How to Find My Task? Chatbot to Assist Newcomers in Choosing Tasks in OSS Projects

Luiz Philipe Serrano Alves, Igor Scaliante Wiese,
Ana Paula Chaves, and Igor Steinmacher
luizphilipe@alunos.utfpr.edu.br,
{igor, chavesana, igorfs}@utfpr.edu.br

Universidade Tecnológica Federal do Paraná, Campo Mourão, Paraná, Brazil

Abstract. Open Source Software (OSS) is making a meteoric rise in the software industry since several big companies have entered this market. Unfortunately, newcomers enter these projects and usually lose interest in contributing because of several factors. This paper aims to reduce the problems users face when they walk their first steps into OSS projects: finding the appropriate task. This paper presents a chatbot that filters tasks to help newcomers choose a task that fits their skills. We performed a quantitative and a qualitative study comparing the chatbot with the current GitHub issue tracker interface, which uses labels to categorize and identify tasks. The results show that users perceived the chatbot as easier to use than the GitHub issue tracker. Additionally, users tend to interpret the use of chatbots as situational, helping mainly newcomers and inexperienced contributors.

Keywords: chatbots · open source software · software engineering · onboarding · skills · barriers · newcomers

1 Introduction

Open Source Software (OSS) projects are open collaboration communities in which geographically distributed people develop software. To remain sustainable and grow, several of these communities rely on volunteers [13]. However, it is known that newcomers often face barriers during the joining process, which may lead them to give up on contributing [27]. Among these barriers, the difficulty to find a task to work on is a crucial one [26, 3] since newcomers have a hard time attempting to match their interests and skill set to the community needs [34]. Still, the communities often expect new members to find their tasks instead of giving them specific work to do [33].

The literature [28] suggests that newcomers need specific guidance since finding the first task depends on the difficulty level, technology and modules affected. To help newcomers find tasks, social coding platforms, such as GitHub encourage the project maintainers to label the issues according to the difficulty level or required skills. However, the labeling process may be complex and time-consuming [4]. To mitigate this problem, Santos *et. al.* [25] proposed an ap-

proach that automatically labels the tasks by predicting the potentially required libraries for each task.

Still, newcomers are unfamiliar with the OSS tools and infrastructure, which bring many challenges when navigating the tasks on a traditional issue tracker interface [16]. Dominic *et al.* [9] suggests that a conversational interface may help engage the newcomers fully by recommending projects, artifacts, and experts as well as choosing an appropriate task. However, chatbots are not yet common in the OSS domain. A large number of GitHub projects adopt bots that assist contributors by performing repetitive tasks [35], such as quality assurance tasks (e.g., automating code reviews, assigning reviewers, reporting failures, etc. [35]) and artifacts recommendation [21]. Although these bots, in some sense, communicate with the community by, for example, posting comments or acting on mentions to them, they are not designed to hold interactive conversations with the community members.

In this paper, we walk toward using a conversational, interactive interface to support newcomers when finding the first task. We developed a chatbot that assists the users in navigating their skill set to select an appropriate task. We evaluated the acceptance of the before-mentioned chatbot by performing a user study that compares this tool to the traditional GitHub issue tracker. Our study focuses on answering the following research question:

RQ. *How does using a chatbot to find tasks compare to the use of the GitHub issue tracker?*

We conducted a survey-based within-subject study with 40 participants, including graduate students and industry practitioners that may or may not have previous experience with GitHub and OSS. Participants assessed each task-finder tool (chatbot vs. GitHub issue tracker) individually and comparatively, in the latter case, by informing which one they preferred. We compared the perceived usefulness and perceived ease of use of the tools according to the Technology Acceptance Model (TAM) [8]. Additionally, we analyzed the participants' qualitative feedback through a thematic analysis.

Our results show that participants perceive both the search feature and the chatbot as similarly useful, but the chatbot as easier to use. This result indicates that chatbots could be an effective way to help newcomers navigate through and find appropriate tasks. The qualitative analysis revealed positive aspects of the chatbot compared to the GitHub issue tracker and improvement opportunities. Our findings demonstrate that using a chatbot to find a task, as proposed by Dominic *et. al.* [9], is feasible and relevant to support newcomers in OSS.

2 Related Work

Advances in natural language processing (NLP) and machine learning (ML) increased the adoption of chatbots in numerous applications. In the past decades, many companies have developed chatbots to the many existing messaging tools, such as Facebook Messenger, Skype, Telegram, and Slack. The website BotList [1]

[1] https://botlist.co/

shows chatbots that are currently available in domains such as entertainment, education, games, leisure, culture, and sports.

The partnership between humans and chatbots makes it possible to create interactions around the activities performed by both [10]. Chatbots have shaped the interaction between companies and customers [5, 19], teachers and learners [6, 37], patients and healthcare providers [12, 17], among many other applications. These examples show that using a chatbot's interactive capabilities is useful for supporting users in their tasks.

In software engineering, chatbots are often called only "bots", regardless of their ability to hold an interactive conversation. Many of the bots for the software engineering domain are "non-conversational," in the sense that their focus is task automation [35, 29]. Even though they often post messages and comments they are not able to respond to the users in a conversational style [36].

Chatbots have been used in OSS projects, especially on GitHub[2], but they appear more timidly than the task-automation bots. The literature presents a couple of examples of chatbots in OSS. For example, Abdellatif and Shihab [2] developed a chatbot called MSRBot that supports users finding answers to questions about specificities of software repositories. They argue that the conversational nature of chatbots potentially lowers barriers that newcomers face when first contributing to a project.

Dominic *et. al* [9] raises a discussion about the ways that newcomers may benefit from interacting with chatbots during the onboarding process. The authors envision the creation of a chatbot to help newcomers find projects and tasks to contribute and provide guidance throughout their first contributions by playing the role of a mentor. Since these claims have not yet been further investigated in the literature, we have hypothesized that a chatbot would increase users' ease of use and perceived usefulness. Therefore, we opted to use an already validated instrument that covers these two constructs, the Technology Acceptance Model [8] (more details in Sect. 3.1).

3 Method

To conduct this research, we first set up the necessary tools and materials for the study; then, we collected the data through a within-subject study and analyzed the outcomes using mixed methods. The research steps are summarized in Figure 1 and detailed below.

3.1 Materials

Task classifier: finding an appropriate task requires ways to filter the available tasks according to the user's expectations. Labeling issues is a way to make it easier for users to find tasks according to their interests. In this study, we used the task classifier created by Santos *et al.* [25] that predicts labels to GitHub issues.

[2] Social Coding Platform, available at https://github.com

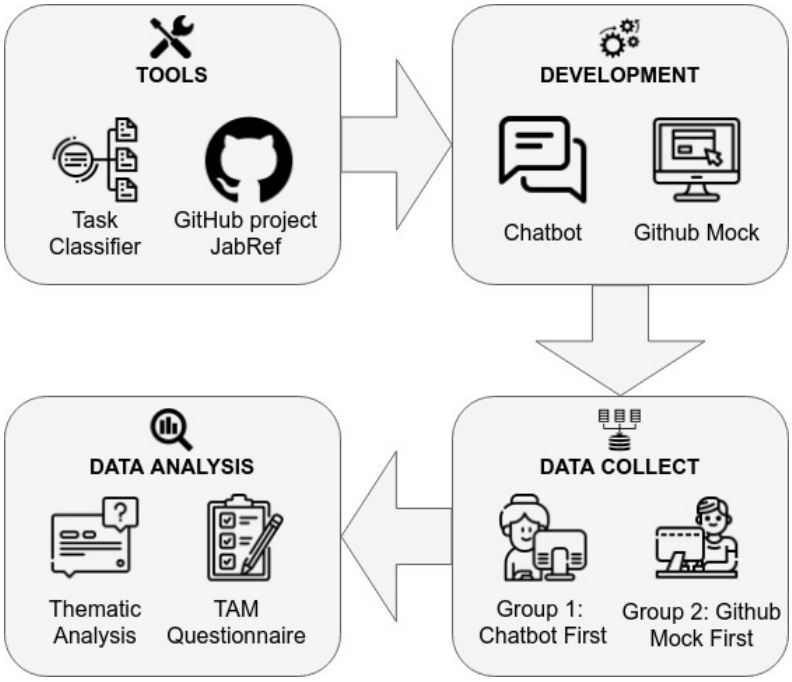

Fig. 1. A summary of the method followed in this study.

We have chosen this task classifier because of its assertiveness when predicting and relevance for newcomers. We used this task classifier to generate the labels used in the GitHub mock and filter the tasks in the chatbot interface.

GitHub project JabRef: to evaluate whether a chatbot help developers find tasks, it is necessary to choose a project which contains a history of open and closed issues. We chose the JabRef project [18], which is an open-source bibliography reference manager developed by a diverse community, including contributors that are not familiar with computer science. Also, the project is mature (created in 2003) and active on GitHub (migrated in 2014) with a large number of commits (15.7k+), 42 releases, 337 contributors, 2.7k closed issues, and 4.1k closed pull requests. JabRef is frequently investigated in other scientific studies [22, 11] and it interacts conveniently with the task classifier used in this study [25] since the classifier was trained with issues extracted from this project.

GitHub Mock: since we are not authorized to add labels to JabRef issue tracker on GitHub, we developed two tools that consume the output produced by the task classifier. The first tool is a mock of JabRef's issue tracker page. We used the mock to control the task labels that a participant sees on the issue tracker page to match with the task labels used in the chatbot environment. To build the mock, we first forked the JabRef project to a new repository. In this new

project, we copied all the issues found in the Santos *et. al.*'s dataset of issues [25] (the same dataset of issues that was used to train the classifier). Before including the issues in the new project, we labeled them with the categories predicted by the task classifier. Figure 2 show the GitHub mock-up resulting from this step.

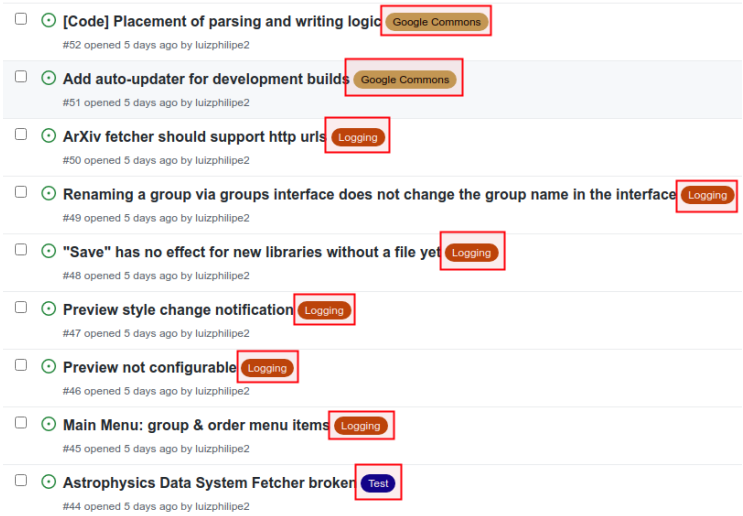

Fig. 2. Labels put on the GitHub cloned project.

Chatbot: besides the GitHub mock, we also developed the task-finder chatbot (in Brazilian Portuguese) using the Watson Assistant platform[3]. We designed the conversational flow based on the hierarchical categorization proposed in Santos *et. al.* [25] (Figure 3), which is based on JabRef's source code and was specifically generated for that project. The chatbot's conversation, as shown in Figure 4, is based on clickable predefined answers (buttons instead of free text) and aims to deliver a link to a task as soon as possible. The chatbot starts the conversation by introducing itself and asking whetherif the user has collaborated on an OSS project before. This question works to acclimate the respondent to the clickable answers. After that, the chatbot asks about the contributor's preference regarding the following possible options: *development*, *systems integration*, and *user interface*. Depending on the chosen option, the chatbot may follow up with options to filter the subcategories. Categories and sub-categories are organized in three layers (*User Interface*, *Development* and *System Integration*), as presented in the Figure 3. The categories and subcategories are specific to this project and were generated automatically by the task classifier. The options were purposely generated to fit the suggested layers and end up one in each layer level.

[3] https://cloud.ibm.com/catalog/services/watson-assistant

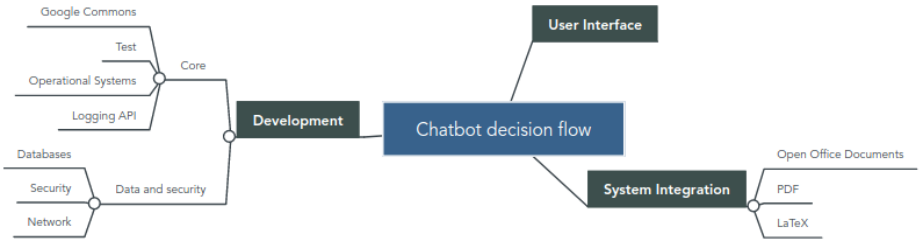

Fig. 3. Mind map of sub-categories that the tasks were categorized.

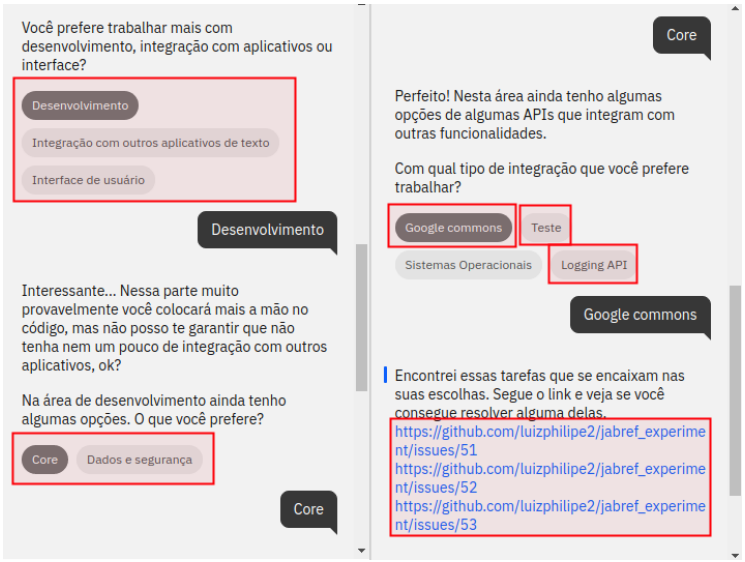

Fig. 4. Chatbot interaction, sub-categories as defined on Table 3 and predefined answers. The last level of the interaction is the same labels as put on Figure 2. The tasks link will conduce to GitHub.

Questionnaire: To answer the question *How does the use of a chatbot to find tasks compare to the use of the GitHub issue tracker?*, we conducted a questionnaire study using Google Forms, divided into three sections. The first two sections contain the instructions to interact with each tool (chatbot or GitHub issue tracker), along with questions about their experience. The experience of the participants was measured using a questionnaire based on version 1 of the TAM instrument [8] followed by open-ended questions aiming to capture general feedback about the pros and insights on how to improve each tool. We present the items in Table 1. The TAM items were measured using a 5-points Likert scale. We evaluated two factors: perceived usefulness (PU—the degree a person believes that the tool would enhance their job performance); and perceived ease

of use (PEOU—the degree to which a person believes that using the system would be free from effort).

Table 1. Items adapted from the Technology Acceptance Model [8] about perceived usefulness and perceived ease of use.

Factor	ID. Item Description
Perceived Usefulness	PU1. Using [tool] would enable me to accomplish tasks more quickly.
	PU2. Using [tool] would improve my job performance.
	PU3. Using [tool] in my job would increase my productivity.
	PU4. Using [tool] would enhance my effectiveness on the job.
	PU5. Using [tool] would make it easier to do my job.
	PU6. I would find [tool] useful in my job.
Perceived Ease of Use	PEOU1. Learning to operate [tool] would be easy for me.
	PEOU2. I would find it easy to get [tool] to do what I want it to do.
	PEOU3. My interaction with [tool] would be clear and understandable.
	PEOU4. I would find [tool] to be flexible to interact with.
	PEOU5. It would be easy for me to become skillful at using [tool].
	PEOU6. I would find [tool] easy to use.

Table 2 presents the descriptive statistics for each of the factors, and each treatment, which demonstrates the consistency of the instrument and that the scales found are reliable (Cronbach's $\alpha \geq 0.8$). It is important to note that the participants answered the same questions just after interacting with each tool. In the third part of the questionnaire, we included one question to ask the preferred tool (GitHub issue tracker, Chatbot, or Any) followed by demographics questions (presented in Figure 5).

Table 2. Descriptive statistics for the Likert Items

Factor	Tool	Mean	Median	SD	Cronbach's α
Perceived Usefulness (PU)	Chatbot	4.19	4	0.94	0.88
	GitHub Search	4.18	4	0.95	0.83
Perceived Ease of use (PEOU)	Chatbot	4.55	5	0.78	0.92
	GitHub Search	4.18	4	0.97	0.93

3.2 Procedures

Participants: We used a convenience sampling approach and openly invited prospective participants from the university where the research has been conducted and from the authors' contacts, kindly asking them to spread the word in their companies. Participants received the invitation link by email and voluntarily responded to it. We received 40 responses to the study, including undergraduate and graduate students as well as software developers. As depicted in Figure 5, 75% of the respondents are men, 50% are between 26-30 years old, and the majority of them have more than 3 years of industry experience and less than 2 years of OSS experience.

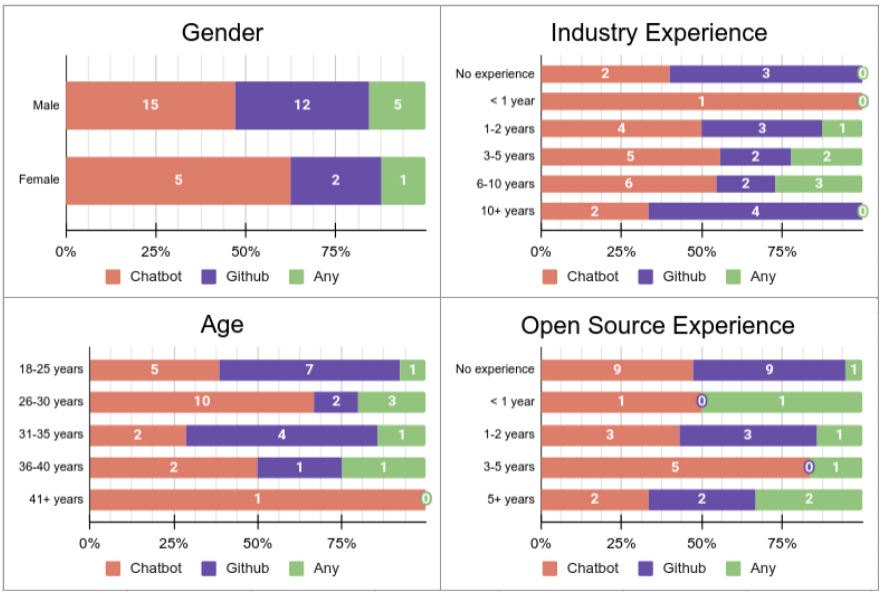

Fig. 5. Demographics of respondents.

Study Task: When the participants clicked on the invitation link, they were redirected to the Google Forms page with the study description. Participants were invited to simulate the scenario of choosing an issue from the JabRef project that they would like to work on. Since we performed a within-subject study, each participant had to complete this scenario twice, using each of the two provided tools: once using the mock of the GitHub issue tracker (control condition) and once using the chatbot (experimental condition). Before each scenario, the participants were introduced to the task they had to perform, followed by the sentence *"Now you are requested to interact with the [TOOL] available at the link below until you find an issue you consider suitable for you"*—replacing [TOOL] with "chatbot" and "GitHub Search." After completing each scenario, participants answered the TAM questionnaire for the ease of use and usefulness of the tool they had just used, along with the two open-ended questions about their impressions of the tool. This portion of the questionnaire was the same for both conditions. To avoid biases, we randomized the order in which the conditions were presented to the participants through a simple redirecting script. Almost half of the participants (19) used the chatbot first, and the remaining 21 used the GitHub issue tracker interface first.

After concluding both scenarios, the participants answered the comparative and demographics questions and submitted the forms, which concluded their participation in the study.

Data Analysis: The last step was to analyze the collected data to understand how using a chatbot to find a task compares to using the GitHub issue tracker and whether the chatbot is useful for supporting newcomers on OSS projects. We first analyzed the participants' answers to the TAM questionnaire using a Cumulative Link Mixed Model (CLMM) for ordinal data [7]. We fitted a model with the rates for each Likert item as the dependent variable, the tool as the independent variable (GitHub mock vs. chatbot), and a factor representing the measured construct (perceived usefulness and perceived ease of use). We also included the individual Likert item as a random effect.

Moreover, we evaluated whether the participants' profiles influence their preferences for one or another condition. To achieve that, we fitted a Generalized Linear Model [14] with binomial class, where the dependent variable is the participant's preferred tool (chatbot vs. GitHub mock), and the independent variables are the participant's age, gender, years of industry experience, and years of OSS experience. The results of the quantitative analysis can be found in Section 4.1.

We also analyzed the open-ended questions through a thematic analysis method [31]. We inspected the answers to identify themes and assigned codes that summarize the key points. By constantly comparing the codes [30], we grouped them into categories to give a high-level representation of the codes. This coding process was conducted by one researcher and constantly discussed with two experienced researchers until reaching a consensus on the codes and categories. These findings are presented in Section 4.2

4 Results

In this section, we present the results of our study. Firstly, we focus on the quantitative analysis, which compares the use of a chatbot to find a task to the use of the traditional GitHub issue tracker. After that, we present our findings related to the qualitative analysis, pointing out the pros and cons of each tool according to the participant's feedback.

4.1 Quantitative analysis

Figures 6 and 7 depict the results for the TAM questions regarding the perceived usefulness (PU) and perceived ease of use (PEOU), respectively. Participants rated both tools positively for their ease of use and usefulness. PU2 and PU3 received the highest number of negative scores. These questions focused on job performance and productivity, which were not sufficiently explored in the study since participants were not required to solve the selected task.

Although participants were mostly positive about both tools, Figure 7 suggests that there is a trend toward the chatbot receiving higher scores than the GitHub mock for the perceived ease of use factor. The results of the CLMM analysis confirmed this inference, as the estimated contrast ($chatbot - GitHub_{mock}$) is 0.42 ($SE = 0.13$, $p - value < 0.001$, $\alpha = 0.05$), which means that one-unit increase in the GitHub mock score would result in $exp(0.42) = 1.52$-unit increase

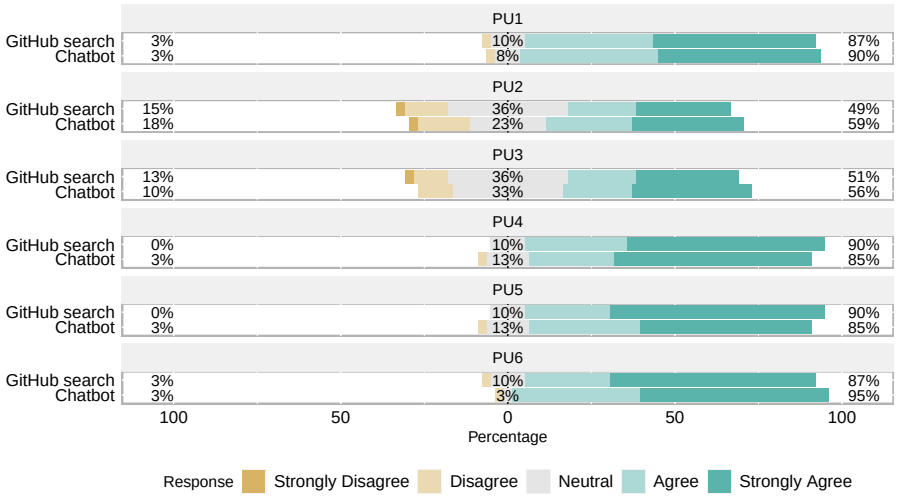

Fig. 6. Perceived usefulness: chatbot vs. GitHub issue search tool

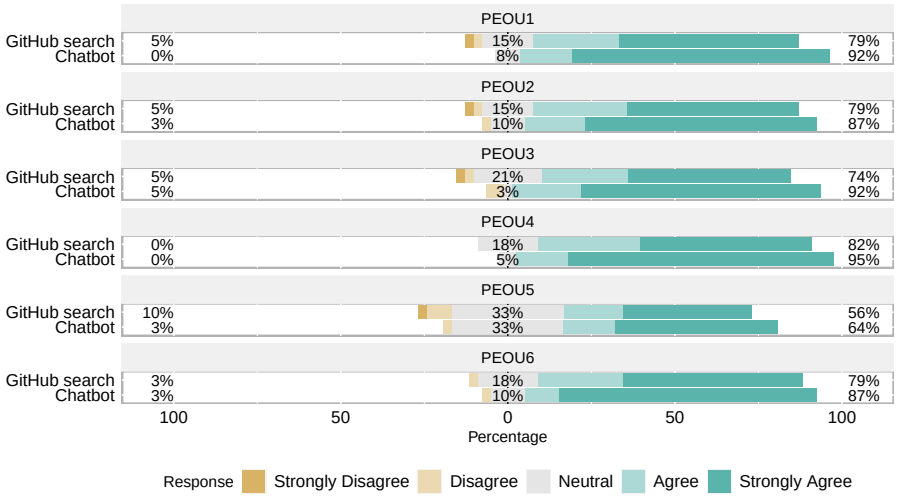

Fig. 7. Perceived ease of use: chatbot vs. GitHub issue search tool

in the chatbot score (odds ratio of higher scores equals to approximately 52%). The contrast per factor showed that the perceived ease of use influenced this difference (estimate=0.92, SE=0.19, p-value< .0001), while we found no difference in the perceived usefulness (estimate < -0.001, SE=0.18 p-value=0.99). Therefore, we concluded that although participants perceived both tools as similarly useful, they perceived the chatbot as easier to use.

When we look at the participant's choice for their preferred tool, half of the respondents (50%) chose the chatbot, 35% chose the GitHub issue tracker, and 15% did not have a preference. The results of the Generalized Linear Model, presented in Table 3, did not identify a particular demographic characteristic that statistically explains the participant's choice. Therefore, we do not have evidence to support that the participant's profile influenced their preference for one interface, which needs further investigation.

Table 3. Table of profile correlation.

	Estimate	**Std. Error**	**z-value**	**Pr** $(>\|z\|)$
Intercept	6.0068	3.9115	1.536	0.1246
Age	-0.2828	0.1601	-1.766	0.0774
Gender	0.1608	1.0647	0.151	0.8800
Industry Experience	0.3152	0.2041	1.544	0.1226
Open Source Experience	-0.1330	0.1468	-0.906	0.3650

4.2 Qualitative analysis of the open-questions

Table 4. Thematic analysis from user's open answers about negative and positive aspects about both tools

Categories		**Chatbot**		**GitHub**	
		Negative	**Positive**	**Negative**	**Positive**
Communicability	context recovery	7			
	lack of information on how to interact	7		4	
	better options to filter		1		
	search efficiency		6		1
UI design	intuitiveness		2		
	interaction mode	1			
	high complexity			10	
	user friendliness		1		
	tool limitation			5	
	ease to use		10		4
Maxim of quantity	information overload			4	
	too few recommendations	6			
	lack of information preview	4			
	more information about the task				7
Profile dependency	personal choice	2	1		1
	situational use		1		
Side results	lack of prioritization	2		1	
	categorization dependency	2		11	

As revealed in our quantitative work in section 4.1, our participants perceived the chatbot as easier to use than the GitHub search. Table 4 summarizes the results of the thematic analysis, which shows that some codes confirm this

inference. We split the codes uncovered by the analysis categories that cluster similar concepts.

The category UI DESIGN refers to the way the tool was designed and built-in terms of user interaction. We further classify the findings under this category into six topics which clearly shows why chatbots are perceived as easier to use. Users found the chatbot has **intuitiveness** because *"the chatbot uses my answers to find what best fits my expectations and abilities"* [P21]. Regarding the **interaction mode**, one participant stated that he *"missed interaction with my answers and not the defined by the options box, the chatbot should not have the text entry box"* [P11]. GitHub **high complexity** was highly commented, as in the quotes: *"people would need to understand the GitHub interface previously because it is not newbie friendly"* [P02] and *"it is complex the search for issues, there are labels that do not show any related issue"* [P11]. In contrast, one participant dropped a comment on the chatbot's **user friendliness**: *"as a newcomer to the programming environment, the chatbot was more user-friendly"*[P02]. We also found some comments regarding **tool limitations**. Some participants mentioned the advanced search options (on GitHub) (e.g., *"there is no way to search using the asterisk as a wildcard"*[P27] and *"I can not store my searches even logged in, I wanted to save predefined filters"*). Participants reported the **ease of use** as a positive aspect for both tools. For example, [P32] reported that the chatbot was *"easy to use, it is prepared to give results directly following the user's profile, probably inexperienced users can not do an open search"*. Regarding the GitHub issue tracker, a participant mentioned that *"the tasks are already on GitHub, so it is easier to use GitHub to filter than a redirecting chatbot"* [P23].

The Grice's MAXIM OF QUANTITY defines the informativeness of a contribution and states that a message should be as informative as required [15]. Four codes are grouped under this category, but we highlight the first one because it shows a negative aspect of GitHub that may have influenced our quantitative result. Some participants found that GitHub's **information overload** leads to a time-consuming experience. For example, one participant complained: *"I had to analyze all the tags to find one that fits my profile"* [P18].

In contrast to what we found so far, our study revealed other aspects of the interaction that may not have influenced the quantitative results. Participants reported that the chatbot had **too few recommendations** and they wanted more (e.g., *"it limits the task options. Using the chatbot, I cannot explore all open issues or choose to detail an issue that already has a discussion"*[P25]. Participants reported **lack of information preview** as a negative aspect of the chatbot, as in the following quote: *"the chatbot does not provide a task preview, so the developer needs to click suggested links to verify"*[P31]. GitHub, on the other hand, offered **more information about the task**: *"in the GitHub search, I have a broader view for a search. Some tasks complement each other, and with the search, it is possible to visualize this. In the chatbot, I had to follow the GitHub link"* [P06].

COMMUNICABILITY, the ability of a tool to communicate, is subdivided into four codes. Users from the chatbot did not like that they could not **recover**

the context of the conversation. For example, one participant said that *"the chatbot should have an option to go back to previous questions in the middle of a conversation without the need to finalize the whole selection flow"*[P15]. Another participant commented that *"changing this path is not easy as going back to the previous question"*[P20]. Besides, there are negative aspects of both chatbot and GitHub interfaces about the **lack of information on how to interact** with the tool. The following quotes demonstrate this need:

"At first I thought I would need to use the keyboard to interact with the chatbot and it gave me no answer at all, I had to ask how to interact to know that I needed to click on options" [P19].

"I would like a basic introduction, few words or images, where the basics of how to use the search tool are shown" [P02].

One participant mentioned that the hierarchical structure of the chatbot conversation provided **more options to filter** and classify the tasks helped him: *"I had the impression that the fact of the chatbot had levels (network, databases) has helped me to identify the tasks according to my interests"* [P11]. Regarding **search efficiency**, [P04] mentioned that *"using the chatbot, I could abstract tasks that would not have caught my attention and choose only among the ones I really would like to do"*. In contrast, another participant reported *"I believe that the search from GitHub is way faster and easier to find my task"*.

We found some topics that depend on the user's **personal choice** and **situational use**, so we categorized them as PROFILE DEPENDENCY. We evidenced that participants who had a negative experience with chatbots before the study leaned toward a traditional interface, as reported by one participant: *"I particularly do not like to use chatbots because my experiences were the worst"* [P36]. Other participants just prefer to search on their own (e.g., *"I prefer search engines to chatbots. Normally until it gives me what I need, I have already found a simple search query."* [P40]), and there were those who just think chatbots are more practical. Regarding the **situational use**, a participant stated that the chatbot would only be useful for a newcomer: *"As I am a newcomer to Open Source Software, the chatbot would be great, but I do not believe I would use this for a long time because the search engine visualization gives me a sense of responsibility and ownership" [P30]*. Another participant explained that choosing between the tools would *"depend on the intention of the moment if it is to explore the tasks or find the task easily"* [P25]. These statements complement what we have found on the quantitative analysis since newcomers would benefit more from a tool easier to use.

SIDE RESULTS correspond to the answers that are not directly related to the research goal but can also help improve its outcome. When talking about the chatbot, participants reported that there is a **lack of prioritization** when displaying the tasks: *"I missed other categories or subcategories like priority and difficulty"* [P11]. While this is related to the lack of information about tasks (under "Maxim of quantity"), it is a problem related to how the development team labeled the tasks. We point out that this is a more general problem related to the project's information architecture rather than the chatbot design. Participants

also mentioned issues with the **categorization dependency**. They reported that *"if the categorization is wrong, the search using labels would be ruined"* [P20], and *"the contributor may give up if the task is not categorized correctly."*. This concern is important and should be taken into consideration. However, it goes beyond the scope of the study, which only used the task classifier as a tool to set up the environment.

5 Discussion

This section provides additional insights on our results and how they impact practice and improve the existing state-of-the-art.

As our quantitative results showed, the chatbot interface was perceived as easier to use. When analyzing the reasons for choosing the preferred tool, the thematic analysis revealed a pattern for those who chose chatbot along the lines of being *"simpler, easier, more intuitive."* This outcome is clearly evidenced in Table 4, in which USER FRIENDLINESS, INTUITIVENESS, and EASE OF USE are heavily reported as chatbot's positive aspects. Going in-depth, we learned that the design of the search engine may negatively affect the perception of ease of use of people who are not used to the tools. As some participants mentioned, *"[GitHub engine] has too much information to process and decide"* [P8] and *"it is complex to understand the issue list"* [P14]. We learned that the simplicity of the chatbot using predefined answers (buttons) makes it intuitive and easy to use, as compared to the UI offered by GitHub.

The results show the benefits of a chatbot for newcomers, supporting the insights from Dominic *et. al* [9] by showing that chatbots support these newcomers in the process of choosing an appropriate task [26]. This was evidenced by several participants who mentioned that *"As a newcomer, I preferred the chatbot because it is simpler"* [P26]. The thematic analysis also revealed that the complexity of the GitHub search engine gives the user more control over the search, but this same characteristic is overwhelming for someone who is not familiar with the issue tracker interface, as reported by [P31]: *"for a newcomer, the chatbot helps a lot...while GitHub tends to facilitate the work of more experienced developers."* As an implication, we suggest that the effort should lay on designing the chatbot to increasingly unveil complexity so that the users can move slowly from the chatbot to the issue tracker interface.

Regarding the usefulness, our quantitative results indicate that both the chatbot and GitHub search engine are not perceived differently. Both of the approaches supported our participants in their tasks. However, the thematic analysis revealed some chatbot negative aspects related to the efficiency in finding appropriate results. This is noticeable in Table 4, when we look at the mentions to TOO FEW RECOMMENDATIONS and LACK OF INFORMATION PREVIEW as negative aspects of the chatbot versus the MORE INFORMATION ABOUT THE TASK as a positive aspect of the GitHub search. At the same time that this is a potential improvement to the chatbot, previous discussions regarding conversational search [20] have already pointed out that an effective result from a conversa-

tional search cannot be as rich as the augmented results from a traditional web search. Furthermore, this may be related to personal preferences or the diversity of information processing styles [23], which deserves further investigation.

The recent research on chatbots for software engineering has mainly focused on exploring the intent discovery [2] and the precision of the information returned [1, 32, 24] rather than looking at the interaction aspects. This study complements the existing literature, shedding light on the importance of understanding not only the precision of the information provided by the chatbots, but also focusing on how the users perceive their usefulness and ease to use.

5.1 Limitations

Participants may be subject to learning effects since they have evaluated two tools to perform the same task. To mitigate that, we designed our experiment to follow a within-subject approach. In this case, the participants assessed the tools in random orders and answered the questionnaires in different orders, thus reducing potential biases by learning effect. Moreover, this enabled us to analyze the tools separately to understand if there is any chance of learning bias that may have caused any impact on the data collected.

Although we used strategies to avoid learning bias, we are still vulnerable to the selection bias since we did not actively search OSS contributors or potential contributors or selected only newcomers. In fact, we selected people randomly in a non-probabilistic way. There was no specific characteristic that we were looking for when selecting participants; the only crucial thing was the software development background.

We have not deployed the chatbot on a server, and we did not track all the paths users used to decide their tasks. We may have missed some important information in this process. Also, the chatbot was outside of GitHub, which does not simulate the real environment that we expect to use, which is the corner of the GitHub official platform. This might have led users to face difficulties that would not be a problem if the chatbot had been deployed inside GitHub.

The GitHub labeling system is not capable of reproducing the hierarchical structure that we created for the chatbot. This may have influenced the participants' experience. However, this is an issue with the GitHub search feature, which we could not change. Thus, we interpret this as a chatbot differential.

We cannot claim that our work is generalizable. We built our experiment considering specific variables: project, label classifier, and GitHub environment. To seek generalizability, other studies need to be conducted with different projects in different domains. Moreover, all participants of this study are Brazilian, which may have introduced a cultural bias. We encourage replications of this study with participants from other countries to assess how it may affect the perception of the chatbot.

6 Conclusion

Our paper proposed a new way to interact with OSS projects. Since there are no studies evaluating chatbots in software engineering, we used the technology acceptance model to understand whether users would find it easy to use and useful. Our results showed that a chatbot could help newcomers to find their first task to work on. This is the first step towards solving this problem already discussed by the literature.

As the chatbot was perceived as easier to use, we claim that this interface would be more appropriate for the newcomers, who may find it harder to onboard with little experience with the GitHub issue tracker interface. Although we could not evidence an association between the participants' choice and their profile, we understand that having a chatbot that helps newcomers filter their interests and deliver a small number of options is a good starting point, which we also evidenced in the qualitative analysis. In short, to answer our research question, we conclude that a chatbot is as useful as the GitHub issue tracker to find a task. Still, the chatbot is easier to use, which primarily benefits the newcomers.

As a future avenue, we will use the results of the qualitative analysis to improve the chatbot design, which may lead to a new round of studies to address the problems we found and mitigate the potential biases that we missed in this research. Another potential future direction is to analyze the preference of people with different learning styles and cognitive facets and explore other aspects of the user's profiles that could influence their preference for one or another interface.

References

1. Abdellatif, A., Badran, K., Costa, D., Shihab, E.: A comparison of natural language understanding platforms for chatbots in software engineering. IEEE Transactions on Software Engineering (2021)
2. Abdellatif, A., Badran, K., Shihab, E.: Msrbot: Using bots to answer questions from software repositories. Empirical Software Engineering **25**(3), 1834–1863 (2020)
3. Balali, S., Annamalai, U., Padala, H.S., Trinkenreich, B., Gerosa, M.A., Steinmacher, I., Sarma, A.: Recommending tasks to newcomers in oss projects: How do mentors handle it? In: OpenSym 2020. pp. 1–14 (2020)
4. Barcomb, A., Stol, K., Fitzgerald, B., Riehle, D.: Managing episodic volunteers in free/libre/open source software communitie. In: IEEE Transactions on Software Engineering. pp. 1–1 (2020)
5. Brandtzaeg, P., Følstad, A.: Chatbots: changing user needs and motivations. Interactions **25**, 38–43 (08 2018)
6. Catania, F., Spitale, M., Cosentino, G., Garzotto, F.: Conversational agents to promote children's verbal communication skills. In: International Workshop on Chatbot Research and Design. pp. 158–172. Springer (2020)
7. Christensen, R.H.B.: ordinal—regression models for ordinal data (2019), https://CRAN.R-project.org/package=ordinal
8. Davis, F.D.: Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Quarterly **13**(3), 319–340 (1989)

9. Dominic, J., Houser, J., Steinmacher, I., Ritter, C., Rodeghero, P.: Conversational bot for newcomers onboarding to open source projects. In: BotSE 2020. pp. 46–50 (2020)
10. Faraooq, U., Grudin, J.: Human-computer integration. In: Interactions 23, 6. pp. 27–32 (2016)
11. Feyer, S., Siebert, S., Gipp, B., Aizawa, A., Beel, J.: Integration of the scientific recommender system mr. dlib into the reference manager jabref. In: European Conference on Information Retrieval. pp. 770–774. Springer (2017)
12. Fitzpatrick, K., Darcy, A., Vierhile, M.: Delivering cognitive behavior therapy to young adults with symptoms of depression and anxiety using a fully automated conversational agent (woebot): A randomized controlled trial. JMIR Mental Health **4**, e19 (2017)
13. Forte, A., Lampe, C.: Defining, understanding, and supporting open collaboration lessons from the literature. American Behavioral Scientist **57**, 535–547 (05 2013)
14. Friedman, J., Hastie, T., Tibshirani, R.: Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software **33**(1), 1–22 (2010)
15. Grice, H.P.: Logic and conversation. In: Cole, P., Morgan, J.L. (eds.) Syntax and Semantics: Vol. 3: Speech Acts, pp. 41–58. Academic Press, New York (1975)
16. Guizani, M., Steinmacher, I., Emard, J., Fallatah, A., Burnett, M., Sarma, A.: How to debug inclusivity bugs? an empirical investigation of finding-to-fixing with information architecture. Tech. rep., EECS, Oregon State University (2020)
17. Höhn, S., Bongard-Blanchy, K.: Heuristic Evaluation of COVID-19 Chatbots, pp. 131–144. Springer (02 2021)
18. JabRef: Jabref project (2019), https://jabref.org/
19. Janssen, A., Cardona, D.R., Breitner, M.H.: More than faq! chatbot taxonomy for business-to-business customer services. In: International Workshop on Chatbot Research and Design. pp. 175–189. Springer (2020)
20. Joho, H., Cavedon, L., Arguello, J., Shokouhi, M., Radlinski, F.: First international workshop on conversational approaches to information retrieval. SIGIR Forum **51**(3), 114–121 (feb 2018)
21. Michael J. Pazzani, D.B.: Content-based recommendation systems. In: The adaptive web. pp. 325–341. Springer (2007)
22. Olsson, T., Ericsson, M., Wingkvist, A.: The relationship of code churn and architectural violations in the open source software jabref. In: Workshop on Software Architecture Erosion and Architectural Consistency. p. 152–158. ACM (2017)
23. Padala, S.H., Mendez, C.J., Dias, L.F., Steinmacher, I., Hanson, Z.S., Hilderbrand, C., Horvath, A., Hill, C., Simpson, L.D., Burnett, M., et al.: How gender-biased tools shape newcomer experiences in oss projects. IEEE Transactions on Software Engineering (2020)
24. Romero, R., Parra, E., Haiduc, S.: Experiences building an answer bot for gitter. In: BotSE 2020. pp. 66–70 (2020)
25. Santos, F., Wiese, I., Trinkenreich, B., Steinmacher, I., Sarma, A., Gerosa, M.A.: Can i solve it? identifying apis required to complete oss tasks. In: IEEE Transactions on Software Engineering. pp. 1–1 (2020)
26. Steinmacher, I., Conte, T., Gerosa, M.A.: Understanding and supporting the choice of an appropriate task to start with in open source software communities. In: 48th Hawaii International Conference on System Sciences. pp. 5299–5308. IEEE (2015)
27. Steinmacher, I., Conte, T.U., Redmiles, D.F., Gerosa, M.A.: Social barriers faced by newcomers placing their first contribution in open source software projects. In: ACM Conference on Computer-Supported Cooperative Work and Social Computing (CSCW 2015). pp. 1–13 (2015)

28. Steinmacher, I., Treude, C., Gerosa, M.A.: Let me in: Guidelines for the successful onboarding of newcomers to open source projects. IEEE Software pp. 1–1 (2018)
29. Storey, M.A., Zagalsky, A.: Disrupting developer productivity one bot at a time. In: Foundations of Software Engineering (FSE). pp. 928–931 (2016)
30. Strauss, A.L., Corbin, J.M.: Basics of qualitative research : techniques and procedures for developing grounded theory. Sage Publications, Thousand Oaks (1998)
31. Terry, G., Hayfield, N., Clarke, V., Braun, V.: Thematic analysis. The SAGE handbook of qualitative research in psychology **2**, 17–37 (2017)
32. Vale, L.d.N., Maia, M.d.A.: Towards a question answering assistant for software development using a transformer-based language model. In: BotSE 2021 (2021)
33. von Krogh, G., Spaeth, S., Lakhani, K.: Community, joining, and specialization in open source software innovation: a case study. Res Policy **32**(7), 1217–1241 (2003)
34. Wang, J., Sarma, A.: Which bug should i fix: Helping new developers onboard a new project. In: 4th International Workshop on Cooperative and Human Aspects of Software Engineering. p. 76–79. CHASE '11 (2011)
35. Wessel, M., de Souza, B.M., Steinmacher, I., Wiese, I.S., Polato, I., Chaves, A.P., Gerosa, M.A.: The power of bots: Characterizing and understanding bots in oss projects. In: ACM on Human Computer Interaction, CSCW. p. 182. vol. 2 (2018)
36. Wessel, M., Wiese, I., Steinmacher, I., Gerosa, M.A.: Don't disturb me: Challenges of interacting with software bots on open source software projects. In: ACM Conference on Computer Supported Cooperative Work and Social Computing (2021)
37. von Wolff, R.M., Nörtemann, J., Hobert, S., Schumann, M.: Chatbots for the Information Acquisition at Universities–A Student's View on the Application Area, pp. 231–244. Springer (2019)