On the Accuracy of Bot Detection Techniques

Mehdi Golzadeh*, Alexandre Decan*, Natarajan Chidambaram*
*Software Engineering Lab,s University of Mons, Belgium
Email: {mehdi.golzadeh,alexandre.decan,natarajan.chidambaram}@umons.ac.be

***Abstract*—Development bots are often used to automate a wide variety of repetitive tasks in collaborative software development. Such bots are commonly among the most active project contributors in terms of commit activity. As such, tools that analyse contributor activity (e.g., for recognizing and giving credit to project members for their contributions) need to take into account the bots and exclude their activity. While there are a few techniques to detect bots in software repositories, these techniques are not perfect and may miss some bots or may wrongly identify some human accounts as bots. In this paper, we present an exploratory study on the accuracy of bot detection techniques on a set of 540 accounts from 27 GitHub projects. We show that none of the bot detection techniques are accurate enough to detect bots among the 20 most active contributors of each project. We show that combining these techniques drastically increases the accuracy and recall of bot detection. We also highlight the importance of considering bots when attributing contributions to humans, since bots are prevalent among the top contributors and responsible for large proportions of commits.**

***Index Terms*—social coding platforms, bot detection techniques, contributor attribution, empirical analysis, GitHub, software repository mining**

I. Introduction

The collaborative nature of open-source software development has led to an increasing rate of creation and use of teamwork development tools [1]. Developers use a number of tools to reduce their workload and increase productivity such as versioning software like Git, social coding platforms such as GitHub and GitLab, DevOps tools, and services like continuous integration and deployment (CI/CD) tools and development bots. Development bots (hereafter referred to as bots) are "automated tools that attempt to free developers from particularly tedious tasks, or support their work in a more general sense" [2]. They perform a wide range of tasks including refactoring, generating bug patches, dependency updating, license checking, and welcoming new contributors [3]–[5]. The behavior of bots may differ depending on the environment they operate in, on the properties of the bot itself and on the interactions of the bot within its environment [2], [6]. While bots are primarily used to increase productivity and improve software quality [7], the presence of bots may have unintended negative consequences. For example, bots may introduce additional communication overhead and can be perceived as distracting and annoying to project contributors [8].

The increasing presence and activity of bots in software repositories makes it challenging for software engineering researchers to study socio-technical aspects of software development since their findings may be biased by not explicitly considering the presence of bots among the contributors [9]. Similarly, it may be important for contributors that their contributions are properly recognized and rewarded since collaborative software development activities are often considered as a criterion for employers when hiring developers [10]. This is especially important when funding or donations are awarded to contributors based on their contributions. While there are tools such as *SourceCred*[1] to support communities in automatically measuring and rewarding value creation, they do not automatically identify bots and their activities so far.

This is where bot identification tools come to the rescue. Such tools aim to distinguish bots from humans in GitHub accounts on the basis of their behaviour. Dey et al. [11] proposed an automatic method to identify bot accounts in git projects based on (i) the presence of the string "bot" at the end of the author name, (ii) commit messages, and (iii) features related to files changed in commits and projects the commits are associated with. Golzadeh et al. [9] proposed an approach and tool to detect bots in GitHub repositories based on the repetitiveness of their comments in issues and pull requests. The approach had been further extended to git commit messages [12].

This paper presents an exploratory study on the accuracy of 5 bot detection techniques on a set of 540 accounts from 27 GitHub projects. We show how prevalent bots and their activities are, and that none of the bot detection techniques are accurate enough to detect bots even among the most active contributors. We also show that combining these techniques increases the accuracy and recall of bot detection but remains insufficient to capture all bots and their activities. This highlights the importance of considering them when conducting socio-technical studies or when attributing contributions, and underlines the need for improved bot detection techniques. In particular, we focus on the following research questions:
RQ1: How accurate are bot detection techniques?
RQ2: How prevalent are bots among the most active contributors?
RQ3: How active are bots in terms of commits?

II. Methodology

Dataset. To carry out our empirical investigation, we selected projects from active software development repositories with a large number of commits and contributors. We relied on the SEART GitHub search tool [13] to filter a set of repositories. We queried repositories that have at least 100 contributors and were not forked and had been active in the

[1]https://sourcecred.io

last 2 months (i.e., in October and December 2021). From these, we randomly selected 27 large and active projects. The selected projects have at least 1,200 commits and 200 contributors. In total, the 27 selected projects account for 175,499 commits from 9,426 contributors and cover a wide variety of programming languages (e.g., Javascript, Python, Java, PHP, Ruby, Rust, Go) and software domains such as software development packages, plugins, and tools.

For each project, we queried the GitHub API to retrieve the 20 most active GitHub accounts in terms of commits, and their respective number of commits. The resulting dataset consists of 540 accounts. Since one of our goals is to evaluate the accuracy of bot detection techniques, we need to determine the correct type (i.e., bot or human) of these accounts. We manually checked these accounts to determine their type, looking for evidence in their profile, their commit activity and their commenting activity. During this process, we found 50 bots out of the 540 considered accounts.

Bot detection techniques. In this paper, we evaluate the accuracy of the following five bot detection techniques:

1) *GitHub account type*. This technique relies on the GitHub API to determine whether a given GitHub account is a bot or not. The GitHub API offers an endpoint[2] to retrieve various metadata for a given GitHub username. Among other, these metadata includes a "type" field that is either "Bot" or "User" depending on whether the corresponding account had been registered as a bot or as a human contributor.
2) *"bot" suffix*. This technique relies on the presence of the string "bot" at the end of the author's name. It has been proposed by Dey et al. [11] as part of an ensemble model, and has notably been used by other researchers [7].
3) *BoDeGHa*. Golzadeh et al. [9] proposed a classification model to identify bots in GitHub pull request and issue activity. Their method measures the similarity of comments and groups them into patterns of similar comments. Bots are then detected based on their lower number of comments patterns. The model has been implemented as part of a tool named BoDeGHa.[3]
4) *BoDeGiC*. Golzadeh et al. [12] further extended the above approach to support git commit messages, and implemented the resulting model as part of a tool named BoDeGiC.[4]
5) *List of bots*. This last technique relies on a predefined list of bots. The list contains the names of 527 known GitHub bot accounts that were manually identified by Golzadeh et al. [9] among 5,000 GitHub accounts.[5]

III. Findings

RQ1: How accurate are bot detection techniques?

We applied the five bot detection techniques on our dataset of 540 contributors. Fig. 1 shows the classifications provided by these techniques. For readability, we only report on the 87 contributors that either correspond to actual bots, or that were classified as bot by at least one of the techniques. Actual bots are shown on the left side of the vertical blue line while actual human contributors are shown on its right. An orange cell indicates that the contributor was identified as a bot by the corresponding technique, while a blue cell indicates that it was identified as a human contributor. Grey cells correspond to cases where there is not enough information for the technique to determine the account type. In the case of BoDeGHa, this corresponds to contributors with less than 10 comments in pull requests or issues. In the case of BoDeGiC, this corresponds to contributors having less than 10 commits made with a committer name matching their GitHub account name.

From this figure, we observe that *list of bots*, *"bot" suffix* and *GitHub account type* are safer techniques, in the sense they do not wrongly classify human contributors as bots. At the same time, they missed many actual bots: from 19 for *list of bots* to 32 for *GitHub account type*. We also observe that *BoDeGiC* effectively captures most bots, but at the same time, wrongly considers several human contributors as bots. *BoDeGHa* exhibits a similar behaviour: it is able to capture 25 out of 50 bots, but wrongly classifies much more humans as bots than *BoDeGiC* (30 versus 9). We note that none of the techniques is perfectly effective in detecting bots. Except for a few cases, the five techniques do not even agree on whether a given account is a bot or not. However, only 4 of the actual bots are not detected as such by any of the techniques, suggesting that a combination of the techniques could lead to an improved bot detection model.

Table I reports on the precision, recall and F1-score of the aforementioned techniques applied on the whole dataset of 540 contributors, distinguishing these scores between bot and human contributors. For completeness, we also report on the overall weighted scores. Given there are far more human contributors than bot contributors in the dataset, these high scores (between 0.898 and 0.966) are mostly driven by the scores obtained for human contributors. To ease the interpretation of these scores, we also provide the scores for a ZeroR model classifying all contributors as human contributors (i.e., the majority class).

The observations that can be made from this table match the ones we made from Fig. 1. In particular, we observe that some techniques (namely *GitHub account type*, *"bot" suffix* and *list of bots*) have a perfect precision but are not able to capture as many bots as *BoDeGiC*. This should not come as a surprise. For example, it is expected that *GitHub account type* has no false positive since it is unlikely that a human contributor would decide to flag his/her own account as a bot. Similarly, *list of bots* relies on a predefined list of bot names that were manually validated by a group of researchers. On the other hand, the precision reached by *"bot" suffix* is surprisingly high since in previous work [9], we found that only around 4% of the contributors having "bot" in their name actually correspond to human contributors.

As observed from Fig. 1, only 4 of the actual bots are

[2]https://docs.github.com/en/rest/reference/users
[3]https://github.com/mehdigolzadeh/BoDeGHa
[4]https://github.com/mehdigolzadeh/BoDeGiC
[5]https://doi.org/10.5281/zenodo.4000388

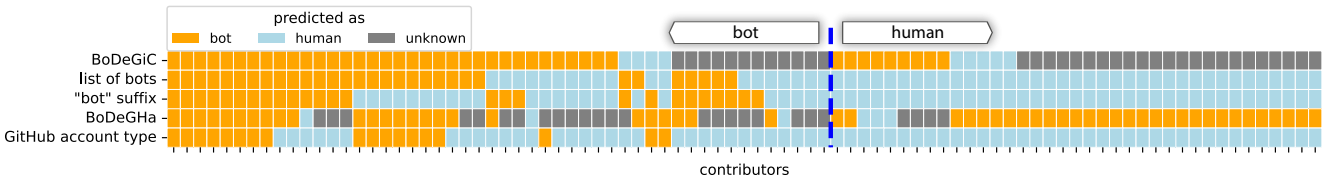

Fig. 1: Classifications ("bot", "human" or "unknown") obtained from the five bot detection techniques. Only actual bots and humans wrongly classified as bot are displayed.

TABLE I: Recall, precision and $F1$-score of bot detection techniques (in ascending order of bot recall).

bot detection technique	bots			humans			overall (weighted scores)		
	recall	precision	$F1$-score	recall	precision	$F1$-score	recall	precision	$F1$-score
Baseline ZeroR	0.000	0.000	0.000	1.000	0.907	0.951	0.907	0.823	0.863
GitHub account type	0.360	**1.000**	0.529	**1.000**	0.939	0.968	0.941	0.944	0.928
BoDeGHa	0.500	0.455	0.476	0.939	0.948	0.944	0.898	0.903	0.900
"bot" suffix	0.520	**1.000**	0.684	**1.000**	0.953	0.976	0.956	0.958	0.949
List of bots	0.620	**1.000**	0.765	**1.000**	0.963	0.981	0.965	0.966	0.961
BoDeGiC	0.680	0.791	0.731	0.982	0.968	0.975	0.954	0.951	0.952
EnsBoD	**0.900**	0.865	**0.882**	0.986	**0.990**	**0.988**	**0.978**	**0.978**	**0.978**

not detected as such by any of the techniques. This suggests that an improved bot detection model can be created by combining the five aforementioned techniques. We build such a model by training a decision tree classifier taking as input the classifications made by each of the five techniques and outputting whether the corresponding contributor is a bot or a human contributor. Since our dataset has a fairly imbalanced number of human and bot contributors, we attributed a class weight inversely proportional to the number of cases. The resulting model is called EnsBoD. We trained and validated it following a 10-fold cross-validation process. The mean scores we obtained are reported on the last row of Table I. Even if it was trained and validated on a small dataset, the EnsBoD model already outperforms any of the five other techniques, with an average recall of 0.9 and an average precision of 0.865 for bots. In the remaining of this paper, we will rely on EnsBoD to separate bots that are correctly identified as bots by a bot detection technique and those that were not, providing an overly optimistic view of the ability to detect bots automatically.

RQ2: How prevalent are bots among the most active contributors?

In Section I we underlined the importance of detecting bots in software repositories, not only for researchers aiming at quantifying and understanding their impact on the development process, but also for properly recognizing and rewarding contributions made by human contributors. This question aims to quantify the prevalence of bots among the 20 most active contributors in the 27 considered projects.

We applied EnsBoD on each of the 540 contributors of our dataset to quantify how many of them can be captured by the bot detection technique. Fig. 2 shows the output of EnsBoD for each project (x-axis) and each contributor (y-axis) sorted by the number of commits they made in the project. In complement to the output of EnsBoD (i.e., "bot" or "human"), we indicate whether the output is correct ("human user" and "correctly classified bot") or not ("human classified as bot" and "missed bot").

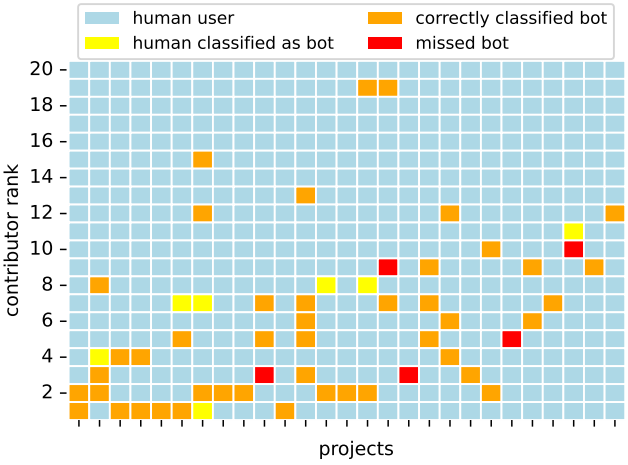

Fig. 2: Rank of top 20 most active contributors in 27 popular open-source software projects.

We observe that all the considered projects are making use of bots, some of them even having 4 different bots among their 20 most active contributors. Interestingly, many of these bots are responsible for most of the activity in the projects. For instance, the most active contributor of 6 projects is a bot, while 18 out of 27 projects have a bot in the top 3 contributors.

We also observe that a non-negligible amount of bots are missed even by our overly optimistic EnsBoD model. For instance, 5 bots are missed and 3 of them are among the 5 most active contributors of the projects. Similarly, a non-negligible amount of actual human contributors are wrongly

classified by EnsBoD: there are 7 human contributors that are detected as bots, of which 1 is the most active contributor in the corresponding project, and 5 others are within the 10 most active contributors.

These findings show the importance of considering bots and their activity in software repositories, not only for conducting empirical research but also when acknowledging or rewarding contributors. While bot detection techniques can help in doing so, even an optimistic combination of them still misses some bots, and still wrongly considers some human contributors as bots.

RQ3: How active are bots in terms of commits?

This question aims to quantify the number of commits made by bots in their respective projects. This is especially important given that tools such as *SourceCred* reward contributors based on their activity, including their commit activity. For each project, we counted the commits made by each of the 20 most active contributors, distinguishing between bot and human contributors. Fig. 3 reports on the proportion of commits made in each commit. As for Fig. 2, we distinguish between human contributors, human contributors classified as bots, bot contributors and bot contributors missed by EnsBoD.

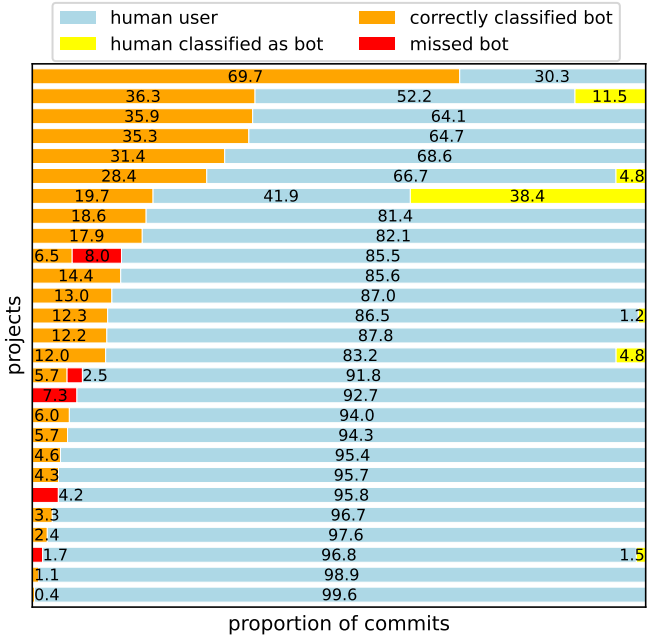

Fig. 3: Proportion of commits made by the 20 most active contributors in each project

The figure shows that the commits made by bots represent up to 69.7% of the commit activity. On average, approximately 16% of the commits in these projects are made by bots (median is 12%), even if bots only account for 9% of the top 20 contributors on average (median is 10%)

While, as observed in previous research question, EnsBoD is able to detect most of the bots, it still misses some of them, and the missed ones are responsible for 8%, 7.3%, 4.2%, 2.5% and 1.7% of the commits in their respective projects (i.e., 4.7% on average). On the other hand, EnsBoD wrongly classified seven human contributors as bots, and these contributors were responsible for 38.4%, 11.5%, 4.8%, 1.5% and 1.2% of the commits (i.e., 10.4% on average).

This again underlines the importance of considering bots when analysing commit activity in software repositories, and highlights the need for better bot detection techniques to do so.

IV. Conclusion

The increasing presence and activity of bots in software repositories makes it challenging for software engineering researchers to study socio-technical aspects of software development since their findings may be biased by not explicitly considering the presence of bots among the contributors. Similarly, it may be important for human contributors that their contributions are properly identified, especially when funding or donations are awarded based on these contributions.

In this paper, we presented an exploratory study on the accuracy of five bot detection techniques on a dataset of 540 contributors corresponding to the top 20 most active contributors in 27 large projects. We found that none of the techniques is perfectly effective in detecting bots. Some of them are accurate, in the sense they generate few (if any) false positives, but at the expense of many bots that remain undetected. However, based on the observation that only a very limited number of bots remain undetected by any of the five techniques, we proposed EnsBoD, a new bot detection model combining these techniques. We evaluated EnsBoD through a 10-fold cross-validation process, and we found that EnsBoD exhibits much better scores. Although this new model has not (yet) been validated on a large dataset, it already shows that combining several bot detection techniques drastically improves bot detection. It correctly captures 45 out of the 50 bots we have in the dataset. On the other hand, it still misclassified 7 human contributors as bot, out of 490.

In a second step, we investigated the prevalence of bots among the top contributors of the 27 considered projects. We found that all the projects make use of a few bots. We also found that bots are commonly found among the most active contributors and are responsible for large proportions of commits in these projects, highlighting the need to consider them when conducting socio-technical studies or when rewarding contributors.

As future work, we plan (1) to extend EnsBoD by integrating additional bot detection techniques (e.g., BIMAN [11]); (2) to compare different ensemble methods to combine them; and (3) to evaluate and validate the approach on a much larger dataset, not only focusing on the 20 most active contributors.

Acknowledgement

This work is supported by DigitalWallonia4.AI research project ARIAC (grant number 2010235), as well as by the Fonds de la Recherche Scientifique – FNRS under grant numbers O.0157.18F-RG43 and T.0017.18.

REFERENCES

[1] F. Thung, T. F. Bissyandé, D. Lo, and L. Jiang. Network structure of social coding in GitHub. In *European Conference on Software Maintenance and Reengineering (CSMR)*, pages 323–326, 2013.

[2] Linda Erlenhov, Francisco Gomes de Oliveira Neto, and Philipp Leitner. An empirical study of bots in software development: Characteristics and challenges from a practitioner's perspective. In *ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, pages 445–455. ACM, 2020.

[3] Marvin Wyrich and Justus Bogner. Towards an autonomous bot for automatic source code refactoring. *International Workshop on Bots in Software Engineering (BotSE)*, pages 24–28, 2019.

[4] Martin Monperrus, Simon Urli, Thomas Durieux, Matias Martinez, Benoit Baudry, and Lionel Seinturier. Repairnator patches programs automatically. *Ubiquity*, (July):1–12, 2019.

[5] S Mirhosseini and C Parnin. Can automated pull requests encourage software developers to upgrade out-of-date dependencies? In *International Conference on Automated Software Engineering (ASE)*, pages 84–94, 2017.

[6] Carlene Lebeuf, Alexey Zagalsky, Matthieu Foucault, and Margaret Anne Storey. Defining and classifying software bots: A faceted taxonomy. *1st International Workshop on Bots in Software Engineering*, pages 1–6, 2019.

[7] Samaneh Saadat, Natalia Colmenares, and Gita Sukthankar. Do bots modify the workflow of github teams? In *2021 IEEE/ACM Third International Workshop on Bots in Software Engineering (BotSE)*, pages 1–5, 2021.

[8] Mairieli Wessel, Igor Wiese, Igor Steinmacher, and Marco Aurelio Gerosa. Don't disturb me: Challenges of interacting with software bots on open source software projects. *Human-Computer Interaction*, 5, 2021.

[9] Mehdi Golzadeh, Alexandre Decan, Damien Legay, and Tom Mens. A ground-truth dataset and classification model for detecting bots in GitHub issue and PR comments. *Journal of Systems and Software*, 175, 2021.

[10] Claudia Hauff and Georgios Gousios. Matching GitHub developer profiles to job advertisements. In *Working Conference on Mining Software Repositories*, pages 362–366. IEEE/ACM, 2015.

[11] Tapajit Dey, Sara Mousavi, Eduardo Ponce, Tanner Fry, Bogdan Vasilescu, Anna Filippova, and Audris Mockus. Detecting and characterizing bots that commit code. In *International Conference on Mining Software Repositories*, pages 209–219. ACM, 2020.

[12] Mehdi Golzadeh, Alexandre Decan, and Tom Mens. Evaluating a bot detection model on git commit messages. In *19th Belgium-Netherlands Software Evolution Workshop (BENEVOL)*, volume 2912. CEUR Workshop Proceedings, 2020.

[13] Ozren Dabic, Emad Aghajani, and Gabriele Bavota. Sampling projects in github for MSR studies. In *18th IEEE/ACM International Conference on Mining Software Repositories, MSR 2021*, pages 560–564. IEEE, 2021.