GFI-Bot: Automated Good First Issue Recommendation on GitHub

Hao He*
Peking University
Beijing, China
heh@pku.edu.cn

Haonan Su*
Peking University
Beijing, China
haonan.su@pku.edu.cn

Wenxin Xiao*
Peking University
Beijing, China
wenxin.xiao@stu.pku.edu.cn

Runzhi He*
Peking University
Beijing, China
rzhe@pku.edu.cn

Minghui Zhou*†
Peking University
Beijing, China
zhmh@pku.edu.cn

ABSTRACT

To facilitate newcomer onboarding, GitHub recommends the use of "good first issue" (GFI) labels to signal issues suitable for newcomers to resolve. However, previous research shows that manually labeled GFIs are scarce and inappropriate, showing a need for automated recommendations. In this paper, we present GFI-Bot (accessible at https://gfibot.io), a proof-of-concept machine learning powered bot for automated GFI recommendation in practice. Project maintainers can configure GFI-Bot to discover and label possible GFIs so that newcomers can easily locate issues for making their first contributions. GFI-Bot also provides a high-quality, up-to-date dataset for advancing GFI recommendation research.

CCS CONCEPTS

• **Software and its engineering → Open source model**.

KEYWORDS

open-source software, onboarding, good first issue, software bot

ACM Reference Format:
Hao He, Haonan Su, Wenxin Xiao, Runzhi He, and Minghui Zhou. 2022. GFI-Bot: Automated Good First Issue Recommendation on GitHub. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '22), November 14–18, 2022, Singapore, Singapore.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3540250.3558922

1 INTRODUCTION

For an open-source software (OSS) project to sustain itself in the long term, it must be able to attract, onboard, and retain newcomers [22, 23]. However, OSS onboarding is known to be very difficult with formidable knowledge, technical, and social barriers [12, 14], among which newcomers report *finding a way to start* as the most challenging barrier to contribute [13]. To make onboarding easier for newcomers, it is generally recommended that OSS projects should curate a list of development tasks for newcomers [15]. For this purpose, GitHub recommends project maintainers to use "good first issue" (GFI) labels to explicitly highlight newcomer contribution opportunities in open issues [3].

Despite the increasing popularity of GFI labels, they are currently added by project maintainers through manual labeling, which has two major limitations [6, 16]. First, manually labeled GFIs tend to be scarce and insufficient for newcomers. For example, in a sample of 46 highly popular GitHub projects that have adopted GFI labels, only 1.5% of their issues are labeled with GFI [6]. Such scarcity limits GFI's usefulness and frustrates newcomers [2], as indicated by a Reddit post [1]: *looking to contribute to open source, but issues labeled good-first-issue are all taken.* Second, project maintainers may miss GFIs or misjudge an issue as a GFI due to the cognitive gap between experts and novices [5, 18]. Tan et al. [16] find that 40.9% of GFIs are not solved by newcomers, 31.2% of newcomers fail to solve a GFI even after several attempts, and newcomers complain most about GFIs being inappropriate. Therefore, we believe it will be extremely beneficial to have an automated recommender system that learns the characteristics of GFIs from historical issue resolution data and recommends likely GFIs from the latest open issues.

In our previous work [21], we proposed RecGFI, a machine learning (ML) powered approach for GFI recommendation. RecGFI models an issue with heterogeneous features from issue content, reporter & repository background, and issue comments and events. It further learns GFI characteristics from issues actually resolved by newcomers using an XGBoost classifier [7]. We have demonstrated the effectiveness of RecGFI through two datasets built from GHTorrent [9] and a preliminary real-world evaluation. However, the practical usefulness of our previous RecGFI implementation is still limited with the following gaps:

- The datasets used in RecGFI rely on GHTorrent and thus cannot be updated regularly (the last GHTorrent dump is released in March 2021). They may also contain noises due to a data gap in GHTorrent [10, 20]. Ideally, a GFI recommender system should be able to learn from an up-to-date dataset of historical resolved issues while maintaining efficiency and scalability on real OSS projects (whose data volume is often large).
- RecGFI does not have a mechanism for project maintainers to integrate the recommendations into their project workflows.
- RecGFI also does not have an interface for newcomers to explore and discover GFIs based on their skills, personal interests, etc.

*All authors are also affiliated with the Key Laboratory of High Confidence Software Technologies, Ministry of Education, Beijing, China
†Corresponding Author

ESEC/FSE '22, November 14–18, 2022, Singapore, Singapore

ACM ISBN 978-1-4503-9413-0/22/11…$15.00
https://doi.org/10.1145/3540250.3558922

In this paper, we present GFI-Bot, a RecGFI-based proof-of-concept implementation to bridge the aforementioned gaps. The key characteristics of GFI-Bot are:

- GFI-Bot builds dataset and trains ML models using only data from GitHub APIs so that GFI-Bot can work on any GitHub project. To minimize GitHub API rate consumption, GFI-Bot supports incremental updates and training based on previously collected repository data and trained ML models.
- GFI-Bot can be configured by project maintainers to perform certain actions on open issues (e.g., adding labels and leaving comments) based on its recommendations.
- GFI-Bot provides an interactive web portal for newcomers to explore GitHub projects and GFI recommendations; project maintainers can also use the web portal for monitoring the status of GFI-Bot (e.g., model performance) on their repositories.

We prototype and demonstrate GFI-Bot on 100 GitHub projects, most of which are renowned OSS projects such as pandas, Scikit-learn, Material UI, VS Code, etc. GFI-Bot can generate GFI recommendations for the 100 projects and perform incremental updates on a regular basis. The trained models can reach a global performance of up to 0.8302 AUC in the current dataset of 159,919 issues. To the best of our knowledge, GFI-Bot is the *first* ML-powered practical tool for automated GFI recommendation in OSS projects.[1]

GFI-Bot is available at https://gfibot.io. Its source code is open-source (licensed under GPL-v3) and accessible at https://github.com/osslab-pku/gfi-bot. The GFI recommendation dataset is available at https://doi.org/10.5281/zenodo.6665931. We provide a short introduction video for GFI-Bot at http://video.gfibot.io.

2 BACKGROUND

2.1 The GFI Recommendation Problem

Given a list of open issues in a GitHub project, the goal of a GFI recommender system is to find a subset of possible GFIs or to rank the issues based on their predicted GFI probability. In either case, the underlying problem is to learn a model $f(\cdot)$ from historical issue data to make GFI predictions on open issues, a typical binary classification problem in machine learning.

There are multiple ways to define ground truth in historical issues. For example, Huang et al. [11] use issues with GFI (or similar) labels as positive samples and the remaining issues as negative samples for model training. However, there is a discrepancy between issues with a GFI label and issues resolved by newcomers [16], so a model trained from this ground truth may be biased toward maintainers' judgment instead of newcomers' perception. In RecGFI, we use issues *actually resolved by newcomers* as positive samples and the remaining issues as negative samples. We consider an issue resolver as a newcomer if they have contributed less than k commits before this issue is resolved. Here k is offered as a hyperparameter due to our observation that newcomers may continue to seek GFIs even if they have successfully made one or two contributions [16].

[1]Existing tools in the wild only provide *label-based* recommendations (i.e., they only list issues with "good first issue" or similar labels), e.g., https://goodfirstissue.dev/, https://goodfirstissues.com/, https://github.com/nodejs/node/contribute. Although GitHub claims to be developing an ML-based GFI recommendation feature [4], we cannot find any evidence of its deployment in practice.

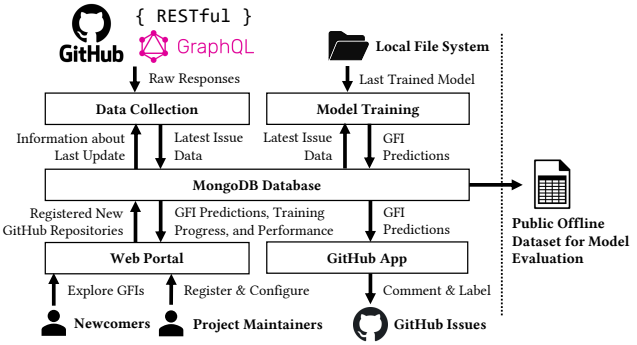


Figure 1: An Overview of the GFI-Bot Architecture

During ground truth construction, it is important to restore issues to their precise state before issue resolution. Otherwise, information from the future may be leaked into the dataset and leads to overly optimistic performance evaluation results [17].

2.2 RecGFI

The remaining challenges of GFI recommendation is to identify viable features from issues and design an appropriate model for learning and prediction. In RecGFI, we have conducted extensive feature engineering and identified a set of features computed from issue content (titles, descriptions, and labels), issue background (reporter and project information), and issue dynamics (comments, events, and participants).[2] Theoretically, all features can be also computed using data from GitHub APIs but features related to GitHub-wide user profiles (e.g., number of stars received in their contributed repositories) require an excessive amount of API rate to compute. To limit API rate consumption and maintain scalability, we exclude these features for issue participants (i.e., commenters and event actors) in the current GFI-Bot implementation.

RecGFI achieves the highest overall performance when using XGBoost [7] as the underlying ML model. XGBoost also bears advantages in production ML systems. For example, it is possible to deploy XGBoost on low-cost cloud instances as its training and inference are highly efficient and CPU-friendly. It can also update trained models incrementally with new training data. In GFI-Bot, we use the same XGBoost configuration as in RecGFI.

3 GFI-BOT IMPLEMENTATION

Figure 1 presents an overview of the current GFI-Bot architecture which has four main modules: data collection, model training, a web portal, and a GitHub App. The four modules work independently as standalone processes and are decoupled by a central MongoDB database, which serves as "a single source of truth" and contains information such as registered repositories, issue datasets, training logs, GFI predictions, etc. A GFI recommendation dataset can be periodically dumped from the MongoDB database for conducting sophisticated offline evaluations or developing new ML models.

GFI-Bot works on a list of GitHub repositories (which can be configured during initial deployment or later added by registered users in the web portal). For each repository, the data collection module collects and incrementally updates necessary issue data

[2]See the original paper [21] for more details.

using GitHub REST and GraphQL APIs. The data will be used by the model training module to update the last trained models and make predictions for all open issues. During data collection and model training, project maintainers can use the web portal to inspect training progress, performance, and the GFI predictions. The same web portal can also be used by newcomers to explore registered repositories and recommended GFIs in these repositories. Finally, project maintainers can configure a GitHub App to automatically comment or label issues in their repository based on the GFI predictions.

At the time of writing, GFI-Bot is deployed on an Oracle Linux server with 4 cores, 32GiB memory, and 96GB storage. All four modules, except the web portal frontend, are implemented with 6,592 lines of Python code. The web portal frontend is implemented with 5,058 lines of TypeScript, 2,289 lines of JavaScript, and 1,163 lines of CSS using the React.js framework. In the remainder of this section, we will introduce each module in more detail.

3.1 Data Collection

For each configured repository, the data collection module needs to collect all necessary data for feature computation. Since the computation of many features requires full repository history (e.g., the # of previous commits by the issue reporter at the time of issue creation), the module collects repository metadata, all commits, all issues, and all pull requests (PRs) using the RESTful API (which consumes less API rate with large pagination). The module also collects user global GitHub profiles for all issue reporters and repository owners (including issues, PRs, commits, code reviews, and stars received) to support the characterization of their OSS expertise in general. This is implemented using the GraphQL API with more functionalities and flexibility. Since GitHub imposes an API rate limit of 5000 per hour for each token, the initial collection may take up to several hours, but all later collections will be done incrementally based on the previously collected data which will be much faster.

The module then outputs a dataset for all resolved issues (for training and prediction) and all open issues (for prediction). If an issue is closed by a PR or a commit, it will be marked as a resolved issue and the PR/commit author will be recorded as the resolver. For each resolved issue, the module computes the features and generates data points at two time points: issue creation and issue resolution. This is because an open issue can be of any state between the two time points and we want the model to also learn the dynamics after issue creation and effectively make predictions for open issues at all stages (e.g., initial report, discussed, labeled, triaged, etc.). Finally, for open issues, the module computes features with their current state and marks them as test data with no ground truth labels.

We build a latest dataset with the same 100 projects used in the RecGFI evaluation. At the time of writing, GFI-Bot have collected 2,032,988 commits, 907,869 issues (114,799 open) and 941,232 PRs. Among the 793,070 closed issues, we have identified 159,919 issues resolved by commits/PRs and they will be used for model training. This dataset can be incrementally updated on a regular basis.

3.2 Model Training

With a curated dataset from the data collection module, the model training module is relatively straightforward. For a list of resolved issues (i.e., training data) and a given k, the module splits issues into batches (to avoid exceeding the memory limit in cloud instances) and iteratively trains an XGBoost model. If an older model is already available (i.e., the case of incremental training), the module identifies newly resolved issues and only uses the new issues to update the old model. To evaluate model performance, the module additionally trains a model using 90% of (older) training data and computes AUC, accuracy, and recall with the remaining 10% of (newer) training data. The performance will be reported both globally and per project so that maintainers can establish confidence on GFI-Bot and decide whether to adopt GFI-Bot in their projects.

Table 1: GFI-Bot Performance on the 100 GitHub Projects used in RecGFI (159,919 Resolved Issues in Total)

k	# Newcomer-Resolved	AUC	Accuracy	Recall
1	17,147	0.8172	0.7082	0.7699
2	21,649	0.8195	0.7042	**0.7892**
3	24,533	**0.8302**	**0.7625**	0.7197
4	26,623	0.8123	0.7599	0.6868
5	28,308	0.8162	0.7626	0.7001

In total, five different models will be trained for $k \in \{1, 2, 3, 4, 5\}$, and five different predictions will be made for each open issue. Project maintainers can configure which k to use for their projects in the web portal (Section 3.3). Our general recommendations are $k = 0$ for projects with high onboarding barriers and a limited number of easy issues, and $k = 3$ for projects with many easy issues and successfully onboarded newcomers.

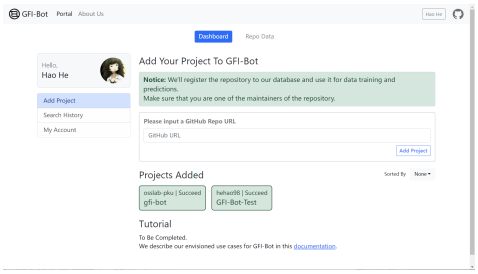
The global performance at the time of writing (for the 100 GitHub projects) is presented in Table 1, showing that the model can learn GFI characteristics well with >0.8 AUCs and 0.7~0.8 accuracy. Within-project performance ranges from 0.6978~1 in terms of AUC. However, we observe that model performance can fluctuate significantly with different batch sizes, training orders, feature preprocessing, model hyperparameters, etc., and we are currently working on the identification of an optimal training strategy.

3.3 Web Portal

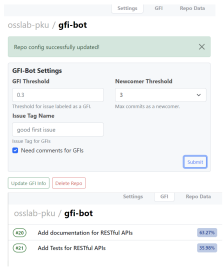
The GFI-Bot web portal serves two purposes: for project maintainers, they use the web portal to register their repositories and monitor the current status of GFI-Bot in their repositories; for newcomers, they use the web portal to browse through repositories indexed by GFI-Bot and find possible GFIs. We implement two main pages in the web portal to serve the two purposes.

The first page is for repository registration, data collection and performance monitoring, and GitHub App configuration. To access this page, a user must log in through an existing GitHub account. Their login API token will be used to perform data collection for their registered repositories. For each registered repository, the page displays data collection and model training progress, model performance, and currently recommended GFIs. It also provides a GitHub App configuration panel for configuring GFI-Bot to perform actions in the registered repository (Section 3.4).

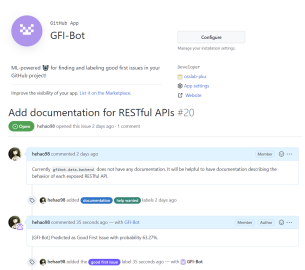
The second page is for listing repositories and recommended GFIs for newcomers. The repositories can be ordered by popularity, programming language, tags, etc., to support newcomer explorations. For each repository, the page lists GFIs by their predicted probability, and users can find more information about a specific issue by clicking and expanding the issue.



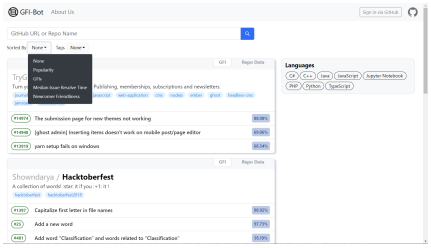
(a) Registering Repositories



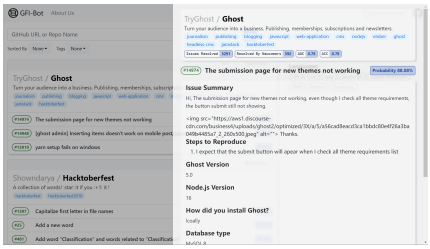
(b) Configuring GFI-Bot



(c) Installing GitHub App



(d) Exploring Repositories and GFIs



(e) Viewing a Specific GFI

Figure 2: GFI-Bot Interfaces for Project Maintainers (2a, 2b, 2c) and Newcomers (2d, 2e)

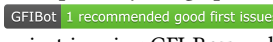
3.4 GitHub App

The final module of GFI-Bot is a GitHub App (available at https://github.com/apps/GFI-Bot) that can be configured by project maintainers to automatically comment or label issues in their repository based on the GFI predictions. To avoid being perceived as noisy, we provide flexible configurations for specifying GFI-Bot's exact behavior on GitHub, following design principles suggested in the software engineering bot literature [8, 10, 19]. Current configuration options include the k, the classification threshold, whether to comment on issues, which specific label to add on GFIs, etc. We plan to add more fine-grained behavior controls (e.g., comment template, when to label & comment) and project integration configurations (e.g., include or exclude issues with certain labels) in the future.

4 USE CASES

In this section, we describe how GFI-Bot can be used in practice by presenting two use cases (one for project maintainers and one for newcomers) and the roles of GFI-Bot in the two cases.

4.1 Project Maintainers

Suppose an OSS project is willing to see contributions from newcomers but its maintainers do not have time to curate and label GFIs. In this case, the maintainer can register their project in GFI-Bot (Figure 2a). Upon registration, GFI-Bot will immediately begin to fetch data, build a dataset, make predictions for open issues in that project, and send a notification when this process has finished. Then, they can find predicted GFIs by clicking the project card and tuning configurations for their project (Figure 2b). If the results are satisfactory, they can then install the GitHub App to label and comment on issues based on their specified configurations (Figure 2c). Finally, they can add a repository badge provided by GFI-Bot in their README (e.g., ) so that newcomers can know their project is using GFI-Bot and click the badge to see the project's GFI recommendations in GFI-Bot web portal.

4.2 Newcomers

Suppose a newcomer is willing to contribute to OSS but has no idea where to start. In this case, they can access the GFI-Bot web portal (https://gfibot.io/) to browse through a list of newcomer-friendly projects with many possible GFIs (Figure 2d). They can further rank or filter the projects that match their personal expertise and interests (based on programming languages, tags, etc.). During exploration, if they have found some projects or issues of interest, they can click on the issue to find out more information (Figure 2e).

5 CONCLUSION AND FUTURE WORK

In this paper, we have presented GFI-Bot, an ML-powered bot prototype for automated GFI recommendation on GitHub. We have demonstrated how GFI-Bot can effectively recommend GFIs in 100 GitHub projects and how it can support OSS newcomer onboarding and help projects that are in need of newcomers.

Currently, GFI-Bot is still an early prototype that needs to be further improved and evaluated in many directions. The data collection module is still costly in terms of API rate consumption and we plan to explore more lightweight approaches. For model training, we plan to explore how feature selection, batch size, dataset balancing, and model hyperparameters may affect model performance to derive an optimal training strategy in the low-resource cloud setting. For the web portal and the GitHub app, significant engineering effort is still needed for improving user experience (e.g., responsiveness and transparency). Finally, after GFI-Bot reaches a certain level of maturity, we plan to conduct real-world user studies with both OSS newcomers and projects as a comprehensive evaluation of a GFI recommender system in practice.

ACKNOWLEDGMENTS

This work is supported by the National Key R&D Program of China Grant 2018YFB1004201 and the National Natural Science Foundation of China Grant 61825201.

REFERENCES

[1] 2020. *Looking to contribute to open source, but issues labeled good-first-issue are all taken.* Retrieved June 15, 2022 from https://www.reddit.com/r/learnprogramming/comments/j48nkn/looking_to_contribute_to_open_source_but_issues/

[2] 2020. *The "good first issue" myth.* Retrieved June 15, 2022 from https://dzhavat.github.io/2020/07/08/the-good-first-issue-myth.html

[3] 2022. *Encouraging helpful contributions to your project with labels.* Retrieved June 14, 2022 from https://docs.github.com/en/communities/setting-up-your-project-for-healthy-contributions/encouraging-helpful-contributions-to-your-project-with-labels

[4] 2022. *How we built the good first issues feature.* Retrieved June 16, 2022 from https://github.blog/2020-01-22-how-we-built-good-first-issues/

[5] 2022. *What makes a good first issue?* Retrieved June 15, 2022 from https://dev.to/cerchie/what-makes-a-good-first-issue-4fn0

[6] Jan Willem David Alderliesten and Andy Zaidman. 2021. An Initial Exploration of the "Good First Issue" Label for Newcomer Developers. In *14th IEEE/ACM International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE@ICSE 2021, Madrid, Spain, May 20-21, 2021*. IEEE, 117–118. https://doi.org/10.1109/CHASE52884.2021.00023

[7] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi (Eds.). ACM, 785–794. https://doi.org/10.1145/2939672.2939785

[8] Linda Erlenhov, Francisco Gomes de Oliveira Neto, and Philipp Leitner. 2020. An empirical study of bots in software development: characteristics and challenges from a practitioner's perspective. In *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*, Prem Devanbu, Myra B. Cohen, and Thomas Zimmermann (Eds.). ACM, 445–455. https://doi.org/10.1145/3368089.3409680

[9] Georgios Gousios and Diomidis Spinellis. 2012. GHTorrent: Github's data from a firehose. In *9th IEEE Working Conference of Mining Software Repositories, MSR 2012, June 2-3, 2012, Zurich, Switzerland*, Michele Lanza, Massimiliano Di Penta, and Tao Xie (Eds.). IEEE Computer Society, 12–21. https://doi.org/10.1109/MSR.2012.6224294

[10] Runzhi He, Hao He, Yuxia Zhang, and Minghui Zhou. 2022. Automating Dependency Updates in Practice: An Exploratory Study on GitHub Dependabot. *CoRR* abs/2206.07230 (2022). https://doi.org/10.48550/arXiv.2206.07230

[11] Yuekai Huang, Junjie Wang, Song Wang, Zhe Liu, Dandan Wang, and Qing Wang. 2021. Characterizing and Predicting Good First Issues. In *ESEM '21: ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, Bari, Italy, October 11-15, 2021*, Filippo Lanubile, Marcos Kalinowski, and Maria Teresa Baldassarre (Eds.). ACM, 13:1–13:12. https://doi.org/10.1145/3475716.3475789

[12] Christopher J. Mendez, Hema Susmita Padala, Zoe Steine-Hanson, Claudia Hilderbrand, Amber Horvath, Charles Hill, Logan Simpson, Nupoor Patil, Anita Sarma, and Margaret M. Burnett. 2018. Open source barriers to entry, revisited: a sociotechnical perspective. In *Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*, Michel Chaudron, Ivica Crnkovic, Marsha Chechik, and Mark Harman (Eds.). ACM, 1004–1015. https://doi.org/10.1145/3180155.3180241

[13] Ifraz Rehman, Dong Wang, Raula Gaikovina Kula, Takashi Ishio, and Kenichi Matsumoto. 2022. Newcomer OSS-Candidates: Characterizing Contributions of Novice Developers to GitHub. *Empir. Softw. Eng.* 27, 5 (2022), 109. https://doi.org/10.1007/s10664-022-10163-0

[14] Igor Steinmacher, Marco Aurélio Graciotto Silva, Marco Aurélio Gerosa, and David F. Redmiles. 2015. A systematic literature review on the barriers faced by newcomers to open source software projects. *Inf. Softw. Technol.* 59 (2015), 67–85. https://doi.org/10.1016/j.infsof.2014.11.001

[15] Igor Steinmacher, Christoph Treude, and Marco Aurélio Gerosa. 2019. Let Me In: Guidelines for the Successful Onboarding of Newcomers to Open Source Projects. *IEEE Softw.* 36, 4 (2019), 41–49. https://doi.org/10.1109/MS.2018.110162131

[16] Xin Tan, Minghui Zhou, and Zeyu Sun. 2020. A first look at good first issues on GitHub. In *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*, Prem Devanbu, Myra B. Cohen, and Thomas Zimmermann (Eds.). ACM, 398–409. https://doi.org/10.1145/3368089.3409746

[17] Feifei Tu, Jiaxin Zhu, Qimu Zheng, and Minghui Zhou. 2018. Be careful of when: an empirical study on time-related misuse of issue tracking data. In *Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2018, Lake Buena Vista, FL, USA, November 04-09, 2018*, Gary T. Leavens, Alessandro Garcia, and Corina S. Pasareanu (Eds.). ACM, 307–318. https://doi.org/10.1145/3236024.3236054

[18] Lev Vygotsky. 1978. Interaction between learning and development. *Readings on the Development of Children* 23, 3 (1978), 34–41.

[19] Mairieli Santos Wessel, Igor Wiese, Igor Steinmacher, and Marco Aurélio Gerosa. 2021. Don't Disturb Me: Challenges of Interacting with Software Bots on Open Source Software Projects. *Proc. ACM Hum. Comput. Interact.* 5, CSCW2 (2021), 1–21. https://doi.org/10.1145/3476042

[20] Marvin Wyrich, Raoul Ghit, Tobias Haller, and Christian Müller. 2021. Bots Don't Mind Waiting, Do They? Comparing the Interaction With Automatically and Manually Created Pull Requests. In *3rd IEEE/ACM International Workshop on Bots in Software Engineering, BotSE@ICSE 2021, Madrid, Spain, June 4, 2021*. IEEE, 6–10. https://doi.org/10.1109/BotSE52550.2021.00009

[21] Wenxin Xiao, Hao He, Weiwei Xu, Xin Tan, Jinhao Dong, and Minghui Zhou. 2022. Recommending Good First Issues in GitHub OSS Projects. In *Proceedings of the 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 21–29, 2022*. ACM. https://hehao98.github.io/files/2022-recgfi.pdf

[22] Minghui Zhou and Audris Mockus. 2012. What make long term contributors: Willingness and opportunity in OSS community. In *34th International Conference on Software Engineering, ICSE 2012, June 2-9, 2012, Zurich, Switzerland*, Martin Glinz, Gail C. Murphy, and Mauro Pezzè (Eds.). IEEE Computer Society, 518–528. https://doi.org/10.1109/ICSE.2012.6227164

[23] Minghui Zhou, Audris Mockus, Xiujuan Ma, Lu Zhang, and Hong Mei. 2016. Inflow and Retention in OSS Communities with Commercial Involvement: A Case Study of Three Hybrid Projects. *ACM Trans. Softw. Eng. Methodol.* 25, 2 (2016), 13:1–13:29. https://doi.org/10.1145/2876443