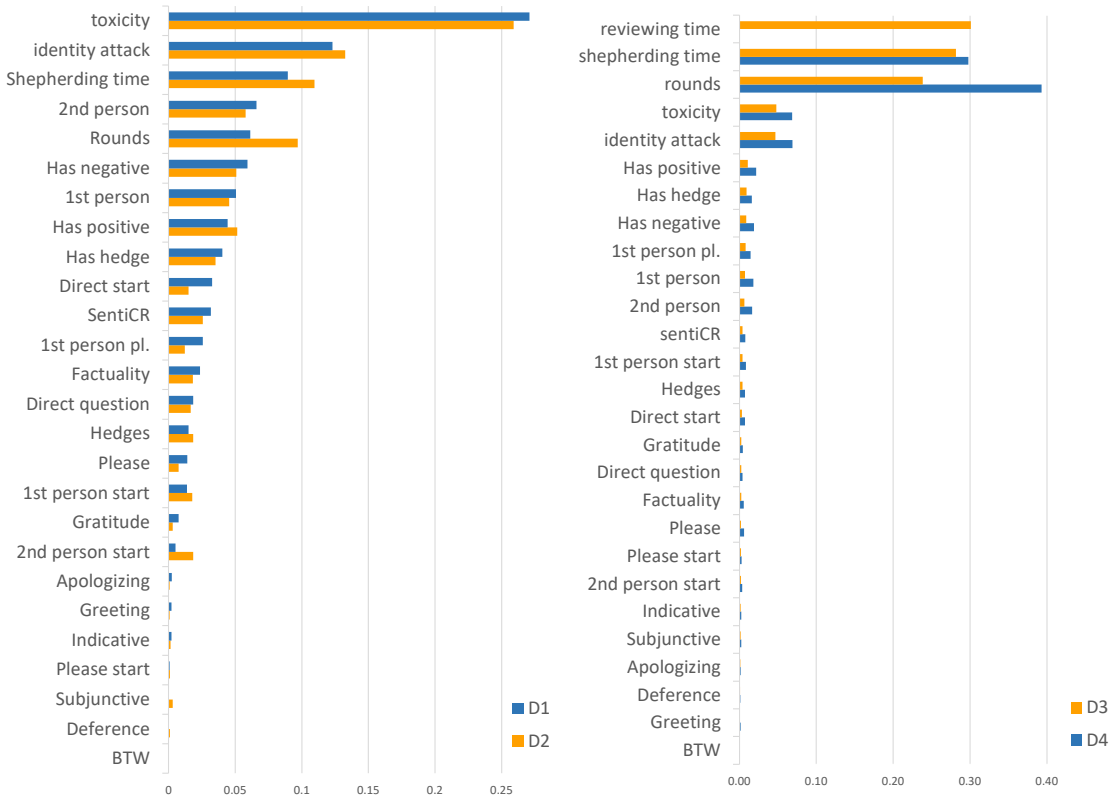


Figure 4.5: Logs-based classifiers' feature importance



(a) Toxicity (D1 and D2) (b) Pushback (D3 and D4)

Figure 4.6: Combined classifiers' feature importance

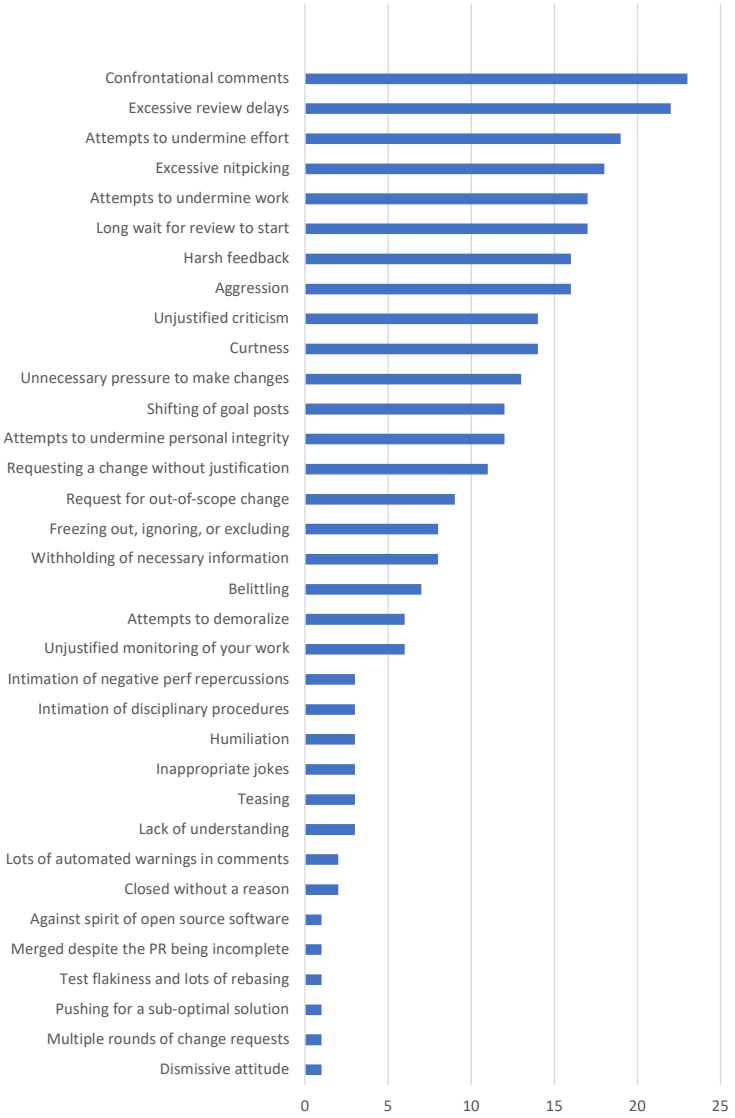


Figure 4.7: Reasons for pushback in OSS

Chapter 5

Intervention: A Dashboard for Maintainers

Chapter 6

Conclusion

This chapter concludes this dissertation by summarizing our contributions and revisiting thesis statement from Chapter 1. Then I will discuss some directions for future research.

6.1 Contributions

In this dissertation, I used the problem of low gender diversity as a starting point and conducted a series of empirical studies to get a better understanding of how to improve diversity and inclusion in each phase of an open-source contributor's career path (Figure 6.1). In the end, I built a dashboard based on the findings from my studies to help maintainers better attract and retain contributors. In this section, I will reiterate my findings on each phase of an open-source contributor.

From a newcomer to a contributor

In this chapter, we used a mixed-methods approach to study how to help new OSS contributors find a suitable project based on signals available on GitHub. We identified a list of signals that GitHub contributors recommend using when assessing projects and estimated logistic regression models to validate each signal's effectiveness in attracting new contributors.

The signals we identified can be roughly classified into three categories: popularity, *e.g.*, the number of stars and recent commits; community, *e.g.*, impolite language and responsiveness; and quality, *e.g.*, a well-structured and thorough README and a contributing guideline. However, not all signals are currently easily observable on GitHub. For example, one can quickly evaluate the quality of a README, but cannot easily infer how friendly the community is or how responsive the maintainers are.

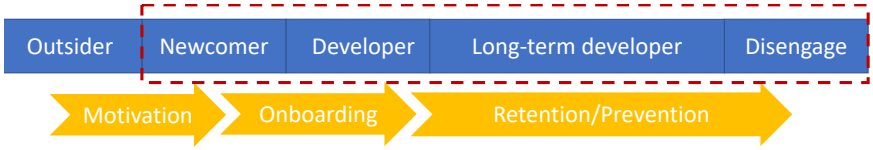


Figure 6.1: An open-source contributor's different phases

This study's results have significant implications for open-source maintainers and the design of social coding environments and intervention tools. We use many of the signals discovered in this study in our climate coach dashboard design.

From a contributor to a long-term contributor

This chapter studied the impact of social capital on sustained participation of open source contributors and, in particular, on gender differences in this impact. We performed a mixed-methods empirical study: we applied survival analysis on a large dataset of OSS contributors and their GitHub collaborators; we also surveyed a subset of these contributors about their perception of social capital to triangulate our findings.

Our results show that being able to obtain more social capital is associated with a higher likelihood of prolonged participation. Bonding social capital, obtained through strong social ties, can provide a sense of belonging and willingness to continue contributing. Bridging social capital, obtained through structure holes, can provide diverse information and more opportunities to continue contributing.

For women, diversity of the project members' expertise becomes more critical to sustain their participation: higher team diversity in terms of prior programming language expertise is associated with decreased risk of disengagement both short- and long-term.

This study reveals valuable signals that are important for improving diversity and inclusion but are currently hard to observe on GitHub, such as recurrent collaboration among teammates and diversity of teammates' technical backgrounds.

Disengagement prevention

This chapter explored how to prevent disengagement by developing automatic tools to detect interpersonal conflict in software development. We cross-pollinated two techniques initially designed for different types of interpersonal conflicts, *i.e.*, pushback and toxicity. Moreover, these two techniques were developed under different contexts, *i.e.*, corporate and OSS, and for different types of discussions, *i.e.*, code review and issues. These two techniques also employed different methods: text-based, *i.e.*, linguistic features, and logs-based, *i.e.*, meta-information.

We constructed new datasets and systematically evaluated the two techniques across contexts and types of discussions. We also tested the combination of the two methods in detecting pushback and toxicity. Our evaluation uncovered insights that can be useful for developing detectors for new contexts or types of conversations. More importantly, this study provides strong signals that can help flag potentially problematic interactions for maintainers to review.

Climate Coach

6.2 Discussion

6.3 Future work

6.3.1 More forms of diversity

Although this dissertation focuses on gender diversity, and in some quantitative analyses, binary gender diversity, future works can explore other types of diversity. For example, from the demographic perspective, future work can also look into racial, ethnic, and cultural origin diversity. Of course, we will still face challenges in obtaining demographic information from a group of contributors that is large enough for us to perform meaningful analysis.

Other than demographics, future work can also explore diversity in terms of geolocation and tenure. There exist some prior work on geolocation diversity [70], but it is mainly a summary of the number of contributors from different geolocations. Future work can dig deeper into the differences among geolocations, such as culture, education, and attitude towards open-source.

Tenure diversity concerns not only new contributors or experienced, long-term contributors but also how they corporate. Vasilescu *et al.* [58] found that a tenure-diverse team is more productive. More work can be done on how contributors of different levels of expertise can collaborate.

6.3.2 Triangulation

This dissertation employed a mixed-methods approach to discover interventions and measure their effectiveness using the rich signals from social coding platforms (Chapter 2). Some of the findings can be deemed intuitive, such as the number of stars being an indicator of whether a project can attract new contributors. In contrast, some others were unclear until we ran the statistical model, *e.g.*, having a contributing guideline can guide contributors but also create overhead.

Future work should further exploit the mixed-methods approach to triangulate the usefulness of intervention or management strategies. Many qualitative studies reported various problems and proposed several solutions. However, little is yet known about whether the proposed methods can solve the problem and, if so, how effective. Knowledge of the effect size of different interventions can not only help practitioners improve their management strategies but also inform tool design and future study directions.

6.3.3 Tool design

This dissertation presented two tools for improving diversity and inclusion: an interpersonal conflict detector (Chapter 4) and a dashboard for project climate (Chapter 5). Although the interpersonal conflict detector was not deployed for public usage, its features, such as the number of reviews and tone of the comments, are included in the climate coach dashboard.

Studies have identified plenty of problems, such as barriers that newcomers face. What is lacking is interventions. We must put more effort into developing effective management strategies into practical tools to aid OSS maintainers. We could produce more effective tools with more studies on measuring and validating the effectiveness of management strategies and interventions.

Future studies can further explore how to incorporate more gamification features in tools. Although we added the comparison feature in our climate coach dashboard, it did not turn out to be very effective and served only as a reference for maintainers. Since past studies demonstrated that gamification features have effectiveness in encouraging contribution [291], future studies can explore how to make a better gamification design.

6.3.4 Tools cater to different genders

When conducting the research on signals that contributors should use (Chapter 2), we made use of the GenderMag framework [29], which describes that people of different genders tend to have different ways of interacting with technology. Unfortunately, this dissertation did not discover significant differences in how men and women contributors choose projects and did not explore the gender difference in interpersonal conflicts during code review. Future studies can attempt these problems and devise design guidelines for gender-inclusive tools.

6.3.5 Social network analysis

This dissertation only slightly touched on social network analysis when employing the social capital theory to understand contributors' sustained participation. However, social network theory is a large treasure trunk for research on human aspects in software engineering. For example, prior study [38] found evidence of gender homophily being a disadvantage of women contributors, yet little is known about the mechanism of gender homophily in OSS. Moreover, future studies can explore more on social network positions and evolution, such as network embeddedness [292] and how OSS social networks have changed over time.

Bibliography

[1] F. Zlotnick, "Github open source survey 2017," http://opensourcesurvey.org/2017/, Jun. 2017. viii, 19, 35

[2] N. Eghbal, *Roads and Bridges: The Unseen Labor Behind Our Digital Infrastructure*. Ford Foundation, 2016. 1, 13

[3] S. Greenstein and F. Nagle, "Digital dark matter and the economic contribution of apache," *Research Policy*, vol. 43, no. 4, pp. 623–631, 2014. 1

[4] D. Izquierdo, N. Huesman, A. Serebrenik, and G. Robles, "Openstack gender diversity report," *IEEE Software*, vol. 36, no. 1, pp. 28–33, 2018. 1, 6

[5] D. Nafus, "'patches don't have gender': What is not open in open source software," *New Media & Society*, vol. 14, no. 4, pp. 669–683, 2012. 1, 2, 5, 45, 62

[6] A. Bosu and K. Z. Sultana, "Diversity and inclusion in open source software (oss) projects: Where do we stand?" in *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 2019, pp. 1–11. 1, 5, 6, 7

[7] J. Terrell, A. Kofink, J. Middleton, C. Rainear, E. Murphy-Hill, C. Parnin, and J. Stallings, "Gender differences and bias in open source: Pull request acceptance of women versus men," *PeerJ Comp Sci*, vol. 3, p. e111, 2017. 1, 4, 5, 6, 9, 45, 47, 61

[8] S. Balali, I. Steinmacher, U. Annamalai, A. Sarma, and M. A. Gerosa, "Newcomers' barriers... is that all? an analysis of mentors' and newcomers' barriers in oss projects," *Computer Supported Cooperative Work (CSCW)*, vol. 27, no. 3, pp. 679–714, 2018. 1, 3, 5, 25, 33, 39, 59

[9] J. Coelho and M. T. Valente, "Why modern open source projects fail," in *Proceedings of the Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*. ACM, 2017, pp. 186–196. 1, 13, 42

[10] M. Valiev, B. Vasilescu, and J. Herbsleb, "Ecosystem-level determinants of sustained activity in open-source projects: A case study of the pypi ecosystem," in *Proceedings of the Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*. ACM, 2018, pp. 644–655. 1, 13, 59

[11] D. Russo and K.-J. Stol, "Gender differences in personality traits of software engineers," *IEEE Transactions on Software Engineering*, 2020. 2

[12] B. Vasilescu, D. Posnett, B. Ray, M. G. van den Brand, A. Serebrenik, P. Devanbu, and V. Filkov, "Gender and tenure diversity in github teams," in *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, 2015, pp. 3789–3798. 2, 62

[13] J. T. Crawford, "Imposter syndrome for women in male dominated careers," *Hastings Women's Law Journal*, vol. 32, no. 2, p. 26, 2021. 2

[14] GitHub, "Open source survey," http://opensourcesurvey.org/2017/. 2, 42

[15] G. C. Vanderheiden, "Curbcuts and computers: Providing access to computers and information systems for disabled individuals." 1983. 2

[16] A. F. Newell and P. Gregor, "Human computer interfaces for people with disabilities," in *Handbook of human-computer interaction*. Elsevier, 1997, pp. 813–824. 2

[17] K. Crowston and J. Howison, "The social structure of free and open source software development," *First Monday*, vol. 10, no. 2, 2005. 2, 3

[18] K. Crowston, K. Wei, Q. Li, and J. Howison, "Core and periphery in free/libre and open source software team communications," in *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*, vol. 6. IEEE, 2006, pp. 118a–118a. 2

[19] A. Cox, "Cathedrals, bazaars and the town council," *Slashdot. org*, 1998. 3

[20] A. Mockus, R. T. Fielding, and J. Herbsleb, "A case study of open source software development: the apache server," in *Proceedings of the 22nd international conference on Software engineering*, 2000, pp. 263–272. 3

[21] A. Hars and S. Ou, "Working for free motivations of participating in open source software projects," *HICSS'04*, pp. 25–31, 2004. 3

[22] K. R. Lakhani and R. G. Wolf, "Why hackers do what they do: Understanding motivation and effort in free/open source software projects," MIT, Tech. Rep. 4425-03, 2003. 3, 6, 13, 16, 54

[23] S. K. Shah, "Motivation, governance, and the viability of hybrid forms in open source software development," *Management science*, vol. 52, no. 7, pp. 1000–1014, 2006. 3

[24] M. Gerosa, I. Wiese, B. Trinkenreich, G. Link, G. Robles, C. Treude, I. Steinmacher, and A. Sarma, "The shifting sands of motivation: Revisiting what drives contributors in open source," in *International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 1046–1058. 3, 6

[25] M. M. Burnett, L. Beckwith, S. Wiedenbeck, S. D. Fleming, J. Cao, T. H. Park, V. Grigoreanu, and K. Rector, "Gender pluralism in problem-solving software," *Interacting with computers*, vol. 23, no. 5, pp. 450–460, 2011. 3

[26] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in GitHub: transparency and collaboration in an open software repository," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, ser. CSCW. ACM, 2012, pp. 1277–1286. 3, 9, 10, 14, 18, 32, 33, 52

[27] A. Trockman, S. Zhou, C. Kästner, and B. Vasilescu, "Adding sparkle to social coding: An empirical study of repository badges in the npm ecosystem," in *Proceedings of the International Conference on Software Engineering*, ser. ICSE. ACM, 2018, pp. 511–522. 3, 9, 15, 17, 22, 38, 39

[28] C. Santos, G. Kuk, F. Kon, and J. Pearson, "The attraction of contributors in free and open source software projects," *The Journal of Strategic Information Systems*, vol. 22, no. 1, pp. 26–45, 2013. 3

[29] M. Burnett, S. Stumpf, J. Macbeth, S. Makri, L. Beckwith, I. Kwan, A. Peters, and W. Jernigan, "GenderMag: A method for evaluating software's gender inclusiveness," *Interacting with Computers*, vol. 28, no. 6, pp. 760–787, 2016. 3, 39, 92

[30] I. Steinmacher, A. P. Chaves, T. U. Conte, and M. A. Gerosa, "Preliminary empirical identification of barriers faced by newcomers to open source software projects," in *2014 Brazilian Symposium on Software Engineering*. IEEE, 2014, pp. 51–60. 3

[31] I. Steinmacher, T. Conte, M. A. Gerosa, and D. Redmiles, "Social barriers faced by newcomers placing their first contribution in open source software projects," in *Proceedings of the 18th ACM conference on Computer supported cooperative work & social computing*, 2015, pp. 1379–1392. 3, 15, 62

[32] S. H. Padala, C. J. Mendez, L. F. Dias, I. Steinmacher, Z. S. Hanson, C. Hilderbrand, A. Horvath, C. Hill, L. D. Simpson, M. Burnett *et al.*, "How gender-biased tools shape newcomer experiences in oss projects," *IEEE Transactions on Software Engineering*, 2020. 3, 5

[33] A. Foundjem, E. E. Eghan, and B. Adams, "Onboarding vs. diversity, productivity, and quality-empirical study of the openstack ecosystem," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 1033–1045. 4

[34] X. Tan, M. Zhou, and Z. Sun, "A first look at good first issues on github," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 398–409. 4

[35] A. Mani and R. Mukherjee, "A study of foss 2013 survey data using clustering techniques," in *2016 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*. IEEE, 2016, pp. 118–121. 4

[36] I. El Asri and N. Kerzazi, "Where are females in oss projects? socio technical interactions," in *Working Conference on Virtual Enterprises*. Springer, 2019, pp. 308–319. 4

[37] Z. Wang, Y. Wang, and D. Redmiles, "Competence-confidence gap: A threat to female developers' contribution on github," in *2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. IEEE, 2018, pp. 81–90. 4

[38] B. Vedres and O. Vasarhelyi, "Gendered behavior as a disadvantage in open source software development," *EPJ Data Science*, vol. 8, no. 1, p. 25, 2019. 4, 92

[39] A. Lee and J. C. Carver, "Floss participants' perceptions about gender and inclusiveness: a survey," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 677–687. 4, 6

[40] N. Imtiaz, J. Middleton, J. Chakraborty, N. Robson, G. Bai, and E. Murphy-Hill, "Investigating the effects of gender bias on github," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 700–711. 4

[41] G. Iaffaldano, I. Steinmacher, F. Calefato, M. Gerosa, and F. Lanubile, "Why do developers take breaks from contributing to OSS projects? a preliminary analysis," in *Proceedings of the International Workshop on Software Health*, ser. SoHeal, 2019. 5, 13

[42] C. Miller, D. G. Widder, C. Kästner, and B. Vasilescu, "Why do people give up flossing? a study of contributor disengagement in open source," in *IFIP International Conference on Open Source Systems*. Springer, 2019, pp. 116–129. 5

[43] T. Wood, "moment().endof('term')," https://medium.com/timrwood/moment-endof-term-522d8965689, 7 2016. 5

[44] N. Lawson, "What it feels like to be an open-source maintainer," *Read the Tea Leaves. https://nolanlawson. com/2017/03/05/what-it-feels-like-to-be-an-open-source-maintainer*, 2017. 5, 62

[45] B. Cannon, A. Stacoviak, and J. Santo, "The changelog, episode 318: A call for kindness in open source," Podcast, 10 2018. 5

[46] N. Raman, M. Cao, Y. Tsvetkov, C. Kästner, and B. Vasilescu, "Stress and burnout in open source: Toward finding, understanding, and mitigating unhealthy interactions," in *International Conference on Software Engineering, New Ideas and Emerging Results*, ser. ICSE. ACM, 2020, pp. 57–60. 5, 11, 62, 63, 64, 65, 66, 67, 68, 72, 73, 79, 80, 81

[47] C. D. Egelman, E. Murphy-Hill, E. Kammer, M. M. Hodges, C. Green, C. Jaspan, and J. Lin, "Predicting developers' negative feelings about code review," in *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE, 2020, pp. 174–185. 5, 11, 62, 63, 64, 65, 66, 67, 68, 69, 72, 79, 81

[48] H. S. Qiu, A. Nolte, A. Brown, A. Serebrenik, and B. Vasilescu, "Going farther together: The impact of social capital on sustained participation in open source," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 688–699. 5, 6, 11

[49] E. Dias Canedo, H. Acco Tives, M. Bogo Marioti, F. Fagundes, and J. A. Siqueira de Cerqueira, "Barriers faced by women in software development projects," *Information*, vol. 10, no. 10, p. 309, 2019. 5

[50] GitHub, "Open source survey," https://opensourcesurvey.org/2017/, 2017, accessed: 2022-03-10. 5, 6

[51] V. Singh and W. Brandon, "Open source software community inclusion initiatives to support women participation," in *IFIP International Conference on Open Source Systems*. Springer, 2019, pp. 68–79. 5, 62

[52] V. Singh, "Women participation in open source software communities," in *Proceedings of the 13th European Conference on Software Architecture-Volume 2*, 2019, pp. 94–99. 5

[53] G. Robles, H. Scheider, I. Tretkowski, and N. Weber, "Who is doing it," *A research on Libre Software developers*, 2001. 6

[54] R. A. Ghosh, R. Glott, B. Krieger, and G. Robles, "Free/libre and open source software: Survey and study," 2002. 6

[55] S. O. Alexander Hars, "Working for free? motivations for participating in open-source projects," *International journal of electronic commerce*, vol. 6, no. 3, pp. 25–39, 2002. 6

[56] P. A. David, A. Waterman, and S. Arora, "Floss-us the free/libre/open source software survey for 2003," *Stanford Institute for Economic Policy Research, Stanford University, Stanford, CA (http://www. stanford. edu/group/floss-us/report/FLOSS-US-Report. pdf)*, 2003. 6

[57] G. Robles, L. A. Reina, J. M. González-Barahona, and S. D. Domínguez, "Women in free/libre/open source software: The situation in the 2010s," in *IFIP International Conference on Open Source Systems*. Springer, 2016, pp. 163–173. 6

[58] B. Vasilescu, D. Posnett, B. Ray, M. G. J. van den Brand, A. Serebrenik, P. Devanbu, and V. Filkov, "Gender and tenure diversity in GitHub teams," in *CHI*, 2015, pp. 3789–3798. 6, 17, 45, 54, 61, 91

[59] StackOverflow, "Developer survey results," https://insights.stackoverflow.com/survey/2017, 2017, accessed: 2022-05-01. 6

[60] M. Medeiros, B. Forest, and P. Öhberg, "The case for non-binary gender questions in surveys," *PS: Political Science & Politics*, vol. 53, no. 1, pp. 128–135, 2020. 6

[61] J. Bethlehem, "Selection bias in web surveys," *International statistical review*, vol. 78, no. 2, pp. 161–188, 2010. 6

[62] V. Kuechler, C. Gilbertson, and C. Jensen, "Gender differences in early free and open source software joining process," in *IFIP International Conference on Open Source Systems*. Springer, 2012, pp. 78–93. 6, 61

[63] B. Vasilescu, A. Capiluppi, and A. Serebrenik, "Gender, representation and online participation: A quantitative study of stackoverflow," in *2012 International Conference on Social Informatics*. IEEE, 2012, pp. 332–338. 6, 7

[64] A. Kofink, "Contributions of the under-appreciated: Gender bias in an open-source ecology," in *Companion Proceedings of the 2015 ACM SIGPLAN International Conference on Systems, Programming, Languages and Applications: Software for Humanity*, 2015, pp. 83–84. 6

[65] B. Vasilescu, A. Serebrenik, and V. Filkov, "A data set for social diversity studies of github teams," in *2015 IEEE/ACM 12th working conference on mining software repositories*. IEEE, 2015, pp. 514–517. 6

[66] E. D. Canedo, R. Bonifácio, M. V. Okimoto, A. Serebrenik, G. Pinto, and E. Monteiro, "Work practices and perceptions from women core developers in oss communities," in *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2020, pp. 1–11. 6

[67] D. Ford, A. Harkins, and C. Parnin, "Someone like me: How does peer parity influence participation of women on stack overflow?" in *2017 IEEE symposium on visual languages and human-centric computing (VL/HCC)*. IEEE, 2017, pp. 239–243. 6, 61

[68] O. Vasarhelyi and B. Vedres, "Gender typicality of behavior predicts success on creative platforms," *arXiv preprint arXiv:2103.01093*, 2021. 6, 8

[69] G. A. A. Prana, D. Ford, A. Rastogi, D. Lo, R. Purandare, and N. Nagappan, "Including everyone, everywhere: Understanding opportunities and challenges of geographic gender-inclusion in oss," *IEEE Transactions on Software Engineering*, 2021. 6

[70] D. Rossi and S. Zacchiroli, "Worldwide gender differences in public code contributions: and how they have been affected by the covid-19 pandemic," *Proceedings of the 44th International Conference on Software Engineering (ICSE 2022) - Software Engineering in Society (SEIS) Track*, 2022. 6, 9, 91

[71] E. Carsenat, "Inferring gender from names in any region, language, or alphabet," *Unpublished*, vol. 10, 2019. 7

[72] P. Sebo, "Performance of gender detection tools: a comparative study of name-to-gender inference services," *Journal of the Medical Library Association: JMLA*, vol. 109, no. 3, p. 414, 2021. 7, 9

[73] B. Vasilescu, A. Capiluppi, and A. Serebrenik, "Gender, representation and online participation: A quantitative study," *Interacting with Computers*, vol. 26, no. 5, pp. 488–511, 2014. 7

[74] S. Foga, "Asf committer diversity survey. 2016," https://cwiki.apache.org/confluence/display/COMDEV/ASF+Committer+Diversity+Survey+-+2016, 2016, accessed: 2022-01-20. 7

[75] D. I. Cortázar, "Gender-diversity analysis of the linux kernel technical contributions," https://speakerdeck.com/bitergia/gender-diversity-analysis-of-the-linux-kernel-technical-contributions?slide=48, 2016, accessed: 2022-01-20. 7

[76] M. Raissi, M. de Blanc, and S. Zacchiroli, "Preliminary report on the influence of capital in an ethical-modular project: Quantitative data from the 2016 debian survey," *Journal of Peer Production*, no. 10, pp. 1–25, 2017. 7

[77] I. E. Asri and N. Kerzazi, "Where are females in oss projects? socio technical interactions," in *Working Conference on Virtual Enterprises*. Springer, 2019, pp. 308–319. 7

[78] H. Carter and J. Groopman, "The linux foundation report on diversity, equity, and inclusion in open source," https://www.linuxfoundation.org/tools/the-2021-linux-foundation-report-on-diversity-equity-and-inclusion-in-open-source/, 2021, accessed: 2022-03-10. 7

[79] R. Dattero and S. D. Galup, "Programming languages and gender," *Communications of the ACM*, vol. 47, no. 1, pp. 99–102, 2004. 8

[80] L. Moldon, M. Strohmaier, and J. Wachs, "How gamification affects software developers: Cautionary evidence from a natural experiment on github," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 549–561. 9

[81] H. Borges and M. T. Valente, "What's in a github star? understanding repository starring practices in a social coding platform," *Journal of Systems and Software*, vol. 146, pp. 112–129, 2018. 9, 19

[82] R. Li, P. Pandurangan, H. Frluckaj, and L. Dabbish, "Code of conduct conversations in open source software projects on github," in *Proceedings of the ACM on Human-Computer Interaction*, vol. 5. ACM, 2021, pp. 1–31. 9

[83] D. Ford, M. Behroozi, A. Serebrenik, and C. Parnin, "Beyond the code itself: how programmers really look at pull requests," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. IEEE, 2019, pp. 51–60. 9

[84] G. Gousios and D. Spinellis, "Ghtorrent: Github's data from a firehose," in *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*. IEEE, 2012, pp. 12–21. 9, 30, 67, 68

[85] M. Spence, "Job market signaling," *The Quarterly Journal of Economics*, vol. 87, no. 3, pp. 355–374, 1973. 10, 17

[86] G. A. Akerlof, "The market for "lemons": Quality uncertainty and the market mechanism," in *Uncertainty in Economics*. Elsevier, 1978, pp. 235–251. 10, 17

[87] A. Zahavi, "Mate selection - a selection for a handicap," *Journal of theoretical Biology*, vol. 53, no. 1, pp. 205–214, 1975. 10, 17

[88] H. S. Qiu, Y. L. Li, S. Padala, A. Sarma, and B. Vasilescu, "The signals that potential contributors look for when choosing open-source projects," *Proceedings of the ACM on Human-Computer Interaction*, vol. 3, no. CSCW, pp. 1–29, 2019. 10, 62

[89] H. S. Qiu, B. Vasilescu, C. Kästner, C. Egelman, C. Jaspan, and E. Murphy-Hill, "Detecting interpersonal conflict in issues and code review: Cross pollinating open- and closed-source approaches," in *2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*. IEEE, 2022, pp. 41–55. 11

[90] G. Avelino, L. Passos, A. Hora, and M. T. Valente, "A novel approach for estimating truck factors," in *2016 IEEE 24th International Conference on Program Comprehension (ICPC)*. IEEE, 2016, pp. 1–10. 13

[91] C. Miller, D. Widder, C. Kästner, and B. Vasilescu, "Why do people give up FLOSSing? a study of contributor disengagement in open source," in *Proceedings of the International Conference on Open Source Systems*, ser. OSS. Springer, 2019, pp. 116–129. 13

[92] S. Krishnamurthy, "On the intrinsic and extrinsic motivation of free/libre/open source (floss) developers," *Knowledge, Technology & Policy*, vol. 18, no. 4, pp. 17–39, 2006. 13

[93] A. Hars and S. Ou, "Working for free? motivations for participating in open-source projects," *International Journal of Electronic Commerce*, vol. 6, no. 3, pp. 25–39, 2002. 13

[94] G. Hertel, S. Niedner, and S. Herrmann, "Motivation of software developers in open source projects: an internet-based survey of contributors to the Linux kernel," *Research Policy*, vol. 32, no. 7, pp. 1159–1177, 2003. 13, 42, 44

[95] J. Lerner and J. Tirole, "Some simple economics of open source," *The Journal of Industrial Economics*, vol. 50, no. 2, pp. 197–234, 2002. 13

[96] J. Donath, "Signals in social supernets," *Journal of Computer-Mediated Communication*, vol. 13, no. 1, pp. 231–251, 2007. 14, 17

[97] J. Marlow, L. Dabbish, and J. Herbsleb, "Impression formation in online peer production: activity traces and personal profiles in GitHub," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*. ACM, 2013, pp. 117–128. 14, 17, 52

[98] K. Crowston, K. Wei, J. Howison, and A. Wiggins, "Free/libre open-source software development: What we know and what we do not know," *ACM Computing Surveys (CSUR)*, vol. 44, no. 2, p. 7, 2012. 15

[99] G. Robles and J. M. Gonzalez-Barahona, "Contributor turnover in libre software projects," in *Proceedings of the International Conference on Open Source Systems (OSS)*. Springer, 2006, pp. 273–286. 15

[100] C. Jergensen, A. Sarma, and P. Wagstrom, "The onion patch: migration in open source ecosystems," in *FSE*, 2011, pp. 70–80. 15

[101] M. Foucault, M. Palyart, X. Blanc, G. C. Murphy, and J.-R. Falleri, "Impact of developer turnover on quality in open-source software," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, 2015, pp. 829–841. 15, 46, 49

[102] B. Lin, G. Robles, and A. Serebrenik, "Developer turnover in global, industrial open source projects: Insights from applying survival analysis," in *2017 IEEE 12th International Conference on Global Software Engineering (ICGSE)*. IEEE, 2017, pp. 66–75. 15

[103] H. S. Qiu, A. Nolte, A. Brown, A. Serebrenik, and B. Vasilescu, "Going farther together: The impact of social capital on sustained participation in open source," in *Proceedings of the International Conference on Software Engineering*, ser. ICSE. IEEE, 2019, pp. 688–699. 15

[104] I. Steinmacher, T. U. Conte, C. Treude, and M. A. Gerosa, "Overcoming open source project entry barriers with a portal for newcomers," in *Proceedings of the International Conference on Software Engineering (ICSE)*. ACM, 2016, pp. 273–284. 15

[105] I. Steinmacher, G. Pinto, I. S. Wiese, and M. A. Gerosa, "Almost there: A study on quasi-contributors in open source software projects," in *ICSE*, 2018, pp. 256–266. 15, 24, 25, 26

[106] H. Zhu, A. Zhang, J. He, R. E. Kraut, and A. Kittur, "Effects of peer feedback on contribution: a field experiment in Wikipedia," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*. ACM, 2013, pp. 2253–2262. 16

[107] I. Steinmacher, M. A. Gerosa, and D. Redmiles, "Attracting, onboarding, and retaining newcomer developers in open source software projects," in *Workshop on Global Software Development in a CSCW Perspective*, 2014. 16, 20

[108] G. J. Link and D. Jeske, "Understanding organization and open source community relations through the attraction-selection-attrition model," in *Proceedings of the International Symposium on Open Collaboration (OpenSym)*. ACM, 2017, p. 17. 16

[109] M. Spence, "Signaling in retrospect and the informational structure of markets," *American Economic Review*, vol. 92, no. 3, pp. 434–459, 2002. 16

[110] J. Marlow and L. Dabbish, "Activity traces and signals in software developer recruitment and hiring," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*. ACM, 2013, pp. 145–156. 17, 42

[111] B. Vasilescu, V. Filkov, and A. Serebrenik, "Perceptions of diversity on GitHub: A user survey," in *Proceedings of the International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 2015, pp. 50–56. 17

[112] A. Kirmani and A. R. Rao, "No pain, no gain: A critical review of the literature on signaling unobservable product quality," *Journal of Marketing*, vol. 64, no. 2, pp. 66–79, 2000. 17

[113] C. A. Lampe, N. Ellison, and C. Steinfield, "A familiar Face(book): profile elements as signals in an online social network," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*. ACM, 2007, pp. 435–444. 17

[114] S. Bakhshi, P. Kanuparthy, and D. A. Shamma, "Understanding online reviews: Funny, cool or useful?" in *Proceedings of the ACM Conference on Computer Supported Cooperative Work & Social Computing*, ser. CSCW. ACM, 2015, pp. 1270–1276. 17

[115] C. M. Liu and J. S. Donath, "Urbanhermes: social signaling with electronic fashion," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*. ACM, 2006, pp. 885–888. 17

[116] B. C. Collier and R. Hampshire, "Sending mixed signals: Multilevel reputation effects in peer-to-peer lending markets," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work & Social Computing*, ser. CSCW. ACM, 2010, pp. 197–206. 17

[117] X. Ma, J. T. Hancock, K. Lim Mingjie, and M. Naaman, "Self-disclosure and perceived trustworthiness of Airbnb host profiles," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work & Social Computing*, ser. CSCW. ACM, 2017, pp. 2397–2409. 17

[118] T. Guilford and M. S. Dawkins, "Receiver psychology and the evolution of animal signals," *Animal Behaviour*, vol. 42, no. 1, pp. 1–14, 1991. 17, 38

[119] N. S. Shami, K. Ehrlich, G. Gay, and J. T. Hancock, "Making sense of strangers' expertise from signals in digital artifacts," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*. ACM, 2009, pp. 69–78. 17

[120] A. Zahavi and A. Zahavi, *The handicap principle: a missing piece of Darwin's puzzle*. Oxford University Press, 1999. 17

[121] J. Sheoran, K. Blincoe, E. Kalliamvakou, D. Damian, and J. Ell, "Understanding watchers on GitHub," in *Proceedings of the International Conference on Mining Software Repositories (MSR)*. ACM, 2014, pp. 336–339. 17, 52

[122] M. J. Lee, B. Ferwerda, J. Choi, J. Hahn, J. Y. Moon, and J. Kim, "GitHub developers use rockstars to overcome overflow of news," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*. ACM, 2013, pp. 133–138. 17

[123] K. Blincoe, J. Sheoran, S. Goggins, E. Petakovic, and D. Damian, "Understanding the popular users: Following, affiliation influence and leadership on GitHub," *Information and Software Technology*, vol. 70, pp. 30–39, 2016. 17

[124] A. Capiluppi, A. Serebrenik, and L. Singer, "Assessing technical candidates on the social web," *IEEE Software*, vol. 30, no. 1, pp. 45–51, 2013. 17

[125] F. Fronchetti, I. Wiese, G. Pinto, and I. Steinmacher, "What attract newcomers to onboard on OSS projects? TL;DR: Popularity," in *Proceedings of the International Conference on Open Source Systems (OSS)*. Springer, 2019, pp. 91–103. 18, 33

[126] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, "Selecting empirical methods for software engineering research," *Guide to Advanced Empirical Software Engineering*, pp. 285–311, 2008. 20, 46

[127] K. A. Ericsson and H. A. Simon, "Verbal reports as data." *Psychological Review*, vol. 87, no. 3, p. 215, 1980. 20

[128] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "The promises and perils of mining GitHub," in *MSR*, 2014, pp. 92–101. 22, 37, 53, 60

[129] A. Strauss and J. M. Corbin, *Basics of qualitative research: Grounded theory procedures and techniques*. Sage Publications, Inc, 1990. 23

[130] A. Begel, J. Bosch, and M.-A. Storey, "Social networking meets software development: Perspectives from GitHub, MSDN, Stack Exchange, and Topcoder," *IEEE Software*, no. 1, pp. 52–66, 2013. 23

[131] H. Hata, T. Todo, S. Onoue, and K. Matsumoto, "Characteristics of sustainable OSS projects: A theoretical and empirical study," in *Proceedings of the International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, 2015, pp. 15–21. 23

[132] K. Yamashita, Y. Kamei, S. McIntosh, A. E. Hassan, and N. Ubayashi, "Magnet or sticky? measuring project characteristics from the perspective of developer attraction and retention," *Journal of Information Processing*, vol. 24, no. 2, pp. 339–348, 2016. 26

[133] P. Tourani, B. Adams, and A. Serebrenik, "Code of conduct in open source projects," in *2017 IEEE 24th international conference on software analysis, evolution and reengineering (SANER)*. IEEE, 2017, pp. 24–33. 29, 62

[134] C. Danescu-Niculescu-Mizil, M. Sudhof, D. Jurafsky, J. Leskovec, and C. Potts, "A computational approach to politeness with application to social factors," *meeting of the association for computational linguistics*, vol. 1, pp. 250–259, 2013. 31

[135] A. Gelman and J. Hill, *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press, 2006. 30, 53

[136] J. Cohen, P. Cohen, S. G. West, and L. S. Aiken, *Applied multiple regression/correlation analysis for the behavioral sciences*. Routledge, 2013. 30, 53

[137] M. R. Veall and K. F. Zimmermann, "Pseudo-R2 measures for some common limited dependent variable models," *Journal of Economic Surveys*, vol. 10, no. 3, pp. 241–259, 1996. 30

[138] R. Blundell and J. L. Powell, "Endogeneity in nonparametric and semiparametric regression models," *Econometric Society Monographs*, vol. 36, pp. 312–357, 2003. 36

[139] W. Abdallah, M. Goergen, and N. O'Sullivan, "Endogeneity: How failure to correct for it can cause wrong inferences and some remedies," *British Journal of Management*, vol. 26, no. 4, pp. 791–804, 2015. 36

[140] J. A. Hausman, "Specification tests in econometrics," *Econometrica: Journal of the Econometric Society*, pp. 1251–1271, 1978. 37

[141] L. R. James and B. K. Singh, "An introduction to the logic, assumptions, and basic analytic procedures of two-stage least squares." *Psychological Bulletin*, vol. 85, no. 5, p. 1104, 1978. 37

[142] E. M. Foster, "Instrumental variables for logistic regression: an illustration," *Social Science Research*, vol. 26, no. 4, pp. 487–504, 1997. 37

[143] R. W. Blundell and J. L. Powell, "Endogeneity in semiparametric binary response models," *The Review of Economic Studies*, vol. 71, no. 3, pp. 655–679, 2004. 37

[144] J. V. Terza, A. Basu, and P. J. Rathouz, "Two-stage residual inclusion estimation: addressing endogeneity in health econometric modeling," *Journal of Health Economics*, vol. 27, no. 3, pp. 531–543, 2008. 37

[145] J. Abrevaya, J. A. Hausman, and S. Khan, "Testing for causal effects in a generalized regression model with endogenous regressors," *Econometrica*, vol. 78, no. 6, pp. 2043–2061, 2010. 37

[146] Z. Guo and D. S. Small, "Control function instrumental variable estimation of nonlinear causal effect models," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 3448–3482, 2016. 37

[147] W. H. Greene, *Econometric analysis*. Pearson Education India, 2003. 37

[148] M. L. Johnson, W. Crown, B. C. Martin, C. R. Dormuth, and U. Siebert, "Good research practices for comparative effectiveness research: Analytic methods to improve causal inference from nonrandomized studies of treatment effects using secondary data sources: The ISPOR Good Research Practices for Retrospective Database Analysis Task Force Report—Part III," *Value in Health*, vol. 12, no. 8, pp. 1062–1073, 2009. 37

[149] J. A. Espinosa, J. N. Cummings, and C. Pickering, "Time separation, coordination, and performance in technical teams," *IEEE Transactions on Engineering Management*, vol. 59, no. 1, pp. 91–103, 2011. 37

[150] M. Ketokivi and C. N. McIntosh, "Addressing the endogeneity dilemma in operations management research: Theoretical, empirical, and pragmatic considerations," *Journal of Operations Management*, vol. 52, pp. 1–14, 2017. 37

[151] C. Gibbs, D. Guttentag, U. Gretzel, J. Morton, and A. Goodwill, "Pricing in the sharing economy: a hedonic pricing model applied to Airbnb listings," *Journal of Travel & Tourism Marketing*, vol. 35, no. 1, pp. 46–56, 2018. 37

[152] H. Zhu, R. Kraut, and A. Kittur, "Effectiveness of shared leadership in online communities," in *Proceedings of the ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW)*. ACM, 2012, pp. 407–416. 37

[153] E. Oster, "Unobservable selection and coefficient stability: Theory and evidence," *Journal of Business & Economic Statistics*, vol. 37, no. 2, pp. 187–204, 2019. 37

[154] D. Gachechiladze, F. Lanubile, N. Novielli, and A. Serebrenik, "Anger and its direction in collaborative software development," in *2017 IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Technologies Results Track (ICSE-NIER)*. IEEE, 2017, pp. 11–14. 38, 62, 64

[155] C. Danescu-Niculescu-Mizil, M. Sudhof, D. Jurafsky, J. Leskovec, and C. Potts, "A computational approach to politeness with application to social factors," *arXiv preprint arXiv:1306.6078*, 2013. 38, 64, 70, 72

[156] N. Novielli, F. Calefato, and F. Lanubile, "The challenges of sentiment detection in the social programmer ecosystem," in *Proceedings of the International Workshop on Social Software Engineering (SSE)*. ACM, 2015, pp. 33–40. 38

[157] R. Jongeling, P. Sarkar, S. Datta, and A. Serebrenik, "On negative results when using sentiment analysis tools for software engineering research," *Empirical Software Engineering*, vol. 22, no. 5, pp. 2543–2584, 2017. 38, 65, 80

[158] B. Lin, F. Zampetti, G. Bavota, M. Di Penta, M. Lanza, and R. Oliveto, "Sentiment analysis for software engineering: How far can we go?" in *Proceedings of the International Conference on Software Engineering (ICSE)*. ACM, 2018, pp. 94–104. 38

[159] C. Mendez, H. S. Padala, Z. Steine-Hanson, C. Hilderbrand, A. Horvath, C. Hill, L. Simpson, N. Patil, A. Sarma, and M. Burnett, "Open source barriers to entry, revisited: A sociotechnical perspective," in *Proceedings of the International Conference on Software Engineering (ICSE)*. ACM, 2018, pp. 1004–1015. 39

[160] Y. Fang and D. Neufeld, "Understanding sustained participation in open source software projects," *J Manage Inform Syst*, vol. 25, no. 4, pp. 9–50, 2009. 42, 45

[161] J. A. Roberts, I.-H. Hann, and S. A. Slaughter, "Understanding the motivations, participation, and performance of open source software developers: A longitudinal study of the apache projects," *Management science*, vol. 52, no. 7, pp. 984–999, 2006. 42

[162] B. Lin, G. Robles, and A. Serebrenik, "Developer turnover in global, industrial open source projects: Insights from applying survival analysis," in *ICGSE*, 2017, pp. 66–75. 42, 46, 49

[163] A. Schilling, S. Laumer, and T. Weitzel, "Who will remain? an evaluation of actual person-job and person-team fit to predict developer retention in floss projects," in *2012 45th Hawaii International Conference on System Sciences*. IEEE, 2012, pp. 3446–3455. 42

[164] Y. Jiang, B. Adams, and D. M. German, "Will my patch make it? and how fast?: Case study on the Linux kernel," in *MSR*, 2013, pp. 101–110. 42

[165] V. J. Hellendoorn, P. T. Devanbu, and A. Bacchelli, "Will they like this?: Evaluating code contributions with language models," in *MSR*, 2015, pp. 157–167. 42

[166] R. Padhye, S. Mani, and V. S. Sinha, "A study of external community contribution to open-source projects on GitHub," in *MSR*, 2014, pp. 332–335. 42

[167] Y. Tao, D. Han, and S. Kim, "Writing acceptable patches: An empirical study of open source project patches," in *ICSME*, 2014, pp. 271–280. 42

[168] G. Gousios, M. Pinzger, and A. v. Deursen, "An exploratory study of the pull-based software development model," in *ICSE*, 2014, pp. 345–355. 42

[169] R. S. Burt, "The gender of social capital," *Rationality and Society*, vol. 10, no. 1, pp. 5–46, 1998. 42, 45

[170] ——, "Structural holes versus network closure as social capital," in *Social Capital: Theory and Research*. De Gruyter, 2001, pp. 31–56. 42, 44

[171] C.-M. Chiu, M.-H. Hsu, and E. T. Wang, "Understanding knowledge sharing in virtual communities: An integration of social capital and social cognitive theories," *Decision Support Systems*, vol. 42, no. 3, pp. 1872–1888, 2006. 42

[172] M. B. Aguilera, "The impact of social capital on labor force participation: Evidence from the 2000 social capital benchmark survey," *Social Science Quarterly*, vol. 83, no. 3, pp. 853–874, 2002. 42

[173] E. Brown and J. M. Ferris, "Social capital and philanthropy: An analysis of the impact of social capital on individual giving and volunteering," *Nonprof Volunt Sec Q*, vol. 36, no. 1, pp. 85–99, 2007. 42

[174] L. Guiso, P. Sapienza, and L. Zingales, "The role of social capital in financial development," *American Economic Review*, vol. 94, no. 3, pp. 526–556, June 2004. 42

[175] J. Hahn, J. Y. Moon, and C. Zhang, "Emergence of new project teams from open source software developer networks: Impact of prior collaboration ties," *Information Systems Research*, vol. 19, no. 3, pp. 369–391, 2008. 43, 45

[176] C. Casalnuovo, B. Vasilescu, P. Devanbu, and V. Filkov, "Developer onboarding in GitHub: The role of prior social links and language experience," in *ESEC/FSE*, 2015, pp. 817–828. 43, 45, 50

[177] M. Lutter, "Do women suffer from network closure? the moderating effect of social capital on gender inequality in a project-based labor market, 1929 to 2010," *American Sociological Review*, vol. 80, no. 2, pp. 329–358, 2015. 43, 45, 50, 51

[178] C. Mendez, H. S. Padala, Z. Steine-Hanson, C. Hilderbrand, A. Horvath, C. Hill, L. Simpson, N. Patil, A. Sarma, and M. Burnett, "Open source barriers to entry, revisited: A sociotechnical perspective," in *ICSE*, 2018, pp. 1004–1015. 44, 45, 59

[179] P. S. Adler and S.-W. Kwon, "Social capital: Prospects for a new concept," *Acad Manage Rev*, vol. 27, no. 1, pp. 17–40, 2002. 44, 45

[180] G. Madey, V. Freeh, and R. Tynan, "The open source software development phenomenon: An analysis based on social network theory," in *AMCIS*, 2002, pp. 1806–1813. 44, 45

[181] J. Wang, "The role of social capital in open source software communities," *AMCIS*, p. 427, 2005. 44

[182] J. S. Coleman, *Foundations of social theory*. Belknap, 1990. 44

[183] M. S. Granovetter, "The strength of weak ties," *Am J Sociol*, vol. 78, no. 6, pp. 1360–1380, 1973. 44

[184] W. E. Baker and A. V. Iyer, "Information networks and market behavior," *Journal of Mathematical Sociology*, vol. 16, no. 4, pp. 305–332, 1992. 44

[185] B. Xu and D. R. Jones, "Volunteers' participation in open source software development: a study from the social-relational perspective," *ACM SIGMIS Database*, vol. 41, no. 3, pp. 69–84, 2010. 44

[186] W. Oh and S. Jeon, "Membership herding and network stability in the open source community: The Ising perspective," *Management Science*, vol. 53, no. 7, pp. 1086–1101, 2007. 44

[187] P. Song and C. W. Phang, "Promoting continuance through shaping members' social identity in knowledge-based versus support/advocacy virtual communities," *IEEE T Eng Manage*, vol. 63, no. 1, pp. 16–26, 2016. 44

[188] S. Toral, M. Martínez-Torres, and F. Barrero, "Analysis of virtual communities supporting OSS projects using social network analysis," *Inform Software Tech*, vol. 52, no. 3, pp. 296–303, 2010. 45

[189] S. Christopherson, "Working in the creative economy: Risk, adaptation and the persistence of exclusionary networks," *Creative labour: Working in the creative industries*, pp. 72–90, 2009. 45

[190] I. Grugulis and D. Stoyanova, "Social capital and networks in film and tv: Jobs for the boys?" *Organization Studies*, vol. 33, no. 10, pp. 1311–1331, 2012. 45

[191] H. Blair, "Active networking: action, social structure and the process of networking," *Creative Labour: Working in the Creative Industries*, pp. 116–134, 2009. 45

[192] A. Portes, "Social capital: Its origins and applications in modern sociology," *Annual Review of Sociology*, vol. 24, no. 1, pp. 1–24, 1998. 45

[193] B. Groysberg, *Chasing Stars: The Myth of Talent and the Portability of Performance*. Princeton University Press, 2010. 45

[194] K. Finley, "Diversity in open source is even worse than in tech overall," https://www.wired.com/2017/06/diversity-open-source-even-worse-tech-overall/. 45

[195] R. S. Geiger, "Summary analysis of the 2017 github open source survey," *arXiv preprint arXiv:1706.02777*, 2017. 45, 54, 60, 62

[196] N. Lin, *Social Capital: A Theory of Social Structure and Action*. Cambridge University Press, 2001. 45

[197] B. Vasilescu, V. Filkov, and A. Serebrenik, "Perceptions of diversity on GitHub: A user survey," in *CHASE*, 2015, pp. 50–56. 45, 54, 57

[198] D. Ford, J. Smith, P. J. Guo, and C. Parnin, "Paradise unplugged: identifying barriers for female participation on Stack Overflow," in *FSE*, 2016, pp. 846–857. 45

[199] L. López-Fernández, G. Robles, and J. González-Barahona, "Applying social network analysis to the information in CVS repositories," in *MSR*, 2004, pp. 101–105. 45

[200] H.-L. Yang and J.-H. Tang, "Team structure and team performance in is development: A social network perspective," *Inform Manage*, vol. 41, no. 3, pp. 335–349, 2004. 45

[201] K. Ehrlich and K. Chang, "Leveraging expertise in global software teams: Going outside boundaries," in *ICGSE*, 2006, pp. 149–158. 45

[202] S. Toral, M. Martínez-Torres, and F. Barrero, "Analysis of virtual communities supporting oss projects using social network analysis," *Inf Sw Tech*, vol. 52, no. 3, pp. 296–303, 2010. 45

[203] D. A. Tamburri, P. Lago, and H. van Vliet, "Uncovering latent social communities in software development," *IEEE Software*, vol. 30, no. 1, pp. 29–36, Jan 2013. 45, 50

[204] M. Zhou and A. Mockus, "What make long term contributors: Willingness and opportunity in oss community," in *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, 2012, pp. 518–528. 46

[205] G. Gousios, "The GHTorrent dataset and tool suite," in *MSR*, 2013, pp. 233–236. 46

[206] I. S. Wiese, J. T. da Silva, I. Steinmacher, C. Treude, and M. A. Gerosa, "Who is who in the mailing list? comparing six disambiguation heuristics to identify multiple addresses of a participant," in *ICSME*, 2016, pp. 345–355. 46

[207] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan, "Mining email social networks," in *MSR*, 2006, pp. 137–143. 46

[208] B. Vasilescu, A. Serebrenik, M. Goeminne, and T. Mens, "On the variation and specialisation of workload—a case study of the gnome ecosystem community," *Empirical Software Engineering*, vol. 19, no. 4, pp. 955–1008, 2014. 46

[209] B. Lin and A. Serebrenik, "Recognizing gender of Stack Overflow users," in *MSR*, 2016, pp. 425–429. 47, 55

[210] F. Karimi, C. Wagner, F. Lemmerich, M. Jadidi, and M. Strohmaier, "Inferring gender from names on the web: A comparative evaluation of gender detection methods," in *WWW Companion*, 2016, pp. 53–54. 47, 48

[211] F. Rangel, P. Rosso, M. Potthast, and B. Stein, "Overview of the 5th author profiling task at pan 2017: Gender and language variety identification in twitter," *Working Notes Papers of the CLEF*, 2017. 47

[212] B. Vasilescu, A. Capiluppi, and A. Serebrenik, "Gender, representation and online participation: A quantitative study," *Interact Comput*, vol. 26, no. 5, pp. 488–511, 2013. 47, 55, 61

[213] R. G. Miller Jr, *Survival analysis*. Wiley, 2011, vol. 66. 49, 52

[214] M. De Vaan, B. Vedres, and D. Stark, "Disruptive diversity and recurring cohesion: Assembling creative teams in the video game industry, 1979–2009," Institute for Social and Economic Research and Policy, Tech. Rep. 3, 2011. 50, 51

[215] M. E. J. Newman, "Scientific collaboration networks II. Shortest paths, weighted networks, and centrality," *Physical Review E*, vol. 64, no. 1, pp. 016 132:1–7, 2001. 50

[216] F. Perretti and G. Negro, "Mixing genres and matching people: a study in innovation and team composition in Hollywood," *J Organ Behav*, vol. 28, no. 5, pp. 563–586, 2007. 51

[217] S. Rodan and C. Galunic, "More than network structure: How knowledge heterogeneity influences managerial performance and innovativeness," *Strategic management journal*, vol. 25, no. 6, pp. 541–562, 2004. 51

[218] C. Zapponi, "Programming languages and github," http://githut.info/, 2017, visited 29 June 2017. 51

[219] B. Vasilescu, A. Serebrenik, and M. G. J. van den Brand, "The Babel of software development: Linguistic diversity in open source," in *SocInfo*, 2013, pp. 391–404. 52

[220] C. Bird, N. Nagappan, B. Murphy, H. Gall, and P. Devanbu, "Don't touch my code!: examining the effects of ownership on software quality," in *FSE*, 2011, pp. 4–14. 52

[221] G. Pinto, I. Steinmacher, and M. A. Gerosa, "More common than you think: An in-depth study of casual contributors," in *SANER*, 2016, pp. 112–123. 53

[222] J. K. Patel, C. Kapadia, and D. B. Owen, *Handbook of statistical distributions*. M. Dekker, 1976. 53

[223] P. M. Grambsch and T. M. Therneau, "Proportional hazards tests and diagnostics based on weighted residuals," *Biometrika*, vol. 81, no. 3, pp. 515–526, 1994. 53

[224] A. Filippova, E. Trainer, and J. D. Herbsleb, "From diversity by numbers to diversity as process: supporting inclusiveness in software development teams with brainstorming," in *ICSE*, 2017, pp. 152–163. 53

[225] W. B. Schaufeli, A. B. Bakker, and M. Salanova, "The measurement of work engagement with a short questionnaire: A cross-national study," *Educ Psychol Meas*, vol. 66, no. 4, pp. 701–716, 2006. 53

[226] N. B. Ellison, C. Steinfield, and C. Lampe, "The benefits of facebook "friends:" social capital and college students' use of online social network sites," *J Comput-Mediat Comm*, vol. 12, no. 4, pp. 1143–1168, 2007. 53, 59

[227] J. Drengner, S. Jahn, and H. Gaus, "Events and loyalty formation: The role of satisfaction, felt community, emotional experience, and frequency of use," in *Stand und Perspektiven der Eventforschung*. Wiesbaden: Gabler, 2010, pp. 151–165. 53

[228] W. B. Schaufeli and A. B. Bakker, "Job demands, job resources, and their relationship with burnout and engagement: a multi-sample study," *J Organ Behav*, vol. 25, no. 3, pp. 293–315, 2004. 53

[229] J. Siegmund, C. Kästner, J. Liebig, S. Apel, and S. Hanenberg, "Measuring and modeling programming experience," *Empir Softw Eng*, vol. 19, no. 5, pp. 1299–1334, 2014. 54

[230] Y. Wang and D. R. Fesenmaier, "Modeling participation in an online travel community," *J Travel Res*, vol. 42, no. 3, pp. 261–270, 2004. 54

[231] M. Razavian and P. Lago, "Feminine expertise in architecting teams," *IEEE Software*, vol. 33, no. 4, pp. 64–71, 2016. 56

[232] J. Cohen, "Statistical power analysis for the behavioral sciences," 1988. 56

[233] Z. Wang, Y. Wang, and D. Redmiles, "Competence-confidence gap: A threat to female developers' contribution on GitHub," in *ICSE*, 2018, pp. 81–90. 56, 60

[234] A. Lee, J. C. Carver, and A. Bosu, "One-time contributors to FLOSS: surveys and data analysis," in *ICSE*, 2017, pp. 187–197. 57

[235] S. Elder and L. J. Johnson, "Sex-specific labour market indicators: What they show," *Int'l Labour Review*, vol. 138, no. 4, pp. 447–464, 2008. 57

[236] G. Robles, L. A. Reina, J. M. González-Barahona, and S. D. Domínguez, "Women in free/libre/open source software: The situation in the 2010s," in *Open Source Systems: Integrating Communities*, 2016, pp. 163–173. 57

[237] J. Romano, J. D. Kromrey, J. Skowronek, and L. Devine, "Exploring methods for evaluating group differences on the NSSE and other surveys: Are the t-test and Cohen's d indices the most appropriate choices?" in *Ann. meeting, South Assoc Institutional Research*, 2006, pp. 1–51. 57

[238] N. Eghbal, *Roads and bridges: The unseen labor behind our digital infrastructure.* Ford Foundation, 2016. 59

[239] A. Trockman, S. Zhou, C. Kästner, and B. Vasilescu, "Adding sparkle to social coding: An empirical study of repository badges in the npm ecosystem," in *ICSE*, 2018. 59

[240] A. Obadimu, E. Mead, M. N. Hussain, and N. Agarwal, "Identifying toxicity within youtube video comment," in *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation.* Springer, 2019, pp. 214–223. 62

[241] L. Aroyo, L. Dixon, N. Thain, O. Redfield, and R. Rosen, "Crowdsourcing subjective tasks: the case study of understanding toxicity in online discussions," in *Companion proceedings of the 2019 world wide web conference*, 2019, pp. 1100–1105. 62

[242] J. Pavlopoulos, J. Sorensen, L. Dixon, N. Thain, and I. Androutsopoulos, "Toxicity detection: Does context really matter?" *arXiv preprint arXiv:2006.00998*, 2020. 62

[243] J. Lehnardt, "Sustainable open source: The maintainers perspective or: How i learned to stop caring and love open source," https://writing.jan.io/2017/03/06/sustainable-open-source-the-maintainers-perspective-or-how-i-learned-to-stop-caring-and-love-open-so html, 2017, [Online; accessed 19-July-2021]. 62

[244] M. Consalvo, "Confronting toxic gamer culture: A challenge for feminist game studies scholars," 2012. 62

[245] S. Wachs, M. F. Wright, and A. T. Vazsonyi, "Understanding the overlap between cyberbullying and cyberhate perpetration: Moderating effects of toxic online disinhibition," *Criminal Behaviour and Mental Health*, vol. 29, no. 3, pp. 179–188, 2019. 62

[246] N. A. Beres, J. Frommel, E. Reid, R. L. Mandryk, and M. Klarkowski, "Don't you know that you're toxic: Normalization of toxicity in online gaming," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–15. 62

[247] M. Squire and R. Gazda, "Floss as a source for profanity and insults: Collecting the data," in *2015 48th Hawaii International Conference on System Sciences.* IEEE, 2015, pp. 5290–5298. 62

[248] D. Schneider, S. Spurlock, and M. Squire, "Differentiating communication styles of leaders on the linux kernel mailing list," in *Proceedings of the 12th International Symposium on Open Collaboration*, 2016, pp. 1–10. 62

[249] I. Ferreira, K. Stewart, D. German, and B. Adams, "A longitudinal study on the maintainers' sentiment of a large scale open source ecosystem," in *2019 IEEE/ACM 4th International Workshop on Emotion Awareness in Software Engineering (SEmotion)*. IEEE, 2019, pp. 17–22. 62

[250] I. El Asri, N. Kerzazi, G. Uddin, F. Khomh, and M. J. Idrissi, "An empirical study of sentiments in code reviews," *Information and Software Technology*, vol. 114, pp. 37–54, 2019. 62

[251] M. Ortu, B. Adams, G. Destefanis, P. Tourani, M. Marchesi, and R. Tonelli, "Are bullies more productive? empirical study of affectiveness vs. issue fixing time," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. IEEE, 2015, pp. 303–313. 62

[252] R. Paul, A. Bosu, and K. Z. Sultana, "Expressions of sentiments during code reviews: Male vs. female," in *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2019, pp. 26–37. 62, 68

[253] A. Alami, M. L. Cohn, and A. Wąsowski, "Why does code review work for open source software communities?" in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 1073–1083. 62

[254] S. Cohen, "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*, 2021, pp. 1669–1671. 62

[255] C. Miller, S. Cohen, D. Klug, B. Vasilescu, and C. Kästner, ""did you miss my comment or what?" understanding toxicity in open source discussions," in *International Conference on Software Engineering (ICSE)*. ACM, 2022. 62

[256] M. Chouchen, A. Ouni, R. G. Kula, D. Wang, P. Thongtanunam, M. W. Mkaouer, and K. Matsumoto, "Anti-patterns in modern code review: Symptoms and prevalence," in *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2021, pp. 531–535. 62

[257] J. Sarker, A. K. Turzo, and A. Bosu, "A benchmark study of the contemporary toxicity detectors on software engineering interactions," in *2020 27th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2020, pp. 218–227. 62, 63

[258] J. Cheriyan, B. T. R. Savarimuthu, and S. Cranefield, "Norm violation in online communities–a study of stack overflow comments," in *Coordination, Organizations, Institutions, Norms, and Ethics for Governance of Multi-Agent Systems XIII*. Springer, 2017, pp. 20–34. 62

[259] D. Jurgens, E. Chandrasekharan, and L. Hemphill, "A just and comprehensive strategy for using nlp to address online abuse," in *57th Annual Meeting of the Association for Computational Linguistics, ACL 2019*. Association for Computational Linguistics (ACL), 2020, pp. 3658–3666. 62

[260] S. Kiritchenko, I. Nejadgholi, and K. C. Fraser, "Confronting abusive language online: A survey from the ethical and human rights perspective," *Journal of Artificial Intelligence Research*, vol. 71, pp. 431–478, 2021. 62

[261] J. Cheriyan, B. T. R. Savarimuthu, and S. Cranefield, "Towards offensive language detection and reduction in four software engineering communities," in *Evaluation and Assessment in Software Engineering*, 2021, pp. 254–259. 63, 80

[262] E. Wulczyn, N. Thain, and L. Dixon, "Ex machina: Personal attacks seen at scale," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 1391–1399. 63

[263] A. Schmidt and M. Wiegand, "A survey on hate speech detection using natural language processing," in *Proceedings of the fifth international workshop on natural language processing for social media*, 2017, pp. 1–10. 63

[264] M. Dadvar, D. Trieschnigg, R. Ordelman, and F. de Jong, "Improving cyberbullying detection with user context," in *European Conference on Information Retrieval*. Springer, 2013, pp. 693–696. 63

[265] G. Park, H. A. Schwartz, J. C. Eichstaedt, M. L. Kern, M. Kosinski, D. J. Stillwell, L. H. Ungar, and M. E. Seligman, "Automatic personality assessment through social media language." *Journal of personality and social psychology*, vol. 108, no. 6, p. 934, 2015. 64

[266] S. O. Sood, E. F. Churchill, and J. Antin, "Automatic identification of personal insults on social news sites," *Journal of the American Society for Information Science and Technology*, vol. 63, no. 2, pp. 270–285, 2012. 64

[267] B. L. Monroe, M. P. Colaresi, and K. M. Quinn, "Fightin'words: Lexical feature selection and evaluation for identifying the content of political conflict," *Political Analysis*, vol. 16, no. 4, pp. 372–403, 2008. 64, 69, 70

[268] P. Brown and S. C. Levinson, *Politeness: Some universals in language usage*. Cambridge university press, 1987, vol. 4. 64

[269] R. Lakoff, "The logic of politeness: Or, minding your p's and q's," in *Proceedings from the Annual Meeting of the Chicago Linguistic Society*, vol. 9, no. 1. Chicago Linguistic Society, 1973, pp. 292–305. 64

[270] ——, "What you can do with words: Politeness, pragmatics and performatives," in *Proceedings of the Texas conference on performatives, presuppositions and implicatures*. ERIC, 1977, pp. 79–106. 64

[271] J. P. Chang, C. Chiam, L. Fu, A. Wang, J. Zhang, and C. Danescu-Niculescu-Mizil, "Convokit: A toolkit for the analysis of conversations," in *Proceedings of Special Interest Group on Discourse and Dialogue*, 2020. 64, 72

[272] J. Zhang, J. P. Chang, C. Danescu-Niculescu-Mizil, L. Dixon, Y. Hua, N. Thain, and D. Taraborelli, "Conversations gone awry: Detecting early signs of conversational failure," *arXiv preprint arXiv:1805.05345*, 2018. 64, 80

[273] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Comput. Linguist*, vol. 35, no. 2, pp. 311–312, 2009. 64

[274] E. Guzman, D. Azócar, and Y. Li, "Sentiment analysis of commit comments in github: an empirical study," in *Proceedings of the 11th working conference on mining software repositories*, 2014, pp. 352–355. 64

[275] F. Calefato, F. Lanubile, F. Maiorano, and N. Novielli, "Sentiment polarity detection for software development," *Empirical Software Engineering*, vol. 23, no. 3, pp. 1352–1382, 2018. 64, 65

[276] T. Zhang, B. Xu, F. Thung, S. A. Haryono, D. Lo, and L. Jiang, "Sentiment analysis for software engineering: How far can pre-trained transformer models go?" in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2020, pp. 70–80. 65, 72

[277] T. Ahmed, A. Bosu, A. Iqbal, and S. Rahimi, "SentiCR: A customized sentiment analysis tool for code review interactions," in *International Conference on Automated Software Engineering (ASE)*. IEEE, 2017, pp. 106–111. 65, 72

[278] N. Novielli, F. Calefato, and F. Lanubile, "A gold standard for emotions annotation in stack overflow," in *Proc. of 15th Int'l Conf. on Mining Software Repositories*, ser. MSR 2018, 2018, pp. 14–17. 65

[279] C. Sadowski, E. Söderberg, L. Church, M. Sipko, and A. Bacchelli, "Modern code review: a case study at google," in *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*, 2018, pp. 181–190. 67, 80

[280] G. Gousios, M. Pinzger, and A. v. Deursen, "An exploratory study of the pull-based software development model," in *Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 345–355. 67, 80

[281] M. M. Rahman, D. Balakrishnan, D. Murthy, M. Kutlu, and M. Lease, "An information retrieval approach to building datasets for hate speech detection," *arXiv preprint arXiv:2106.09775*, 2021. 67

[282] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001. 73

[283] H. Ishwaran, "Variable importance in binary regression trees and forests," *Electronic Journal of Statistics*, vol. 1, pp. 519–537, 2007. 73, 74

[284] S. J. Kazemitabar, A. A. Amini, A. Bloniarz, and A. Talwalkar, "Variable importance using decision trees," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 425–434. 73, 74

[285] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 233–240. 74

[286] A. Lees, D. Borkan, I. Kivlichan, J. Nario, and T. Goyal, "Capturing covertly toxic speech via crowdsourcing," in *Proceedings of the First Workshop on Bridging Human–Computer Interaction and Natural Language Processing*, 2021, pp. 14–20. 75

[287] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018. 80

[288] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," in *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, vol. 11, no. 3. NIH Public Access, 2017, p. 269. 80

[289] M. J. Howard, S. Gupta, L. Pollock, and K. Vijay-Shanker, "Automatically mining software-based, semantically-similar words from comment-code mappings," in *2013 10th working conference on mining software repositories (MSR)*. IEEE, 2013, pp. 377–386. 80

[290] X. Han and Y. Tsvetkov, "Fortifying toxic speech detectors against veiled toxicity," *arXiv preprint arXiv:2010.03154*, 2020. 80

[291] B. Vasilescu, A. Serebrenik, P. T. Devanbu, and V. Filkov, "How social Q&A sites are changing knowledge sharing in open source software communities," in *17th ACM Conference on Computer Supported Cooperative Work and Social Computing*, ser. CSCW. ACM, 2014, pp. 342–354. 92

[292] M. T. Rivera, S. B. Soderstrom, and B. Uzzi, "Dynamics of dyads in social networks: Assortative, relational, and proximity mechanisms," *Annual Review of Sociology*, vol. 36, no. 1, pp. 91–115, 2010. 92