

Evaluating a bot detection model on git commit messages

Mehdi Golzadeh
Software Engineering Lab
University of Mons, Belgium
mehdi.golzadeh@umons.ac.be

Alexandre Decan
Software Engineering Lab
University of Mons, Belgium
alexandre.decan@umons.ac.be

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

Abstract—Detecting the presence of bots in distributed software development activity is very important in order to prevent bias in socio-technical empirical studies. In previous work, we proposed a classification model to detect bots in GitHub repositories based on the pull request and issue comments of GitHub accounts. The current study generalises the approach to git contributors based on their commit messages. We train and evaluate the classification model on a large dataset of 6,922 git contributors. The original model based on pull request and issue comments obtained a precision of 0.77 on this dataset, whereas re-training the classification model on git commit messages increased the precision to 0.80. As a proof-of-concept, we implemented this model in BoDeGiC, an open source command-line tool to detect bots in git repositories.

Index Terms—bot detection, distributed software development, classification model, empirical analysis

I. INTRODUCTION

Many open source software projects rely on *development bots* or *DevBots* (hereafter referred to as *bots* for short) to reduce the workload of project maintainers and contributors [1]. These bots play a prominent role and in the software development process, especially in online coding platforms. Bots submit pull requests (PR) to update dependencies [2], merge PRs [3], submit issues for different problems [3], help in code review [4], and commit code directly to the codebase [5].

Today, no one can dispute the significance of the presence of bots in social coding platforms, but to understand how they impact the software development process it is necessary to study how they are used. Some studies are proposing new bots [6]–[11] and some are trying to analyze the adoption of bots [4], [12], [13]. A prerequisite for understanding the presence of bots in software repositories is the ability to detect them reliably. Previous studies have proposed methods to automatically identify bots by taking into account specific characteristics of bots. Dey et al. [5] proposed an ensemble technique to identify bots relying on git commit information such as commit messages, contributors name, and commit files. In a prior study [14], we proposed an automatic classification model to identify bots in git repositories hosted in GitHub based on the content of pull request (PR) and issue comments.

This research is supported by the joint FNRS / FWO Excellence of Science project SECO-ASSIST and FNRS PDR T.0017.18.

This paper aims to evaluate the generalisability of the classification model we proposed in [14] by applying it to git commit messages. To do so, we evaluate the model on a set of 6,922 contributors extracted from Dey et al.’s dataset [5]. As will be shown, our classification model is able to attain a good precision, even if it was trained on PR and issue comments rather than on git commit messages. Retraining the classifier specifically on git commit messages slightly improves its precision. Finally, to ease the use of our model by practitioners and researchers, we propose BoDeGiC, a command-line tool that detects bots in git repositories.

II. BACKGROUND

In previous work [14], we developed and proposed a classification model and associated tool, called BoDeGHa¹ to identify bots in GitHub repositories based on the comments they made in pull requests and issues. To train and evaluate this model, we created a ground-truth dataset of GitHub contributors that were manually labeled as bots and humans by three raters. The evaluation of the model on the test set achieved a very high overall precision.

In parallel to this prior research, Dey et al. [5] conducted a similar study aiming to identify bots in git repositories based on git commit data. They created a ground-truth dataset based on the World of Code (WoC) dataset [15] containing 73M git repositories. Their ground-truth dataset is composed of 13,150 bots and 13,150 humans that were identified using BIN (for Bot Identification by Name), a technique relying on the presence of some keywords (e.g., “bot”) to identify bots. Based on this ground-truth dataset, they proposed BIMAN (for Bot Identification by commit Message, commit Association, and contributor Name) to detect bots based on commit activity.

Although both aforementioned studies pursue the same goal, they are different in essence. The most prominent difference is the type of data on the basis of which bots are being identified. In [14], the identification of bots is based on comments made in PRs and issues, while BIMAN relies on git commit information to identify bots from humans. As a consequence, the set of considered contributors is different since contributors active in PR or issue comments are not necessarily active in code commits, and vice versa. Another difference is that we

¹<https://github.com/mehdigolzadeh/BoDeGHa>

restricted our dataset to contributors active in GitHub while BIMAN works on git repository, even if they are not hosted on GitHub. Moreover, when a contributor is active in more than one repository, we considered each repository individually while BIMAN aggregates the activity of multiple repositories for each contributor.

III. METHOD AND APPROACH

A. Initial classification model

In prior work [14], we proposed an approach to distinguish bots from humans using their PR and issue commenting activities in GitHub repositories. We developed a model (and associated command-line tool) with a high precision to predict whether a contributor is a bot or human based on the comments made in issues and PRs. The underlying idea was that bots perform automated tasks, therefore, they are assumed to have more repetitive comments than humans. To capture this repetition of comments the model was trained using four features related to the comments associated to a contributor: (i) we measured the number of comment patterns on the basis of a compound comment similarity metric. We hypothesized that the less comment patterns, the more likely a contributor is a bot; (ii) we computed the Gini coefficient to capture the inequality of the number of comments in patterns; (iii) we counted the number of comments since it allows to distinguish between contributors having a similar number of comment patterns; (iv) and we counted the number of empty comments, driven by the assumption that bots are supposed to have meaningful, non-empty, messages. We refer to [14] for the rationale behind these features and how we computed them.

To evaluate the performance of the model, we relied on a ground-truth dataset composed of 5,000 distinct GitHub contributors. To create such a ground-truth dataset, we manually labeled each contributor as a “bot” or a “human” with high inter-rater agreement. The final dataset contains 527 bots and 4,473 humans. We trained and compared various classification models, and we achieved highest results ($F1 = 0.99$) with a random forest classifier. Not only does the model perform well in general, it also achieves high precision and recall for each class (i.e., for “bot” and for “human”): a precision of 0.91 for bots and 0.99 for humans, recall of 0.96 for bots and 0.99 for humans, and $F1$ of 0.94 for bots and 0.99 for humans. Only 8 out of 211 bots and 20 out of 1,789 humans were misclassified by the model.

B. Ground-Truth Commit Message Dataset

Our previous model performed very well to identify bots based on the repetitiveness of their comments, a text-based activity. Since git commit messages are also text-based, and since we can expect that bots active in commits exhibit a similar kind of similarity, it seems natural to apply our model on git commit messages as well.

In this paper, we will first evaluate how our model trained on PR and issue comments performs when applied as-is to git commit messages. Then, we will evaluate the approach we developed in [14] applied on git commit messages, by

TABLE I
SUMMARY OF THE DATASET CHARACTERISTICS.

original commit dataset from [5]	number
# git repositories	6,394
# commits	311,622
# distinct contributors	6,922
↪ # bots	3,380
↪ # humans	3,542

training a new model. To do so, we need a labeled dataset of contributors and their commit messages. We rely on the dataset of git commits that was used by [5] and that has been made publicly available.²

We transformed the dataset to conform to the input format required by our classification model. First of all, as explained in previous section, our model expects the set of features of a contributor to be computed on one repository at a time. Therefore, if a contributor is active in more than one repository, we split its activity by repository. Our approach being based on the assumption that bots exhibit more repetitive activities, it cannot be applied for contributors that do not have enough activities. In [16], we observed that the performance of the model decreased when less the number of comments was below 10. Similarly, in this work, we will only consider contributors that have at least 10 commit messages. As in previous work, we rely on 100 commit messages to compute the set of features. This upper bound significantly reduces the computational cost, and has been shown to be more than sufficient to achieve a very high precision in [14]. Not imposing an upper bound would require to consider all commits for each contributor-repository pair, and some pairs have more than 20,000 commits. Even for pairs with over 1,000 commit messages the performance begins to slow down considerably.

After having performed these steps, the resulting dataset contains 311,622 commit messages from 6,922 contributors, 3,380 whom have been labeled as bots and 3,542 as humans. This accounts for around 25% of the original dataset. The dataset characteristics are summarized in Table I.

IV. RESULTS

In this section, we will first evaluate how the classification model trained on PR and issue comments performs when applied as-is to git commit messages. Then, we will evaluate how the approach developed in [14] applies to git commit messages, by training a new classification model.

We start by applying the model that was trained on PR and issue comments to see how it performs when applied on a new kind of data (i.e., on git commit messages). For each of the 6,922 contributors in the dataset, we computed the features required by the model, and we asked the model to predict whether the contributor is a bot or a human. We then compared the prediction with the ground-truth, enabling us to compute the precision P of the model, its recall R and its $F1$ -score. The results are reported in Table II.

²<https://zenodo.org/record/4042126>

TABLE II
EVALUATION OF THE CLASSIFICATION MODEL OF [14].

	classified as bot	classified as human	P	R	F1
Bot	2,631 (TP)	749 (FN)	0.76	0.78	0.77
Human	848 (FP)	2,694 (TN)	0.78	0.76	0.77
weighted avg			0.77	0.77	0.77

The model achieved a precision of 0.77, with about 22.1% (749 out of 3,380) false negatives (FN) of bots misclassified as humans, and about 23.9% (848 out of 3,542) false positives (FP) of humans misclassified as bots. Most contributors are correctly classified as bot or human by the existing classification model even though the model was not trained on git commit messages but on PR and issue comments.

To see how the approach developed in [14] behaves on git commit messages, we trained a new model on git commit messages as opposed to the previous model that was trained on PR and issue comments. To do so, we divided the ground-truth dataset into two disjoint subsets. 60% of the data are used to perform grid-search cross-validation to select the classifier and its parameters. A test set composed of the remaining 40% is used to evaluate the selected classifier on unseen data.

At the end of the cross-validation set, we obtained 91 different classifiers. The performance of these classifiers was measured using traditional performance metrics of precision P , recall R , and $F1$ -score for the population of each class (i.e., for bots B and human H). We report the highest $F1$ -score for each classifier in Table III, in descending order. We retained the random forest (RF) classifier, as it slightly outperforms the other classifiers. Its score was obtained with the *entropy* split criterion, 20 estimators (i.e., trees) and a tree depth of 8.

We evaluated the selected classifier on the test set containing the remaining 40% data. This test set includes 2,769 identities, of which 1,417 correspond to humans and 1,352 correspond to bots. The evaluation results are reported in Table IV. With this retrained model about 24.6% of bots (333 out of 1,352) are misclassified as humans (FN), and about 15.9% of humans (226 out of 1,417) are misclassified as bots (FP). Compared to the previous model, the model trained on commit messages therefore more accurately detects humans, while the converse can be observed for bots. We a value of 0.80, the precision of the retrained model is slightly higher than the previous one.

V. A BOT DETECTOR TOOL FOR GIT REPOSITORIES

In order to enable practitioners to use the classification model, we implemented it through BoDeGiC (Bot Detector for Git)³, a command-line tool to detect bots in given git repositories. The tool analyses the commit messages of each contributor in the specified git repositories and predicts whether the contributor is a bot or a human. BoDeGiC is implemented Python 3.7 and is easily installable through `pip`, the official package manager for Python.

³<https://github.com/mehdigolzadeh/BoDeGiC>

```
$ bodegic -h
bodegic [-h] [REPOSITORY [REPOSITORY ...]] [--include [NAME [NAME ...]]]
[--mapping [MAPPING]] [--start-date START_DATE] [--verbose]
[--min-commits MIN_COMMITS] [--max-commits MAX_COMMITS]
[--committer] [--only-predicted] [--text | --csv | --json]
```

Fig. 1. List of command-line arguments for BoDeGiC (version 0.2.0).

BoDeGiC works in three steps. The first step consists of extracting all commit information from the specified git repository using `git log`. This step results in a list of commits, authors and their corresponding commit messages. The second step consists of computing the features to feed the classification model. Features consists of the total number of messages, the number of empty messages, the number of message patterns and the inequality between the number of messages within patterns. In the third step, we apply the classifier that was trained on these features. The tool outputs, for each contributor, the prediction made by the model.

The command-line interface of BoDeGiC is summarized in Fig. 1. The output and the behaviour of BoDeGiC can be adapted by many optional command-line arguments in several different ways. By default, BoDeGiC relies on the author names in git commits, but the committer names can be used instead by specifying `--committer`. The list of names to consider can be explicitly specified with `--include`. BoDeGiC also supports identity merging (i.e., when a contributor uses multiple names) through the `--mapping` parameter. This parameter expects a path to a CSV file specifying how to map names to identities. This file can also be used to ignore specific names, by mapping them to the special “IGNORE” identity. Since the model (and by extension, the tool) requires at least 10 commits for a contributor to generate a prediction, contributors that have commits less than this number are predicted as “Unknown”. These cases can be excluded from the output by means of the `--only-predicted` parameter. Additionally, The minimum and maximum number of commits to consider can be changed respectively with `--min-commits` and `--max-commits`. By default, BoDeGiC outputs one line per author with the predicted class. The set of computed features can be included in this output by adding the `--verbose` parameter. Finally, the output of BoDeGiC can be exported in text (by default) or in JSON or as a CSV.

Fig. 2 presents the output of BoDeGiC on a randomly chosen git repository that was analysed on 2020-10-14. The first columns shows contributors name, the second column

```
$ bodegic --start-date 2020-10-14 --verbose --committer
committer      messages patterns dispersion prediction
Travis CI[bot] 20             1             0.026         Bot
greenkeeper[bot] 10            3             0.141         Bot
greenkeeper     69            58            0.040         Human
snyk-bot        5             NaN           NaN           Unknown
```

Fig. 2. Example of running BoDeGiC (version 0.2.0).

TABLE III
PRECISION, RECALL AND $F1$ OF THE BEST-PERFORMING CLASSIFIERS PER CLASSIFIER FAMILY (IN DECREASING ORDER OF $F1$).

classifier family	bots		humans		overall		
	$P(B)$	$R(B)$	$P(H)$	$R(H)$	$P(B \cup H)$	$R(B \cup H)$	$F1(B \cup H)$
random forest (RF)	0.817	0.748	0.775	0.837	0.817	0.748	0.793
decision trees	0.845	0.698	0.750	0.876	0.845	0.698	0.787
support vector machine	0.798	0.735	0.762	0.819	0.798	0.735	0.777
logistic regression	0.807	0.720	0.755	0.832	0.807	0.720	0.776
k-nearest neighbours	0.831	0.653	0.722	0.872	0.831	0.653	0.761

TABLE IV
EVALUATION OF THE RETRAINED CLASSIFICATION MODEL.

	classified as bot	classified as human	P	R	F1
Bot	1,019 (TP)	226 (FP)	0.82	0.75	0.78
Human	333 (FN)	1,191 (TN)	0.78	0.84	0.81
weighted avg			0.80	0.80	0.80

shows the number of extracted commit messages, the third column the number of computed message patterns, and the fourth column the statistical dispersion of the number of comments per pattern as computed by the Gini inequality index. The last column provides the predicted class of each contributor.

VI. DISCUSSIONS AND THREATS

The main threats to validity and mitigation strategies mentioned in our prior work [14] also apply to the current study.

A distinct threat to construct validity stems from the ground-truth dataset that has been used to train and evaluate the models. The dataset was created by other researchers and we have assumed it was correctly built and validated. Any presence of mislabeled items in that dataset could negatively affect the results of applying our original classification model (i.e., that was trained on PR and issue comments) as well as the retrained model that was directly trained on this dataset.

In order to assess to what extent this threat holds, we selected a subset of contributors from the dataset and manually verified whether they are actually humans or bots. We randomly selected 25 contributors from each category of correctly and incorrectly classified contributors (i.e., from TP, TN, FP and FN). Each of these cases was independently evaluated and labeled by the three authors of this paper. At the end of this process, we compared the labelings and observed agreement on all 100 cases. We then compared our own labelings with the actual labels found in the dataset, and observed a disagreement for 19 out of 100 cases. Among these cases, 13 corresponded to bots and 6 to human contributors. The prediction made by our classification model for these 19 cases matched our own labeling, i.e., the model was able to correctly predict them.

While manually looking at some other contributors in the dataset, we encountered a few cases we could not agree on because they combine both bot-like and human-like behaviours. We already encountered such “mixed” contributors in our previous study [14] where we found that some contributors

were occasionally relying on tools or bots to automate part of their activity. The presence of such mixed cases in git commit messages reinforces our belief that a better definition of “what a bot is” is required, with a clearer boundary between humans and bots. We also believe, in view of these mixed cases, that it might be interesting to identify bots not at the level of a contributor but at the level of its activity. In other words, the question “Is this contributor a bot?” would become “Is this contributor activity produced by a bot?”.

VII. CONCLUSION

In prior work [14] we proposed a classification model to predict bots based on features extracted from the PR and issue comments of GitHub accounts. The motivation behind the model was that bots tend to carry out repetitive tasks, implying that they are more likely to use repetitive messages. In this paper, we evaluated to which extent this approach performs on git commit messages, another text-based activity for which we expect bots to exhibit a similar, repetitive, behaviour.

We first evaluated how the model we developed and trained on GitHub PR and issue comments performs on this new type of data. We then generalised the classification model by retraining it on a ground-truth dataset composed of commit messages for git 6,922 contributors. We found that the highest precision was achieved by a random forest classifier, improving the precision of the previous model by 3% to 0.80.

We implemented BoDeGiC, a command-line tool to enable practitioners to take advantage of this classification model. It analyzes the commit messages of a git repository and predicts whether the contributors are bot or human.

Although promising, the obtained results suggest that, to achieve a precision comparable to the one we obtained for GitHub PR and issue comments, it will be necessary to use other data and features in complement to git commit messages. This is reminiscent of the BIMAN approach [5] which, in addition to analysing commit messages (the BIM part of BIMAN) combines two other independent models based on the contributor names (BIN) and the files changed in a commit (BICA). Given that our model is able to accurately capture more bots (+8%) than BIM, it would be interesting to substitute BIM by our classification model in the BIMAN approach.

REFERENCES

- [1] L. Erlenhov, F. Gomes de Oliveira Neto, R. Scandariato, and P. Leitner, "Current and future bots in software development," in *International Workshop on Bots in Software Engineering*, May 2019, pp. 7–11.
- [2] 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*, 2017, pp. 84–94.
- [3] M. Wessel, B. M. De Souza, I. Steinmacher, I. S. Wiese, I. Polato, A. P. Chaves, and M. A. Gerosa, "The power of bots: Understanding bots in OSS projects," *The ACM on Human-Computer Interaction*, 2018.
- [4] M. Wessel, A. Serebrenik, I. Wiese, I. Steinmacher, and M. A. Gerosa, "Effects of Adopting Code Review Bots on Pull Requests to OSS Projects," 2020.
- [5] T. Dey, S. Mousavi, E. Ponce, T. Fry, B. Vasilescu, A. Filipova, and A. Mockus, "Detecting and characterizing bots that commit code," in *Int'l Conf. Mining Software Repositories*, 2020.
- [6] R. Romero and E. Parra, "Experiences building an answer bot for Gitter," *International Workshop on Bots in Software Engineering*, pp. 66–70, 2020.
- [7] J. Dominic, J. Houser, I. Steinmacher, C. Ritter, and P. Rodeghero, "Conversational bot for newcomers onboarding to open source projects," in *International Workshop on Bots in Software Engineering*. New York, NY, USA: ACM, 2020, pp. 46–50. [Online]. Available: <https://doi.org/10.1145/3387940.3391534>
- [8] C. Brown and C. Parnin, "Sorry to bother you again: Developer recommendation choice architectures for designing effective bots," 2020.
- [9] M. Wyrich and J. Bogner, "Towards an autonomous bot for automatic source code refactoring," *International Workshop on Bots in Software Engineering*, pp. 24–28, 2019.
- [10] C. Matthies, F. Dobrigkeit, and G. Hesse, "An additional set of (automated) eyes: Chatbots for agile retrospectives," *International Workshop on Bots in Software Engineering*, pp. 34–37, 2019.
- [11] M. Wessel, I. Steinmacher, I. Wiese, and M. A. Gerosa, "Should i stale or should I close?: An analysis of a bot that closes abandoned issues and pull requests," in *International Workshop on Bots in Software Engineering*. IEEE Press, 2019, pp. 38–42.
- [12] M. A. Storey and A. Zagalsky, "Disrupting developer productivity one bot at a time," in *ACM SIGSOFT Symp. Foundations of Software Engineering*, 2016, pp. 928–931.
- [13] M. Wessel and I. Steinmacher, "The inconvenient side of software bots on pull requests," 2020.
- [14] M. Golzadeh, A. Decan, D. Legay, and T. Mens, "A ground-truth dataset and classification model for detecting bots in GitHub issue and PR comments," *Journal on Systems and Software [Submitted for review]*, 2020. [Online]. Available: <http://arxiv.org/abs/2010.03303>
- [15] Y. Ma, C. Bogart, S. Amreen, R. Zaretski, and A. Mockus, "World of code: An infrastructure for mining the universe of open source VCS data," in *International Conference on Mining Software Repositories*. IEEE Press, 2019, pp. 143–154. [Online]. Available: <https://doi.org/10.1109/MSR.2019.00031>
- [16] M. Golzadeh, D. Legay, A. Decan, and T. Mens, "Bot or not? detecting bots in GitHub pull request activity based on comment similarity," *International Workshop on Bots in Software Engineering*, pp. 31–35, 2020.