Matching Skills, Past Collaboration, and Limited Competition: Modeling When Open-Source Projects Attract Contributors

Hongbo Fang
hongbofa@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, PA, USA

James Herbsleb
jdh@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, PA, USA

Bogdan Vasilescu
bogdanv@cs.cmu.edu
Carnegie Mellon University
Pittsburgh, PA, USA

ABSTRACT

Attracting and retaining new developers is often at the heart of open-source project sustainability and success. Previous research found many intrinsic (or endogenous) project characteristics associated with the attractiveness of projects to new developers, but the impact of factors external to the project itself have largely been overlooked. In this work, we focus on one such external factor, a project's *labor pool*, which is defined as the set of contributors active in the overall open-source ecosystem that the project could plausibly attempt to recruit from at a given time. How are the size and characteristics of the labor pool associated with a project's attractiveness to new contributors? Through an empirical study of over 516,893 Python projects, we found that the size of the project's labor pool, the technical skill match, and the social connection between the project's labor pool and members of the focal project all significantly influence the number of new developers that the focal project attracts, with the competition between projects with overlapping labor pools also playing a role. Overall, the labor pool factors add considerable explanatory power compared to models with only project-level characteristics.

CCS CONCEPTS

• **Software and its engineering** → **Open source model**; • **Human-centered computing** → *Empirical studies in collaborative and social computing*.

KEYWORDS

open source, sustainability, labor pool

ACM Reference Format:
Hongbo Fang, James Herbsleb, and Bogdan Vasilescu. 2023. Matching Skills, Past Collaboration, and Limited Competition: Modeling When Open-Source Projects Attract Contributors. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '23), December 3–9, 2023, San Francisco, CA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3611643.3616282

1 INTRODUCTION

The importance and economic value of open-source software are, by now, undeniable [25]. Open source is used in all domains, by companies big and small, public institutions, scientific organizations, etc. There is a wealth of open-source libraries, frameworks, and tools that can be reused and built upon to facilitate innovation and increase software development productivity [59].

Much like any software [15], open source also requires considerable effort to develop and maintain, e.g., to respond to evolving user needs [5], unexpected bugs or issues [67], and ever-changing dependencies [11]. However, since much of open source is still being developed and maintained by volunteers [7, 55], for projects, being able to attract and retain new contributors is essential [53]. Attracting and onboarding new contributors preserves the continuity of project development and maintenance, and can even save the project from being abandoned when core developers leave [4]. The arrival of new contributors may also bring new knowledge and perspectives to the project team, and thus help to produce software of higher quality [54]. Yet, open-source projects often struggle to find the right contributors [4] and this is aggravated by the voluntary nature of many open-source contributions — turnover rates are high [31], disengagement of core project developers is common [30], and the reasons people drop out are often unavoidable, e.g., a change of job or life status [48].

There has been considerable prior research to understand the characteristics of projects that succeed in attracting and onboarding new contributors, as well as the barriers faced by people placing their first contributions in open-source projects [63]. For example, there is empirical evidence that project popularity [13], age, size in terms of the number of contributors [64], license [57], the presence and quality of documentation [51], and even activity on social media [29], are all associated with a project's likelihood of attracting new contributors. However, much less is known about how factors external to the project, related to its position and role in the overall open-source ecosystem, impact the process of attracting and retaining new contributors.

Meanwhile, there is mounting evidence that open-source contributors, especially volunteers, have ample freedom to choose which projects to contribute to [34], typically join existing communities over starting new projects [34], and often jump around between projects and programming language ecosystems [20, 39, 73], bringing with them knowledge and skills. Therefore, it is insufficient to consider open-source projects as independent — their success or failure to attract and retain new contributors is likely influenced by the broader context they are part of and their relationships to other projects in the overall open-source ecosystem.

With this holistic view, in this paper we take a step towards better understanding an open-source project's **labor pool**, defined as *the set of contributors active in the overall open-source ecosystem that a given project could plausibly attempt to recruit from at a*

ESEC/FSE '23, December 3–9, 2023, San Francisco, CA, USA

ACM ISBN 979-8-4007-0327-0/23/12.
https://doi.org/10.1145/3611643.3616282

given time. Using longitudinal data from a large sample of 516,893 Python projects, and holding constant project-level characteristics previously reported in the literature, we show that 1) the level of technical skill match between potential contributors and a focal project, 2) the strength of their connection to existing team members from past collaborations, and 3) the amount of 'competition' from other similar projects in the overall ecosystem, are all statistically associated with a project's attractiveness to new contributors.

Our methodology and results can help open source practitioners, community managers, platform designers, and funders to better allocate typically already scarce resources, target promotional campaigns, and monitor the health and sustainability of projects. By illuminating, with quantitative empirical evidence, possible new mechanisms through which open-source projects can attract (or may struggle to attract) new contributors, our work also contributes to our theoretical understanding of the ecosystem-level dynamics in open source, an area in much need of additional research attention, in our opinion.

2 RELATED WORK

The sustainability of open-source projects is one of the key questions emerging since the early days of open-source development. Software development is a complex, effort-intensive process [15, 49], and the maintenance of such products can be no less costly [8]. Because open-source teams often consist of volunteer developers, being able to sustain contributions becomes especially challenging. Multiple studies have looked at how developers become involved in open-source projects. Krogh et al.proposed a "joining script" of developer participation in the open-source mailing list and described the practice it took for a newcomer to become established in the community [74]. Crowston and Howison studied an onion-like structural model of open-source teams, with the transition between different layers being a result of team interaction and developer contribution [21]. Steinmacher and others described the developer contribution to a project as a multi-stage process that starts from an initial motivation and attraction phase, then moves to the retention phase in the later stage [64]. The developers' advancement towards becoming long-term contributors was also studied. Researchers found that the developers' relative sociality [77], their attitudes [78], and the project environmental factors such as the availability of work opportunities and project popularity [78, 79] all play a role to influence the likelihood of long-term contributions. Those works helped to support more sustainable open-source development by providing guidance on tool design [6, 35, 60, 61, 66] and project or task recommendations [58].

Attracting new developers to contribute to an open-source project is an important part of the project's sustainability and the early stage of developer involvement in the project. Researchers have identified several factors that should help with attracting and onboarding new developers. For example, Hahn et al.reported that developers tend to join projects where they have past collaboration experience with the existing developers [36, 37], with a similar result found by Casalnuovo et al. [18]. Tan and others found that beginner-friendly tasks and their characteristics impact developer onboarding [68]. More recently, studies looked at attracting new contributors through the lens of signaling theory, reporting that better READMEs [51], the adoption of badges [69], the project popularity metrics (e.g., the number of stars [13, 32]), and the time to review pull requests [32] all help to make the project more attractive to new contributors.

Our work provides an alternative perspective to understanding the new contributors to a project by shifting the focus from the project itself to the characteristics of potential contributors. Our results also connect to the research on social media promotion [14, 29] for open-source projects and open-source project diffusion in general [42], as we show that the characteristics of the community that the promotion reaches impact the successful attraction of new contributors. We further identify several such characteristics that open-source promoters may pay attention to.

3 THEORETICAL FRAMEWORK AND RESEARCH HYPOTHESES

Our study investigates the relationship between characteristics of a project's labor pool and its ability to attract new contributors. As mentioned above, we define *labor pool* as the set of developers active in the overall open-source ecosystem around the same time, that a given project could *plausibly* attempt to recruit new contributors from. That is, in our definition labor pools are always *tied to specific projects*. For example, we expect that a text processing project's labor pool is different from a bioinformatics project's labor pool, although the two may overlap if, for example, they involve applications of machine learning. In addition, we require a notion of *plausible awareness* of the focal project from developers in the labor pool. Indeed, at the very least, one would need to be aware of a project and possible opportunities to contribute before deciding to do so. We discuss our operationalization later, in Section 4.1.

With this definition, we hypothesize about differences in tendencies to join (i.e., start contributing changes to) a focal project for developers in the labor pool, and the relationships (we theorize the direction of influence as well) between such factors external to a project and the project's success at attracting new contributors. We will operationalize and more formally test these hypotheses below, in Section 5. Note also that we use the terms *developer* and *contributor* interchangeably, and inclusively of all types of contribution, not just code.[1]

To begin with, the size of the project's labor pool at some point in time should be an important predictor of the number of new contributors the project engages in the near future. A larger labor pool size should indicate that more developers are aware of the project, thus the possibility that some of those developers will be interested to contribute increases. Therefore, we hypothesize:

H$_1$. *The number of developers in a project's labor pool is positively associated with the number of new contributors the project receives in the near future.*

For potential contributors in the labor pool, a social connection with current project developers should reduce their uncertainty about the project, as they may trust the information about the project provided by their 'friends' and may be in a better position to evaluate the outcome of contributing [44]. In addition, familiarity

[1]Although in our operationalization below we consider only contributions in the form of commits (including pull request commits), to keep our analysis tractable. Commits can still touch non-code parts of a project, e.g., documentation files.

with the current project members should help potential developers to understand the working norms and the way of collaboration better [26]. Finally, the focal project's current developers are generally more likely to accept contributions from people they know about [70]. We thus hypothesize:

H$_2$. *The strength of social connections between existing project members and others in the labor pool is positively associated with the number of new developers the project receives in the near future.*

The development of open-source software is skilled work. To provide valuable contributions to the project, developers need to be familiar with the programming languages, technologies, and coding style in the project [17], and align with the project's goals and other such constraints. The requirement for technical programming skills is often a barrier for new contributors to join a project [65], and the contributions of people with less of a track record of activity in open source are also less likely to be accepted [70]. Going one step further, we hypothesize that it's not enough to have open source contribution experience, but rather that the experience should be technically relevant to the focal project:

H$_3$. *The degree of similarity (or fit) between the technologies used in a project and the technical background of the developers in the project's labor pool is positively associated with the number of new developers the project receives in the near future.*

Finally, contributing to open-source projects takes a lot of effort [15, 49]. Given that the amount of time anyone can invest in contributing to open source is unavoidably limited (at the very least by the laws of physics), one can expect that the total number of projects one can join should be limited as well. And while it is common for people to contribute to multiple open-source projects even during the same day [72], and one's capacity for 'multitasking' across projects increases when the projects share the same programming languages [72], there can still be more projects available than one has capacity for. Therefore, a developer part of the labor pool of multiple projects may be forced to choose from among them. Stated differently, a developer's tendency to join a project may not only be influenced by the characteristics of the focal project, but also by the amount of other 'competing' projects the developer is exposed to. As competition for new contributors can exist between projects with overlapping labor pools, we hypothesize:

H$_4$. *A project will engage fewer new contributors in the near future the more overlap there is between its labor pool and labor pools of other projects.*

H$_5$. *The project will receive more new contributors if it is relatively attractive compared to other projects that developers in its labor pool are also possibly exposed to.*

We describe our study design for testing these hypotheses next.

4 METHODS

This section first gives the high-level intuition behind our research design, and then dives into the technical details for each step.

4.1 Key Study Design Decisions and Tradeoffs

Testing the hypotheses above requires a macro, ecosystem-level analysis. Given a focal project at a given time, we need to operationalize its labor pool from among those people active across the entire open-source 'universe' at that time (**H$_1$**, **H$_2$**). Then, for every such person, we need to make inferences about their past experience with different technologies across the open-source 'universe.' We then need to use these estimates to compute the people's technical fit with the focal project (**H$_3$**), as well as to compute, pairwise, their technical fit with all other available projects in the open-source 'universe', to identify the ones which might be competing for their attention (**H$_4$**, **H$_5$**).

Given the obvious computational complexity of such analysis, we made several tradeoffs between computational feasibility, on the one hand, and realism and statistical power, on the other hand. This resulted in the following three key study design decisions.

Study Context: The Python Ecosystem. To keep our analysis tractable we choose to analyze *only one open-source ecosystem* — the Python programming language ecosystem — but consider *all projects* part of the ecosystem. Python is among the most popular and widely used programming languages in open source [9, 22], and it is the language behind libraries and frameworks used in a diversity of application domains.[2] At the same time, the Python open-source ecosystem contains a large set of projects, enabling us to draw conclusions with high statistical power. Thus, a study of the Python ecosystem provides important practical implications and also significant scientific value.

Labor Pool Sampling Frame: The Co-commit Network. We operationalize the labor pool of a focal project at a given time based on relationships between nodes in the project's *collaboration network*. Nodes in the network represent individual developers, and edges indicate that those two developers made commits to a same project in the same time period, i.e., a co-commit relationship. For simplicity, we compute the collaboration network in yearly snapshots, with each snapshot capturing the activity in the respective previous full calendar year $y-1$, i.e., from 1st of January, $y-1$ to 31st of December, $y-1$. We construct the network starting from the set of all contributors to the focal project in year $y-1$. Then we expand outwards, to include all their 'collaborators' from all other projects each contributed to in year $y-1$. Then, transitively, we collect all *their* 'collaborators' in the same year $y-1$ and so forth, up to three hops away from the starting set of the focal project's contributors.[3]

The goal here is to approximate the number of developers who might be aware of the focal project at a given time — awareness is a first step towards deciding to contribute. Clearly, there are myriad online and offline ways to learn about new projects, including social media [12, 28, 62], recommendation systems [46], directly following other developers on social coding platforms like GitHub [10], meetings [14], and many other channels [50].

Identifying the exact set of developers who are aware of the project at a given time is impossible. Instead, we use co-committing relationships as a proxy for awareness of the focal project. Prior research suggests that the collaboration between developers creates opportunities for communication, and the information about a project is likely to diffuse as a result of developer interactions [2].

[2] https://www.python.org/about/apps/

[3] Note that all subsequent outcome measures in our models are computed in year y, to avoid soundness issues caused by the possibly reversed chronology of the 'collaboration' and focal project joining events.

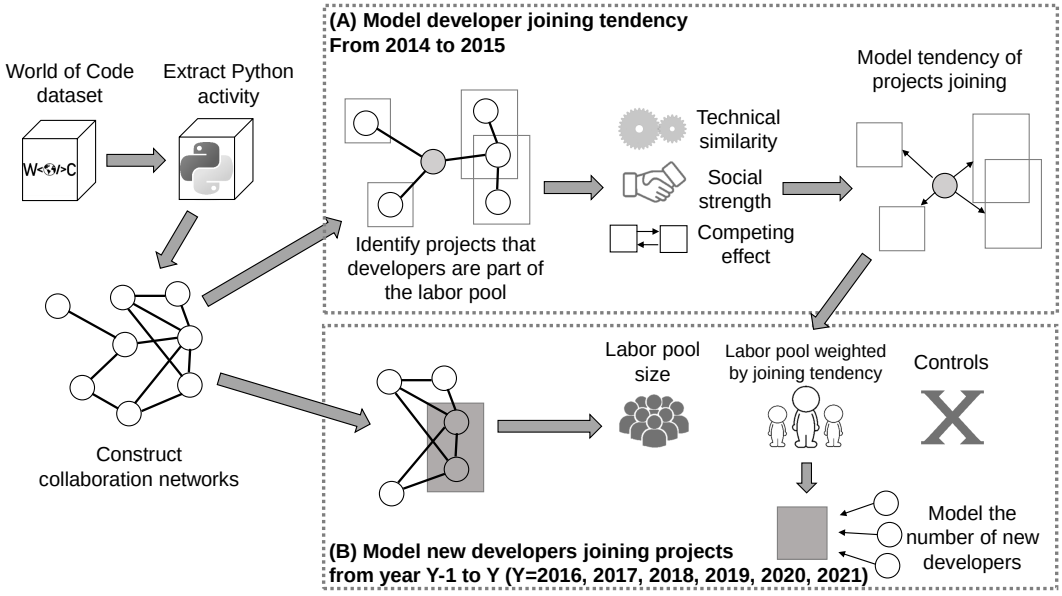

Figure 1: Summary of our data preparation and analysis process.

Co-committing to a common project does not guarantee direct collaboration or interaction [40], but is a prerequisite for both for contributors who make changes to a project's codebase — at the very least, it provides an *opportunity* for interaction and direct collaboration. Moreover, not all such co-committing relationships will result in information about the focal project being diffused, especially as the number of hops from the focal project's developers increases. However, we expect that, on average, the more such opportunities for interaction exist, the more likely it becomes for information about the focal project to be diffused.

This operationalization also has the advantage of being agnostic to the platform where the source code repository is being hosted, thereby allowing us to more easily scale the analysis beyond any single platform where additional, platform-specific forms of interaction could also be considered, e.g., interactions via issue tracker comments. While currently the dominant one, GitHub is not the only platform for hosting open source.

Labor Pool Operationalization: People One-Hop Away From Current Project Contributors in the Co-commit Network. To further reduce the computational complexity of the analysis, we operationalize the labor pool of a focal project at a given time only as the set of developers *within one hop away* from current project contributors in the collaboration network, who have never contributed to the project before. The rationale is twofold.

First, information diffusion theory suggests that the closer nodes are to the information source in a network, the more likely they are to receive the information [56, 76]. Thus, one can expect that awareness of the focal project should decrease substantially the more hops away one is from the focal project's current developers. This is consistent with prior software engineering research [29] reporting a relatively low tendency to join the focal project in the future for people who saw it mentioned on Twitter — there is arguably considerable distance between a focal open-source project hosted on the GitHub platform and an average Twitter user.

Second, we analyzed the yearly snapshots of the collaboration network between new contributors and the existing project developers across all projects in our sample, and computed the network distances (number of hops) from the existing project developers. Figure 2 visualizes the ratio $\frac{\sum_p N_{dpy}}{\sum_p N_{py}}$, where N_{dpy} is the number of new contributors to project p in year y who are d-hops away from the existing developers in the collaboration network in year $y-1$,[4] and N_{py} is the total number of new contributors to project p in year y.[5] As expected, the number of new contributors identifiable in the collaboration network decreases sharply with the number of hops, with the one-hop distance capturing most of them — around 61-65% of everyone identifiable within three hops. Given the exponential increase in complexity with network distance, we choose to restrict the operationalization of a project's labor pool to developers one-hop away in the collaboration network, as this distance captures most identifiable developers.

Note that overall, the one-hop distance captures only around 19-23% of all new contributors, i.e., the majority of new contributors are unidentifiable within one hop of the current project contributors in the yearly collaboration graph (similarly, also within three hops). There are many possible reasons. Perhaps the one-year horizon is insufficient to capture all ties, and ties surely also form through many other means than our relatively strict operationalization of co-committing to the same repository can capture. However, given the considerable fraction of new contributors that our relatively simple, but most scalable, operationalization can capture, we expect that our resulting sample contains sufficient signal for our subsequent

[4]Recall that outcome measures are computed in subsequent years from collaboration network edges, cf. footnote 3.
[5]We apply the same minimum activity filter as in Section 4.5, for consistency.

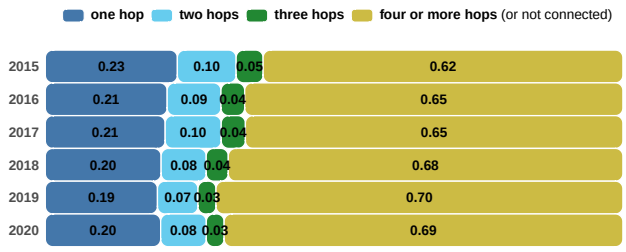

Figure 2: Percentage of new contributions from different network distances to the existing developers, across years.

modeling needs, while at the same time keeping the data volume and the computation needed for the analysis tractable.

4.2 Overview of the Analysis

Given these key design decisions, we structure our study in two parts, as summarized in Figure 1. Both parts involve regression models explaining the tendency and number of new developers joining projects in a next time period as a function of the sets of factors we formulate explicit hypotheses about, via their corresponding variables computed in a current time period. In the first part we take a *developer-centric view* — from the perspective of an individual developer, they typically have a choice of projects they could contribute to and a range of projects they're in the labor pool for, based on past collaborations. In the second part we take a *project-centric view*, aggregating individual-level effects to the level of the whole ecosystem, to reason about project labor pool characteristics and competition effects.

First, we estimate the relative importance of the three sets of factors we formulate explicit hypotheses about[6] — the strength of social connections to existing project members ($\mathbf{H}_2$), the fit between one's technical background and the focal project ($\mathbf{H}_3$), and the amount of competition (or choice one has) between available projects with similar technical fit ($\mathbf{H}_4$, $\mathbf{H}_5$) — at the individual level. To do this, we start by computing a data frame of (labor-pool-developer, focal-project) pairs, with measurements of the relevant variables (details below) for every developer in a given project's labor pool, across all projects in our sample; we also record a binary outcome variable indicating whether or not that developer joined the project in the next period. Using this data, we then construct a logistic regression model explaining the developers' tendency to join a focal project in the next year as a function of the variables of interest; the labor pool is operationalized as described above, i.e., people one-hop away from the focal project's developers. We refer to variations of this logistic regression model (under different specifications) as **individual models**.

Note that the goal here is not to make individual predictions about any one developer's tendency to join a given project in the next time period. Rather, the goal is to estimate the *relative importance* of the three sets of factors of interest, on average, across a large sample, such that we can reuse these 'weights,' i.e., the estimated β coefficients from the logistic regression model, in the second part of our analysis. For example, we estimate how much the technical background fit explains the joining tendency of an *average* developer, compared to the strength of social connections and the amount of competition from other projects, over a large sample. Because we estimate the logistic regression over a very large sample, we can assume that these coefficients are stable,[7] so we estimate only one set of individual models[8] to be used as input for the second part.

Next, we lift[9] the individual-level analysis to the project level by estimating regressions that explain the number of new developers joining projects in a next year as a function of their labor pool characteristics (and control variables) in the current year. We refer to these models as **project models** and we use them to formally test all our hypotheses $\mathbf{H}_1$–$\mathbf{H}_5$.

To ensure the robustness of our conclusions, we repeat this analysis for all the complete pairs of consecutive years in our data, from 2015–2016 to 2020–2021. In the end, we quantify the amount of variance that the labor pool characteristics explain when modeling the number of new contributors a project will receive, interpret the results, and discuss the implications of our findings.

4.3 Data Collection and Filtering

We mine our data from the World of Code (WoC) dataset [45], which contains the git commit traces for all public projects hosted on GitHub, Gitlab, Bitbucket, SourceForge, and many other smaller ones. We expect that World of Code should give better coverage of open-source development compared to other datasets typically used in prior research.

To begin with, we define the open-source Python ecosystem as containing all repositories with over 50% of their files written in the Python language. We then apply several filters to de-noise the data, as typical with mining software repositories research [41].

First, we filter out repositories with fewer than 10 commits that involve changes to library import statements, i.e., adding or removing dependencies. This step is needed because we later use this dependency information to characterize the technical needs of projects, i.e., we assume that a project using certain libraries requires contributors with experience in those libraries. We chose the threshold arbitrarily, balancing a desire to retain a large sample, on the one hand, with an attempt to filter out trivial projects (code dumps, homework solutions, etc) and a need for 'enough' data for the subsequent embeddings-based approach to work. Similarly, we filter out developers from labor pools if they authored fewer than 10 commits that involve changes to library import statements, for analogous reasons. As a robustness validation, we run the same analysis over datasets where projects and authors with less than 100 commits that involve change of packages are removed, and the results are qualitatively similar (See the replication package for validation study).

Second, we made sure to use the de-aliased activity records from the World of Code dataset, which provides both raw data on commit authors as well as data on de-aliased commit authors, after merging developer identities when they use different aliases; see

[6] Excluding $\mathbf{H}_1$, which refers to the labor pool size, rather than its composition.

[7] We discuss robustness checks for this assumption below.

[8] We computed all independent variables in 2014 and the outcome variable in 2015.

[9] The estimated β coefficients from the individual models enable this aggregation.

Table 1: The definitions of variables in the individual model

Variables related to social connections	
Social strength	The total number of projects one has worked on with any of the current project developers.
Variables related to technical fit	
Technical similarity	The similarity between one's technical background and the project's technologies.
Variables related to competition effects	
Number of competing projects	The total number of projects one is in the labor pools for.
Relative advantage in social connection	The percentile of the *Strength of social connection* variable defined above.
Relative advantage in technical similarity	The percentile of the *Technical fit* variable defined above.

Fry et al. [33] for details on the random forest model used to merge developer aliases based on their user IDs. It is important to use the de-aliased activity records because the volume of developer aliases in such data may skew our measurements of project contributors and experience with Python libraries [3].

Finally, we also make a best effort attempt to filter out bots and unidentifiable accounts together with their associated commit activities, for similar reasons [75]. Specifically, we use three heuristics to identify bot-like accounts. First, we reuse a list of 13,169 bot accounts in the World of Code dataset compiled by Dey et al. [24] after developing a machine learning classifier for this purpose, based on author names, commit messages, files, and projects modified by the suspected bot account. Second, we convert all account usernames into lowercase characters and use string matching to flag as bots those with the last part of their username being *-bot* or *-robot*. Third, we order all developer accounts in our dataset based on the number of commits they made, and we manually evaluate the top 100 accounts. This revealed a few additional bot-like and unidentifiable accounts such as *GitHub Merge Button <merge-button@github.com>*. Overall, all these commit authors are excluded from our analysis.

4.4 Part I: Individual Models

As discussed briefly above, we use logistic regression to model the factors associated with the individual tendency to join a focal project, across all (labor-pool-developer, project) pairs in our sample. The full model is specified as:

$$P(J_{ipy}) = logit(\beta_0 P_{py-1} + \beta_1 S_{ipy-1} + \beta_2 T_{ipy-1} + \beta_3 C_{ipy-1}), \quad (1)$$

where $P(J_{ipy})$ is the likelihood that developer i joins project p in year y, and independent variables S_{ipy-1}, T_{ipy-1}, and C_{ipy-1} represent the social connection between potential contributor i and the existing developers of project p, the technical background fit between developer i and project p, and the factors relating to the competitive advantage of project p among the set of projects developer i can potentially join, respectively, all computed in year $y-1$. Table 1 gives definitions of the variables in the model; we expand on how we operationalized the variables below.

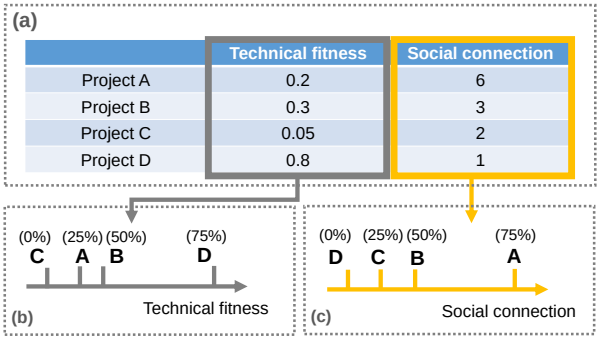

Figure 3: Illustration of the project relative advantage.

Modeling Considerations. For simplicity, since the estimated β coefficients are stable, we compute only one individual model for $y-1=2014$ and reuse the coefficients throughout Part II.

We also restrict our sample only to the labor-pool developers who were active (i.e., made at least one commit) in 2014, because developers who are inactive for more than one year tend to have a low probability to make commits in future years [16]. Until the end of 2014, there are 104,899 Python developers in our sample who made at least 10 valid commits with changes to import statements, and were active in 2014. For each developer, we identify the projects whose labor pools the developer was part of, and model their tendency to join those projects in 2015. Since some developers may be in the labor pools of a large number of projects, for each developer, we randomly sample 30% of the labor pools they are part of. Consequently, we also exclude developers who are part of the labor pools of less than four projects, to ensure that at least one project per person is sampled. In total, we have 47,788 developers and 5,778,144 (developer–project) observations in our sample.

Finally, given the inherently nested structure of our data (the same developer being in the labor pool for multiple projects), we make clustering adjustments in the standard errors at both the project and developer levels to account for the possible within-cluster correlation [1].

Measuring the Technical Fit Between Projects and Developers. The fit between project technical requirements and individual technical background is hypothesized to be an important factor influencing the developer joining behavior. We use the packages (or libraries) a project imports to measure the *technical requirement* of a project, and the packages imported in past code commits of a developer to measure their *technical skills*. While prior research used the programming language as a proxy for technical skills [18], this coarse-grained measure is not suitable for our study as all the projects in our sample are mostly written in Python.

The World of Code dataset contains dependency information extracted from each commit (i.e., the packages that a commit imports).[10] Therefore, we can obtain the packages that a project depends on, and the packages that a developer has used in their past commits. Following Dey et al. [23], we then train a Doc2Vec model to obtain the technical skill embedding of developers and projects.

[10] https://github.com/woc-hack/tutorial

Table 2: The definitions of the variables used in the project model.

Outcome Variable	
Number of new developers	The number of new developers joining the project in the next year. New developers are defined as developers who have never committed to the project before the study period, and made their first commit in the following year. We only count the new contributors that have at least 10 valid commits in the past across the entire WoC dataset.
Control variables	
Project age	The number of days elapsed since the first commit of the project.
Total number of developers	The number of all contributors to the project in the past. Contributors are defined as the developers who made at least one commit to the project.
Number of recent developers	The number of all contributors who committed to the project in the past year.
Total number of commits	The number of commits developers made to the project in the past.
Number of recent commits	The number of commits developers made to the project in the past year.
Has license	A binary variable indicating whether the project had any license by the time of the study.
Has readme	A binary variable indicating whether the project had any README file by the time of the study.
Labor pool variables	
Labor pool size	The number of potential developers in the labor pool of the project. See Section 4.1 for the definition and operationalization.
Labor pool effective size, non-competing variables	The size of the labor pool with each individual weighted by their tendency to join the project. The individual model does not include the competition-related variable.
Labor pool effective size, full variables	The size of the labor pool with each individual weighted by their tendency to join the project. The individual model includes all effects as hypothesized in Section 3.

Analogous to a Doc2Vec model in natural language processing [43], we consider the package names to be the tokens, and the developers (projects) to be the documents (consisting of tokens). We can thus learn a vector embedding of each token (package), and each document (developer and project). The cosine distance between vector embeddings will be small if the two documents are similar in terms of the tokens they contain, or the packages they use. Therefore, the cosine distance between the developer's and the project's embedding is a good proxy for technical skill fit.

Measuring the Project Relative Advantage. To operationalize possible competition between projects over potential contributors, we rank projects, from the perspective of an individual potential contributor, in terms of strength of social connection and technical similarity — we expect that given multiple options, on average, developers will typically choose projects for which they are a better socio-technical fit. For a potential developer, we obtain all projects whose labor pools they are part of. For each project, we compute the characteristics that we consider to be influential on the developer joining behavior as hypothesized in Section 3. The relative advantage of each project is defined as the percentile of their characteristics, among the set of projects where the focal developer is part of the labor pool.

In Figure 3 (a), for example, assume that the focal developer is in the labor pool of four projects (i.e., A, B, C, and D). We first order projects based on their technical fit with the focal developer, and the resulting rank (or percentiles, as labeled on top of the project ID in Figure 3 (b)) is the relative advantage of projects in terms of their skill fit. Similarly, we compute the relative advantage of projects based on their strength of social connection with the potential developer, as shown in Figure 3 (c).

4.5 Part II: Project Models

Given the previous individual models estimating the probability of a labor-pool developer to join a focal project from Section 4.4, summing over all developers who are in the project's labor pool gives the mathematical expectation of the number of new contributors joining at the project level:

$$P_{py} = \sum_i P(J_{ipy}), \tag{2}$$

where P_{py} is the tendency of all developers i in the project p's labor pool to join the project in year y, and $P(J_{ipy})$ is the tendency for any individual developer i to join project p, given that developer i is part of p's labor pool.

Our modeling strategy is hierarchical regression, i.e., we estimate separate individual models that incorporate different sets of factors, ranging from a baseline model with control variables only to a full model that includes all hypothesized factors. Since the aggregation to project level incorporates, therefore, the characteristics of developers in the labor pool, we refer to this outcome variable as the "effective (labor pool) size" in Table 2. For example, the effective labor pool of a project could be much smaller than the number of one-hop potential contributors would suggest, if the technical fit is relatively low or there is high competition from other projects.

Practically, we regress the number of new developers a project receives in year y. Similarly to the individual model, we restrict our sample to only active projects (i.e., having at least one commit) in year y, as projects that are inactive for more than one year should have lower probability of attracting new contributors in the next year compared to active projects [52] and we do not want our sample to be biased by the large number of inactive projects which attract no new contributors. New developers are those who made their first ever commit to the project in year y. As before, when measuring the number of new contributors (i.e., the outcome variable), we only count the new contributors who made at least 10 commits involving import statements in the past. For new project contributors without enough commit history to infer their technical skills, we cannot estimate their 'future' commit tendency, and thus we are unable to predict whether they will join a new project or not in this study.

The main result of this paper is reported for y = 2021, as this was the most recent complete snapshot in our sample and also the largest dataset; the 2021 snapshot contains 516,893 active Python projects. Since the results are consistent for other values of y, i.e., the regression coefficients point in the same direction and have similar scale and statistical significance, we don't discuss the other models in detail but include the results in our replication package.

5 RESULTS

5.1 Models of Individual Joining Tendency

We first present the result for modeling the developers' tendency to join other projects, conditioned on them being part of the project's labor pool, in Table 3.

In model I, we include social and technical variables that correspond to $\mathbf{H}_2$ and $\mathbf{H}_3$. Both the social connection between potential new developers and the existing developers, and the technical similarity between potential developers' skill sets and the project's technical requirements are statistically significantly and positively associated to the likelihood of individuals joining new projects, which confirms $\mathbf{H}_2$ and $\mathbf{H}_3$.

In model II, we include the competition effects on the basis of model I, as discussed in $\mathbf{H}_4$ and $\mathbf{H}_5$. $\mathbf{H}_4$ is confirmed, as indicated by the negative effect of variable *Number of competing projects (log)*, which suggests that the more projects a developer was potentially exposed to (the more labor pools they are part of), the lower the likelihood that the developer will join any specific one of them. $\mathbf{H}_5$ was also confirmed by the significant positive coefficient for variables *Social strength percentile* and *Technical similarity percentile*, suggesting that the project's relative advantage among the set of projects that a developer is potentially exposed to also influences the developer's tendency to join the project. Interestingly, the effect of absolute technical similarity disappears after controlling for the relative similarity effect, suggesting that the relative technical fit among exposed projects is more important to influence the developers' joining behavior compared to the absolute fitness. Compared to model I, model II has a lower AIC (Akaike information criterion) value overall which indicates a better match between the predicted value and the ground truth.

To better analyze the impact of social-technical factors on individual joining tendencies, we examine project-developer pairs where the developer belongs to the project's labor pool and is a potential new contributor. We categorize these pairs into four groups based on whether the developer's technical similarity to the focal project and their social connection to existing developers are above or below the median values observed in our sample. We then calculate the joining probability for each project-developer pair within each group. Figure 4 summarizes the results. Among (labor pool developer, project) pairs where developers have strong social connections (top 50%) and high technical similarity (top 50%) with the project, we find an average of 8.7 new developers joining per 10,000 project-developer pairs. In contrast, for pairs where developers have weak social connections and low technical similarity, this number drops to only 0.6 new developers per 10,000 project-developer pairs.

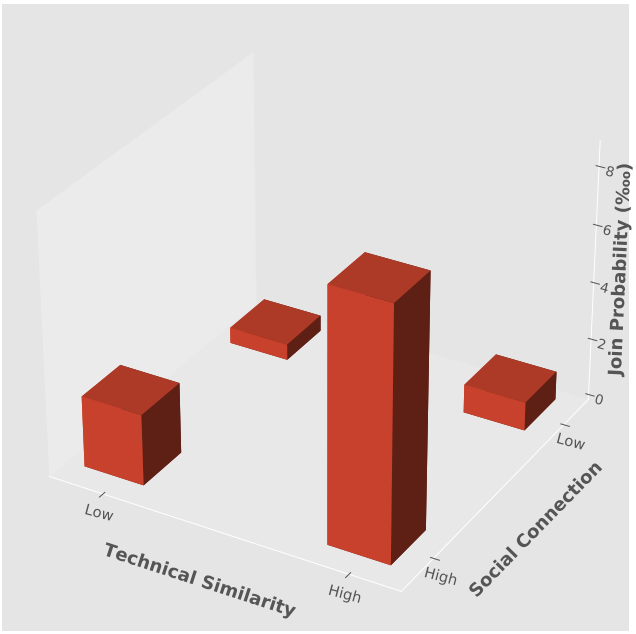

Figure 4: Illustration of the social-technical effects on developer joining tendency (y=2021).

5.2 Models of the Number of New Contributors

Table 4 summarizes the models explaining the expected number of new contributors that a project receives in the next year; note the log-scaled outcome variable. Model I is our base model, where we explain the number of new developers a project receives only with the control variables (project-level characteristics). As the model suggests, the current and historical size of the project's developer team is positively associated with the number of future developers that a project receives, which suggests the existence of a preferential attachment effect that developers tend to attach to popular and well-known projects. This results is consistent with the one reported by prior work [57, 71]. In addition, the use of licenses and the project age both have a significant positive effect on the number of new developers. The effect of the current project size (or the number of commits) is also significant, though it is highly correlated with the historical project size and commits, as expected. Overall, this model explains 12.9% of the variance.

Next, we include the labor pool size variable in model II to explain the number of new developers a project will receive. As hypothesized in $\mathbf{H}_1$, the size of the project's labor pool is positively associated with the number of new developers, and the model explains 14.8% variance in total, or about 2% of the overall variance more than model I.

The simple measurement of the project's labor pool size treats all developers in the labor pool equally, and fails to capture their different tendencies to join. In models III and IV, each developer is weighed differently based on their estimated tendency to join, as computed by the individual model. For the variable *Effective size, no-competition variables*, the estimated joining tendency for each developer is computed with non-competing variables only (model I

Table 3: Modeling the project-joining tendency of developers

	Model I	Model II
Non-competition effects		
Social strength (log)	$1.17(0.02)^{***}$	$0.89(0.03)^{***}$
Technical similarity	$2.03(0.21)^{***}$	$-0.14(0.25)$
Competition effects		
Num. competing projects (log)		$-0.68(0.02)^{***}$
Social strength percentile		$2.03(0.07)^{***}$
Technical similarity percentile		$0.93(0.13)^{***}$
Observations	5,778,144	5,778,144
Akaike Inf. Crit.	64,668.72	60,465.68

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

in Table 3), and the variable *Effective size, all variables* is computed based on both competition and no-competition variables (model II in Table 3). Controlling for the simple measurement of the project's labor pool size, the effective labor pool size is still positively associated with the number of new contributors that a project receives, and it provides additional explanatory and predictive power for our outcome variable. Model III which includes the effective size computed with only non-competing variables explains 14.9% variance, slightly more variance compared to model II, and by adding the effective size computed by both competition and no-competition effects, model IV explains 16.4% variance in total, and 1.5% more of the overall variance than model III – a sizeable increase. Therefore, we conclude that all hypotheses in Section 3 are confirmed at the project level.

5.3 Analysis on the Generalizability of Labor Pool Factors

To better understand when labor pool factors fail to explain the number of new contributors, we plot in Figure 5 the number of new developers that projects in our sample attract and the corresponding effective labor pool size (computed with the full-variables individual model). For better visualization, high-leverage points are removed and we log-scale the effective labor pool size. The red dot represents the mean of effective labor pool size with a given number of newly attracted developers.

We focus on two areas of the graph. First, projects in area A of Figure 5 have a large effective labor pool size, but they attract very few new contributors, if at all, which differs from what we would predict with the effective labor pool. In contrast, projects in area B attract a large number of new developers in the next year but seem to have a very small effective labor pool size. It is also interesting to understand how they attract a large number of new developers given a relatively small effective labor pool size. Therefore, we manually explore the projects in area A and B in Figure 5 and list possible factors that may lead to model misclassification.

First, we find that many projects in area A (large effective labor pool and a low number of new contributors) seem to be used for storage purposes, such as workshops or student competition projects (e.g., *ai2cm/dsl_workshop*, *PlatziMaster/challenge-prework-ds*), rather than more traditional open-source development. Those

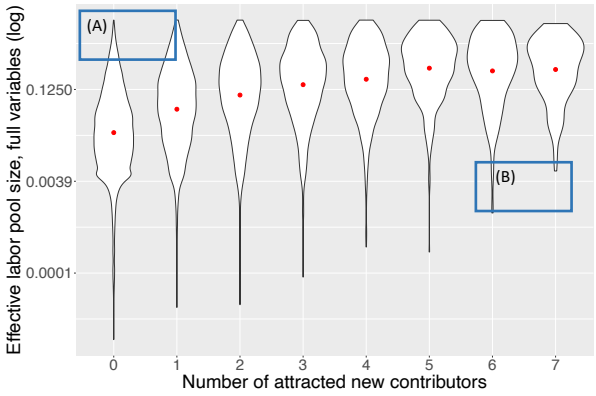

Figure 5: The relationship between the effective labor pool size and the attracted new contributors (Y=2021, excluding outlier projects attracting more than eight developers).

projects tend to stop receiving new contributors after the workshop or competition ends, thus their contribution is relatively independent of the effective labor pool size. Second, projects in area A are likely to receive no commits at all in the next year, being that not only do they attract no new developers, but also they receive little to no commits from the existing developers in the next year. This suggests that the existing developers, which is the basis for identifying a project's labor pool in our study, may not be sending signals to their collaborators and attracting new developers as they have abandoned the project themselves. In addition to some existing developers being "forced" to leave the project because of unavailability, as also reported by prior research [48], another possible reason is the completeness of project features or a decision to stop the development, which leads to a natural end of the project development. We find evidence for such projects as they have been archived [47] (e.g., *ansible-community/molecule-azure*), or they receive no new issues and updates in the future (e.g., *FergusYip/DrinkMoreApp*).

For projects in area B (low effective labor pool and a large number of new contributors), we find the most common reason is that those projects are owned by organizational accounts (e.g., *skit-ai/dialogy*, *panther-labs/panther-analysis*, *ZJU-OpenKS/OpenKS*), and the labor pool measured based on developer networks may not be an accurate reflection of developers who are aware of the project.

Overall, we found that factors such as project completion and the disengagement of existing developers may complicate the estimation of a project's labor pool, which could bias predictions and explanations of the number of future contributors. We further acknowledge that it is harder to estimate the labor pool size based on our current operationalization for projects owned by organizational accounts, as we do not incorporate the organization influence into our labor pool operationalization. However, we recommend that future researchers and developers also consider these factors when estimating a project's effective labor pool or the number of future contributors that may be expected. In addition, despite having considerable explanatory power on the number of future developers, the regression model with the labor pool and other controls in our study has not yet produced practically usable predictions on individual future contributors. We estimate that with the individual

Table 4: Modeling the number of new developers a project will receive in the next year ($y = 2021$)

	Model I	Model II	Model III	Model IV
Control variables				
Project age (log)	$0.01(0.0004)^{***}$	$0.005(0.0004)^{***}$	$0.005(0.0004)^{***}$	$0.005(0.0004)^{***}$
Project total developer size (log)	$0.07(0.001)^{***}$	$0.06(0.001)^{***}$	$0.06(0.001)^{***}$	$0.06(0.001)^{***}$
Project recent developer size (log)	$0.05(0.001)^{***}$	$0.04(0.001)^{***}$	$0.04(0.001)^{***}$	$0.02(0.001)^{***}$
Project total commits (log)	$0.001(0.0004)^{*}$	$0.0003(0.0004)$	$0.001(0.0004)$	$0.001(0.0004)^{*}$
Project recent commits (log)	$0.01(0.0004)^{***}$	$0.01(0.0004)^{***}$	$0.01(0.0004)^{***}$	$0.01(0.0004)^{***}$
Has readme	$-0.01(0.001)^{***}$	$-0.01(0.001)^{***}$	$-0.01(0.001)^{***}$	$-0.01(0.001)^{***}$
Has license	$0.03(0.001)^{***}$	$0.02(0.001)^{***}$	$0.02(0.001)^{***}$	$0.03(0.001)^{***}$
Labor pool variables				
Labor pool size (log)		$0.02(0.0002)^{***}$	$0.01(0.0002)^{***}$	$0.01(0.0002)^{***}$
Effective size, no-competing variables (log)			$0.10(0.004)^{***}$	
Effective size, full variables (log)				$0.42(0.004)^{***}$
Observations	516,893	516,893	516,893	516,893
Adjusted R^2	0.129	0.148	0.149	0.164

Note: $^{*}p < 0.05$; $^{**}p < 0.01$; $^{***}p < 0.001$

model (i.e., predicting whether a given developer will join a project in the next year or not), making a prediction has 7% precision and 14% recall on average, and with the project model (i.e., predicting the number of new contributors to a project in the next year) making a prediction has 0.19 average difference, or the prediction on the number of new contributors for each project will be off by 0.19 compared to the ground truth on average. This accuracy may be insufficient for *prediction* tasks in practice, which go beyond the scope of this work in which we focus on *explanation* instead, and better understanding the underlying mechanisms.

6 DISCUSSION

In this paper, we explored the influence of the projects' labor pool on attracting new developers. We summarize the main results and discuss limitations, the scientific value, and the practical implications of our results below.

6.1 Labor Pool as an Important Factor for Project Sustainability

The prevailing empirical studies on open-source sustainability, and attracting new contributors in particular, have focused on the influence of project-level characteristics. In our paper, we provide an alternative, ecosystem-level perspective by suggesting the project's labor pool as an important factor.

The labor pool of a project corresponds to the communities of developers that may be of help to the development and maintenance of the project. They consist of developers who learn about the project through many possible channels, like recommendations from their friends or collaborators, exposure on social media spaces, recommendation systems, web search engines, and many others. They are the contribution resources potentially 'accessible' by the project at a given time.

Considering labor pool factors following a methodology similar to ours can help open-source stakeholders to better understand and predict the amount of contributions projects may have in the future, which helps with the project management and resource allocation. Moreover, users who are seeking sustainable open-source projects to adopt can use labor pool factors to better evaluate the sustainability of candidate projects.

6.2 The Size of the Labor Pool Is Important, but It Is Not All That Matters

It is not surprising that expanding the size of the project's labor pool helps to attract new developers, which is why so much effort has been devoted to promoting open-source projects to a larger audience [14, 29, 62]. However, our results indicate that the characteristics of the developers in the project's labor pool, in addition to the raw number of developers, also have a significant effect to explain the number of new developers that a project will receive.

With many previous studies about attracting and retaining open-source contributors focusing on expanding the influence of projects and reaching out to a large community, our work adds to the conventional wisdom by suggesting that expanding project influence to a *proper* audience also matters. In addition, our work is the first, to the best of our knowledge, to provide empirical evidence of the competition effects between projects, and suggests the practice of recruiting new developers is more complex beyond simply reaching out to a large community.

6.3 Towards Targeted Project Promotion

For open-source project promoters seeking to attract more attention and effort to their projects, our results suggest that it is not only the number of developers that the promotion reaches that matters, but also the characteristics of those developers. Therefore, a targeted promotion that diffuses the project information to a developer community that possesses the related technical skills and is socially close to the project's existing developers may be more efficient and effective to attract new developers compared to promotions to a wide audience, as they reach a community that is more likely to join the project as contributors. In addition, as the promotion of projects

becomes more targeted, developers may be less overwhelmed by the wide range of projects hosted online as the information and promotion they are exposed to will be more relevant to their skills, social connections, and needs. It can reduce the cognitive workload for individual developers and should help to better allocate attention and effort at the open-source ecosystem level [19].

Some project promotion channels or tools are potentially suitable for targeted promotions. For example, project promotion on social media such as Twitter may easily reach socially connected groups as the diffusion of information depends on the social connections; and the project recommendation tools can incorporate technical relatedness and social connections as factors for project recommendations. With most open-source promotion research so far concentrating on the size of developers that a promotion reaches [14, 29], we argue for the importance of studying the characteristics of developers that those promotions reach, and reconsider the value of promotion campaigns based on their effectiveness to reach a targeted audience instead of reaching a large developer community.

6.4 The Relationship Between Projects and the Success of Open-Source Ecosystems

The significant competition effects between projects revealed by our models suggest that the attraction of developers to projects is not a local question. Given two projects with overlapping labor pools, any effort to make one project more attractive is likely to have a negative effect on the sustainability of the other, when they are competing for the same effort pool.

This finding raises the important question about the allocation of effort among open-source projects, and the role of individual projects in the success of the open-source ecosystem. While open-source developers and researchers have devoted much effort to making individual projects more successful and sustainable, little was discussed about the influence of those efforts on other projects in the ecosystem. Our result is one of the first works to understand the relationship and competition between projects, and we call upon future research in this direction to further study the sustainability of individual projects jointly with that of the open-source ecosystem.

6.5 Validations and Robustness Checks

Validate the Robustness of Our Results Over Years. We evaluate the effectiveness of labor pool factors over years and run the same project model for data in three other years (i.e., selecting $y = 2016, 2017, 2018, 2019, 2020$ and 2021). The result suggests that the labor pool factors are important predictors of the new developer attraction for four consecutive years, see our replication package for full results.

Validate the Results With a Negative Binomial Regression Model. Negative binomial regression is a powerful tool to model discrete outcome variables such as the number of new developers [38]. To validate the influence of labor pool factors, we use a negative binomial regression model to estimate the number of new developers that a project receives, with all the independent variables the same as the ordinary least squares (OLS) regression used to report the main result; see replication package.

All labor pool factors are still significantly associated with the outcome variable, with the direction of effects qualitatively similar except for the effect from *Effective size, no-competition variables* in model III, which changes from positive in the OLS regression model to the current negative effect. Further analysis shows that the effect is only negative when controlling for the *Labor pool size* variable, but significantly positive otherwise. Despite this inconsistency, the effective labor pool size with the individual model of full variables still shows a significant positive effect, and the Akaike information criterion (AIC) value decreases when adding labor pool variables, which indicates better model fit. Therefore, we conclude the results with negative binomial regression are generally consistent with the OLS regression.

7 CONCLUSIONS

In this work, we show that an open-source project's labor pool, defined as the set of developers who are possibly aware of the project and may serve as potential future project contributors, is an important factor that can impact project sustainability. We found that adding the labor pool factors, being both the amount and characteristics of developers in a project's labor pool, explains 27% more variance compared to baseline models that only include project characteristics. We also show that the technical fit, the social connection, and the project competition effects are three factors that can affect how developers move between projects. Our work contributes to the scientific understanding of what leads to the attraction of new open-source developers beyond individual project-level factors, and provides important implications to open-source stakeholders for better project promotion and community management.

8 DATA AVAILABILITY

The data and scripts to reproduce our results are available in the replication package [27]. DOI 10.5281/zenodo.8252560

ACKNOWLEDGMENTS

The authors kindly acknowledge partial support from the NSF awards 1901311, 2107298, and 2120323, the Google Open Source Programs Office, Google Faculty Research Awards, and a Google Award for Inclusion Research. We are additionally grateful to Audris Mockus and Mahmoud Jahanshahi, for their help with data collection and discussions around our experimental design.

REFERENCES

[1] Alberto Abadie, Susan Athey, Guido W Imbens, and Jeffrey Wooldridge. 2017. *When should you adjust standard errors for clustering?* Technical Report. National Bureau of Economic Research.
[2] Divyakant Agrawal, Ceren Budak, and Amr El Abbadi. 2011. Information diffusion in social networks: Observing and influencing societal interests. *Proceedings of the VLDB Endowment* 4, 12 (2011), 1512–1513.
[3] Sadika Amreen, Audris Mockus, Russell Zaretzki, Christopher Bogart, and Yuxia Zhang. 2020. ALFAA: Active Learning Fingerprint based Anti-Aliasing for correcting developer identity errors in version control systems. *Empirical Software Engineering* 25 (2020), 1136–1167.
[4] Guilherme Avelino, Eleni Constantinou, Marco Tulio Valente, and Alexander Serebrenik. 2019. On the abandonment and survival of open source projects: An empirical investigation. In *2019 International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 1–12.
[5] Aida Azadegan, K Nadia Papamichail, and Pedro Sampaio. 2013. Applying collaborative process design to user requirements elicitation: A case study. *Computers in Industry* 64, 7 (2013), 798–812.
[6] Sogol Balali, Umayal Annamalai, Hema Susmita Padala, Bianca Trinkenreich, Marco A Gerosa, Igor Steinmacher, and Anita Sarma. 2020. Recommending tasks

to newcomers in oss projects: How do mentors handle it?. In *Proceedings of the 16th International Symposium on Open Collaboration*. 1–14.

[7] Ann Barcomb, Andreas Kaufmann, Dirk Riehle, Klaas-Jan Stol, and Brian Fitzgerald. 2018. Uncovering the periphery: A qualitative survey of episodic volunteering in free/libre and open source software communities. *IEEE Transactions on Software Engineering* 46, 9 (2018), 962–980.

[8] Pankaj Bhatt, Gautam Shroff, and Arun K Misra. 2004. Dynamics of software maintenance. *ACM SIGSOFT Software Engineering Notes* 29, 5 (2004), 1–5.

[9] Tegawendé F Bissyandé, Ferdian Thung, David Lo, Lingxiao Jiang, and Laurent Réveillere. 2013. Popularity, interoperability, and impact of programming languages in 100,000 open source projects. In *2013 IEEE 37th annual computer software and applications conference*. 303–312.

[10] Kelly Blincoe, Jyoti Sheoran, Sean Goggins, Eva Petakovic, and Daniela Damian. 2016. Understanding the popular users: Following, affiliation influence and leadership on GitHub. *Information and Software Technology* 70 (2016), 30–39.

[11] Christopher Bogart, Christian Kästner, James Herbsleb, and Ferdian Thung. 2016. How to break an API: cost negotiation and community values in three software ecosystems. In *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*. 109–120.

[12] Hudson Borges, Andre Hora, and Marco Tulio Valente. 2016. Understanding the factors that impact the popularity of GitHub repositories. In *International Conference on Software Maintenance and Evolution (ICSME)*. 334–344.

[13] Hudson Borges and Marco Tulio Valente. 2018. What's in a github star? understanding repository starring practices in a social coding platform. *Journal of Systems and Software* 146 (2018), 112–129.

[14] Hudson Silva Borges and Marco Tulio Valente. 2019. How do developers promote open source projects? *Computer* 52, 8 (2019), 27–33.

[15] Frederick P Brooks Jr. 1995. *The mythical man-month: essays on software engineering*. Pearson Education.

[16] Fabio Calefato, Marco Aurelio Gerosa, Giuseppe Iaffaldano, Filippo Lanubile, and Igor Steinmacher. 2022. Will you come back to contribute? Investigating the inactivity of OSS core developers in GitHub. *Empirical Software Engineering* 27, 3 (2022), 76.

[17] Gerardo Canfora, Massimiliano Di Penta, Rocco Oliveto, and Sebastiano Panichella. 2012. Who is going to mentor newcomers in open source projects?. In *Proceedings of the 20th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*. 1–11.

[18] Casey Casalnuovo, Bogdan Vasilescu, Premkumar Devanbu, and Vladimir Filkov. 2015. Developer onboarding in GitHub: the role of prior social links and language experience. In *Proceedings of the 23th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*. 817–828.

[19] Johan SG Chu and James A Evans. 2021. Slowed canonical progress in large fields of science. *Proceedings of the National Academy of Sciences* 118, 41 (2021), e2021636118.

[20] Eleni Constantinou and Tom Mens. 2017. Socio-technical evolution of the Ruby ecosystem in GitHub. In *2017 IEEE 24th international conference on software analysis, evolution and reengineering (SANER)*. 34–44.

[21] Kevin Crowston and James Howison. 2005. The social structure of free and open source software development. *First Monday* (2005).

[22] Daniel P Delorey, Charles D Knutson, and Christophe Giraud-Carrier. 2007. Programming language trends in open source development: An evaluation using data from all production phase sourceforge projects. In *Second International Workshop on Public Data about Software Development (WoPDaSD'07)*.

[23] Tapajit Dey, Andrey Karnauch, and Audris Mockus. 2021. Representation of developer expertise in open source software. In *Proceedings of the 43rd International Conference on Software Engineering (ICSE)*. 995–1007.

[24] Tapajit Dey, Sara Mousavi, Eduardo Ponce, Tanner Fry, Bogdan Vasilescu, Anna Filippova, and Audris Mockus. 2020. Detecting and characterizing bots that commit code. In *Proceedings of the 17th International Conference on Mining Software Repositories (MSR)*. 209–219.

[25] Nadia Eghbal. 2016. *Roads and bridges: The unseen labor behind our digital infrastructure*. Ford Foundation.

[26] Alberto Espinosa, Robert Kraut, Javier Lerch, Sandra Slaughter, James Herbsleb, and Audris Mockus. 2001. Shared mental models and coordination in large-scale, distributed software development. (2001).

[27] Hongbo Fang, James Herbsleb, and Bogdan Vasilescu. 2023. Replication package. https://doi.org/10.5281/zenodo.8252560

[28] Hongbo Fang, Bogdan Vasilescu, and James Herbsleb. 2023. Understanding information diffusion about open-source projects on Twitter, HackerNews, and Reddit. In *Proceedings of the 16th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE)*. 56–67.

[29] Hongbo Fang, Bogdan Vasilescu, Hemank Lamba, and James Herbsleb. 2022. "This Is Damn Slick!" Estimating the Impact of Tweets on Open Source Project Popularity and New Contributors. In *Proceedings of the 44th International Conference on Software Engineering (ICSE)*.

[30] Fabio Ferreira, Luciana Lourdes Silva, and Marco Tulio Valente. 2020. Turnover in Open-Source Projects: The Case of Core Developers. In *Proceedings of the 34th Brazilian Symposium on Software Engineering*. 447–456.

[31] Matthieu Foucault, Marc Palyart, Xavier Blanc, Gail C Murphy, and Jean-Rémy Falleri. 2015. Impact of developer turnover on quality in open-source software. In *Proceedings of the 23th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*. 829–841.

[32] Felipe Fronchetti, Igor Wiese, Gustavo Pinto, and Igor Steinmacher. 2019. What attracts newcomers to onboard on oss projects? tl; dr: Popularity. In *IFIP International Conference on Open Source Systems*. Springer, 91–103.

[33] Tanner Fry, Tapajit Dey, Andrey Karnauch, and Audris Mockus. 2020. A dataset and an approach for identity resolution of 38 million author ids extracted from 2b git commits. In *Proceedings of the 17th International Conference on Mining Software Repositories (MSR)*. 518–522.

[34] Marco Gerosa, Igor Wiese, Bianca Trinkenreich, Georg Link, Gregorio Robles, Christoph Treude, Igor Steinmacher, and Anita Sarma. 2021. The shifting sands of motivation: Revisiting what drives contributors in open source. In *Proceedings of the 43rd International Conference on Software Engineering (ICSE)*. 1046–1058.

[35] Mariam Guizani, Thomas Zimmermann, Anita Sarma, and Denae Ford. 2022. Attracting and retaining oss contributors with a maintainer dashboard. In *Proceedings of the 44th International Conference on Software Engineering (ICSE): Software Engineering in Society*. 36–40.

[36] Jungpil Hahn, Jae Yoon Moon, and Chen Zhang. 2006. Impact of social ties on open source project team formation. In *IFIP International Conference on Open Source Systems*. Springer, 307–317.

[37] Jungpil Hahn, Jae Yun Moon, and Chen Zhang. 2008. Emergence of new project teams from open source software developer networks: Impact of prior collaboration ties. *Information Systems Research* 19, 3 (2008), 369–391.

[38] Joseph M Hilbe. 2011. *Negative binomial regression*. Cambridge University Press.

[39] Corey Jergensen, Anita Sarma, and Patrick Wagstrom. 2011. The onion patch: migration in open source ecosystems. In *Proceedings of the 19th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*. 70–80.

[40] Mitchell Joblin, Wolfgang Mauerer, Sven Apel, Janet Siegmund, and Dirk Riehle. 2015. From developer networks to verified communities: A fine-grained approach. In *Proceedings of the 37th International Conference on Software Engineering (ICSE)*. 563–573.

[41] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M German, and Daniela Damian. 2014. The promises and perils of mining github. In *Proceedings of the 11st International Conference on Mining Software Repositories (MSR)*. 92–101.

[42] Hemank Lamba, Asher Trockman, Daniel Armanios, Christian Kästner, Heather Miller, and Bogdan Vasilescu. 2020. Heard it through the Gitvine: an empirical study of tool diffusion across the npm ecosystem. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*. 505–517.

[43] Jey Han Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368* (2016).

[44] Daniel Z Levin and Rob Cross. 2004. The strength of weak ties you can trust: The mediating role of trust in effective knowledge transfer. *Management science* 50, 11 (2004), 1477–1490.

[45] Yuxing Ma, Chris Bogart, Sadika Amreen, Russell Zaretzki, and Audris Mockus. 2019. World of Code: An infrastructure for mining the universe of open source VCS data. In *Proceedings of the 16th International Conference on Mining Software Repositories (MSR)*. 143–154.

[46] Tadej Matek and Svit Timej Zebec. 2016. Github open source project recommendation system. *arXiv preprint arXiv:1602.02594* (2016).

[47] Courtney Miller, Christian Kästner, and Bogdan Vasilescu. 2023. "We Feel Like We're Winging It:" A Study on Navigating Open-Source Dependency Abandonmenment. In *Proceedings of the ACM SIGSOFT Joint Meeting on the Foundations of Software Engineering (ESEC/FSE)*.

[48] Courtney Miller, David Gray Widder, Christian Kästner, and Bogdan Vasilescu. 2019. Why do people give up FLOSSing? A study of contributor disengagement in open source. In *IFIP International Conference on Open Source Systems*. 116–129.

[49] Audris Mockus, David M Weiss, and Ping Zhang. 2003. Understanding and predicting effort in software projects. In *Proceedings of the 25th International Conference on Software Engineering (ICSE)*. 274–284.

[50] Gang Peng. 2019. Co-membership, networks ties, and knowledge flow: An empirical investigation controlling for alternative mechanisms. *Decision Support Systems* 118 (2019), 83–90.

[51] Huilian Sophie Qiu, Yucen Lily Li, Susmita Padala, Anita Sarma, and Bogdan Vasilescu. 2019. The signals that potential contributors look for when choosing open-source projects. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (2019), 1–29.

[52] Huilian Sophie Qiu, Alexander Nolte, Anita Brown, Alexander Serebrenik, and Bogdan Vasilescu. 2019. Going farther together: The impact of social capital on sustained participation in open source. In *Proceedings of the 41st International Conference on Software Engineering (ICSE)*. 688–699.

[53] Israr Qureshi and Yulin Fang. 2011. Socialization in open source software projects: A growth mixture modeling approach. *Organizational Research Methods* 14, 1 (2011), 208–238.

[54] Sam Ransbotham and Gerald C Kane. 2011. Membership turnover and collaboration success in online communities: Explaining rises and falls from grace in Wikipedia. *Mis Quarterly* (2011), 613–627.

[55] Dirk Riehle, Philipp Riemer, Carsten Kolassa, and Michael Schmidt. 2014. Paid vs. volunteer work in open source. In *2014 47th Hawaii International Conference on System Sciences*. 3286–3295.

[56] Kazumi Saito, Ryohei Nakano, and Masahiro Kimura. 2008. Prediction of information diffusion probabilities for independent cascade model. In *International conference on knowledge-based and intelligent information and engineering systems*. Springer, 67–75.

[57] Carlos Santos, George Kuk, Fabio Kon, and John Pearson. 2013. The attraction of contributors in free and open source software projects. *The Journal of Strategic Information Systems* 22, 1 (2013), 26–45.

[58] Fabio Santos, Bianca Trinkenreich, João Felipe Pimentel, Igor Wiese, Igor Steinmacher, Anita Sarma, and Marco A Gerosa. 2022. How to choose a task? Mismatches in perspectives of newcomers and existing contributors. In *2022 International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 114–124.

[59] Jan-Felix Schrape. 2019. Open-source projects as incubators of innovation: From niche phenomenon to integral part of the industry. *Convergence* 25, 3 (2019), 409–427.

[60] André Luis Schwerz, Rafael Liberato, Igor Scaliante Wiese, Igor Steinmacher, Marco Aurélio Gerosa, and João Eduardo Ferreira. 2012. Prediction of developer participation in issues of open source projects. In *2012 Brazilian Symposium on Collaborative Systems*. 109–114.

[61] Luiz Philipe Serrano Alves, Igor Scaliante Wiese, Ana Paula Chaves, and Igor Steinmacher. 2021. How to Find My Task? Chatbot to Assist Newcomers in Choosing Tasks in OSS Projects. In *International Workshop on Chatbot Research and Design*. Springer, 90–107.

[62] Leif Singer, Fernando Figueira Filho, and Margaret-Anne Storey. 2014. Software engineering at the speed of light: how developers stay current using twitter. In *Proceedings of the 36th International Conference on Software Engineering*. 211–221.

[63] Igor Steinmacher, Tayana Conte, Marco Aurélio Gerosa, and David Redmiles. 2015. 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*. 1379–1392.

[64] Igor Steinmacher, Marco Aurélio Gerosa, and David Redmiles. 2014. Attracting, onboarding, and retaining newcomer developers in open source software projects. In *Workshop on Global Software Development in a CSCW Perspective*.

[65] Igor Steinmacher, Marco Aurelio Graciotto Silva, Marco Aurelio Gerosa, and David F Redmiles. 2015. A systematic literature review on the barriers faced by newcomers to open source software projects. *Information and Software Technology* 59 (2015), 67–85.

[66] Xiaobing Sun, Wenyuan Xu, Xin Xia, Xiang Chen, and Bin Li. 2018. Personalized project recommendation on GitHub. *Science China Information Sciences* 61 (2018), 1–14.

[67] Lin Tan, Chen Liu, Zhenmin Li, Xuanhui Wang, Yuanyuan Zhou, and Chengxiang Zhai. 2014. Bug characteristics in open source software. *Empirical software engineering* 19, 6 (2014), 1665–1705.

[68] Xin Tan, Minghui Zhou, and Zeyu Sun. 2020. A first look at good first issues on GitHub. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*. 398–409.

[69] Asher Trockman, Shurui Zhou, Christian Kästner, and Bogdan Vasilescu. 2018. Adding sparkle to social coding: an empirical study of repository badges in the npm ecosystem. In *Proceedings of the 40th International Conference on Software Engineering (ICSE)*. 511–522.

[70] Jason Tsay, Laura Dabbish, and James Herbsleb. 2014. Influence of social and technical factors for evaluating contribution in GitHub. In *Proceedings of the 36th International Conference on Software Engineering (ICSE)*. 356–366.

[71] Marat Valiev, Bogdan Vasilescu, and James Herbsleb. 2018. Ecosystem-level determinants of sustained activity in open-source projects: A case study of the PyPI ecosystem. In *Proceedings of the 26th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*. 644–655.

[72] Bogdan Vasilescu, Kelly Blincoe, Qi Xuan, Casey Casalnuovo, Daniela Damian, Premkumar Devanbu, and Vladimir Filkov. 2016. The sky is not the limit: Multitasking across GitHub projects. In *Proceedings of the 38th International Conference on Software Engineering (ICSE)*. ACM, 994–1005.

[73] Bogdan Vasilescu, Alexander Serebrenik, Mathieu Goeminne, and Tom Mens. 2014. On the variation and specialisation of workload—a case study of the Gnome ecosystem community. *Empirical Software Engineering* 19, 4 (2014), 955–1008.

[74] Georg Von Krogh, Sebastian Spaeth, and Karim R Lakhani. 2003. Community, joining, and specialization in open source software innovation: a case study. *Research policy* 32, 7 (2003), 1217–1241.

[75] Mairieli Wessel, Bruno Mendes De Souza, Igor Steinmacher, Igor S Wiese, Ivanilton Polato, Ana Paula Chaves, and Marco A Gerosa. 2018. The power of bots: Characterizing and understanding bots in OSS projects. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (2018), 1–19.

[76] Chengxi Zang, Peng Cui, Chaoming Song, Christos Faloutsos, and Wenwu Zhu. 2017. Quantifying structural patterns of information cascades. In *Proceedings of the 26th International Conference on World Wide Web Companion*. 867–868.

[77] Minghui Zhou and Audris Mockus. 2011. Does the initial environment impact the future of developers?. In *Proceedings of the 33rd International Conference on Software Engineering (ICSE)*. 271–280.

[78] Minghui Zhou and Audris Mockus. 2012. What make long term contributors: Willingness and opportunity in OSS community. In *Proceedings of the 34th International Conference on Software Engineering (ICSE)*. 518–528.

[79] Minghui Zhou and Audris Mockus. 2014. Who will stay in the floss community? modeling participant's initial behavior. *IEEE Transactions on Software Engineering* 41, 1 (2014), 82–99.

Received 2023-02-02; accepted 2023-07-27