An Initial Analysis of Newcomers based on Labels in Contemporary GitHub Projects

Fadhlih Hasan Setiawan
Fakultas Komunikasi dan Informatika
Universitas Muhammadiyah Surakarta
Surakarta, Indonesia
fhasansetiawan@gmail.com

Yusuf Sulistyo Nugroho
Fakultas Komunikasi dan Informatika
Universitas Muhammadiyah Surakarta
Surakarta, Indonesia
yusuf.nugroho@ums.ac.id

Raula Gaikovina Kula
Graduate School of Science and Technology
Nara Institute of Science and Technology
Nara, Japan
raula-k@is.naist.jp

2024 IEEE International Conference on Data and Software Engineering (ICoDSE) | 979-8-3315-0640-7/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICODSE63307.2024.10829876

***Abstract*—Due to the nature of Open Source Software (OSS), the need to attract, onboard, and retain newcomers is critical to ensure the sustainability of those projects. As such, initiatives such as the Good First Issue (GFIs) allow developers to target specialized issues that might be easier for a newcomer to handle. While previous studies have indicated that GFIs often require more time to resolve, are rarely addressed by newcomers, and may not be as straightforward as initially assumed, their characteristics have not been compared with other types of issues. Thus, this paper aims to investigate the differences between newcomer-addressed issues (such as GFIs) and those handled by experienced contributor. We focus on all closed issues extracted from both issues and pull requests from four Python projects on GitHub, that are 18,212 issues from AWS-CDK, 259 issues from Six, 4,646 issues from Flask, and 19,934 issues from NumPy. Using an automatic clustering technique, we are able to cluster and distinguish between newcomers and experts closed issues. The results show that, although the labels are assigned differently in all projects, there is no difference between issues that are targeted at newcomers. This suggests new potential future research, such as exploring whether newcomer topics vary in difficulty levels rather than focusing solely on specialized subjects, which could impact the long-term sustainability of these projects.**

***Index Terms*—closed issues, GFI, github, labels, newcomers**

I. Introduction

Open Source Software (OSS) projects face the risk of becoming obsolete, as they are unable to attract and retain newcomer [1]. With the increasing popularity of the GitHub platform, OSS projects have a great chance to attract potential contributions through the platform. GitHub reported around 10 million-plus new users in 2020 and allows more than 40 million developers to showcase their skills to the world's largest community (44 million upstream repositories).[1] With this upsurge in user activity, it could be possible to find and attract newcomers.

Prior studies have explored the motivations and attractiveness of OSS developers [2]–[5] and examined the transition from a newcomer to a core member of a project among developers [6]–[9]. These highlight the motivation and project appeal that can attract outsiders and draw them into the participation process. Another work by Choi et al. [10] identified seven socialization tactics that influence newcomers' commitment to online groups. According to Lakhani and Wolf [11], external benefits such as better jobs and career advancement, motivate primarily new contributors, as do intrinsic, code-based challenges, and improved programming skills. Recent studies also focus on analyzing and characterizing the first contributions generated by a surge of newly-joined developers on GitHub as a guideline for prospective newcomers to help them contribute towards OSS [12]–[14]. Beside, newcomers are also essential to the survival, long-term sustainability, and continuity of OSS [15]. Although prior studies have highlighted various strategies to onboard new contributors, however, there is little evidence on how labeling issues such as Good First Issues (GFIs) on GitHub [16] differentiate between issues suitable for newcomers and those for experienced contributors. Projects that use issue labels such as GFI indicate that the issues are appropriate for newcomers [16].

Based on the background, this study aims to explore the newcomer issue topics that are implemented and how they differ from expert-labeled data. To facilitate our study, we distinguish the issues by newcomers and experts based on their labels. By using LDA clustering techniques, we are able to explore the differences between these two groups.

From our results, we make two observations. First, there seems no clear difference between topics that could be assigned to newcomers and second, there is no difference between the newcomers' issue topics and the experts' issue topics. Based on these results, we understand that more analysis might be needed to understand the differences that exist between the newcomers and experts issues.

[1]Statistics from https://octoverse.github.com accessed in January 2020

II. Method

A. Research Questions

This case study aims to understand the differences of issue topics shared by newcomers and experts in contemporary GitHub projects. To guide our study, we formulate two research questions:

- RQ_1: What are the issue topics shared by newcomers in contemporary GitHub projects?
- RQ_2: Are the issue topics shared by newcomers different than those by experts in contemporary GitHub projects?

TABLE I
DESCRIPTION OF THE PROJECTS UTILIZED IN OUR ANALYSIS PER APRIL 2023

Project name	#stars	#contributors	#all issues[a]	#users	#labels
AWS-CDK	10.1K	1,193	22,598	25.1K	315
Six	937	58	261	1.6m	14
Flask	62.5K	697	4,836	1.4m	23
NumPy	23.2K	1,465	21,367	1.5m	89

[a]Combination between GitHub issues and pull requests.

TABLE II
NUMBER OF CLOSED ISSUES AND LABELS OF BOTH NEWCOMERS AND EXPERTS IN AWS-CDK

Category	Sample Labels	#Closed Issues	#Labels	
Newcomers	effort/small, good first issue, help wanted	1,825	3	(1.12%)
Experts	auto-approve, blocked, breaking-change, cause/not-a-bug, close-for-staleness, closing-soon, conflicts, contributions/core, duplicate, effort/large, feature-request, feature/coverage-gap, feature/enhancement, feature/new-construct, feature/new-language, feature/pattern, feature/pfr, feature/service-integration, guidance, in-progress, investigating, management/ci-cd, management/devenv, management/repo, management/rfc, management/roadmap, management/tracking, needs-cfn, needs-design, needs-discussion, needs-reproduction, needs-triage,no-autoclose, pr/auto-approve, pr/blocked, pr/breaking-change, pr/do-not-merge, pr/forward-merge, pr/needs-review, pr/needs-tests, pr/no-squash, pr/ready-to-merge, pr/requires-two-approvers, pr/work-in-progress, response-requested, SECURITY, service-api, testing, wontfix	11,807	49	(18.42%)
Others	-	4,580	214	(80.45%)
Total		18,212	266	(100.00%)

B. Data Extraction

To facilitate our study, we used four GitHub projects as described in Table I, which reflects the project descriptions in April 2023, namely AWS-CDK,[2] Six,[3] Flask,[4] and NumPy.[5] These projects are considered to be popular projects on GitHub, based on their high number of stars [17], issues, and contributors [18]. In Table I, all issues represent issues and pull requests, which are closed and opened. We believe that these projects are appropriate for our analysis.

In this study, we only focus on closed issues from each repository since they indicate the problems that have been resolved or fixed. These datasets were extracted from four GitHub repository projects in 2022. We used the API from GitHub to extract closed issues and pull requests, ending up with 18,212 issues for AWS-CDK, 259 issues for Six, 4,646 issues for Flask, and 19,934 issues for NumPy.

C. Identify Newcomers and Experts labels

We carried out manual analyses to differentiate the issues between two developer categories, that are, newcomers and experts. First, we extracted all labels from the datasets. Then, we removed the duplicate labels from the list. Next, we conducted a manual classification of the extracted label list. The initial filtration is performed by using the same approach as Tan et al. [16] to identify the newcomers' issues. In the next iteration, we added other labels that seem to require more maintainer access and tackle more severe issues, such as security and breaking fixes. In addition, we added management, new features, approval, and other labels that relate to the experience.

[2]https://github.com/aws/aws-cdk accessed April 2023
[3]https://github.com/benjaminp/six accessed April 2023
[4]https://github.com/pallets/flask accessed April 2023
[5]https://github.com/numpy/numpy accessed April 2023

TABLE III
NUMBER OF CLOSED ISSUES AND LABELS OF BOTH NEWCOMERS AND EXPERTS IN SIX

Category	Sample Labels	#Closed Issues	#Labels	
Newcomers	minor, trivial	39	2	(22.22%)
Experts	major, enhancement, critical, blocker	95	4	(44.44%)
Others	-	125	3	(33.33%)
Total		259	9	(100.00%)

TABLE IV
NUMBER OF CLOSED ISSUES AND LABELS OF BOTH NEWCOMERS AND EXPERTS IN FLASK

Category	Sample Labels	#Closed Issues	#Labels	
Newcomers	good first issue	36	1	(5%)
Experts	blocker, blueprints, discussion, logging, packaging, security, session, testing	132	8	(40%)
Others	-	4,478	11	(55%)
Total		4,646	20	(100%)

As can be seen in Table II, in AWS-CDK, 1.12% labels are grouped in the newcomers category and 18.42% labels are in the experts category. Experts category has a high variety of labels from different issues, from `auto-approve`, `blocked`, to `wontfix`. Finally, we then extracted 1,825 issues for newcomers, 11,807 issues for experts, and 4,580 issues for the remaining data.

Table III shows that, in the Six project, there are 22.22% labels in the newcomers category and 44.44% labels in the experts category. Moreover, the experts labels vary, such as `major`, `enhancement`, `critical`, and `blocker`. At last, we then extracted 39 issues for newcomers, 95 issues for the experts and the remaining amount to 125 issues.

As can be seen in Table IV, the label classification for Flask describes that there are 5% labels in the newcomers category

TABLE V
NUMBER OF CLOSED ISSUES AND LABELS OF BOTH NEWCOMERS AND EXPERTS IN NUMPY

Category	Sample Labels	#Closed Issues	#Labels
Newcomers	Priority: low	7	1 (1.17%)
Experts	Enhancement, Maintenance, Testing, Deprecation, Discussion, WIP, Compiler, Benchmark, API, Reversion, Cross-compilation, Duplicate, In progress, Needs decision, Needs work, Ready for review, Needs tests	5,492	24 (28.23%)
Others	-	14,435	60 (70.58%)
Total		19,934	85 (100.00%)

TABLE VI
FIVE TOPICS EXTRACTED FROM NEWCOMERS ISSUES IN AWS-CDK

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
featur	*https*	*new*	*build*	group
https	*stack*	*const*	file	subnet
request	*deploy*	*error*	asset	*creat*
chang	*new*	*creat*	*deploy*	*self*
implement	bug	*import*	*run*	bug
new	*creat*	*scope*	command	secur
support	*error*	*stack*	imag	*https*
may	set	true	*pipelin*	*type*
might	issu	prop	directori	instanc
break	doe	lambda	*instal*	vpc

and 40% labels in the experts category. It also indicates that the experts labels vary from different topics, from `blocker`, `blueprints`, to `testing`. Finally, we then extracted 36 issues for newcomers, 132 issues for the experts and the remaining amount of 4,478 issues.

Table V shows the label classification for newcomers and experts of NumPy. The newcomers category covers 1.17% labels, while the experts category has 28.23% labels. In addition, the experts labels have a high variety of labels, from `enhancement`, `maintenance`, to `needs tests`. We then extracted 7 issues for newcomers, 5,492 issues for experts, and the remaining data accounting for 14,435 issues.

D. Clustering Issues Using LDA

To group the topics into both newcomers and experts, we applied the Latent Dirichlet Allocation (LDA), which allows us to extract and cluster words from the textual data of the title and body of GitHub.

The steps to implement the LDA are as follows:

1) We separated the dataset of issue into newcomers, experts, and combined group.
2) From each issue, we extracted the title and body using the html parser of `BeautifulSoup`[6] then concatenated them into a sentence.
3) In each sentence, we then applied a regular expression to tokenize the sentence into the list of the single English word.
4) The stemming technique was applied on each word by using `SnowballStemmer`.[7] We removed the stop words by using `stopwords` from `NLTK`.[8]
5) After the filtering and tokenizing process, we preprocessed the cleaned dataset by using pos_tag from tag on `NLTK`.
6) The `LDA Mallet` model was created by using the `Gensim`[9] to extract and cluster the topics. We experimented with topic numbers (k) based on coherence score, with k = 5 which achieved the best score.
7) Finally, we analyzed the combined topics by using `pyLDA`.[10]

E. Online Appendix

To facilitate the reproducibility of this work, we have made our research artefacts publicly available on https://github.com/Fadhlih2001/NewcomersAnalysisBasedOnLabels.

III. RESULTS AND DISCUSSION

In this section, we discuss the results of our study. To notice, the different font color displayed in each table shows the duplication of one keyword in a topic for newcomers and expert topics in each project. In one project, we cluster the issues into 5 topics where each topic is a group of fixed vocabulary of words, in which each word has a probability that it belongs to that topic.

A. Issue Topics of Newcomers and Experts

1) RQ_1: What are the issue topics shared by newcomers in contemporary GitHub projects?:

> **Observation of RQ_1**: There seem to be no clearly defined topics within the set of newcomers issues.

Table VI shows that some keywords in AWS-CDK project are duplicates across other topic groups, i.e., *https*, *new*, *stack*, *creat*, and *error*. The *https*, *new* and *creat* appears to be the most used keywords. The *https* appears in topic 1, 2, and 5, *new* in topic 1, 2, and 3, and *creat* in topic 2, 3, and 5. In contrast, all keywords in topic 4 seem to be not related to other topics. Since each topic seems to be slightly different from the others, we consider that the keywords in each topic are not related enough to define them as a topic of issues for newcomers.

Similarly, some keywords in Six project are duplicates across other topic group, as described in Table VII. For example, both *return* and *implement* appear in topic 1 and 2, and *modul* appears in topic 2 and 3. On the other hand, all keywords in topic 4 and 5 seem to be not related to each other. Similar to prior finding, we consider that the keywords in each topic are not related enough for defining them as a topic of issue for newcomers.

[6]https://pypi.org/project/beautifulsoup4/ accessed September 2022
[7]https://pypi.org/project/snowballstemmer/ accessed September 2022
[8]https://www.nltk.org/ accessed September 2022
[9]https://pypi.org/project/gensim/ accessed September 2022
[10]https://github.com/bmabey/pyLDAvis accessed September 2022

TABLE VII
FIVE TOPICS EXTRACTED FROM NEWCOMERS ISSUES IN SIX

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
loop	return	compat	except	getcwd
move	iter	modul	suit	support
best	iterkey	print	test	unicod
per	modul	file	compat	python2
timeit	self	aaugustin	packag	bboe
byte	dir	function	differ	fail
return	itervalu	attribut	run	check
implement	test_move_item	method	cjwatson	python-six
ord	msbramo	fix	import	tkinter
buf	implement	test	deprec	maxgrenderjon

TABLE VIII
FIVE TOPICS EXTRACTED FROM NEWCOMERS ISSUES IN FLASK

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
import	document	client	file	instal
test	mode	request	mode	yum
pip	https	http	test	env
easy_instal	doc	document	contain	version
autocorrect	link	test	object	python-virtualenv
datetim	develop	empti	zero-pad	virtualenv
context	read	string	tree	static
doc	sure	werkzeug	warn	instead
https	secur	session	url	exampl
timezon	extens	side	properti	problem

Some keywords in Flask project, as shown in Table VIII, are also duplicates across other topic group, that are, *test* appears in topic 1, 3, and 4, and both *https* and *doc* are included in topic 1 and 2. However, all keywords in topic 5 does not relate to other topics. This result indicates that the keywords in each topic in Flask are not related enough to determine as a topic of issue for newcomers.

In contrast to the results of the previous analyses, there are no duplicated keywords in all topics in Numpy project, as illustrated in Table IX. Thus, we consider that the keywords in each topic are not related for defining as a topic of issue for the newcomers.

2) RQ_2: Are the issue topics shared by newcomers different than those by experts in contemporary GitHub projects?:

TABLE IX
FIVE TOPICS EXTRACTED FROM NEWCOMERS ISSUES IN NUMPY

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
shape	benchmark	generat	improv	rais
int	test	string	bad	return
mismatch	function	file	benefit	anyth
error	lie	docstr	better	assign
valu	multipli	filenam	difficult	except
exampl	run	first	editors/develop	faili
array	singi	loadtxt	environ	http
broadcast	suit	yield	experi	isnan
incorrect	becaus	argument	haystack	nan
ind	time	.bz2	hit	rational

TABLE X
FIVE TOPICS EXTRACTED FROM EXPERTS ISSUES IN AWS-CDK

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
new	request	featur	npm	silli
creat	submit	deploy	self	https
const	pull	stack	error	extract
stack	contribut	implement	line	chore
https	confirm	may	import	build
resourc	term	break	depend	instal
import	licens	build	build	fetch
deploy	test	might	construct	release
error	fix	incur	run	add
type	chang	pipelin	scope	final

TABLE XI
FIVE TOPICS EXTRACTED FROM EXPERTS ISSUES IN SIX

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
line	loop	class	modul	import
import	import	pass	import	version
modul	return	assert	example	support
return	iter	type	run	line
last	rang	print	thread	work
call	per	version	jaraco	git
recent	timeit	function	renam	return
except	iterkey	move	get	call
tracback	implement	base	function	http
class	tkinter	attribut	line	differ

Observation of RQ_2: Although the result of RQ_1 shows differences between topics for newcomers, there seems to be no difference when compared to expert issues.

As shown in Table X, the topics from experts issues in AWS-CDK seem to have the same interpretation as newcomers issues, i.e., (i) some keywords are duplicated across each other, (ii) the keywords in each topic are not related to each other, and (iii) each topic seems to be different from each other. Moreover, there are several same keywords between Table VI and Table X, which shows the issues from newcomers and experts are not too different.

Similar to topics of experts issues in AWS-CDK, the topics of experts issues in Six, Flask, and NumPy, as described in Table XI, Table XII, and Table XIII, respectively, have the same insight as the newcomers issues. We can also interpret

TABLE XII
FIVE TOPICS EXTRACTED FROM EXPERTS ISSUES IN FLASK

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
log	app	blueprint	test	line
line	blueprint	version	blueprint	https
return	error	app	doc	header
app	self	dependabot	chang	e
configur	import	import	regist	loader
session	line	line	session	request
debug	return	onli	fix	except
test	handler	default	app	log
call	rais	request	pass	return
valu	code	close	issu	access

TABLE XIII
FIVE TOPICS EXTRACTED FROM EXPERTS ISSUES IN NUMPY

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
array	dependabot	*test*	compil	compil
function	version	line	*file*	option
https	close	*error*	work	-pthread
test	merg	*file*	warn	-c
type	https	import	gcc	extra
code	updat	*http*	copi	-msse2
chang	reopen	*function*	build	-msse3
return	rebas	call	librari	*test*
dtype	stop	build	*int*	meshgrid
import	request	ticket	remov	refer

TABLE XIV
TOPICS FROM COMBINED ISSUES IN AWS-CDK. THE HIGHLIGHTED TOPICS INDICATE OVERLAPPED TOPICS.

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
https	new	pull	build	action
featur	stack	contribut	npm	forward
request	deploy	confirm	depend	s3
chang	const	term	version	merge'mast
new	creat	licens	file	info'v2-main
support	error	submit	error	aws/cdk-opschor
implement	import	test	line	polici
construct	fail	fix	asset	log
creat	resourc	silli	import	size
may	bug	chore	packag	packages/

TABLE XV
TOPICS FROM COMBINED ISSUES IN SIX. THE HIGHLIGHTED TOPICS INDICATE OVERLAPPED TOPICS.

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
line	line	import	return	except
import	modul	modul	iter	traceback
loop	class	return	import	modul
return	move	function	iterkey	doe
best	return	method	support	version
per	base	differ	run	support
timeit	import	version	http	import
modul	function	func	renam	rais
pass	metaclass	provid	itervalu	tri
class	doe	line	peterjc	differ

TABLE XVI
TOPICS FROM COMBINED ISSUES IN FLASK. THE HIGHLIGHTED TOPICS INDICATE OVERLAPPED TOPICS.

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
line	blueprint	test	document	import
log	test	blueprint	dependabot	line
request	error	doc	default	test
app	call	self	request	app
session	app	add	close	issu
blueprint	https	fix	set	exampl
configur	handler	env	session	function
applic	client	chang	file	code
import	class	version	pass	blueprint
onli	context	issu	templat	easier

that, (i) some keywords are duplicated across each other, (ii) the keywords in each topic are not related to each other, and (iii) each topic seems to be different from each other. There are also some duplicated keywords in both newcomers and experts issues of the same project, by looking at the topics in Table VII and Table XI, topics in Table VIII and Table XII, and topics in Table IX and Table XIII, They indicate that issues from the newcomers and experts are also similar.

B. Issue Topics of the Combined Newcomers and Experts

As shown in Table XIV, topic 1 and 2 from combined issues in AWS-CDK contain duplicated keywords, for instance, *new* and *creat*. This also occurs for topic 2 and 4, such as *error* and *import*. A closer look at each topic in Table XIV, some topics have similar set of the most used keywords found in Table VI and Table X, e.g., (i) topic 3 in Table VI and topic 2 in Table XIV, and (ii) topic 1 in Table X and topic 2 in Table XIV. Even, these topics appear in both issues for newcomers and experts. This indicates that the issues of newcomers and experts are not different in our study.

In Six project, we observed the topics extracted from combined issues, as shown in Table XV. Topic 3 and 5 contain duplicated keywords, for instance, *import*, *modul*, *differ*, and *version*. In addition, some topics have similar set of the most used keywords found in Table VII and Table XI, e.g., (i) topic 4 in Table VII and topic 5 in Table XV, and (ii) topic 1 in both Table XI and Table XV. These topics also appear in both newcomers and experts issues. Therefore, we can interpret that the issues of newcomers and experts are similar in our study.

Similar to previous findings in AWS-CDK and Six projects, some topics have similar set of the most used keywords found in Table VIII and Table XII for Flask, as described in Table XVI. This also occurs for NumPy, as detailed in Table XVII, showing that some of the topics have similar set of the most used keywords found in Table IX and Table XIII. For example, in Flask, we found the similar topics between (i) topic 1 in Table VIII and topic 5 in Table XVI, and (ii) topic 5 in Table XII and topic 1 in Table XVI. We also underscore the equal topics in NumPy between (i) topic 2 in Table IX and topic 1 in Table XVII, and (ii) topic 3 in Table XIII and topic 1 in Table XVII. Interestingly, these topics appear in both issues from newcomers and experts in each project. Based on these results, we can conclude that the issues of newcomers and experts are not so different in our study.

TABLE XVII
TOPICS FROM COMBINED ISSUES IN NUMPY. THE HIGHLIGHTED TOPICS INDICATE OVERLAPPED TOPICS.

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
test	dependabot	array	line	warn
array	version	function	compil	array
function	close	https	work	gcc
code	merg	type	import	int
https	https	return	file	type
fix	reopen	dtype	http	compil
chang	updat	annot	version	nan
need	rebas	valu	build	copi
issu	stop	method	install	line
import	request	import	option	void

C. Implications

We highlight two takeaways and a research outlook, as follows:

- *Newcomers and Experts tackle issues that have the same topics.* The first takeaway message from our analysis is that the topics of newcomers and experts issues are not so different. Although there might be differences in the topics themselves as resulted in RQ_1, there seem to be no differences from the LDA model. There could be two explanations. First, our method might have to include other features and metrics of the issue, such as lines of code, filename, file path, and types of changes (e.g. documentation, code, configuration, bug fix, or security). Second, the limitation of our datasets. We selected AWS-CDK, Six, Flask, and NumPy as the mature projects highly dependent upon the PyPI ecosystem. As such, we believe that these early results provide insights into more potential future work.
- *The topics might be similar, but maybe Newcomers are assigned at different difficulty levels.* The key implication of RQ_2 is the topics are similar, which suggests that the amount of expertise required to fix the issues may be different from the topic. Thus, we may need a deeper manual analysis to understand the differences between experts and newcomers issues. One assumption that we could speculate is maybe newcomers issues affect less severe issues when compared to the experts issues. Thus, studying the filepath and the location of issues in the project might provide insights.

D. Limitations

The first threat in our study is regarding the dataset. Since we only focus on four projects, that are AWS-CDK, Six, Flask, and NumPy, we could not generalize the results for other software projects. In addition, our analysis is based on the titles and bodies of the issue. Thus, the results could not be generalize to other features. For future work, we might need to add other features of the issues to make better correlations. Another key threat is our manual classification of the labels. This might be due to misunderstanding of the authors to labeling the dataset. However, we discussed the first round of labeling to reach the consensus between all authors before classifying the remaining data.

IV. Conclusion

This paper investigated closed issues from four GitHub projects, namely AWS-CDK, Six, Flask, and NumPy. We applied quantitative and qualitative analyses to differentiate between issues that are appropriate for newcomers and experienced developers.

The results provide insight into the differences between newcomer and expert issues in GitHub OSS projects, showing that although topics do not have a significant difference, the difficulty of the issue likely plays a larger role. This opens up future research directions to explore other features of issues (e.g., file paths, lines of code changed) and utilize increasingly complex techniques to categorize issues according to their level of difficulty. These findings have important implications for OSS project maintainers on GitHub, providing suggestions for better ways to categorize and assign suitable tasks to newcomers so that newcomers' onboarding experience is improved as well as retention rates, and overall project sustainability.

References

[1] I. Steinmacher, C. Treude, and M. A. Gerosa, "Let me in: Guidelines for the successful onboarding of newcomers to open source projects," *IEEE Software*, vol. 36, no. 4, pp. 41–49, 2018.

[2] P. Meirelles, C. Santos Jr, J. Miranda, F. Kon, A. Terceiro, and C. Chavez, "A study of the relationships between source code metrics and attractiveness in free software projects," in *2010 Brazilian symposium on software engineering*. IEEE, 2010, pp. 11–20.

[3] C. Santos, G. Kuk, F. Kon, and J. Pearson, "The attraction of contributors in free and open source software projects," *J. Strateg. Inf. Syst.*, vol. 22, no. 1, p. 26–45, Mar. 2013.

[4] S. Shah, "Motivation, governance, and the viability of hybrid forms in open source software development," *Management Science*, vol. 52, pp. 1000–1014, 07 2006.

[5] Y. Ye and K. Kishida, "Toward an understanding of the motivation open source software developers," in *Proceedings of the 25th International Conference on Software Engineering*, ser. ICSE '03. USA: IEEE Computer Society, 2003, p. 419–429.

[6] N. Ducheneaut, "Socialization in an open source software community: A socio-technical analysis," *Computer Supported Cooperative Work (CSCW)*, vol. 14, pp. 323–368, 2005.

[7] Y. Fang and D. Neufeld, "Understanding sustained participation in open source software projects," *Journal of Management Information Systems*, vol. 25, no. 4, pp. 9–50, 2009.

[8] J. Marlow, L. Dabbish, and J. Herbsleb, "Impression formation in online peer production: activity traces and personal profiles in github," in *Proceedings of the 2013 conference on Computer supported cooperative work*, 2013, pp. 117–128.

[9] G. Von Krogh, S. Spaeth, and K. R. Lakhani, "Community, joining, and specialization in open source software innovation: a case study," *Research policy*, vol. 32, no. 7, pp. 1217–1241, 2003.

[10] B. Choi, K. Alexander, R. E. Kraut, and J. M. Levine, "Socialization tactics in wikipedia and their effects," in *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, 2010, pp. 107–116.

[11] K. Lakhani and R. Wolf, "Why hackers do what they do: Understanding motivation and effort in free/open source software projects," *Perspectives on Free and Open Source Software*, 09 2003.

[12] I. Rehman, D. Wang, R. G. Kula, T. Ishio, and K. Matsumoto, "Newcomer candidate: Characterizing contributions of a novice developer to github," in *2020 IEEE international conference on software maintenance and evolution (ICSME)*. IEEE, 2020, pp. 855–855.

[13] V. N. Subramanian, I. Rehman, M. Nagappan, and R. G. Kula, "Analyzing first contributions on github: What do newcomers do," *IEEE Software*, 2020.

[14] I. Rehman, D. Wang, R. G. Kula, T. Ishio, and K. Matsumoto, "Newcomer oss-candidates: Characterizing contributions of novice developers to github," *Empirical Software Engineering*, vol. 27, no. 5, pp. 1–20, 2022.

[15] R. G. Kula and G. Robles, *The Life and Death of Software Ecosystems*. Springer, 2019, pp. 97–105.

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

[17] O. Jarczyk, B. Gruszka, S. Jaroszewicz, L. Bukowski, and A. Wierzbicki, "Github projects. quality analysis of open-source software," pp. 80–94, 2014.

[18] J. Han, S. Deng, X. Xia, D. Wang, and J. Yin, "Characterization and prediction of popular projects on github," *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1, pp. 21–26, 2019.