RABBIT: A tool for identifying bot accounts based on their recent GitHub event history

Natarajan Chidambaram
Software Engineering Lab
University of Mons
Mons, Belgium
natarajan.chidambaram@umons.ac.be

Tom Mens
Software Engineering Lab
University of Mons
Mons, Belgium
tom.mens@umons.ac.be

Alexandre Decan
F.R.S.-FNRS Research Associate
Software Engineering Lab
University of Mons
Mons, Belgium
alexandre.decan@umons.ac.be

ABSTRACT

Collaborative software development through GitHub repositories frequently relies on bot accounts to automate repetitive and error-prone tasks. This highlights the need to have accurate and efficient bot identification tools. Several such tools have been proposed in the past, but they tend to rely on a substantial amount of historical data, or they limit themselves to a reduced subset of activity types, making them difficult to use at large scale. To overcome these limitations, we developed RABBIT, an open source command-line tool that queries the GitHub Events API to retrieve the recent events of a given GitHub account and predicts whether the account is a human or a bot. RABBIT is based on an XGBoost classification model that relies on six features related to account activities and achieves high performance, with an AUC, F1 score, precision and recall of 0.92. Compared to the state-of-the-art in bot identification, RABBIT exhibits a similar performance in terms of precision, recall and F1 score, while being more than an order of magnitude faster and requiring considerably less data. This makes RABBIT usable on a large scale, capable of processing several thousand accounts per hour efficiently.

KEYWORDS

GitHub events, classification model, bot identification

ACM Reference Format:
Natarajan Chidambaram, Tom Mens, and Alexandre Decan. 2024. RABBIT: A tool for identifying bot accounts based on their recent GitHub event history. In *21st International Conference on Mining Software Repositories (MSR '24), April 15–16, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3643991.3644877

1 INTRODUCTION

The GitHub social coding platform enables collaborative software development, involving many activities such as updating dependencies, opening new issues and pull requests, performing code reviews, committing code, creating branches and tags, and publishing releases [4, 6]. As some of these activities can be repetitive and error-prone, software repositories tend to use automation tools such as GitHub Actions workflows, GitHub Apps and bot accounts. Bot accounts (referred to as bots in the remainder of this paper) are regular GitHub accounts that use some automated software agent to perform activities in the repositories they are involved in [13].

Bots have been shown to belong to the top contributors in GitHub repositories [10]. Disregarding the contributions of such automated agents could lead to wrong decisions for organisations aiming to accredit the top human contributors in their projects, as well as to incorrect conclusions during empirical socio-technical analyses [2].

To distinguish bots from human contributors, several bot identification techniques have been proposed in the past [1, 7, 8]. However, a large amount of data of different nature (e.g., commit messages, comments in issues and pull requests, as well as account metadata such as login, name, bio, number of followers, account tag and so on) is required by these techniques to identify bots, making them difficult to use at large scale. Furthermore, most of these techniques detect bots by considering a limited subset of activity types only, even though bots have been active in as many as 24 distinct activity types [4].

Our goal is to create a bot identification tool that takes all these activity types into account, while being able to process thousands of accounts on an hourly basis with a limited amount of data. To achieve this goal, we developed RABBIT, a recursive acronym for "RABBIT is an Activity-Based Bot Identification Tool". RABBIT is publicly released on GitHub[1] as an open source Python tool that can be easily installed using pip. It takes as input a GitHub account, extracts its recent events from the GitHub Events API, and considers the account's activities to quickly determine if it corresponds to a *bot* or a *human*.

2 RELATED WORK

Many bot-based studies have been carried out in the past, for different purposes. Some focused on categorizing bots in GitHub repositories [12, 13], or identifying the challenges in using bots during collaborative software development [11, 14]. To assist such studies, several bot identification techniques and tools have been proposed [1, 7–9]. This section presents some of these studies and bot identification techniques.

Categorising bots. By analysing the activities of 48 different bots in 93 GitHub projects, Wessel et al. [13] identified 12 different tasks that bots perform in those projects. Those tasks include reviewing pull requests, running automated tests, building projects, analysing

[1] https://github.com/natarajan-chidambaram/RABBIT

MSR '24, April 15–16, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0587-8/24/04…$15.00
https://doi.org/10.1145/3643991.3644877

and updating dependencies, and creating issues. They identified significant differences in 44 GitHub projects in terms of number of commits, number of changed files, number of comments and so on before and after bot adoption. Wang et al. [12] identified 201 bots in 613 GitHub projects and grouped them into 6 categories based on their tasks: (1) CI assistance, (2) issue and pull request management, (3) code review assistance, (4) dependency and security analysis, (5) developer and user community support, and (6) documentation generation. They identified that 60% of these projects use at least one bot to automate routine tasks and 74 out of 201 bots belong to more than one category.

Challenges in using bots. Storey et al. [11] studied the unintended negative impacts that bots might create in software projects and among developers. Although bots automate repetitive and error-prone tasks, they might not adapt to the cultural changes in the organisation, reduce interaction between team members, and bring interruptions and distractions. Wessel et al. [14] interviewed 21 practitioners and identified 25 challenges that development bots bring to software projects. They categorised these challenges into three categories: interaction with bots (e.g., intimidating newcomers, providing non-comprehensive feedback, impersonating developers), bot adoption (e.g., burden to set up configuration files, limited configurations, handling technical failures) and bot development (e.g., building multitasking bots, restricted bot actions by GitHub API, hosting and deploying bots).

Bot identification tools. Dey et al. [7] developed BIMAN, an ensemble model that identifies bot involved in commits based on a combination of three different approaches (i) having 'bot' in the account name (e.g., *jenkins-bot*), (ii) text similarity and patterns in commit messages, and (iii) features related to files modified in commits (e.g., number of unique file extensions). They achieved a precision of 0.667, recall of 0.866 and F1-score of 0.754. The main limitation of this approach is that it only considers commit activities.

Golzadeh et al. [8] developed BoDeGHa, a classification model and associated tool to identify bots based on features related to comments in issues and pull requests. They also developed BoDeGiC [9] a model and associated tool to identify bots based on commit messages. For BoDeGHa, they created a manually labelled ground-truth dataset of 5,000 GitHub contributors of which 527 were bots, and for BoDeGiC they trained the model on 6,922 accounts that made commits in git repositories. The limitations of these approaches are that (i) bot identification requires data related to comments in commits, issues and pull requests, and (ii) predictions are made on a single-repository level, implying that predictions may diverge across repositories [3].

Abdellatif et al. [1] developed BotHunter, a state-of-the-art and accurate bot identification tool combining features used by BoDeGHa, BIMAN and some additional ones. Since the model is using features based on data of different nature (such as account bio, comments, issues, pull requests and comments) it needs to download a large amount of data to provide its prediction, requiring hours to process thousands of contributors.

3 GROUND-TRUTH DATASET

To train and test a classification model for our bot identification tool RABBIT, we need a sufficiently large ground-truth dataset of GitHub accounts that are labeled as bots or humans. We started the dataset creation by combining (i) labelled accounts that were published in our previous work [4] and (ii) accounts that were published by Wyrich et al. [15]. From these datasets, we removed the accounts that performed less than 5 events (a condition required by RABBIT's classification model). We also ignored GitHub Apps as they are already marked as 'Bot' by GitHub. Applying both filters resulted in a dataset of 234 bots and 502 humans from [4] and 629 accounts from [15].

These 629 accounts were independently labelled as *bot*, *human* or *not sure* by two researchers, relying on information derived from the account's GitHub profile, the GitHub activity available for this account, and the account's event sequence that was extracted through the GitHub Events API. The label *not sure* was used in case a rater did not have enough information to come to a decision, or when the account exhibited a behaviour that mixed human and bot activity. Cohen's kappa (κ) was used to measure inter-rater agreement. The raters agreed on the label for 561 accounts and disagreed for 68 accounts, leading to a *substantial agreement* ($\kappa = 0.79$). A third researcher was involved to discuss all 68 cases on which the first two raters disagreed, as well as the 13 cases where both raters were *not sure*. After this discussion, agreement was reached to label 410 accounts as *bot*, 189 as *human*, while 30 accounts were discarded since at least one rater was still *not sure*.

Overall, we identified 1,335 accounts, of which 644 are bots and 691 are humans.

4 ACCOUNT CLASSIFICATION MODEL

RABBIT relies on an account classification model for identifying bots. To create such a model we derived a set of features from the activity sequences extracted for each GitHub account in the ground-truth dataset. In previous work [5], we identified distinguishing characteristics for bot and human users. Carrying out some further experimentation we came up with 45 potential distinguishing features and after feature selection we identified the six most important features for the classification model:

(1) the number of distinct activity types carried out by the account,
(2) the mean number of activities carried out (by the account) per activity type,
(3) the median time between two consecutive activities of different types carried out by the account,
(4) the mean number of activities carried out (by the account) per repository to which the account has contributed,
(5) the number of distinct owners of all repositories the account has contributed to,[2]
(6) the (Gini) inequality in the time between consecutive activities carried out by the account.

To create a classification model based on these six features, we split the dataset into a *training/validation* set containing 60% of the accounts. The remaining 40% were kept for testing the model on

[2]Any GitHub repository has the form www.github.com/OWNER/REPO where OWNER is the repository owner, and REPO is the repository name.

Table 1: Performance scores of the best model for seven considered classifiers and the ZeroR baseline, in descending order of weighted F1 score

	bots		humans		weighted			
classifier	P	R	P	R	P	R	F1	AUC
XGBoost	**0.939**	0.933	0.941	**0.940**	**0.940**	**0.937**	**0.937**	0.974
GB	0.934	0.933	0.939	0.938	0.937	0.936	0.936	**0.978**
RF	0.929	**0.938**	**0.944**	0.931	0.937	0.935	0.934	0.976
LDA	0.915	0.905	0.915	0.919	0.915	0.912	0.912	0.893
DT	0.917	0.833	0.860	0.929	0.888	0.983	0.882	0.958
SVM	0.799	0.864	0.865	0.798	0.833	0.830	0.829	0.937
CNB	0.751	0.944	0.933	0.707	0.845	0.821	0.819	0.894
ZeroR	0.000	0.000	0.519	1.000	0.269	0.519	0.354	0.500

unseen data. We used stratified splitting to have a similar proportion of bots and humans in both sets. This resulted in a training set of 801 accounts composed of 386 bots (i.e., 48.2%) and 415 humans, and a test set of 534 accounts composed of 258 bots (i.e., 48.3%) and 276 humans.

We used the training/validation set to determine the best classification model, following a grid-search hyper-parameter tuning 10-fold cross-validation process. We evaluated seven commonly used classifiers as candidates for the model: Decision Trees (DT), Random Forest (RF), Gradient Boosting (GB), eXtreme Gradient Boosting (XGB), Support Vector Machine (SVM), Linear Discriminant Analysis (LDA), and Complement Naive Bayes (CNB). We compared these classifiers against the ZeroR baseline, a classifier that predicts the majority class (i.e., *human*) for all accounts and hence does not have any actual predictive power.

To find the best model, we varied the input parameters of these classifiers, resulting in 3,734 candidate models. To evaluate the performance of these models on the training set, we used Area Under the ROC Curve (AUC) and the weighted version of precision (P), recall (R), and F1 score. Table 1 reports the performance scores for the best model for each considered classifier. XGBoost performed better than the other models with an AUC of 0.974, weighted precision of 0.940, weighted recall of 0.937, and weighted F1 of 0.937. The second and third best models GradientBoost and Random Forest had a performance that is quite close to XGBoost. The evaluation of the model on the test set with 40% of unseen data is presented in Section 6.

5 TOOL DESCRIPTION

Using the classification model identified in Section 4, we developed RABBIT as an open source command-line tool to allow researchers and practitioners to easily determine the bot nature of GitHub accounts. RABBIT is publicly released on https://github.com/natarajan-chidambaram/RABBIT and can be installed using pip install git+https://github.com/natarajan-chidambaram/RABBIT.

The mandatory inputs are a GitHub API key and either the name of a single account or a text file of GitHub accounts (one per line) to be classified. For each account, RABBIT first checks whether it corresponds to a GitHub App by verifying that the account name ends with [bot][3] and the type returned by the GitHub Users API is Bot. For the remaining user accounts, RABBIT queries the GitHub Events API to extract up to 300 events, converts these event sequences into activity sequences, computes the six features required by the classification model, and outputs for each account the prediction and its confidence.

[3]GitHub reserves the suffix [bot] for GitHub Apps only.

Listing 1 gives an example of using RABBIT in practice, providing eight login names in the input file `names.txt`. (Login names for human users have been anonymised to comply with the GDPR regulations.) RABBIT predicts the type for each account: *app*, *bot*, *human*, *unknown* or *invalid*. If the account made less than the default minimum of five events it is reported as *unknown*, whereas it is reported as *invalid* if the account does not exist in GitHub.

```
% rabbit --input_file names.txt --key <MYAPIKEY>
             account      prediction          confidence
 github_actions[bot]             app                 1.0
      johnpbloch_bot             bot               0.998
  tensorflow_jenkins             bot               0.993
        <ANONYMISED>           human                 1.0
             codecov             bot               0.986
        <ANONYMISED>           human               0.995
              cxbot         unknown                 NaN
           renovate         invalid                 NaN
```

Listing 1: Example of RABBIT usage

RABBIT provides a *confidence* score for the reported predictions. For *app*, the confidence is always 1.0. For *unknown* and *invalid* it is NaN. For bots and humans, the confidence is based on the *probability* (between 0 and 1) that the model classifies the account as a bot. A probability > 0.5 leads to a *bot* decision, while a probability ≤ 0.5 leads to a *human* decision. The *confidence* (between 0 and 1) is computed as $|probability - 0.5| * 2$.

RABBIT has several optional arguments to change its default behaviour. A full description of these optional arguments can be obtained by typing `rabbit --help` in the terminal. In particular,

`--min-events` specifies the minimum number of events that need to be considered for making a prediction (default is 5).

`--max-queries` specifies how many API queries should be made for each account, either 1, 2 or 3 (default is 3).

`--verbose` outputs additional information, such as the values of all features that were used to make the prediction.

`--csv` and `--json` can be used to save the output as a comma-separated-values or a JSON file, respectively.

6 TOOL EVALUATION

6.1 Predictive power

We evaluated RABBIT on a test set corresponding to 40% of the ground-truth dataset (258 bots and 276 humans). Table 2 reports the confusion matrix and performance metrics. One can observe that 21 bots are misclassified as *human* (FN) and 21 humans are misclassified as *bot* (FP). This resulted in an overall precision, recall and F1 of 0.919 each and AUC of 0.921.

Table 2: Confusion matrix and performance on test set

	confusion matrix		weighted scores			
	classified as bot	classified as human	P	R	F1	AUC
258 bots	TP: 237	FN: 21	.919	.919	.919	-
276 humans	FP: 21	TN: 255	.924	.924	.924	-
534 accounts	258	276	.919	.919	.919	.921

When analysing those misclassifications, we observed that they could be considered as outliers with atypical behaviour. For example, some of the misclassified accounts performed very few events.

We hypothesised that the number of misclassifications is mostly driven by the number of available events for these accounts. For this reason, we clustered the accounts in three bins based on the number of events they performed: (0,100], (100,200] and (200,300]. For each bin, we computed the confusion matrix. Table 3 provides the results. In the test set, among 534 accounts, 248 (85 bots and 163 humans) have performed at most 100 events, 86 (53 bots and 33 humans) accounts have performed more than 100 events and at most 200 events and the remaining 200 accounts (120 bots and 80 humans) have performed more than 200 and at most 300 events. One can observe that the number and proportion of correct classifications (TP and TN) increase as the number of events increases. The opposite observation can be made for misclassifications (FN and FP). By considering accounts that have performed (0,100] events, 86.3% (214 out of 248) of them are correctly identified, while this proportion increases to 94.2% (81 out of 86) when considering the accounts that have performed (100,200] events, and to 98.5% for accounts with more events.

6.2 Runtime evaluation

We compared the performance of RABBIT with the state-of-the-art bot identification tool BotHunter [1] by executing it on 23 January 2024 on the test set of 534 accounts. This resulted in an overall precision of 0.939, a recall of 0.899 and an F1-score of 0.919. These results are very similar to those obtained with RABBIT.

We also compared the execution time and the amount of data required to make these predictions. The execution time was measured on an Intel Xeon W-1290P with 3.7 GHz CPU. We relied on the network traffic monitoring tool NetHogs 0.8.7 to measure the amount of data required and the number of API queries needed for both tools.

Table 4 reports on the results. We observe that BotHunter requires 3.5 times more data than RABBIT. With respect to execution time, RABBIT only took 8 minutes to process the 534 accounts, while BotHunter took more than 50 times longer (7 hours and 19 minutes). During these 7 hours and 19 minutes, BotHunter was idle for around 70 minutes because it repeatedly exceeded GitHub's hourly API rate limit (5,000 queries per hour). Indeed, BotHunter required more than 37K queries on the GitHub API to retrieve the data for these 534 accounts. In contrast, RABBIT only needed 1,040 queries. By extrapolating this number, RABBIT should be able to process more than 2,500 accounts per hour while staying under GitHub's hourly API rate limit.

Table 3: Confusion matrix w.r.t. the number of events

	number of events		
	(0,100]	(100,200]	(200,300]
bots	85	53	120
TP	68 (80%)	50 (94.3%)	119 (99.2%)
FN	17	3	1
humans	163	33	80
TN	146 (89.6%)	31 (93.9%)	78 (97.5%)
FP	17	2	2

In summary, RABBIT is able to achieve similar prediction performance than the state-of-the-art bot identification tool BotHunter, while being more than an order of magnitude faster and requiring considerably less data.

Table 4: Runtime evaluation of RABBIT and BotHunter

Tool	execution time	data received	API queries
BotHunter	7h19m	214.1MB	37,130
RABBIT	**8m**	**61.5MB**	**1,040**

6.3 Limitations

A first limitation is that RABBIT relies only on an account's public events, since the API does not provide access to events in private repositories. Second, RABBIT only relies on information obtained from the GitHub Events API. One could consider extracting other event types through additional APIs, such as GitHub's Issues API, but this would require more API queries, more data to be downloaded, and more execution time. Third, since the Events API returns the most recent events only, RABBIT is unable to predict accounts that were not recently active. Finally, as for any known bot identification tool or even for human raters, it is impossible to predict so-called *mixed accounts* that combine both human and bot behaviour [2].

7 CONCLUSION

During socio-technical empirical studies, the contributions made by GitHub accounts should be treated differently depending on their nature (bot or human). Existing bot identification tools are often impractical at scale since they usually require large amount of data, take a lot of time to process thousands of accounts, or are based on a limited set of characteristics (such as the presence of "bot" in the name). We therefore proposed RABBIT, an open source bot identification tool that accurately predicts whether a GitHub account is a bot or a human, by relying on its recent events made on GitHub. RABBIT uses an XGBoost classification model to predict the account type. Trained on a ground-truth of 644 bots and 691 human accounts, the model achieves a very good performance on unseen data, reaching a precision, recall, F1-score of 0.919 each and AUC-ROC of 0.921. To achieve this precision, the model requires only six different features related to the amount and duration of activities, to activity types and to the repositories the account has contributed to. Compared to the state-of-the-art, RABBIT is more than 50 times faster and requires 3.5 times less data. RABBIT can process thousands of accounts per hour while staying under GitHub's API hourly rate limit, making it suitable for large scale analysis.

ACKNOWLEDGMENTS

This work is supported by Service Public de Wallonie Recherche under grant number 2010235 - ARIAC by DigitalWallonia4.AI, by the Fonds de la Recherche Scientifique – FNRS under grant numbers J.0147.24, T.0149.22, and F.4515.23. We thank Marvin Wyrich for sharing his account dataset, and Youness Hourri for helping in establishing the ground-truth dataset.

REFERENCES

[1] Ahmad Abdellatif, Mairieli Wessel, Igor Steinmacher, Marco A. Gerosa, and Emad Shihab. 2022. BotHunter: An Approach to Detect Software Bots in GitHub. In *International Conference on Mining Software Repositories (MSR)*. 6–17. https://doi.org/10.1145/3524842.3527959

[2] Nathan Cassee, Christos Kitsanelis, Eleni Constantinou, and Alexander Serebrenik. 2021. Human, bot or both? A study on the capabilities of classification models on mixed accounts. In *International Conference on Software Maintenance and Evolution (ICSME)*. IEEE. https://doi.org/10.1109/ICSME52107.2021.00075

[3] Natarajan Chidambaram, Alexandre Decan, and Mehdi Golzadeh. 2022. Leveraging Predictions from Multiple Repositories to Improve bot Detection. In *International Workshop on Bots in Software Engineering (BotSE)*. IEEE. https://doi.org/10.1145/3528228.3528403

[4] Natarajan Chidambaram, Alexandre Decan, and Tom Mens. 2023. A Dataset of Bot and Human Activities in GitHub. In *International Conference on Mining Software Repositories (MSR)*. IEEE/ACM, 465–469. https://doi.org/10.1109/MSR59073.2023.00070

[5] Natarajan Chidambaram, Alexandre Decan, and Tom Mens. 2023. Distinguishing Bots From Human Developers Based on Their GitHub Activity Types. In *Seminar on Advanced Techniques & Tools for Software Evolution (SATToSE)*. CEUR.

[6] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. 2012. Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository. In *International Conference on Computer Supported Cooperative Work (CSCW)*. ACM, 1277–1286. https://doi.org/10.1145/2145204.2145396

[7] Tapajit Dey, Sara Mousavi, Eduardo Ponce, Tanner Fry, Bogdan Vasilescu, Anna Filippova, and Audris Mockus. 2020. Detecting and Characterizing Bots That Commit Code. In *International Conference on Mining Software Repositories (MSR)*. ACM, 209–219. https://doi.org/10.1145/3379597.3387478

[8] Mehdi Golzadeh, Alexandre Decan, Damien Legay, and Tom Mens. 2021. A ground-truth dataset and classification model for detecting bots in GitHub issue and PR comments. *Journal of Systems and Software* 175 (2021). https://doi.org/10.1016/j.jss.2021.110911

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

[10] Mehdi Golzadeh, Tom Mens, Alexandre Decan, Eleni Constantinou, and Natarajan Chidambaram. 2022. Recognizing Bot Activity in Collaborative Software Development. *IEEE Software* 39, 5 (2022), 56–61. https://doi.org/10.1109/MS.2022.3178601

[11] Margaret-Anne Storey and Alexey Zagalsky. 2016. Disrupting Developer Productivity One Bot at a Time. In *International Symposium on Foundations of Software Engineering (FSE)*. ACM SIGSOFT, 928–931. https://doi.org/10.1145/2950290.2983989

[12] Zhendong Wang, Yi Wang, and David Redmiles. 2022. From Specialized Mechanics to Project Butlers: The Usage of Bots in Open Source Software Development. *IEEE Software* 39, 5 (2022), 38–43. https://doi.org/10.1109/MS.2022.3180297

[13] Mairieli Wessel, Bruno Mendes De Souza, Igor Steinmacher, Igor S. Wiese, Ivanilton Polato, Ana Paula Chaves, and Marco A. Gerosa. 2018. The power of bots: Understanding bots in OSS projects. *International Conference on Human-Computer Interaction (CHI)* (2018). https://doi.org/10.1145/3274451

[14] Mairieli Wessel, Igor Wiese, Igor Steinmacher, and Marco A Gerosa. 2021. Don't Disturb Me: Challenges of Interacting with Software Bots on Open Source Software Projects. *International Conference on Human-Computer Interaction (CHI)* 5 (2021). https://doi.org/10.1145/3476042

[15] Marvin Wyrich, Raoul Ghit, Tobias Haller, and Christian Müller. 2021. Bots Don't Mind Waiting, Do They? Comparing the Interaction With Automatically and Manually Created Pull Requests. In *International Workshop on Bots in Software Engineering (BotSE)*. 6–10. https://doi.org/10.1109/BotSE52550.2021.00009