*Intelligent and Converged Networks* ISSN 2708-6240
2023, 4(3): 181−197 DOI: 10.23919/ICN.2023.0006

# BDGOA: A bot detection approach for GitHub OAuth Apps

**Zhifang Liao, Xuechun Huang, Bolin Zhang, Jinsong Wu*, and Yu Cheng**

**Abstract:** As various software bots are widely used in open source software repositories, some drawbacks are coming to light, such as giving newcomers non-positive feedback and misleading empirical studies of software engineering researchers. Several techniques have been proposed by researchers to perform bot detection, but most of them are limited to identifying bots performing specific activities, let alone distinguishing between GitHub App and OAuth App. In this paper, we propose a bot detection technique for OAuth App, named BDGOA. 24 features are used in BDGOA, which can be divided into three dimensions: account information, account activity, and text similarity. To better explore the behavioral features, we define a fine-grained classification of behavioral events and introduce self-similarity to quantify the repeatability of behavioral sequence. We leverage five machine learning classifiers on the benchmark dataset to conduct bot detection, and finally choose random forest as the classifier, which achieves the highest F1-score of 95.83%. The experimental results comparing with the state-of-the-art approaches also demonstrate the superiority of BDGOA.

**Key words:** Github; DevBots; machine learning; text similarity

## 1 Introduction

The rapid development of social coding platforms, such as GitHub, has attracted a large number of excellent software developers. These developers participate in the collaborative development of open-source software through mechanisms, like issue, Pull Request (PR), comment, and review. While the collaborative mode increases the efficiency of development, it also significantly increases the workload of repository maintainers to communicate with contributors, review codes, merge PR, run test cases, etc.

• Zhifang Liao, Xuechun Huang, and Bolin Zhang are with the School of Computer Science and Engineering, Central South University, Changsha 410083, China. E-mail: zfliao@csu.edu.cn; snowona@foxmail.com; wolfbolin@foxmail.com.
• Jinsong Wu is with the School of Artificial Intelligence, Guilin University of Electronic Technology, Guilin 541004, China, and also with the Department of Electrical Engineering, University of Chile, Santiago 8320000, Chile. E-mail: jwu_soc@qq.com.
• Yu Cheng is with Hunan Glozeal Science and Technology Co., Ltd., Changsha 410083, China. E-mail: yucheng@tinoy.cn.
* To whom correspondence should be addressed.
Manuscript received: 2023-02-23; revised: 2023-03-08; accepted: 2023-04-01

To reduce the workload and focus on core development and maintenance tasks, repository maintainers usually seek some automated programs to perform some repetitive tasks for themselves. For example, developers have employed bots to update software dependencies[1], execute test cases[2, 3], perform code refactoring[4], help newcomers quickly integrate into the team[5, 6], and so on. Generally, we refer to these automated tools as DevBots, and various types of DevBots are expected to be powerful tools for solving problems in increasingly complex software development work.

While DevBots bring convenience to software maintainers, they can also bring unexpected negative effects. Researchers in software engineering often analyze developers, in order to understand their cultural activities[7−10], assess team size[11], and make estimates of developer productivity[12]. Their empirical studies rely on a large number of account information and historical behavioral records mined from software repositories. The built-in automated nature of bots makes them more productive, threatening the validity of some metrics and even causing analysis results to deviate significantly from the estimates. Dey et al.[13]

pointed out that the first step to reduce the impact of bots on software engineering researches is to identify those bots. There are indeed many popular and highly regarded developing bots existing in the GitHub community, such as Dependabot and Renovate, but not all of them are easily recognizable, well designed, and maintained. In an empirical study conducted by Wessel et al.[14], respondents reported that bots may scare newcomers with mechanistic comments, which causes some newcomers to close their PRs. For newcomers or inexperienced developers, it is more frustrating to receive a comment like "you let coverage go down" than "thanks for your contribution". Ambiguous perception of the identity of bot accounts and their working logic reinforces the fear of the unknown, reduces developers' enthusiasm for code contribution. From both the perspective of researchers and developers, effective recognition of developing bots is a necessary task.

There are two types of bot accounts on GitHub: GitHub Apps and OAuth Apps. In earlier days, developers used OAuth Apps for assistance in software development. OAuth Apps can be authorized and act as actual GitHub users, and perform most development activities with API. As the demand for automation grows among developers, GitHub Apps and GitHub actions are officially launched on GitHub for a higher level of automation. GitHub Apps can be installed by an organization owner or individual repository administrator, and be given access to specific content, such as read permission to repository content, access to source code, management of issues, PRs, and tags, etc. GitHub Apps use their own identities when performing activities, and it is very convenient to use GitHub API to identify GitHub Apps (https://api.github.com/apps). Considering that OAuth App can perform most activities with the identity of human account, its bot identity is more invisible. Therefore, we focus on the identification of OAuth Apps.

Several methods have been proposed to identify bot accounts on GitHub[13, 15−18], which apply supervised learning techniques to enable automated bot detection, and achieve high precision on labeled dataset. However, most of the methods are limited to identifying bots performing specific activities, let alone distinguish between two different bot accounts. Moreover, the utilization of behavioral data stays in simple numerical statistics, without in-depth mining of repetitive behavior patterns.

In this paper, we present an approach, BDGOA, to identify bot accounts on GitHub, especially for OAuth Apps. Through exploratory data analysis, we select 24 effective features, which can be divided into three dimensions: account information, account activity, and text similarity. In contrast to existing research, we define a fine-grained classification for behavioral events, and also introduce a self-similarity feature to quantify the repeatability of behavioral sequences to better extract behavioral features. Meanwhile, we expand the range of selected text contents and improve the calculation of text similarity by taking into account the content format differences of different types of texts. Finally, we train five machine learning classifiers on the benchmark dataset to conduct bot detection, and finally choose the random forest classifier, which achieves the highest F1-score of 95.83%. The experimental results comparing with other state-of-the-art methods also demonstrate the superiority of our approach.

The remainder of the paper is organized as follows. Section 2 lists related work of our study. In Section 3, we conduct an exploratory data analysis to extract effective features. We review the selected features according to three principles of generality, stability, and robustness, and present the overall architecture of BDGOA approach in Section 4. Section 5 presents the dataset and evaluation metrics. Section 6 presents the experimental results. We discuss our findings in Section 7 and conclude the whole paper in Section 8.

## 2 Related work

### 2.1 DevBots in GitHub

To reduce workload, maintainers of open-source project often use automated script tools to help them perform some predefined repetitive tasks. Such automated tools are called DevBots. Existing researches on DevBots focus on the practical

application of employing bots in software engineering and the influence of bots on activities in software repository.

Mirhosseini and Parnin[1] analyzed 7470 GitHub projects that use notification mechanisms, like automated PRs and project badges, to update dependencies, and found that the update frequencies of software dependency were increased by 1.6 times and 1.4 times, respectively. Erlenhov et al.[3] believed that the key to better test bots is to design more efficient test cases, and avoid misleading conclusions from the test results. A case study was conducted by them to identify challenges faced in test design, and guidelines are given for test design patterns. Urli et al.[2] introduced the repairator bot, which constantly monitors the failed tests and reproduced bugs, and runs the program repair tools for each reproduced bug. Repairator bot has studied 11 523 test failures for 1609 open-source projects, and generated patches for 15 projects. Dominic et al.[6] proposed a chatbot that can recommend open-source projects for newcomers, and assist them to participate in the community with resources, like human mentors and Stack Overflow (S.O.). related information. Similarly, Alves et al.[5] proposed a chatbot which can filter tasks to help newcomers choose tasks that match their skills. Wessl et al.[19] randomly selected 351 popular projects on GitHub and found that 26% of these projects employ software bots, in which bots perform lots of tasks. Before and after bots were used, statistical differences were found in comment, commit, file changes, and close time of PR. Moreover, Wessel et al.[20] observed that the introduction of code review bots can increase the number of merged PRs per month, but reduce the communication among developers to some degree. Phaithoon et al.[21] proposed a FixMe bot that helps developers detect and monitor on-hold SATD in their repositories. FixMe bot monitors SATD comments as well as referenced issues, and developers will receive notifications when issues are resolved. Similarly, Mohayeji et al.[22] found that the TODO Bot does have an encouraging effect on developers to introduce TODO comments, and also increase the visibility of TODO comments. Romero et al.[23] proposed an answering bot, named GitterAns, which can automatically detect technical questions asked by developers in Gitter, and provide answers by utilizing contents from the S.O. website. To realize automated code refactoring, Wyrich and Bogner[4] developed Refactoring-Bot, which automatically eliminates odors in java code and commits PRs containing source code changes to developers.

### 2.2 Bots detection techniques in GitHub

In most research works which investigate the influence of introducing DevBots in software development, researchers adopted manual inspection to identify bot accounts, which lacks an automated solution for identification. Due to the limitations of manual labeling, the representativeness and validity of the empirical study results that have not been conducted on a large-scale bot accounts are not guaranteed. Considering that the identification of bot accounts is the basis for empirical studies, researchers have begun to seek solutions.

Dey et al.[13] first proposed a method to automaticly detect commit code bots, BIMAN. They constructed an dataset containing 461 bots and 13 762 430 commits. Features in three dimensions: account login, commit messages, and changed files associated with commit are selected. Following this, a series of researches were conducted by Golzadeh et al.[15], who hypothesized that the PR comments[24] submitted by bots contain repetitive text patterns. Based on 20 090 PR comments collected, they combined the Levenshtein edit distance with the Jaccard distance for cluster analysis, and found that bots have fewer patterns in their comments than humans. To validate on a larger and more representative dataset, Golzadeh et al.[15] constructed a ground-truth dataset containing PR and issue comments from 5000 GitHub accounts[15], of which 527 are labeled bots. Considering the existence of mixed accounts with two identities, Golzadeh et al.[16] proposed to identify human and bot at a more fine-grained level of comments. They proposed a method named BoDeGHa, which uses bag-of-words, as well as Term Frequency-Inverse Document Frequency (TF-IDF) techniques, to encode comments as vectors, and

train a binary classifier to predict the type of comments. Similarly, Golzadeh et al.[17] applied the same model to Git commit messages and demonstrated its excellent generalization performance.

Although the above methods of identifying bots in GitHub have achieved promising results, they are restricted to bots performing specific activity, such as PR, issue, and commit. Abdellatif et al.[18] proposed BotHunter, a method to detect bots from multiple activity types. They extracted 19 features from three dimensions: user profile, account activity, and text similarity, and merged the datasets constructed by Dey et al.[13] and Golzadeh et al.[15] A random forest classifier was used on a dataset containing 5107 GitHub accounts, and achieved F1-score of 92.4% and Area Under Curve (AUC) of 98.7%. Chidambaram et al.[25] noted that the BoDeGHa model utilized only activities in current repositories, which might lead to lead to inconsistent and incomplete prediction. They proposed method named Woc-P, which relied on the BoDeGHa for prediction and introduced the principle of group wisdom to correct, improving the accuracy by 6.9%.

## 3 Exploratory data analysis

In order to comprehensively characterize an account on GitHub, we have to extract a large number of features from multiple perspectives, such as account profile information and account activity. Based on the benchmark dataset provided by Golzadeh et al.[15], and account information and behavioral event records provided by GitHub archive, we launch an exploratory data analysis to investigate personal information and historical activity records for each account, thus mining features that can effectively distinguish humans and bots. Considering the time-sensitive nature of account information, we correct the public account set. Within the modified account set, 4462 are marked as human, and 512 are bot accounts. We select a total of 24 distinguishing features based on data analysis, which are divided into 3 dimensions: account information, account activity, and text similarity.

### 3.1 Account information

Personal homepage on GitHub provides basic account information, including (1) profile information, like name, bio, and school; (2) social information, like number of followers and followings, and affiliated organizations; and (3) personal contribution, such as the number of repositories, commits, and opened issues. We hypothesize that these features would distinguish well between humans and bots, and carry out statistical analysis on them to confirm.

**(1) Login, name, and bio**

Login is the primary identifier of a GitHub account. Users can also set an account name as alias. Bio is a short description which can briefly introduce user interests and skills. Figure 1 shows the profile of a bot account with login “renovate-bot”, whose identity is not explicitly declared in name but with clear statements in bio. We also find the “bot” string existing in human’s bio but it is rare to see. Dey et al.[13] first proposed the BIN model, which uses regular expressions to identify bots based on whether their name contains the string “bot”. Abdellatif et al.[18] found the presence of “bot” string in login, name, and bio accounts for 48%, 40%, and 35% of in their inspected accounts, respectively. We also analyze the performance of the above features in our account set and find that bot accounts with “bot” string in login, name, and bio account for 74.3%, 46.4%, and 18.2% of all bot accounts, respectively, while the corresponding percentage for humans are 2.6%, 1.1%, and 0.3%,

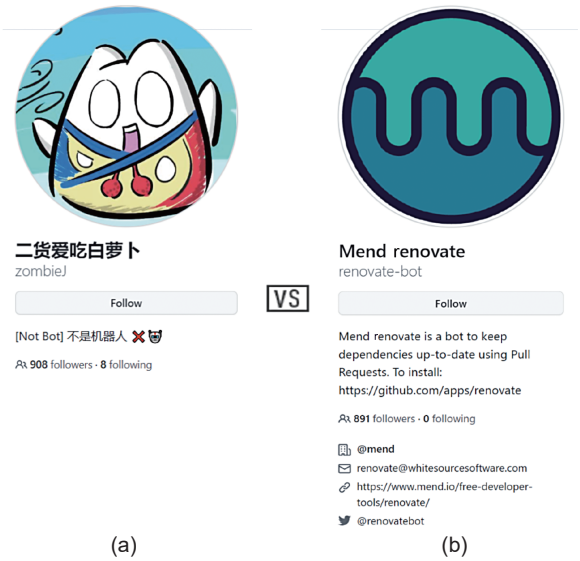



**Fig. 1 Examples of profile information with “bot” string.**

respectively. It can be seen that containing "bot" string in login, name, and bio is common for bot accounts, while humans rarely have such habits.

**(2) Information integrity**

There is a lot of optional profile information on user's homepage like email, blog, and website. Information integrity stands for the degree of completeness of profile information filled by users. We hypothesize that the information integrity on personal homepages could effectively distinguish between human and bot, and investigate the integrity of humans and bots on 7 pieces of information: name, bio, email, status, company, location, and website. Statistics show that the mean value of information integrity of bot accounts is 0.63%, which is significantly higher than that of human at 0.42%. Actually, we find many bot accounts, such as renovate, greenkeeper, and depfu, are widely used in the community. Their companies and teams design websites to introduce their products, provide user manuals, maintain twitter accounts to expand their social influence, and provide communications by phone or email to give after-sales support. The tool and product nature of bots require a higher level of information integrity to introduce, promote, and serve.

**(3) Number of followings and followers**

The number of followings and followers are considered as significant social information for GitHub users. Users can follow accounts of their interests and receive notifications of activity events from their followings. The number of followers is a good reflection of account's popularity and activity level. Abdellatif et al.[18] proved in statistics that humans are more likely to perform social activities. Among the accounts we inspected, we find the median number of human followings is 8, while bots have an average of only 1. What is more, the median number of human followers is 55, while that of bots is only 2. We think the number of followings and followers are critical social features that provide good distinction between humans and bots.

**(4) Level of personal contribution**

We choose several features to measure the individual contribution of GitHub accounts: (1) Total issue contribution—how many issues the user opened; (2) Total PR contribution—how many pull requests the user opened; (3) Total commit contribution—how many commits are made by the user; (4) Total repository contribution—how many repositories the user created. We calculate the value of above four features for each inspected account, and draw a box plot in Fig. 2. Due to large variation in value, we take logarithms of the total number. In terms of median, features of four dimensions are distinguishable.

### 3.2 Account activity

We also try to explore valuable features from historical behavior records. More specifically, we focus on the type, number, and time intervals of activities performed by GitHub users over a period of time. We hypothesize the above information can reflect user's role as part of the team, as well as time and effort they invested.

To obtain behavioral data of the inspected GitHub accounts, we download the data from the GitHub archive in 2020 and filter all the behavioral event records for each user in corrected account set by login. Furthermore, we need a taxonomy of GitHub activities to better extract behavioral features. GitHub event API returns detailed information of events triggered by account activities, and the corresponding API documentation provides basic classification for events, as Level 2 listed in Table 1. To obtain a more fine-grained classification of behavioral events, we further parse the "payload" object of each event record to subdivide the event type. Finally, We divide all event types into five dimensions: user, repository, commit, issue, and PR, with 17 types in Level 2, and 38 types in Level 3, as shown in Table 1.

Based on the fine-grained behavioral event

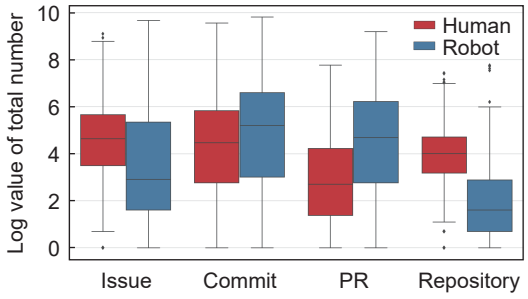



**Fig. 2 Distribution of statistical features of personal contribution.**

**Table 1 Fine-grained classification of behavioral events.**

| Dimension | Classification level 2 | Classification level 3 |
|---|---|---|
| User | ForkEvent | USER-fork_repo |
| | WatchEvent | USER-star_REPO |
| Commit | PushEvent | COMMIT-create_commit |
| | CommitCommentEvent | COMMIT-create_comment |
| Repository | CreateEvent | REPO-create_repo |
| | | REPO-create_tag |
| | | REPO-create_branch |
| | DeleteEvent | REPO-delete_repo |
| | | REPO-delete_tag |
| | | REPO-delete_branch |
| | PublicEvent | REPO-public_repo |
| | ReleaseEvent | REPO-create_release |
| | MemberEvent | REPO-create_member |
| | GollumEvent | REPO-update_wiki |
| Issue | IssuesEvent | ISSUE-create_issue |
| | | ISSUE-update_issue |
| | | ISSUE-delete_issue |
| | | ISSUE-reopen_issue |
| | | ISSUE-assign_issue |
| | | ISSUE-cancel_issue |
| | | ISSUE-mark_issue |
| | | ISSUE-unmark_issue |
| | IssueCommentEvent | ISSUE-create_comment |
| | | ISSUE-edit_comment |
| | | ISSUE-delete_comment |
| PR | PREvent | PR-create_pull |
| | | PR-update_pull |
| | | PR-delete_pull |
| | | PR-reopen_pull |
| | | PR-assign_pull |
| | | PR-cancel_pull |
| | | PR-mark_pull |
| | | PR-unmark_pull |
| | | PR-sync_pull |
| | | PR-ask_for_review |
| | | PR-give_up_review |
| | PRReviewEvent | PR-create_review |
| | PRReviewCommentEvent | PR-create_comment |

classification, we extract and sort each user's behavioral events by timestamp. and form a sequence of behavioral events for the whole year.

**(1) Total number of activities**

Abdellatif et al.[18] considered bots and humans have different activity patterns and chose total number of four groups of activities as key feature. We agree with them and think this feature can directly reflect the overall work level of GitHub users. Unlike their work, we refer to the classification of behavioral events in Table 1 and count the total number of behavioral events in 5 dimensions: user-related, repository-related, commit-related, issue-related, and PR-related. For each account inspected, we count total number of five dimensions of behavioral events performed in 2020.

In Fig. 3, we compare the total number of events performed by humans and bots in above five dimensions. Since the value of total number of activities varies widely, we take logarithmic values for ease of presentation. It can be seen that the median value of activities performed by bots is higher than that of humans in four dimensions, but significantly lower than that of humans in the user dimension. We think that bots are always pre-defined to perform certain functional tasks, while user-related events, such as star and fork, are subjective behaviors of humans.

**(2) Type number of activities**

Bots have access to limit resources and can only perform several pre-defined activities, while humans are free in their activities. Therefore, we think the type number of behavioral events is also a good indicator to distinguish between humans and bots. We count the type number of activities performed by bots and humans in five dimensions, and draw a box plot. As shown in Fig. 4, the median of type number of behavioral events performed by human accounts is always higher than that of bot accounts, as well as the maximum.

**(3) Median number of activities per day**

Abdellatif et al.[18] found bots have a higher level of daily activity than humans and computed the median

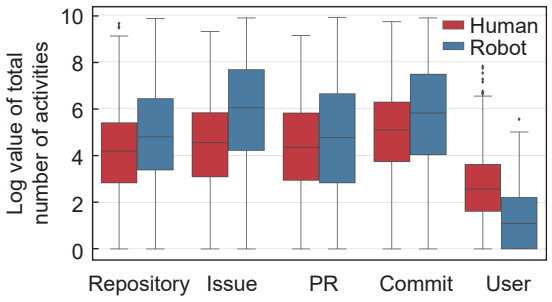



**Fig. 3 Distribution of total number of activities in 5 dimensions.**

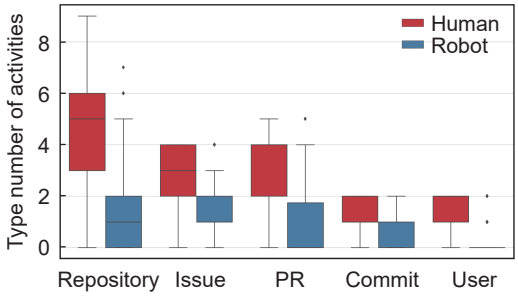



**Fig. 4 Distribution of type number of activities in 5 dimensions.**

number of activities per day for each account. Similarly, we count the number of behavioral events performed per day in 2020 for all the inspected accounts and draw the count-frequency scatter plot. As shown in Fig. 5a, the frequency of bots is significantly higher than that of humans at a high level of daily activity. Moreover, the number of behavioral events performed by bots per day in some cases even exceeds the maximum that humans can achieve.

**(4) Median response time per day**

Abdellatif et al.[18] observed that bots generally have faster response time than humans, and they explained bots are triggered by predefined events and continuously monitor the dynamics of repository. They compared accounts' median response time to PR & issue creation events. Here, we record the time intervals between two consecutive events in behavioral sequences of all the inspected accounts and draw the interval-frequency scatter plot as Fig. 5b. Under the same occurrence frequency condition, the time interval of bots is always smaller than that of humans.

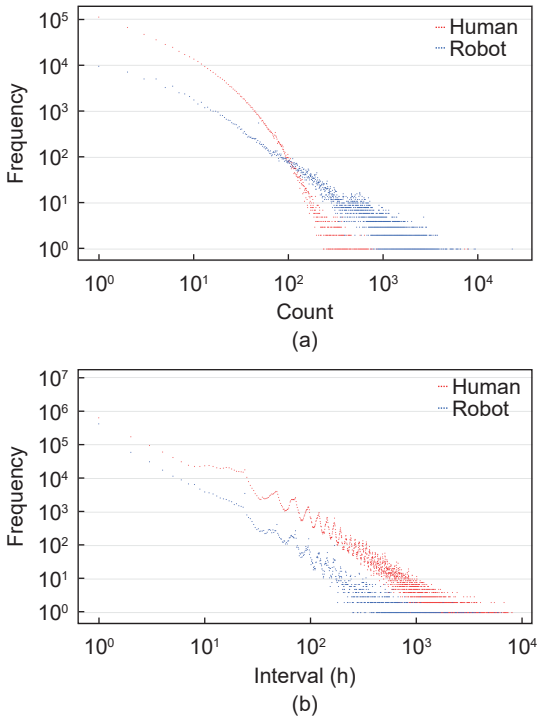



**Fig. 5 Relationship between features and their relative occurrence frequency.**

**(5) Self-similarity**

From the above analysis, it appears that bots with a limited type number of activities perform more activities than humans. Besides, bots are driven by computer programs and they are bound to perform lots of repetitive actions. To quantify the repetitiveness in behavioral event sequences, we try to resort to some sequence mining methods rather than simple numerical statistics.

We introduce the self-similarity feature proposed by Lee et al.[26] to measure the repetitiveness of account activity. Firstly, we group and sort each account's behavioral events by timestamp, then set a reasonable time period, followed by converting the behavioral events within the interval into a vector by count encoding. Then, we calculate the cosine similarity between each behavioral vector and the unit vector. After all behavioral vectors are converted into a set of cosine similarities, we calculate the standard deviation and use it as the value of self-similarity. Detailed calculation process is described in Section 4.1.1. The self-similarity value is close to 1 when the behavioral event sequence is highly repetitive; conversely, it will be close to 0.5.

To verify the validity of the self-similarity metrics, we calculate the self-similarity value of the inspected accounts following the above procedure and draw box plots for humans and bots. In terms of the median value, the self-similarity value of bots is obviously higher than those of humans. It can be seen in Fig. 6, some bot accounts achieve a self-similarity of 1, which indicates they perform single activity.

## 3.3 Text similarity

There exists lots of unstructured text on GitHub, such as user comments, titles of PR & issue, content of readme files, etc. Human comments are usually subjective and arbitrary. However, bots are usually

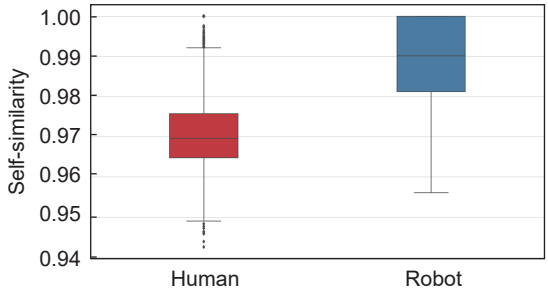



**Fig. 6 Comparison of self-similarity between humans and bots.**

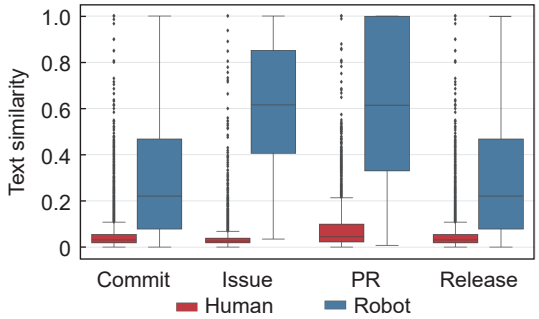



**Fig. 7 Comparison of text similarity between humans and bots of four types of text content.**

triggered to make comments after specified events (e.g., creation of PR & issue), to indicate the results or influence of events. The well-known renovate-bot is widely used to update outdated software dependencies and create PRs to provide patch. It will submit a comment under each PR created, stating the software dependency involved in the current update, and change in version number, e.g., "This PR contains the following updates: .....".

Previous research[15, 16, 24] assumed that comments published by bot accounts would show more similarity than humans, and used text distance measures, such as Jaccard distance[27] and Levenshtein distance[28], to quantify the similarity of comments in terms of content and structure. Text information studied in their research are mainly PR & issue comments and commit messages.

Our research extends the scope of text. We extract text information contained in eight types of behavioral events: commit-create_commit, commit-create_comment, issue-create_issue, issue-create_comment, pr-create_pull, pr-create_review, pr-create_comment, and repo-create_release. Supplementary text information includes contents of PR & issue, release notes, PR reviews, etc. Considering the unique text format of different categories, we divide the text information into four dimensions: issue-related; PR-related; commit-related, and release-related. We compute text similarity in four dimensions for all the inspected accounts and draw the box plot in Fig. 7. Detailed calculation process is described in Section 4.1.2. It can be seen that higher median value of text similarity is achieved by bots in all four dimensions. This is in line with our assumptions and consistent with findings of previous studies.

## 4 Bot account detection model

We have found some features that can effectively distinguish between humans and bots during exploratory data analysis. In this section, we describe some detailed extraction process of features and perform feature review based on three criteria to obtain the final feature set. Moreover, the overall architecture of our proposed approach BDGOA is also presented.

### 4.1 Feature extraction

We introduce the specific computation process for some features in this section.

#### 4.1.1 Self-similarity

Taking a GitHub account for example, whose self-similarity is calculated as follows.

Firstly, generate a set of behavioral vectors. We collect records of behavioral events for each user, sort them by timestamp, and form a behavioral sequence. We set the time interval to minute and split the behavioral sequence per minute. The behavioral events within one minute are transformed by count encoding into a vector, which consists of event id and its frequency.

Secondly, calculate the cosine similarity of behavioral vectors. We calculate the cosine similarity between each behavioral vector and a unit vector, and obtain a set of cosine similarities. The cosine similarity between two vectors is defined as follows:

$$\cos(A,B)=\frac{AB}{|A||B|}=\frac{\sum_{i=0}^{n}A_iB_i}{\sqrt{\sum_{i=1}^{n}A_i^2}\sqrt{\sum_{i=1}^{n}B_i^2}} \tag{1}$$

Thirdly, compute the self-similarity value $H$ of a GitHub account. We calculate the standard deviation of

this set of cosine similarities and convert them according to

$$H = 1 - \frac{1}{\sigma} \tag{2}$$

where $\sigma$ is the value of standard deviation.

#### 4.1.2 Text similarity

As mentioned in Section 3.3, we divide the text information from 8 behavioral events into 4 categories: issue-related, PR-related, commit-related, and release-related, according to the relevance in terms of content and structure. To give a example, we calculate the text similarity of commit comments and commit messages, and take the mean value of the two as final text similarity of commit-related text.

We choose the Jaccard distance as the text distance metric to measure the text similarity. Assuming that there are two comments $C_1$ and $C_2$, the Jaccard distance $J(C_1, C_2)$ is defined as follows:

$$J(C_1, C_2) = 1 - \frac{vocab(C_1) \cap vocab(C_2)}{vocab(C_1) \cup vocab(C_2)} \tag{3}$$

where $vocab(C_1)$ and $vocab(C_2)$ denote the set of words that appear in comment $C_1$ and $C_2$, respectively. It quantifies the content similarity of two comments at the word level. We collect all the commit comments for a given account, calculate the Jaccard distance between any two comments, and use the mean value as the text similarity of commit comments.

### 4.2 Feature review

In order to successfully apply these features to the bot detection task on GitHub, we review them based on the following considerations.

#### 4.2.1 Generality

We expect the selected features with the ability to distinguish between human and bot accounts, especially for OAuth Apps. Account tag was chosen as feature used in BotHunter[18], and the authors in Ref. [18] explained that they could examine whether an account is used as a GitHub App by API. It did achieve 100% identification of bot accounts of GitHub Apps type. However, it failed to distinguish between human accounts and OAuth Apps, which did not satisfy our demands. Besides, we removed three features: login, name, and bio. We considered the "bot" string in the profile information to be a strong hint of the bot identity. Just like the account tag feature, it is more like a label rather than a feature.

#### 4.2.2 Stability

The GitHub platform is in the process of expanding and evolving. New and existing accounts will have unavoidable differences in their individual contribution features due to the length of time they have been registered. In order to minimize the impact, some time-insensitive behavioral features are also introduced. Compared with account information and text contents, we think behavior records are the most impossible to miss, and are intuitive and easy to obtain. Therefore, more attention should be paid to extracting behavior features.

#### 4.2.3 Robustness

Utilization of behavioral data in previous research is limited to simple numerical statistics. Although these features have been proved distinguishable, they still remain staying in the overall level. The self-similarity value we proposed focus on the frequency of the repeated activity patterns rather than the frequency of the activities themselves. Furthermore, it can better reflect the repetitiveness of behavioral sequences and reveal the behavior patterns of accounts. It can be said that, the introduction of the self-similarity value enhances the robustness of our bot detection model.

Finally, we list all the 24 features used in our model in Table 2.

### 4.3 Model structure

The overall architecture of BDGOA is presented in Fig. 8. The corrected account set, as well as the account information and behavioral records extracted from GitHub archive are our data source. We divide account information into 3 categories, namely social information, profile information, and personal contribution. Behavioral records are parsed to get text contents contained in behavioral events, and behavioral events are sorted into behavioral sequences by timestamp. Some numerical statistics or sequence mining approaches are adopted by us to perform feature extraction.

All extracted features are reviewed to get the final

**Table 2 Feature set of the bot detection model.**

| Dimension | Feature |
|---|---|
| Account profile | Profile information integrity |
| | Number of followers |
| | Number of followings |
| | Total issue contributions |
| | Total commit contributions |
| | Total PR contributions |
| | Total repository contributions |
| Account behaviour | Total number of user activities |
| | Total number of repository activities |
| | Total number of issue activities |
| | Total number of PR activities |
| | Total number of commit activities |
| | Type number of user activities |
| | Type number of repository activities |
| | Type number of issue activities |
| | Type number of PR activities |
| | Type number of commit activities |
| | Median activities per day |
| | Median response time per day |
| | Self-similarity |
| Text similarity | Issue-related text similarity |
| | Commit-related text similarity |
| | PR-related text similarity |
| | Release-related text similarity |

valid feature set. We take these features and choose some mainstream machine learning classifiers for account classification, and evaluate the classification performance on the corrected account set.

## 5 Experiment

### 5.1 Dataset

To evaluate performance of our proposed method on bot identification task, we fetch the ground-truth dataset (gTruth) released by Golzadeh et al.[15], who first obtained links to over 3.3 million GitHub repositories associated with 37 software package registries provided by library.io. They randomly selected 136 000 repositories as the basis for building the dataset. By filtering out accounts with fewer than 10 comments, 79 342 GitHub accounts with 6 307 489 comments distributed across 42 492 software repositories are available. They randomly sampled a subset containing 5082 accounts and manually labeled them according to PR and Issue comments. Finally, a dataset containing 5000 GitHub accounts is constructed, with 527 bot accounts and 4473 human accounts.

Considering the behavioral data of these 5000 accounts needed to be utilized for feature extraction, we download the data from GitHub archive in 2020, which records public activities of all accounts. By filtering on the base of the login name, we collect for each account all of its activity records in 2020. However, we find 136 accounts with empty activity data when collecting data, which indicates that gTruth is highly time-sensitive and required manual verification and correction.

There are 70 bot accounts and 66 human accounts

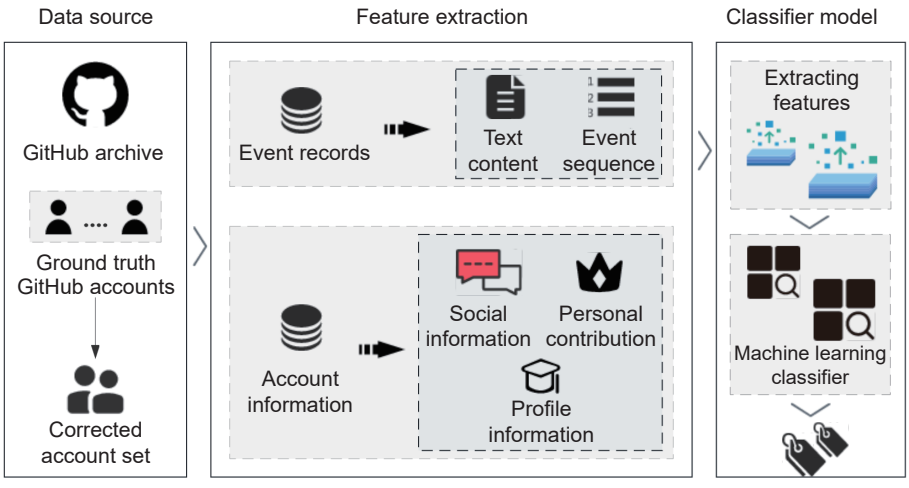



**Fig. 8 Overall architecture of BDGOA.**

among the 136 invalid accounts. We search in the GitHub community based on the login name and project information provided by gTruth. 54 human accounts have changed logins, and other reasons for invalid are distributed as follows: deleted, not existing, public or private GitHub Apps, etc.

We update the renamed accounts, remove deleted and non-existent accounts to get the corrected gTruth-2020 dataset. Detailed distribution of accounts is shown in Table 3.

### 5.2 Performance evaluation

Taking the 24 features of Table 2 as inputs, five mainstream machine learning classifiers: Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM) and XGBoost (XGB) are chosen to perform bot accounts detection. K-fold cross-validation is applied to evaluate the performance of our model. The gTruth-2020 dataset is divided into $k$ subsets of the same size. One of the $k$ sets is selected as test set, and the remaining $k-1$ sets are taken as training set, and the above process is repeated $k$ times. Average value of $k$-times model result is taken as the final performance of K-fold cross-validation, which can effectively avoid over-fitting and under-fitting.

Precision, recall, and F1-score are chosen as evaluation metrics. Precision is the probability of the number of accounts that are true bot relative to the number of accounts that are predicted to be bot. Recall is the probability of the number of accounts correctly predicted to be bot relative to the number of accounts that are true bot. F1-score is the harmonic mean of the two. Considering the imbalance of the dataset, i.e., the number of bot accounts is significantly smaller than the number of human accounts, we also plot the Receiver Operating Characteristic (ROC) curve. The horizontal coordinate of the ROC curve is the False Positive Rate (FPR) and the vertical coordinate is the True Positive Rate (TPR). We can get a point (FPR, TPR) based on the performance of a classifier on the test set, and by continuously adjusting the classification threshold, we can get an ROC curve passing through the points (0, 0) and (1, 1).

**Table 3 Account distribution in gTruth-2020.**

| Account type | Count |
|---|---|
| Human | 4462 |
| Bot (GitHub Apps) | 47 |
| Bot (OAuth Apps) | 465 |

## 6 Result

In this section, we present the result of our research. The motivation and results for each question and objective analysis are described in detail.

**(1) RQ1: Which classifier achieves the best performance on bot detection?**

The motivation for RQ1 is to choose the one that performs best on our task among mainstream machine learning classifiers. A small portion of accounts with missing features are deleted in the process of feature extraction, and there are 4520 valid GitHub accounts left in our dataset. We leverage all 24 features defined in Table 2 and conduct the experiment on 4520 GitHub accounts. During the experiment, default parameters setting on top of scikit-learn[29] are used to implement the model. What is more, we use 10-fold cross-validation to test the accuracy of the model and take the mean value of 10 experimental results to get the final model performance.

We list the precision, recall, F1-score, accuracy, and the AUC value of different machine learning classifiers in our experiment. Among these five evaluation metrics, we pay more attention to F1-score. The experimental results are shown in Table 4. Overall, the above classifiers perform very well, achieving the average F1-score of more than 0.8 and the average AUC value value of more than 0.9. Compared with other machine learning classifiers, the RF and XGB model achieve the highest F1-score. DT shows the worst performance of all five metrics.

We have performed a detailed review of test results. In the experiments using Random Forest and XGBoost,

**Table 4 Performance of machine learning classifiers. (%)**

| Classifier | Precision | Recall | F1-score | AUC | Accuracy |
|---|---|---|---|---|---|
| LR | 87.69 | 80.28 | 83.82 | 99.05 | 97.57 |
| SVM | 85.92 | 85.92 | 85.92 | 99.21 | 97.79 |
| DT | 45.19 | 85.92 | 59.22 | 88.52 | 90.71 |
| RF | 94.52 | 97.18 | 95.83 | 99.89 | 99.34 |
| XGB | 95.71 | 94.37 | 95.04 | 99.83 | 99.23 |

for 904 GitHub accounts included in the test set, we find that most accounts are correctly classified. The experimental result is shown in Table 5. Only 3 human accounts (4 in XGBoost) are incorrectly classified as bots, and only 2 bot accounts are incorrectly classified as humans. After manually analyzing all wrong classified GitHub accounts, we find that all wrong classifications are due to low frequency of account activity, resulting in meaningless statistics. There is also a case where the bot is wrongly classified because the user mainly performs commit activities and has low text similarity. After observation, we find that this bot account automatically pushes commits from other code hosting platforms to GitHub, and its main work does not center around GitHub. The text content is still written manually.

**(2) RQ2: Which features are of high importance score?**

There are many ways to compute the feature importance for the Random Forest classifier, such as the built-in feature importance and permutation-based importance[30].

The importance score of feature is a built-in attribute of random forest, we can get the relative importance score of each feature by accessing this attribute. The feature importance score is obtained by computing the Gini coefficient, which can reflect the purity of the segmented data. The permutation_importance is a function provided by scikit-learn. The principle behind permutation-based importance is more intuitive. A column of features will be randomly sorted at a time to compare the change in model performance before and after the feature change, and the feature that has the greatest impact on model performance is the most important feature.

We use the permutation_importance function and the built-in feature_importances_ attribute to compute the feature importance. Table 6 presents the top-10 features and their importance scores in the two evaluation metrics. It also provides us with a better understanding of which features we should focus on.

As we can see from Table 6, the feature "issue-related text similarity" ranks highest in both lists, and the "profile information integrity", "self-similarity", and "number of followings" features are in the second to fourth places, with a small gap with the first place. It proves part of our assumptions and analytical results in Section 3. Content similarity of text contents, repetitiveness of behavioral sequences, completeness of profile information, and level of socialization are the key indicators to distinguish bots from humans.

In order to demonstrate the effectiveness of all dimensional features and also to prove that using features of all dimensions is more effective than using features of a single dimension, we divide all features into four groups: profile information, text similarity, account activity, and self-similarity. We test the classification performance of each group of features under five classifiers separately, and the experimental

**Table 5 Confusion matrix of RF and XGBoost.**

| Class | RF | | XGBoost | |
|---|---|---|---|---|
| | Human | Bot | Human | Bot |
| Human | TPS: 830 | FNS: 4 | TPS: 829 | FNS: 4 |
| Bot | FPS: 3 | TNS: 67 | FPS: 2 | TNS: 69 |

**Table 6 Importance scores for the top-10 features.**

| feature_importances_ | | permutation_importance | |
|---|---|---|---|
| Feature | Score | Feature | Score |
| Issue-related text similarity | 0.1775 | Issue-related text similarity | 0.0158 |
| Profile information integrity | 0.1608 | Profile information integrity | 0.0060 |
| Self-similarity | 0.0967 | Number of followings | 0.0034 |
| Number of followings | 0.0618 | Self-similarity | 0.0019 |
| Total number of issue activities | 0.0600 | Number of followers | 0.0016 |
| Number of followers | 0.0486 | Median activities per day | 0.0009 |
| Median response time per day | 0.0448 | PR-related text similarity | 0.0009 |
| Total number of commit activities | 0.0419 | Total number of issue activities | 0.0007 |
| Type number of commit activities | 0.0346 | Type number of PR activities | 0.0006 |
| Median activities per day | 0.0327 | Type number of commit activities | 0.0005 |

results are shown in Fig. 9. As can be seen from the ROC curves and AUC values, the group of account activity performs the best, but the performance of each classifier varies significantly. In contrast, the group of self-similarity has the worst performance, but there is little difference between different classifiers.

The experimental results show that the classification performance using single dimensional feature is not superior to the performance of the classifier combining all features. Therefore, the BDGOA model selects all the above features as the basis for classification, which takes into account the feature validity and stability, and thus achieves better classification performance.

**(3) RQ3: How effective is BDGOA compared with the state-of-the-art techniques?**

In RQ1 and RQ2, we evaluate the performance of the proposed method on benchmark datasets and finally choose RF as the classifier, which achieves an F1-score of 95.83% in detecting bots. To demonstrate the performance advantage of the proposed BDGOA method, we compare it with other bot detection techniques in GitHub. Considering that the GitHub API can accurately identify the bot identity of GitHub Apps, we also take it as a technique for performance comparison. Table 7 presents the performance of our approach against other state-of-the-art methods.

**GitHub App API**※**:** Due to the iteration of GitHub features, we can now determine whether an account is a GitHub Apps directly from the results returned by the GitHub API. Since many OAuth Apps in the dataset have been converted to GitHub Apps, GitHub Apps account for a large percentage of bot accounts. GitHub API can accurately identify GitHub Apps and judge all non-GitHub Apps accounts as humans, which means

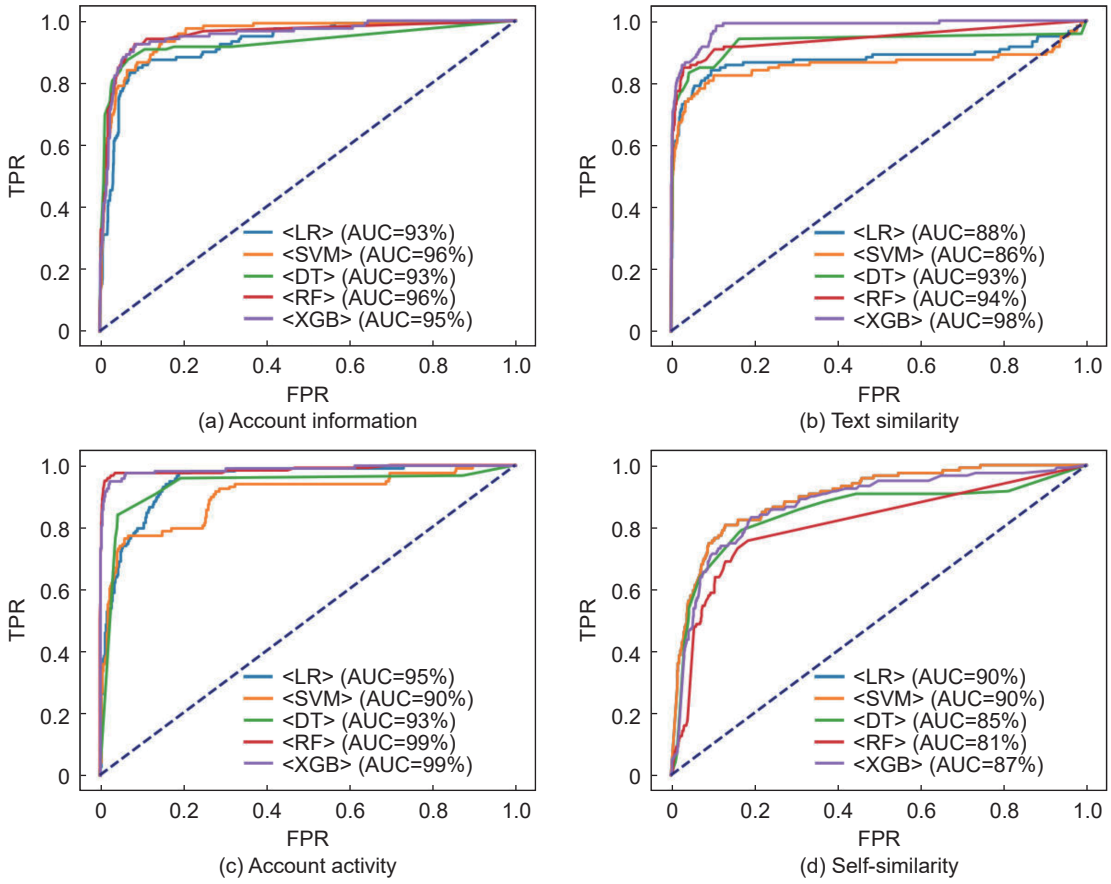



**Fig. 9 ROC curves of five classifiers. The dotted line connecting the points (0, 0) and (1, 1) is the reference line. The ROC curves are usually on the upper left of the reference line, and the farther away from the reference line, the better the classification result.**

※https://docs.github.com/en/rest/apps/apps#get-an-app

**Table 7 Precision, recall, F1-score, and AUC of bot detection techniques. (%)**

| Approach | Precision | Recall | F1-score | AUC |
|---|---|---|---|---|
| GitHub API | 90.0 | 99.9 | 94.7 | 53.1 |
| BIMAN | 83.6 | 99.4 | 90.8 | 93.5 |
| BoDeGHa | 81.6 | 95.2 | 89.3 | 89.5 |
| BotHunter | 92.8 | 84.6 | 90.5 | 81.0 |
| BDGOA | 96.5 | 94.1 | 95.3 | 99.6 |

only a small percentage of OAuth Apps will be wrong classified. It also explains its ability to achieve 90% precision, but with AUC value of 53.1%.

**BIMAN[13] and BoDeGHa[16]:** These two models measure the textual similarity of account comments and group them into repeated comment patterns. Due to the sparsity of text content, they do not perform well on the benchmark dataset.

**BotHunter[18]:** Similar to BDGOA, BotHunter extracts features in three dimensions: account information, account behavior, and text similarity. Due to the limitations of the GitHub API, which only extracts records of account activities within the last three months. We apply the open-source BotHunter tool to our constructed benchmark dataset, which covers account behavior data for the whole year of 2020, but find the classification performance of the model decreases with larger data volume. Meanwhile, we find that BotHunter relies heavily on the feature of whether the profile information contains the "bot" string. After we remove all GitHub Apps, we find that the probability of wrong classification is further increased. To further observe the impact of this feature, we use uniform random numbers to interfere with the feature value and perform the test 10 times. The mean value of F1-score and AUC value are 90.5% and 81.0%, respectively, which indicates that the stability of the classifier will be at risk when a small number of features occupy a large importance weight.

The bot detection model we proposed, BDGOA, will fully learn from the findings of the above study. BDGOA no longer identifies bot accounts of GitHub Apps after the GitHub feature upgrade, and such accounts are classified as self-admitted bot accounts whose existences are reasonable and do not interfere with the user's usage order. Inspired by BIMAN, BoDeGHa, and BotHunter, BDGOA no longer relies on text similarity or other single-dimension features. We extract features with universality, stability, and robustness from three dimensions: account profile, account activity, and text similarity to avoid misleading results from a single feature. Based on the more comprehensive features and high-performance classifier, BDGOA achieves better performance on the benchmark dataset.

## 7 Discussion

Taking into account the discussion of wrong classification in previous research, we have organized and enumerated all possible reasons of wrong classification.

### 7.1 Accounts that cannot be classified

**Lack of profile information**: Some users do not maintain their accounts after registration or fill in detailed profile information. Their main development activities are not based on GitHub, but only use GitHub as a tool to fork code or download code. They have no contribution to open-source projects and may not even be software developers.

**Low frequency of account activity**: Some users have no activity records in GitHub for a long time, and their development work may rely on other platforms (e.g., Gitee and Gitlab) or Git management tools in company. They contribute less to the open-source community, and most of their behaviors are mainly star, fork, and issue. Their low activity level results in sparse statistics and no valid information to provide judgment.

**Renamed or logged out**: Some users will modify their id or log out of their accounts, even if they were once a key participant in some project or left a record in the event stream. Some early OAuth Apps may have been converted to GitHub Apps and have modified their id and visibility. Due to the changes in id and permissions, the uniqueness and activity visibility of the account cannot be located by the program and therefore does not have classification value.

### 7.2 Accounts with classification difficulties

**Mixed accounts**: Users control OAuth Apps to

perform specific actions in the selected repository, such as deleting branches, closing issues, and merging code periodically. The repository is also maintained manually by the user, including code updates, participation in issue discussions, and review PRs. It is difficult to classify at the account level and requires a more fine-grained level of activity.

**Copy human activities:** There is some need for users to deploy programs related to cloning activities through OAuth Apps. Such programs copy code and tags from other repositories in real time or periodically in the current repository. We have observed three reasons for this phenomenon. One is that the user's development activities are not performed in GitHub, but the code needs to be open sourced through GitHub. Second, users add external references that do not belong to GitHub in a replicated manner, or to avoid accidental modification of external references. Third, users attempt to distribute content in a replicated manner that may be legally risky.

**Improvement of automatic dialogue capability**: With the development of the GPT model, bots already have the ability to participate in discussions in issue and PR discussions in issue and PR, even if these accounts are not mentioned. Therefore, traditional text similarity calculation methods based on text distance may fail.

## 8 Conclusion

In this paper, we propose an approach to identify GitHub OAuth Apps, named BDGOA. This approach introduces the latest bot operation mechanism of GitHub, removes GitHub Apps that are easily identified, and optimizes the design for GitHub OAuth applications. We extract features from three dimensions: account information, account activity, and text information, and finally filter 24 features according to three criteria: generality, stability, and robustness. We introduce the self-similarity feature to quantify the repetitiveness of behavioral sequence and improve the calculation of text similarity to enhance the perception of natural languages. We evaluate the performance of five mainstream machine learning classifiers and finally select the RF as the classifier according to the stability and accuracy principles. BDGOA has excellent classification performance on the benchmark dataset, with F1-score of 95.83%.

As the GitHub community is rapidly iterating and many new features are introduced, like actions, star list, and sponsors, they will continuously bring new challenges to our model. In future work, we plan to build a larger dataset of bot accounts based on the BDGOA tool. We hope to model account behavior with more sophisticated techniques, such as neural network models, to help developers and researchers better classify GitHub account. Meanwhile, we will conduct an empirical study of bot accounts to quantitatively observe its impact on existing research, as well as the GitHub community.

## References

[1] S. Mirhosseini and C. Parnin, Can automated pull requests encourage software developers to upgrade out-of-date dependencies? in *Proc. 32nd IEEE/ACM Int. Conf. on Automated Software Engineering*, Urbana, IL, USA, 2017, pp. 84–94.

[2] S. Urli, Z. Yu, L. Seinturier, and M. Monperrus, How to design a program repair bot? Insights from the repairnator project, in *Proc. IEEE/ACM 40th Int. Conf. on Software Engineering*: *Software Engineering in Practice Track*, Gothenburg, Sweden, 2018, pp. 95–104.

[3] L. Erlenhov, F. G. De O. Neto, M. Chukaleski, and S. Daknache, Challenges and guidelines on designing test cases for test bots, in *Proc. IEEE/ACM 42nd Int. Conf. on Software Engineering Workshops*, Seoul, Republic of Korea, 2020, pp. 41–45.

[4] M. Wyrich and J. Bogner, Towards an autonomous bot for automatic source code refactoring, in *Proc. IEEE/ACM 1st Int. Workshop on Bots in Software Engineering*, Montreal, Canada, 2019, pp. 24–28.

[5] L. P. S. Alves, I. S. Wiese, A. P. Chaves, and I. Steinmacher, How to find my task? Chatbot to assist newcomers in choosing tasks in OSS projects, in *Proc. 5th Int. Workshop on Chatbot Research and Design*, Virtual Event, 2022, pp. 90–107.

[6] J. Dominic, J. Houser, I. Steinmacher, C. Ritter, and P. Rodeghero, Conversational bot for newcomers onboarding to open source projects, in *Proc. IEEE/ACM 42nd Int. Conf. on Software Engineering Workshops*, Seoul, Republic of Korea, 2020, pp. 46–50.

[7] S. Amreen, B. Bichescu, R. Bradley, T. Dey, Y. Ma, A. Mockus, S. Mousavi, and R. Zaretzki, A methodology for measuring FLOSS ecosystems, in *Towards Engineering*

*Free/Libre Open Source Software* (*FLOSS*) *Ecosystems for Impact and Sustainability*, B. Fitzgerald, A. Mockus, M. Zhou, eds. Singapore: Springer, 2019, pp. 1–29.

[8] T. Dey, Y. Ma, and A. Mockus, Patterns of effort contribution and demand and user classification based on participation patterns in NPM ecosystem, in *Proc. Fifteenth Int. Conf. on Predictive Models and Data Analytics in Software Engineering*, Recife, Brazil, 2019, pp. 36–45.

[9] T. Dey and A. Mockus, Deriving a usage-independent software quality metric, *Empir. Software Eng.*, vol. 25, no. 2, pp. 1596–1641, 2020.

[10] T. Dey and A. Mockus, Which pull requests get accepted and why? A study of popular NPM packages, arXiv preprint arXiv: 2003.01153, 2020.

[11] T. Bhowmik, N. Niu, W. Wang, J. R. C. Cheng, L. Li, and X. Cao, Optimal group size for software change tasks: A social information foraging perspective, *IEEE Trans. Cybern.*, vol. 46, no. 8, pp. 1784–1795, 2016.

[12] M. Zhou and A. Mockus, Developer fluency: Achieving true mastery in software projects, in *Proc. Eighteenth ACM SIGSOFT Int. Symp. on Foundations of Software Engineering*, Santa Fe, NM, USA, 2010, pp. 137–146.

[13] T. Dey, S. Mousavi, E. Ponce, T. Fry, B. Vasilescu, A. Filippova, and A. Mockus, Detecting and characterizing bots that commit code, in *Proc. 17th Int. Conf. on Mining Software Repositories*, Seoul, Republic of Korea, 2020, pp. 137–146.

[14] M. Wessel, A. Serebrenik, I. Wiese, I. Steinmacher, and M. A. Gerosa, What to expect from code review bots on GitHub? A survey with OSS maintainers, in *Proc. XXXIV Brazilian Symp. on Software Engineering*, Natal, Brazil, 2020, pp. 457–462.

[15] 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, *J. Syst. Software*, vol. 175, p. 110911, 2021.

[16] M. Golzadeh, A. Decan, E. Constantinou, and T. Mens, Identifying bot activity in GitHub pull request and issue comments, in *Proc. 2021 IEEE/ACM Third Int. Workshop on Bots in Software Engineering*, Madrid, Spain, 2021, pp. 21–25.

[17] M. Golzadeh, A. Decan, and T. Mens, Evaluating a bot detection model on git commit messages, in *Proc. 19th Belgium-Netherlands Software Evolution Workshop*, Virtual Event, http://arxiv.org/abs/2013.11779, 2021.

[18] A. Abdellatif, M. Wessel, I. Steinmacher, M. A. Gerosa, and E. Shihab, BotHunter: An approach to detect software bots in GitHub, in *Proc. IEEE/ACM 19th Int. Conf. on Mining Software Repositories*, Pittsburgh, PA, USA, pp. 6–17, 2022.

[19] M. Wessel, B. M. De Souza, I. Steinmacher, I. S. Wiese, I. Polato, A. P. Chaves, and M. A. Gerosa, The power of bots: Characterizing and understanding bots in OSS projects, *Proc. ACM Human-Comput. Interact.*, vol. 2, no. CSCW, p. 182, 2018.

[20] M. Wessel, A. Serebrenik, I. Wiese, I. Steinmacher, and M. A. Gerosa, Effects of adopting code review bots on pull requests to OSS projects, in *Proc. 2020 IEEE Int. Conf. on Software Maintenance and Evolution*, Adelaide, Australia, 2020, pp. 1–11.

[21] S. Phaithoon, S. Wongnil, P. Pussawong, M. Choetkiertikul, C. Ragkhitwetsagul, T. Sunetnanta, R. Maipradit, H. Hata, and K. Matsumoto, FixMe: A GitHub bot for detecting and monitoring on-hold self-admitted technical debt, in *Proc. 36th IEEE/ACM Int. Conf. on Automated Software Engineering*, Melbourne, Australia, 2021, pp. 1257–1261.

[22] H. Mohayeji, F. Ebert, E. Arts, E. Constantinou, and A. Serebrenik, On the adoption of a TODO bot on GitHub: A preliminary study, in *Proc. IEEE/ACM 4th Int. Workshop on Bots in Software Engineering*, Pittsburgh, PA, USA, 2022, pp. 23–27.

[23] R. Romero, E. Parra, and S. Haiduc, Experiences building an answer bot for gitter, in *Proc. IEEE/ACM 42nd Int. Conf. on Software Engineering Workshops*, Seoul, Republic of Korea, 2020, pp. 66–70.

[24] M. Golzadeh, D. Legay, A. Decan, and T. Mens, Bot or not? Detecting bots in GitHub pull request activity based on comment similarity, in *Proc. IEEE/ACM 42nd Int. Conf. on Software Engineering Workshops*, Seoul, Republic of Korea, 2020, pp. 31–35.

[25] N. Chidambaram, A. Decan, and M. Golzadeh, Leveraging predictions from multiple repositories to improve bot detection, in *Proc. IEEE/ACM 4th Int. Workshop on Bots in Software Engineering*, Pittsburgh, PA, USA, 2022, pp. 6–9.

[26] E. Lee, J. Woo, H. Kim, A. Mohaisen, and H. K. Kim, You are a game bot! Uncovering game bots in MMORPGs via self-similarity in the wild, in *Proc. 23rd Annu. Network and Distributed System Security Symp.*, San Diego, CA, USA, 2016, pp. 1–15.

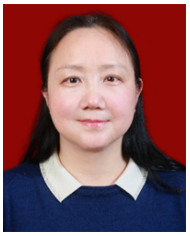
[27] P. Jaccard, The distribution of the flora in the alpine zone, *New Phytol.*, vol. 11, no. 2, pp. 37–50, 1912.

[28] V. I. Levenshtein, Binary codes capable of correcting deletions, insertions and reversals, *Sov. Phys. Dokl.*, vol. 10, pp. 707–710, 1966.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in Python, *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[30] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, Permutation importance: A corrected feature importance measure, *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, 2010.

**Jinsong Wu** received the PhD degree from Queens University, Canada in 2006. He is a professor at the School of Artificial Intelligence, Guilin University of Electronic Technology, China, and the Department of Electrical Engineering, University of Chile, Santiago, Chile. He received the 2020 IEEE Green Communications and Computing Technical Committee Distinguished Technical Achievement Recognition Award, for his outstanding technical leadership and achievement in green wireless communications and networking, and the 2017 IEEE Green Communications and Computing Technical Committee Excellent Services Award for his excellent technical leadership and services in the green communications and computing community. He is the proposer (2021), founder (2022) and founding editor-in-chief (2022–present) for the international journal Green Technologies and Sustainability (GTS), co-published by Elsevier and China Science Publishing & Media Ltd., jointly called KeAi. He was the leading editor and a co-author of the comprehensive book, entitled Green Communications: Theoretical Fundamentals, Algorithms, and Applications, published by CRC Press in September 2012, which has been recognized as the very first comprehensive published book effort on green communications covering broad topics of green wireless communications, green wireline communications, general relevant green topics and applications in one book using a research perspective. He won the 2017, 2019, and 2021 IEEE System Journal Best Paper Awards, and the 2018 IEEE Green Communications and Computing Technical Committee Best Magazine Paper Award. He was the founder (2011) and founding chair (2011–2017) of the Technical Committee on Green Communications and Computing (TCGCC). He is the current Chair (2022–present), the co-founder (2014), and the founding vice-chair (2014–2022) of IEEE Technical Committee on Big Data (TCBD), which was the very first technical organization on big data within the whole IEEE. He was elected as the vice chair, Technical Activities, IEEE Environmental Engineering Initiative (EEI) (2017–2022), a pan-IEEE effort under IEEE Technical Activities Board (TAB) across 25 IEEE societies, councils, and organization units (OU). He was the very first proposer of IEEE journals/transactions on green information and communications technologies (ICT) (2011–2012). He was the proposer (2012), founder (2014), and series editor (2014–2018) on green communication and computing networks in IEEE Communications Magazine. He was an area editor (2016–2020) of IEEE Transactions on Green Communications and Networking. He was series editor (2014–2016) of IEEE Journal of Selected Areas on Communications (JSAC) series on green communications and networking.



**Zhifang Liao** received the BEng degree in industry control engineering and the MEng degree in computer science both from the Changsha Railway Institute, China in 1990 and 1998, respectively, and the PhD degree in computer technology and application from Central South University, China in 2008. From 1990 to 1997, she was an engineer at Hunan Computer Factory. She is currently an associate professor at Central South University. Her research interests include open source software, open source software ecosystems, and data mining.

**Xuechun Huang** received the BEng degree in computer science and technology from China University of Mining and Technology, China in 2020. She is currently a master student in computer science and technology at Central South University. Her research interests include data mining, bot account detection, and analysis of user behavior in open source community.

**Bolin Zhang** received the BEng degree in software engineering from Central South University, China in 2020. He is currently a master student in software engineering at Central South University, China. His main research interests include open source software, open source ecosystem, and data mining.

**Yu Cheng** received the BEng degree from Hunan University, China in 2010. He is currently working at Hunan Glozeal Science and Technology Co., Ltd. as a research & development engineer. His research interests include open source software and knowledge graph.