A ground-truth dataset and classification model for detecting bots in GitHub issue and PR comments*

Mehdi Golzadeh[a,*,1], Alexandre Decan[a,2], Damien Legay[a,3] and Tom Mens[a,4]

[a] Software Engineering Lab, University of Mons, Avenue Maistriau 15, 7000 Mons, Belgium

ARTICLE INFO

Keywords:
distributed software development
bot identification
GitHub repositories
text similarity
classification model

ABSTRACT

Bots are frequently used in Github repositories to automate repetitive activities that are part of the distributed software development process. They communicate with human actors through comments. While detecting their presence is important for many reasons, no large and representative ground-truth dataset is available, nor are classification models to detect and validate bots on the basis of such a dataset. This paper proposes a ground-truth dataset, based on a manual analysis with high interrater agreement, of pull request and issue comments in 5,000 distinct Github accounts of which 527 have been identified as bots. Using this dataset we propose an automated classification model to detect bots, taking as main features the number of empty and non-empty comments of each account, the number of comment patterns, and the inequality between comments within comment patterns. We obtained a very high weighted average precision, recall and F1-score of 0.98 on a test set containing 40% of the data. We integrated the classification model into an open source command-line tool to allow practitioners to detect which accounts in a given Github repository actually correspond to bots.

1. Introduction

The collaborative nature of software development has inherently made it a social phenomenon, which has led to the advent of *social coding* platforms such as GitHub, BitBucket, and GitLab [14]. These online platforms have taken the collaborative nature of open source software development to a new level, by integrating mechanisms such as issue reporting, pull requests (PR), commenting and reviewing support into distributed version control tools [33, 67, 68]. The pull-based development process is the primary means for integrating code from thousands of developers in distributed development platforms such as GitHub [33]. This model has had a significant impact on the development of open-source software, but at the same time has significantly increased the workload of repository maintainers to communicate with other contributors, review source code, deal with contributor license agreement issues, explain project guidelines, run tests and build code, and merge pull requests [34].

To reduce this workload, developers have been adopting automated tools to perform repetitive tasks in the development process [71], such as updating dependencies [57] (e.g. *dependabot*) and fixing vulnerabilities (e.g. *snykbot*), improving code reviews (e.g. *Review bot*) [4] and documenting code refactorings [60]. Such tools are commonly known as *DevBots* [24], or *bots* for short. They are generally seen as a promising approach to deal with the ever-increasing complexity of contemporary distributed software development.

While the use of bots in open source software repositories can alleviate maintainer workload, their presence poses challenges for empirical software engineering researchers that aim to study socio-technical aspects of software development. For example, in a previous study we analysed the impact of discussions on pull request (PR) decisions in GitHub repositories [31] by studying these discussions in 188K PRs of GitHub repositories. We ignored the presence of bots in that study, deferring it to future work. Repeating the same analysis taking into account the bots allowed us to discover that 20% of those comments belong to bots, and that bots were involved in 31% of all PRs. Bots were responsible of accepting or rejecting 25% of all PRs. Moreover, we found that the proportion of successfully integrated PRs was twice as high for PRs involving bots.

Other empirical socio-technical analyses based on historical software repository data are likely to have been biased as well by not considering the presence of bots. Some empirical studies explicitly acknowledge the presence of bots, and attempt to remove them during data preprocessing (e.g. filtering out bots) or postprocessing (e.g. removing outliers) [16, 53]. It is therefore important to consider the presence of bots in such studies, and to treat them differently than humans.

A prerequisite for considering bots is the ability to identify their presence in software development activities. This is not a simple task because, depending on the considered data source, bots often do not have a distinct representation in social coding platforms, and may look, act like or even impersonate humans. Our review of the research literature (see Section 2) revealed a few attempts to manually identify and classify bots. We only came across one study attempting to automate the bot identification process based on commit activity in GitHub repositories [17]. The current paper has a similar focus, but based on a different data source, namely PR and issue comments in GitHub repositories.

* This work is supported by the Fonds de la Recherche Scientifique – FNRS [Grants number T.0017.18, O.0157.18F-RG43 and J.0151.20]

*Corresponding author
ORCID(s): 0000-0003-1041-439X (M. Golzadeh); 0000-0002-5824-5823 (A. Decan); 0000-0001-6811-6585 (D. Legay); 0000-0003-3636-5020 (T. Mens)
[1] mehdi.golzadeh@umons.ac.be
[2] alexandre.decan@umons.ac.be
[3] damien.legay@umons.ac.be
[4] tom.mens@umons.ac.be

As a first major contribution, we propose a large and reliable ground-truth dataset, consisting of 5,000 distinct GitHub accounts of which 527 were manually identified as bots based on their PR and issue commenting contents. As a second major contribution, we use this ground-truth to create and evaluate a classification model that relies on comment-related features to accurately classify accounts as either bot or human. As a third contribution, we propose an open-source tool based on the classification model to allow GitHub contributors to detect which accounts in their repositories actually correspond to bots.

The remainder of this paper is structured as follows: Section 2 presents the related work. Section 3 explains the steps to create the ground-truth dataset. Section 4 details which features we selected for the classification model. Section 5 explains the workflow to select an appropriate classification model and evaluates the selected classification model. Section 6 presents an open source tool implementing this model. Section 7 discusses the results. Section 8 presents the main threats to validity of the research. Section 9 outlines future research avenues and Section 10 concludes.

2. Related Work

The earliest idea of computer software imitating humans dates back to the ideas by Alan Turing in 1950 [69]. In recent years, the development of AI and machine learning has led to a proliferation of automated tools that substitute humans to perform particular repetitive tasks [15]. For example, chatbots imitate natural language to communicate with humans through a conversational interface [46], and automated social actors (ASA) automatically create content on social networks [1]. Bots are also widely used in other contexts such as education [44, 6, 28], e-commerce [5, 66], customer services [30, 41], peer production communities such as Wikipedia [13, 29], and social networks (such as Twitter). Social network bots are generally aimed at spamming and sending fake news and hence they seek to hide their true nature. Because of this, numerous studies are focused on identifying them [56, 20, 61, 3]. These studies have proposed machine learning models which aim to identify bots based on features such as profile specification (e.g., age, location and biography), tweet content (e.g., hashtags, URLs and similarity of sentiments), tweet time (e.g., burstness and average tweets per day), and user network profile (e.g., interaction between users). Such features either do not have any equivalent in social coding platforms (e.g., hashtags) or require a lot of effort to collect (e.g., user network profiles). Moreover, bots in social networks appear to be quite different from bots in social coding platforms (i.e., DevBots), whose purpose is to help developer teams carry out automated activities in the software development process. We did not find any evidence of intentionally malicious use of DevBots.

In the context of software development, bots are automated software agents that perform repetitive well-defined tasks that support and integrate with the activities of human developers [71, 26]. They are capable of communication and decision making [65] and carry out tasks that involve interactions with humans [48]. They support both technical and social activities [51] to coordinate collaborative software development [59], such as improving feedback on code contributions [39], repairing continuous integration build failures [70], and deployment and evaluation of software engineering analysis techniques [7].

Recent research has focused on the practical value of bot adoption in software engineering, such as how bots increase software development productivity [65], how bots enable faster software dependency updates [57] and how bots can help reduce the friction points software developers face when working collaboratively [47]. Other studies have introduced new bots and analysed their effect on software repository activities such as test bots [23], bots to improve newcomers' experience and help them to better engage in the project [18], bots for answering developer questions using historical Q&A data [63], bots for assisting in the development of microservice architecture and the use of NLP [52].

A prerequisite for studying the impact of bots on software production processes is the ability to identify such bots in the first place. We found very few studies trying to identify and categorise bots. Wessel et al. [71] conducted a study about prevalence and effect of bots in GitHub repositories. They manually analysed 351 repositories and found that 26% of them use bots. By manual inspection of GitHub accounts they identified 48 different bots in 93 projects. They found statistical differences regarding the number of commits, number of changed files, and closing time of PRs between projects before and after bot adoption. They reported both positive and negative challenges of bot adoption from integrators and contributors' viewpoints. In another study, they discuss six useful bots in GitHub's PR process [72]. They analysed the negative aspects of bots in code contributions and introduce a meta-bot that acts as a middleman to mitigate this effect.

Erlenhov et al. [22] presented a taxonomy that classifies 11 existing development-related bots in GitHub and Slack. Lebeuf [49] provided a multi-faceted classification of bots (including many well-known examples of bots), combining their properties and behaviour. None of these studies proposes an automated approach to identify bots.

Dey et al. [17] did propose an automatic method to identify bot accounts in git projects. Each identity in their dataset consists of an author name and email address. They studied three different approaches to find bots, based on (i) the presence of the string "bot" at the end of the author name, (ii) commit messages, and (iii) features related to files changed in commits and projects the commits are associated with. They combined these three different approaches into a single ensemble model that was validated on a dataset of 67 bots of which 58 cases (85%) were effectively captured by the model.

Their study is fundamentally different from ours, since their dataset is based on commit data in GitHub repositories, whereas we will focus exclusively on GitHub issue and PR comments. They also identified authors based on the author name and email address, whereas we rely on the GitHub account name exclusively. Both datasets are quite complementary, as we found many examples of bots that are only involved in commit activity and others that are only involved in issue and PR activities. Moreover, the nature and contents of commit comments is quite different from issue and PR comments, requiring other features to establish an accurate classification model.

3. Ground truth dataset

In order to be able to evaluate an automated algorithm to detect bots based on their commenting activity in GitHub issues and pull requests, a ground truth dataset is required. Such a ground truth dataset indicates, given a contributor commenting in an issue or a pull request, whether this contributor is a human or a bot. To be effective and representative, the ground truth dataset should be large enough, i.e., it should cover a considerable number of GitHub repositories, contributors, issues and pull requests.

Since we did not encounter any such representative ground truth dataset in the research literature, we set out to create it ourselves. To do so, we downloaded and manually examined comments from thousands of issues and pull requests, labelling each contributor either as a bot or a human commenter. Despite the considerable effort needed to create such a dataset, it was a worthwhile endeavour, since it will be a valuable resource for other researchers as well.

This section explains how we proceeded to create and validate our ground truth dataset, from the raw data we downloaded to the process of rating and labelling each contributor.

3.1. Terminology

In the context of this paper, we will consistently use the following terminology. We use the term ***bot*** to refer to a GitHub bot, defined by Wessel [71] as "*a task-oriented bot, responsible for automating well-defined tasks on GitHub repositories. A GitHub bot behaves like a human user, serving as an interface between users and services.*"

Since our study focuses on distributed software development on GitHub, we use the term ***repository*** to refer to a GitHub repository. Contributors to a repository can be identified by their unique (GitHub) ***account***. Contributions to a repository can take different forms, such as code ***commits***, ***issues*** and ***pull requests (PR)***. The focus of this paper will be on issues and PRs.

Contributors can add (uniquely identifiable) ***comments*** to PRs and issues in a repository. We use the term ***commenter*** to refer to the GitHub account having provided this comment. We also use the term comment to refer to its actual textual content. Since a commenter can be either a bot or a human contributor, we will refer to them as ***bot commenter*** and ***human commenter***, which we will abbreviate to ***bot*** and ***human***, respectively.

3.2. Data extraction

Our goal is to identify bot and human commenters based on the comments they made in issues and pull requests of collaborative software development repositories on GitHub. GitHub is one of the leading online collaborative development platform. As of November 2020, GitHub reported having over 48 million users and more than 195 million repositories (including at least 37 million public repositories).

Following the guidelines provided by Kalliamvakou et al. [42], we want to avoid repositories that have been created merely for experimental or personal reasons, or that only show sporadic traces of issue and PR comments. Moreover, since our focus is on software development repositories, we want to exclude repositories that are not related to software development. To comply with these constraints, we relied on libraries.io [43], a monitoring service indexing information for several million packages being distributed through 37 software package registries, such as npm, PyPI, etc.

We downloaded the data dump of January 2020[5] containing, among others, links to the GitHub repositories related to these distributed software packages. Since it contains more than 3.3 million GitHub repositories, we randomly selected around 136K of them as the starting point of our dataset creation process. For each of these repositories, we extracted on 16 February 2020 the last 100 comments of the last 100 issues and pull requests using GitHub's GraphQL API. This resulted in over 10 million comments covering a period of more than 10 years (ranging from 17 December 2009 to 15 February 2020). These comments were made by more than 837K distinct contributors, corresponding to more than 3.5 million issues and pull requests. The extracted comments also include the textual description of each considered PR. While the GitHub API does not consider PR descriptions as comments, we do, since the GitHub web interface does not visually distinguish them from other comments.

Since our goal is to distinguish between bots and human contributors based on their comments, we require a sufficiently large number of comments for each commenter. Hence, we decided to exclude commenters who made fewer than 10 comments based on a threshold we identified in a previous study [32]. At this stage of the process, the dataset contains 6,307,489 comments belonging to 79,342 contributors, spanning 42,492 repositories.

Since this is too much data to process manually, we extracted a subset covering 5,082 commenters. This subset was composed of 4,644 randomly selected commenters to which we manually added 438 extra commenters that are more likely to correspond to bots based on previous studies [32, 71] (52 cases), or because they contained a specific substring in their GitHub account name (386 cases). The substrings we considered were "bot", "ci", "cla", "auto", "logic", "code", "io" and "assist". By doing so, we increased the likelihood of having a sufficient number of bots in the dataset.

The resulting subset contains 5,082 commenters and covers 3,975 repositories, 186,991 issues and pull requests, and contains 301,557 comments. Table 1 summarizes the main characteristics of the considered datasets.

[5]Version 1.6 on `http://doi.org/10.5281/zenodo.3626071`

Table 1
Summary of the dataset characteristics.

raw dataset	**number**
GitHub repositories	136,529
↪ from # distinct owners	84,983
issues	1,588,363
pull requests (PR)	1,951,705
issue and PR comments	10,874,611
↪ from # distinct commenters	873,489
selected subset	
GitHub repositories	3,975
↪ from # distinct owners	3,425
issues	50,241
pull requests (PR)	136,750
issue and PR comments	301,557
↪ from # distinct commenters	5,082

3.3. Data labelling and rating process

The next step to create a ground truth dataset is to manually identify bots and humans. To ease this process, we developed a web application through which the list of comments of each commenter was presented to at least two raters (among the four authors of this paper). Comments were displayed by batches of 20, starting with the most recent comments first, and the rater had an option to display more comments if needed. The account name of the commenter was not revealed to avoid bias, as the goal was to classify commenters based on their comments only. The rater could select whether the commenter is considered as a "Bot" or a "Human". In case a rater was uncertain whether the commenter was a bot of a human being, a third option could be selected: "I don't know". Furthermore, the rater was asked to select a difficulty level among "Very easy", "Easy", "Difficult" and "Very difficult" for his decision.

Fig. 1 shows a screenshot of the rating application in action. For the specific example being shown, raters could easily decide that the commenter is a bot based on the content and repetitiveness of all visible comments.

In total, 5,082 commenters were rated, ending up with exactly 5,000 commenters after having filtered out 82 commenters during the following process. The rating process was performed in two steps to come with an optimal inter-rater agreement, relying on Landis agreement levels [45]. The rating process is summarized in Fig. 2. Each commenter was initially rated by two distinct raters. All cases that were agreed either as bot or human were included in the ground-truth dataset. In order to assess the reliability of the ground-truth dataset, we computed the inter-rater reliability (IRR) [11] between each pair of ratings based on Cohen's kappa κ [55]. The results are presented in Table 2.

The first step of the rating process ended up with 472 bots and 4,364 humans, with a "*substantial*" agreement ($\kappa = 0.84$) between raters. At the end of this step, there were 246 cases because they were either not agreed (177 cases) or agreed as "I don't know" (69 cases). Additionally, 91 cases

Table 2
Summary of two-step rating process.

	first step	**second step**
commenters agreed as *bot*	472	527
↪ from # repositories	457	505
commenters agreed as *human*	4,364	4,473
↪ from # repositories	3,425	3,515
proportion of bots	9.8%	10.5%
commenters agreed as "I don't know"	69	–
commenters without agreement	177	4
commenters agreed as "mixed"	–	78
κ agreement score	0.84	0.96

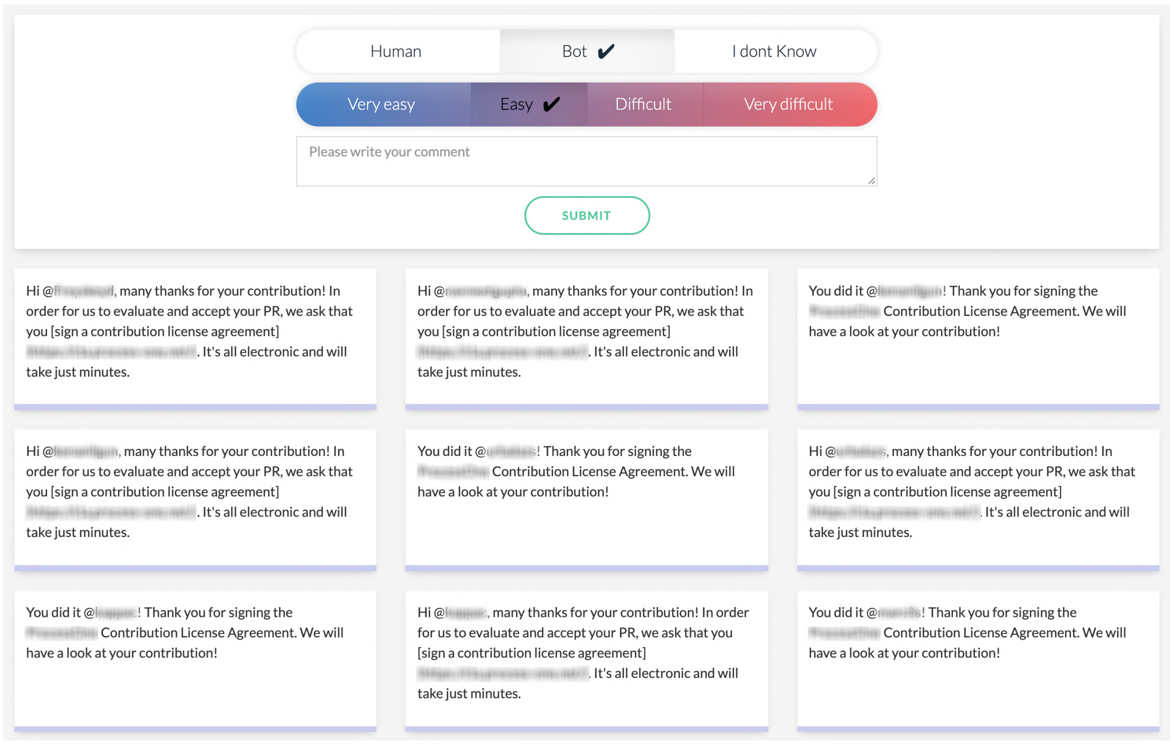
evaluated as "difficult" or "very difficult", leading to a total of 268 cases for the second step.

In a second step, we involved a third rater for the cases that were identified as "difficult" or "very difficult" during the first step. We then discussed all together all cases for which an agreement could not be achieved, or the cases where the third rater disagreed with one of the two former ones. During these discussions, we sometimes relied on additional information (e.g., we looked at the GitHub account of the commenter, at time intervals between comments, the overall activity of the account, etc.) to come to a decision.

The large majority of cases we discussed were resolved on the basis of an unanimous decision between raters, leading to an "*almost perfect*" inter-rater reliability ($\kappa = 0.96$). At the end of the second step, only 82 cases were left out of the ground-truth dataset, either because no agreement could be reached (4 cases), or because we agreed on the "mixed" nature of these commenters. These "mixed" commenters correspond to human commenters that relied on automatic tools to generate comments, therefore "mixing" the behaviour of a human and a bot at the same time.

For example, some of these accounts rely on an automated tool to facilitate code review by sending PRs to *Reviewer*, a code review tool for GitHub. Other examples include the use of tools such as *StyleCI* to improve code style, or *semantic-release* to automatically determine the next version number of a release, generate release notes and publish a package. We will discuss these "mixed" commenters in more details in Section 7.

This left us with 5,000 commenters, of which 527 (i.e., 10.5%) are bots. Table 3 summarizes the characteristics of final ground-truth dataset. Since we believe such a ground-truth dataset is valuable for the research community (e.g., to have a list of known bots, to study their characteristics or to train other models), we share it publicly on `http://doi.org/10.5281/zenodo.4000388`. This dataset contains the name of the repository, the name of the commenter and whether it is a bot or a human. Due to GDPR regulations and in order to protect GitHub users' privacy, we do not provide additional information (e.g., their comments).


Human | [✓ Bot] | I dont Know

Very easy | [✓ Easy] | Difficult | Very difficult

Please write your comment

SUBMIT

Hi @[redacted], many thanks for your contribution! In order for us to evaluate and accept your PR, we ask that you [sign a contribution license agreement] ([redacted]). It's all electronic and will take just minutes.

Hi @[redacted], many thanks for your contribution! In order for us to evaluate and accept your PR, we ask that you [sign a contribution license agreement] ([redacted]). It's all electronic and will take just minutes.

You did it @[redacted]! Thank you for signing the [redacted] Contribution License Agreement. We will have a look at your contribution!

Hi @[redacted], many thanks for your contribution! In order for us to evaluate and accept your PR, we ask that you [sign a contribution license agreement] ([redacted]). It's all electronic and will take just minutes.

You did it @[redacted]! Thank you for signing the [redacted] Contribution License Agreement. We will have a look at your contribution!

Hi @[redacted], many thanks for your contribution! In order for us to evaluate and accept your PR, we ask that you [sign a contribution license agreement] ([redacted]). It's all electronic and will take just minutes.

You did it @[redacted]! Thank you for signing the [redacted] Contribution License Agreement. We will have a look at your contribution!

Hi @[redacted], many thanks for your contribution! In order for us to evaluate and accept your PR, we ask that you [sign a contribution license agreement] ([redacted]). It's all electronic and will take just minutes.

You did it @[redacted]! Thank you for signing the [redacted] Contribution License Agreement. We will have a look at your contribution!

Figure 1: Anonymised screenshot of the rating application in action.

Table 3
Summary characteristics of final ground truth dataset.

number of...	**bot**	**human**	**total**
commenters	527	4,473	5,000
repositories with at least 1 commenter	505	3,515	3,909
comments	28,287	268,504	296,791
issues with at least 1 commenter	2,749	46,959	49,623
PRs with at least 1 commenter	16,937	118,896	134,208

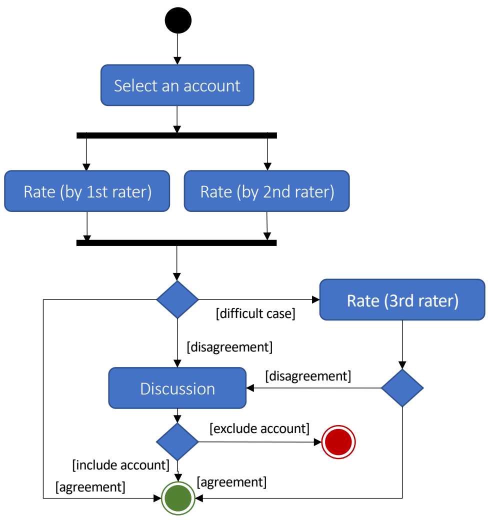


Figure 2: Workflow of the rating process.

4. Feature selection

In this section, we explain the features that will be used by the classification model to distinguish bots from human commenters. These features include the number of comment patterns, the number of (empty) comments, and the number of comments within each pattern. The following subsections explain these features and the rationale behind their selection.

4.1. Text distance between comments

Based on the assumption that bots perform more repetitive and automated tasks, we hypothesise that bot commenters exhibit more repetitive comments than human commenters. Consequently, we expect comments belonging to a bot to exhibit more similarity than comments belonging to a human commenter. In order to measure the similarity between comments of each commenter, both in terms of content and structure, we rely on text distance metrics that are commonly used for this purpose in natural language processing. The two metrics we consider are the Jaccard [40] and

Levenshtein [50] distances. The first one aims to quantify the similarity of two texts based on its content, and the second one captures the structural difference by counting single character edits.

More precisely, the Jaccard distance $J(C_1, C_2)$ measures the distance between two texts C_1 and C_2 by comparing the number of distinct common words in C_1 and C_2 with the total number of distinct words in C_1 and C_2. If $words(C)$ denotes the set of words in C, then $J(C_1, C_2)$ is computed as:

$$\mathcal{J}(C_1, C_2) = 1 - \frac{|\, words(C_1) \cap words(C_2) \,|}{|\, words(C_1) \cup words(C_2) \,|}$$

The second distance we consider is the Levenshtein edit distance $lev(C_1, C_2)$ that measures the difference between two character sequences C_1 and C_2 by counting the minimum number of single-character edits (insertion, deletion, or substitution) required to convert C_1 into C_2. We rely on its normalized version, computed as:

$$\mathcal{L}(C_1, C_2) = \frac{lev(C_1, C_2)}{max(|C_1|, |C_2|)}$$

To support our assumption that comments made by a bot have higher similarity than comments made by a human, we computed for each commenter in the ground truth dataset the Jaccard and Levenshtein distances between all pairs of comments belonging to that commenter. In order to compute the Jaccard distance, we first needed to split comments into words, a process also known as *tokenization*. To do so, we relied on spaCy, an "industrial-strength natural language processing library"[6] that notably offers a fast but robust tokenization algorithm, among others.

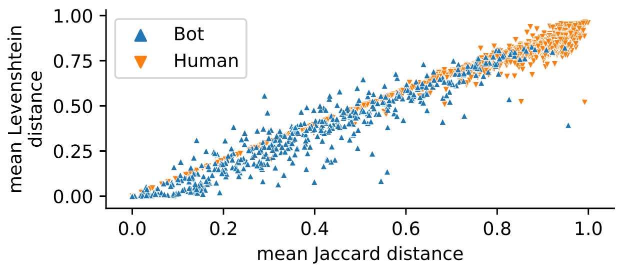


Figure 3: Mean Levenshtein and Jaccard distances between pairs of comments, per commenter.

Fig. 3 shows the mean Levenshtein and Jaccard distances for each commenter, distinguishing between bots (blue triangles) and humans (orange triangles). We observe that many humans are grouped in the top right part of the figure, i.e., they have high mean values for both distances. On the other hand, most bots have lower values for their mean distances. For instance, 91.6% of all bots have mean Jaccard and Levenshtein distances below 0.75. For comparison, only 7.2% of all human commenters exhibit mean Jaccard and Levenshtein distances below 0.75.

[6]`https://spacy.io`

Despite this, there is still a lot of overlap between bots and humans in Fig. 3, indicating that the mean distances are not enough to properly distinguish between bots from humans. By manually inspecting the comments belonging to bots having high mean distances, we found that their comments usually form sets of similar comments. Even if the distance between comments in a set (i.e., intra-set distance) is low, the distance between comments belonging to different sets (i.e., inter-set distance) is high. As a consequence, the overall mean distances between all comments tends to remain high, rivalling the distances observed for most human commenters.

We found many of these cases. One example is the bot that was identified in Fig. 1. We observe that it has two different sets of similar comments. The first set consists of comments of the form "*You did it @…! Thank you for signing the … Contribution License Agreement. We will have a look at your contribution!*". The second set consists of comments of the form "*Hi @…, many thanks for your contribution! In order for us to evaluate and accept your PR, we ask that you [sign a contribution license agreement] … It's all electronic and will take just minutes.*". The mean distance between pairs of all 20 comments belonging to the first set (i.e., intra-set distance) is very low (0.06 and 0.08 for Levenshtein and Jaccard distance respectively) and even lower (0.04 and 0.05 respectively) for the second set of 27 comments. However, the intra-set distance (i.e., the distance obtained by comparing comments from the first pattern with comments for the second pattern) is much much higher (0.70 and 0.81 for Levenshtein and Jaccard distance respectively). Consequently, the overall mean distances between all pairs of comments are 0.37 for Levenshtein and 0.43 for Jaccard distance. These distances are usually observed for human commenters, not for bots.

4.2. Repetitive comment patterns

Since high mean distances between comments of a commenter could correspond to either a human or, in many cases, to a bot having sets of similar comments, we cannot exclusively rely on these mean distances to distinguish between bots and humans. However, we observed that bots tend to have sets of many similar comments (i.e., they follow comment patterns), while we found that most comments from humans are unique and only a few of them seem to follow a pattern (e.g., "*Thank you!*", "*LGTM*"[7] or "*+1*"[8]).

Based on this observation, we expect bots to have a lower number of comment patterns than humans. In order to capture these comment patterns, we rely on a clustering algorithm. Clustering aims to group items into sets ("clusters"), in such a way that items belonging to the same cluster are more similar than items belonging to different clusters.

We selected DBSCAN (Density Based Spatial Clustering of Applications with Noise) [25], a well-known density-based clustering algorithm that notably has the ability (i) to

[7]Shorthand for "Looks Good To Me", a common way among GitHub users to agree with what is proposed in a pull request.

[8]This is another common way of expressing agreement with what was proposed in the previous comment or in the issue or PR description.

generate clusters of unequal size (i.e., we can have patterns with unequal numbers of comments), (ii) to generate a single cluster if needed (e.g., a commenter whose comments are all the same), and (iii) to generate single item clusters (e.g., a commenter whose comments are all very different). Additionally, DBSCAN permits not to specify the number of clusters in advance, fitting our use case wherein we do not know the number of patterns of each commenter in advance.

Since we aim to capture both the structural and content distance between comments, we rely on a combination of the Levenshtein and Jaccard distance, defined as follows:

$$\mathcal{D}(c_1, c_2) = \frac{\mathcal{L}(c_1, c_2) + \mathcal{J}(c_1, c_2)}{2}$$

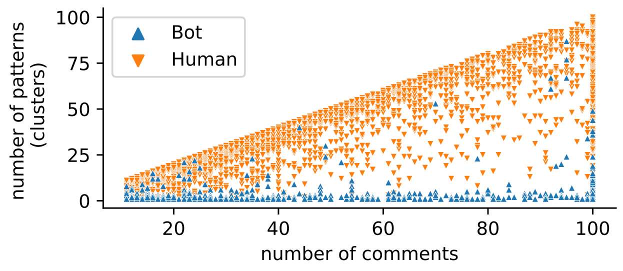


Figure 4: Number of comment patterns (clusters) and number of considered comments per commenter.

For each commenter, we computed $\mathcal{D}(c_i, c_j)$ for each pair (c_i, c_j) of comments. The resulting distance matrix, one per commenter, is then passed to DBSCAN to group the comments based on their similarity. Fig. 4 reports on the number of patterns (i.e., clusters), distinguishing between bot and human commenters. Since the number of patterns could depend on the number of comments, we report on the number of patterns relative to the number of considered comments.

Compared to Fig. 3 we can observe a much clearer separation between bots and humans based on the number of comment patterns and the number of comments, although it is not perfect. We observe that most humans are along the diagonal line which indicates that the number of patterns is close to the number of comments, and that almost all bots are along the horizontal axis. This means that the number of comment patterns for bots remains stable, and low, regardless of the number of comments they made. This confirms our assumption that bots have a limited set of comment patterns, contrarily to humans that seems to make much more varied comments.

4.3. Inequality between comments in patterns

Although we expected human comments to be mostly non-repetitive (i.e., each comment corresponds to a different pattern), we found instances in which a human commenter had a non-negligible number of repetitive comments (e.g., "*Thank you!*", "*LGTM*" or "*+1*") alongside other messages. This leads to having human commenters whose number of comment patterns is much lower than the number of comments, which is exactly the assumption we had for bots due to their repetitive comments. However, we found that those human commenters correspond to cases having at the same time a few patterns with many comments and many patterns with a few (mostly single) comments. On the other hand, bots exhibit single comment patterns less often. For instance, among the 2,431 patterns corresponding to bots, 50% are composed of a single comment, while this proportion is much higher (95.9%) for the 230,711 patterns we have for humans.

This observation lead us to consider the inequality in the number of comments in each pattern as a supplementary feature to distinguish between bots and humans. The Gini coefficient [19] provides a way to quantify the inequality (i.e., the distribution) of the number of comments for each pattern. A value of 0 expresses perfect equality (i.e., each comment pattern consists of the same number of comments). A value of 1 expresses maximal inequality among values (i.e., a few patterns capture many comments, and the remaining comments are spread into many single-comment patterns).

Let us consider the example of a specific human commenter in our dataset. This human made 73 comments belonging to 12 patterns. 9 of these patterns have exactly one comment. The other ones correspond to "*LGTM*" (37 comments), "*##Fixes{Number}*" (22 comments) and "*lgtm*" (5 comments). As a result, the Gini coefficient for this commenter is very low 0.04, since most patterns (9 out 12) have the same number of comments. Let us compare this to a bot in our dataset with a similar number of comments (61) and comment patterns (10). The number of comments in each pattern is more unequally distributed, ranging from 1 to 49 comments per pattern, a consequence of much more repetitive messages. As a result, its Gini coefficient is much higher, namely 0.52.

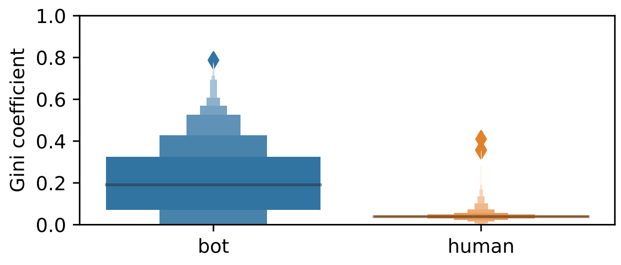


Figure 5: Distribution of Gini coefficient for bot and human commenters.

Fig. 5 shows the distribution of the Gini coefficient for all bots and humans in our dataset, by means of boxen plots [38]. We observe that humans exhibit a lower inequality than bots with respect to the spread of comments within patterns. We statistically compared these distributions using a Mann-Whitney-U test [54]. The null hypothesis, stating that the two distributions are the same was rejected ($p < 0.001$), indicating a statistically significant difference between the two distributions. The effect size turned out to be *large* (Cliff's delta $|d| = 0.58$) [12, 62]. This confirms that humans tend to have a lower inequality than bots, a consequence of many of their patterns containing a single comment. Therefore, the Gini coefficient can help in distinguishing between bots and humans.

4.4. Number of comments and empty comments

In addition to the number of patterns and the unequal distribution of comments within patterns, we also consider the number of comments made by each commenter as a feature for our model. This feature makes it possible to distinguish between commenters having a similar number of patterns. Indeed, consider for example two commenters having exactly 10 patterns. Assume they have respectively 10 and 100 comments. The first commenter is likely to be a human (since it has 10 patterns each containing exactly one comment, i.e., all comments are different), while the second one is more likely to be a bot.

We also consider the number of empty comments as a feature for our model. Indeed, during the rating process we found that a non-negligible proportion (6.5%) of the considered comments were empty. The presence of such comments in the dataset may seem strange. Even if the GitHub user interface does not allow empty comments in a discussion, it does not prevent comments to be composed of white characters. Moreover, the GitHub user interface allows the creation of pull requests whose description is empty. Since this description is the very first comment of a pull request, it explains why we found empty comments in the dataset.

Interestingly, we found that empty comments are mostly created by human commenters and not by bots. For instance, only 7% of all bots generated at least one such comment, whereas this proportion reaches 41.2% for human commenters. This should not come as a surprise, since one could expect bots mainly to generate informative comments and, by definition, empty comments are uninformative. Consequently, we decided to consider the number of empty comments as a feature of our classification model.

In summary, based on the analysis in this section we decided to use four distinct features for commenters to train the classification model: (i) the number of comment patterns; (ii) the inequality between comments in patterns; (iii) the total number of comments for the commenter; and (iv) the number of empty comments.

5. Classification model

All the scripts and data used to carry out the experiments in this section are available in a replication package on: https://github.com/mehdigolzadeh/IdentifyBots_ReplicationPackage

5.1. Classifier selection

A wide variety of algorithms can be used to construct a classification model. In this section we compare different classification algorithms to determine which one is the most appropriate to distinguish between bot and human commenters. Among the classifiers having the ability to perform binary classification, we consider decision trees (DT) [64], random forest (RF) [8, 27], support vector machines (SVM) [36], logistic regression (LR) [10], and k-nearest neighbours (kNN) [2]. Since the performance of these classifiers could depend on the input parameters, we follow a standard workflow of hyper-parameter tuning using a grid-search cross-validation process [73] (see Fig. 6). To do so, we rely on scikit-learn [58], a well-known machine learning library for Python.

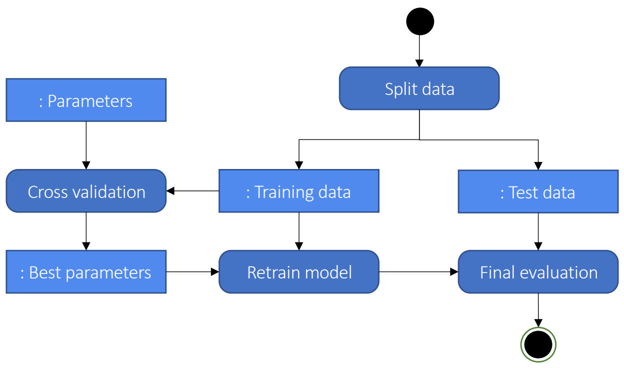


Figure 6: Standard workflow for grid-search cross-validation

We first divided the ground-truth dataset into two disjoint sets: a training set containing 60% of the data that will be used in a grid-search cross-validation process to determine the best input parameters and the best classifier, and a test set composed of the remaining 40% that will be used to evaluate the performance of the selected classifier and parameters on new data. Since we have many more humans than bots in our datasets, we relied on a stratified train-test split method to create these two sets with the same ratio of bots and humans.

Selecting an appropriate model with the best possible parameters requires hyper-parameter tuning. Based on the supported parameters of each classifier, we implemented a grid-search process based on a limited set of values for each parameter. For example, DT and RF were evaluated by setting the split criterion to *Gini* and *entropy*, among others. Doing so resulted in 91 different classifiers. To address the class imbalance problem [37] and avoid affecting the performance of the classifiers [35], we rely on a cost-sensitive learning approach [21]. Practically, this means we set the *class weight* parameter in scikit-learn to *balanced* for each supported classifier.

We then trained and evaluated the performance of all classifiers using a 10-fold cross-validation process. This approach splits the dataset into 10 subsets of equal size, and for each fold a model is trained using 9 subsets and is evaluated on the remaining one. The overall performance of the model is averaged from the performance of these 10 models. To ensure that the created subsets preserve the same proportion of bots and humans as in the complete training set, we relied on a *stratified shuffle split* to create them.

The performance of the resulting models is measured using the classical metrics of precision P, recall R and $F1$-score. We use these metrics for the population of each class

Table 4
Definitions of precision, recall and $F1$-score.

population	precision P	recall R	$F1$-score
bots B	$\frac{TP}{TP+FP}$	$\frac{TP}{TP+FN}$	$\frac{2\times P(B)\times R(B)}{P(B)+R(B)}$
humans H	$\frac{TN}{TN+FN}$	$\frac{TN}{TN+FP}$	$\frac{2\times P(H)\times R(H)}{P(H)+R(H)}$
$B \cup H$	$\frac{P(B)\times\|B\|+P(H)\times\|H\|}{\|B\|+\|H\|}$	$\frac{R(B)\times\|B\|+R(H)\times\|H\|}{\|B\|+\|H\|}$	$\frac{2\times P\times R}{P+R}$

Table 5
Precision, recall and $F1$-score of the best classifiers per family of classifiers (in descending order of $F1$-score).

	bots		humans		overall		
classifier	$P(B)$	$R(B)$	$P(H)$	$R(H)$	$P(B \cup H)$	$R(B \cup H)$	$F1(B \cup H)$
RF	0.932	0.916	0.990	0.992	0.984	0.984	0.984
kNN	0.943	0.853	0.983	0.994	0.978	0.979	0.978
SVM	0.876	0.925	0.991	0.984	0.979	0.978	0.978
DT	0.882	0.884	0.986	0.985	0.975	0.974	0.974
LR	0.839	0.931	0.992	0.978	0.975	0.973	0.974
ZeroR	-	0.000	0.893	1.000	0.798	0.893	0.843

(i.e., for bots B and humans H). For the entire population we computed the *weighted* version of these metrics to take into account the class imbalance. We aim to achieve an as high $F1$-score as possible. Since our goal is to identify bots, we also strive to keep bot recall $R(B)$ high enough, given that the population of bots is significantly smaller than the population of humans, and that it is much easier and faster to recover from humans misclassified as bots than the opposite. All these metrics are summarized in Table 4, and are defined in terms of the number of *true positives* ***TP*** (the number of bots that are correctly classified as such by the model), *true negatives* ***TN*** (the number of humans that are correctly classified as such by the model), *false positives* ***FP*** (humans that are wrongly classified as bots), and *false negatives* ***FN*** (bots that are wrongly classified as humans).

Following the grid-search cross-validation process described above, we trained and obtained 91 classifiers. For each of them, we computed the resulting *bot*, *human* and *overall* precision, recall and $F1$-score. Table 5 reports on these metrics, in descending $F1$-score order. To ease readability, rather than reporting on all 91 classifiers, we selected for each classifier category (e.g., DT, RF, ...) the instance whose parameters resulted in the highest $F1$-score. We also compared the precision, recall and $F1$-score of these classifiers against a baseline classifier, ZeroR. ZeroR is a very simple classifier that has no predictive power: it ignores the features and always predicts the majority class (i.e., "human" in our case). We observe that all classifiers exhibit a high overall performance (in terms of precision, recall and $F1$) and surpass ZeroR by a wide margin.

The overall scores for R, P and $F1$ of all classifiers are consistently higher than the ZeroR baseline, and range between 0.974 and 0.984. Even though the best SVM and LR classifiers have higher bot recall $R(B)$ than the best RF classifier (0.925 and 0.931 compared to 0.916, respectively), the overall R, P and $F1$ scores are highest for the RF classifier. We therefore decided to use the best RF classifier, which was obtained with the *entropy* split criterion, 10 estimators (i.e., trees) and a maximum depth of 10 for these trees.

Table 6
Evaluation of the classification model using the test set.

	classified as bot	classified as human	P	R	F1
Bot	TP: 192	**FN: 19**	0.94	0.91	0.92
Human	**FP: 13**	TN: 1776	0.99	0.99	0.99
weighted avg			0.98	0.98	0.98

5.2. Evaluation

In this subsection, we aim to evaluate the actual performance of that model on data that were not used to train the model, i.e., on new data contained in the test set. Following the workflow presented in Fig. 6, we start by constructing a new classification model instance based on the selected RF classifier, its parameters, and the training set containing 60% of the ground-truth dataset.

We evaluate and report the accuracy of the model based on the test set, corresponding to the remaining 40% of the ground-truth dataset. This test set includes 2,000 commenters, of which 1,789 are humans and 211 are bots. The evaluation results are reported in Table 6.

We see that most bots and humans are correctly classified by the model. For instance, only 19 out of 211 bots were misclassified as humans (**FN**), and only 13 out of 1789 humans were misclassified as bots (**FP**). The overall $F1$-score is very high (0.98), a consequence of the high precision (0.98) and high recall (0.98) of the model. Thanks to the fact that we

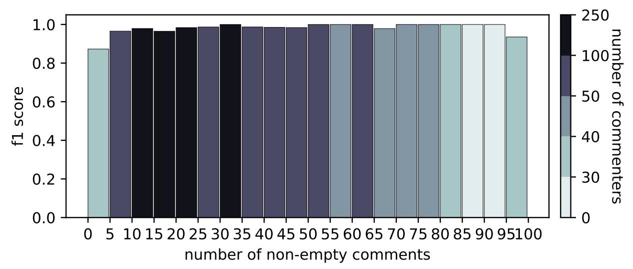


Figure 7: $F1$-score of the model when applied on commenters, grouped by their number of non-empty comments. The colour indicates the number of commenters in each bin.

have taken into account class imbalance during the training phase, these high scores can also be observed individually for each class, even if the precision and recall for bots is slightly lower than for humans. These results confirm what we already observed in previous section, that is, the model is effective in identifying bots and humans.

Scrutinising all 19 misclassified bots (***FN***) we found that ten of them were already problematic during the first step of the manual rating process, where they were rated as either "Human" or "I don't know" by one of the raters. Moreover, the final decision to classify them as bots during the discussion session among raters was based on additional information that is not available in the comments themselves, explaining why the model is not able to classify them correctly.

The model also misclassified 13 humans. The fact that the model misclassified these humans as bots is not surprising given that, during the first step of the rating process, 10 out of 13 cases were manually rated as *difficult* or *very difficult*, 2 cases as "I don't know" by both raters and one case was even rated as a bot by one of the raters. Section 7 provides a detailed analysis of these misclassified commenters.

Since the model relies on features computed on comments to distinguish bots from humans, it is worthwhile to consider and measure the impact of the number of considered comments on the performance of the model. In particular, we aim to identify the minimal number of non-empty comments required to reliably classify bots and humans. To this end, we evaluated our model and computed the $F1$-score for commenters in the test set, grouped by their number of non-empty comments.

Fig. 7 shows the resulting $F1$-scores of the model grouped by bins based on the number of non-empty comments. The colour of a bin indicates how many commenters there are in that bin. The bins with 10 to 24 and 30 to 34 non-empty comments have the highest number of commenters, while bins between 85 to 94 have the lowest number of commenters. The $F1$-score increases from 0.87 (bin 0-4) and becomes stable around 0.96 to 1.00 after 10 non-empty comments are reached (from bin 10-14). This suggests that having at least 10 non-empty comments is enough to achieve good performance with the model.

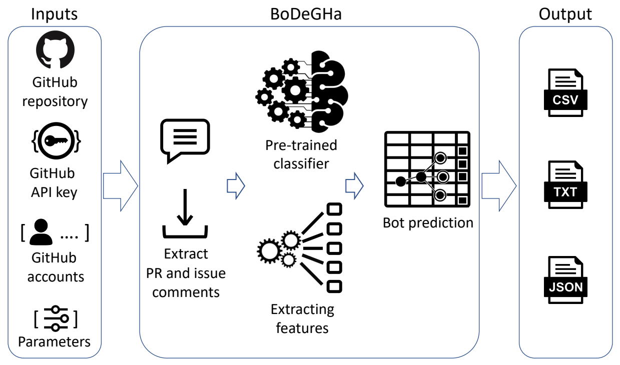


Figure 8: The BoDeGHa architecture.

```
$ bodegha -h
usage: bodegha [-h] repository [--accounts [ACCOUNT [ACCOUNT ...]]]
       [--exclude [ACCOUNT [ACCOUNT ...]]] [--start-date START_DATE]
       [--verbose] [--min-comments MIN_COMMENTS] [--only-predicted]
       [--max-comments MAX_COMMENTS] [--text | --csv | --json]
        --key APIKEY
```

Figure 9: List of command-line arguments for BoDeGHa 1.0.1.

6. The BoDeGHa bot detector tool

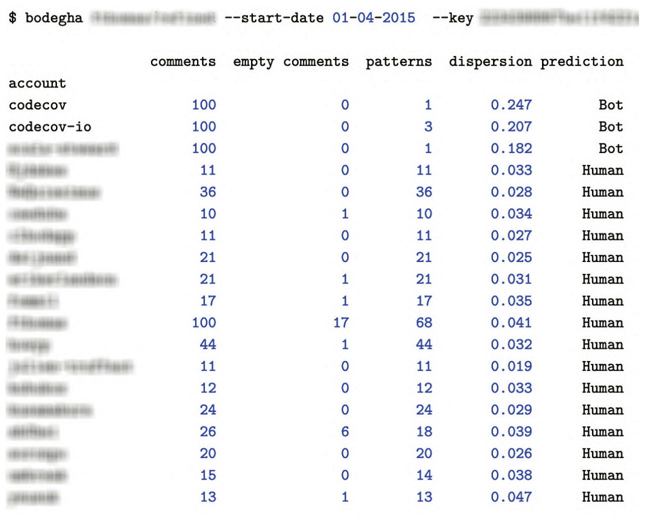
Since the classifier we trained to identify bots presents very good performance, we implemented it as part of a tool. The tool is called BoDeGHa (Bot Detector for GitHub activity), is developed for Python 3.7 and is easily installable through pip, the official package manager for Python.[9] BoDeGHa can be used by any researcher or practitioner to classify accounts of a given GitHub repository either as bot or as human based on their issue and PR comments.

In its simplest form, BoDeGHa accepts the name of a GitHub repository and a GitHub API key. BoDeGHa computes its output in three steps, summarized in Fig. 8. The first step consists of downloading all comments from the specified repository thanks to GitHub's GraphQL API. This step results in a list of commenters and their corresponding comments. The second step consists of computing the number of comments, empty comments, comment pattern and inequality between number of comments within patterns (i.e., the features of the classification model). The third step simply applies the pre-trained model on these examples, and outputs the prediction made by the model.

BoDeGHa supports several additional parameters. The minimum and maximum number of comments to download and to consider can be specified, as well as the start date from which to consider comments. It is also possible to provide a list of specific accounts for the tool to consider. To ease its reuse by other tools, it is also possible to export the results either as comma-separated values or JSON. The command-line interface of BoDeGHa is summarized in Fig. 9.

Fig. 10 presents the output of BoDeGHa for a randomly chosen GitHub repository. The output shows, for each GitHub account (first column), the number of extracted

[9]Using `pip install git+https://github.com/mehdigolzadeh/BoDeGHa`



```
$ bodegha [redacted] --start-date 01-04-2015  --key [redacted]

                comments  empty comments  patterns  dispersion prediction
account
codecov              100               0         1       0.247        Bot
codecov-io           100               0         3       0.207        Bot
[redacted]           100               0         1       0.182        Bot
[redacted]            11               0        11       0.033      Human
[redacted]            36               0        36       0.028      Human
[redacted]            10               1        10       0.034      Human
[redacted]            11               0        11       0.027      Human
[redacted]            21               0        21       0.025      Human
[redacted]            21               1        21       0.031      Human
[redacted]            17               1        17       0.035      Human
[redacted]           100              17        68       0.041      Human
[redacted]            44               1        44       0.032      Human
[redacted]            11               0        11       0.019      Human
[redacted]            12               0        12       0.033      Human
[redacted]            24               0        24       0.029      Human
[redacted]            26               6        18       0.039      Human
[redacted]            20               0        20       0.026      Human
[redacted]            15               0        14       0.038      Human
[redacted]            13               1        13       0.047      Human
```

Figure 10: Example of running BoDeGHa.

comments (second column), the number of empty comments (third column), the number of computed comment patterns (fourth column), and the inequality among them (fifth column). The last column provides the predicted class of each account. This example shows that three commenters are identified as bots, and all remaining commenters as humans.

7. Discussion

The evaluation of the classifier revealed several commenters that the model was not able to properly classify. We specifically look at the commenters that have been misclassified by the model. During the evaluation of the model on the test set, we found 19 bots and 13 humans that were misclassified. In order to have a more complete categorisation of misclassified commenters, we also applied the model on the training set and obtained 4 additional bots and 12 additional humans that are misclassified.

Starting with the 24 (19+5) bots, we found that in most cases they correspond to bots that use, convert or copy text that was initially produced by humans. Even if these bots perform repetitive tasks (i.e., copy information) and even if some of these bots use templates to transfer or copy comments that are recognizable to the human eye (e.g., "*Jira issue originally created by user {username}: {content of the issue}*"), it is difficult for an automated algorithm to detect such cases.

Copy from humans (9 bots): We found some instances of bots whose comments were generated based on content made by humans (e.g., taskcat-ci, trax-robot). Since our model solely relies on features derived from comments, bot comments originating from human messages increase the likelihood of an incorrect classification. Among these cases, we found several bots that transfer data (including issues, PRs and their associated comments) to GitHub from issue trackers, code review support tools, email etc. For example, neos-bot transfers all issues from a Jira issue tracker, suchabot duplicates comments and issues from another system to GitHub, and wallabag-bot migration from email content to GitHub.

Insufficient comments (9 bots): We found 9 bots (e.g., devtools-bot and egg-bot) that were wrongly identified as humans due to the lack of a sufficient number of non-empty comments. Since our model relies on comment contents, bots with too few non-empty comments may lead to incorrect predictions even if these comments have similar comment patterns. We do not see any direct way to overcome this, since bots are expected to provide relevant information about what they are doing, and as such, one can expect their comments to be informative and non-empty.

Diverse comments (6 bots): We found 6 cases of bots that are used for the purpose of reporting, logging, or proposing code changes. The variation of comments in these bots increases the number of comment patterns, which prevents the model from identifying these bots. The source of the comment diversity comes from the reports they send for each task. For example sentry-io creates an issue each time an error occurs in the software project, along with the details of this error (e.g., stack trace). Another example is violinist-bot that submits a PR to update outdated dependencies and to report about the changes of this update. Despite these comments starting with a similar sentence (e.g., "*Sentry Issue:*" or "*If you have a high test coverage index, and your tests for this pull request are passing, it should be both safe and recommended to merge this update. Here is a list of changes between the version you use, and the version this pull request updates to:*"), they mainly consist of details related to the submitted issue or PR (i.e., stack traces for sentry-io and list of issues for violinist-bot) and are considered as different comment patterns.

We also looked at the 17 (13+4) humans that were misclassified as bots, and created the following categories:[10]

Repetitive comments (8 humans): We found 8 instances of human commenters whose comments are mostly composed of repetitive messages, such as *thank you* or *LGTM* and that have nearly no other comments. Since repetitive messages are usually indicative of the presence of a bot, the model failed to correctly classify these commenters.

Insufficient comments (3 humans): We found 3 humans with few comments, most of them being empty. Most of these comments were created in the context of a pull request whose title was already sufficiently informative. Since these empty comments are grouped in a single comment pattern, and since they form the large majority of the comments made by these commenters, they were wrongly considered as being generated by a bot due to their repetitive nature. We also found instances where the comment content is too short or there are too few non-empty comments. This prompts our

[10]To comply with GDPR regulations, we cannot provide the account names for these cases.

algorithm to group them into a small number of patterns and consequently provide wrong predictions.

Mostly unfilled issue templates (3 humans): It is not unusual in GitHub repositories to require commenters to follow a comment template or a checklist when creating issues or pull requests.[11] We found 3 commenters whose comments were mostly composed of unfilled or barely filled templates, leading these comments to be considered as a single pattern, and leading the model to misclassify them as bots. Relying on an analysis of the content of such comments could prevent them from being misclassified, by taking into account the presence of such templates.

Others (3 humans): These cases do not fall into any of the above categories, and we have found no specific reason to explain their misclassification. Some of them have a small number of comments, while others only have a few patterns (e.g., due to the presence of similar long URLs in comments) despite the fact that they do not seem to have duplicated or similar comments.

Most commenters that were misclassified by the classification model were also hard to recognize by the raters during the process of creating the ground-truth dataset. In the test set, about 84.6% (11 out of 13) of the humans that were misclassified as bots and about 63.1% (12 out of 19) of the bots that were misclassified as humans were originally rated as "I don't know", "difficult", or "very difficult" by at least one of the raters. In contrast, among the correctly classified commenters, a much lower percentage of bots (12.5%, 24 out of 192) and humans (9.5%, 169 out of 1772) were rated as such.

Furthermore, during the creation of the ground-truth dataset, we encountered several examples of commenters whose features and comments were reminiscent of both humans and bots. Such so-called "mixed" commenters are the result of GitHub accounts belonging to humans allowing automatic tools to use their account for carrying out certain specific tasks. Hence, the comments of such commenters include both human-like and bot-like behaviour. We identified 78 such commenters out of 5,082 commenters (i.e., 1.5%) during the rating phase and we consistently excluded them from the ground-truth dataset since we could not decide whether these commenters should be classified as bots or humans.

Nevertheless, it is interesting to report how our model behaves when exposed to these specific "mixed" cases. Out of these 78 identified "mixed" commenters, 21 were classified as bots (26.9%) and 57 as a humans (73.1%). The fact that the proportion of "mixed" commenters classified as bots is higher than the one in the training set (10.3%) suggests that their behaviour is perceived to be closer to that of a bot than a human by the classification model.

The presence of mixed accounts as well as the categories of bots that have been misclassified as humans suggests that it is not easy to come up with a single definition for a bot. Two persons could easily disagree on whether a given account is a bot or a human if they have a different interpretation of what it means to be bot. This calls for a more precise definition of bots. Erlenhov et al. [24] started doing so based on qualitative interviews with developers. This enabled them to identify three distinct DevBot *personas* that differ in terms of features like autonomy, chat interfaces, and smartness. This more fine-grained classification of DevBots and their characteristics paves the way for more sophisticated classification models.

The approach presented in this paper is not the first one to have been proposed in the literature to detect bots in social coding platforms. Dey et al. [17] proposed three different approaches for identifying bot accounts in GitHub projects, mostly based on their commit messages. One of them consists of checking for the presence of the string "bot" in the account name of the committer. We partially relied on this heuristic to add more potential bot candidates during our data collection. However, solely relying on it to identify bots is likely to lead to a large number of both false positives and false negatives. To confirm this, we applied their approach on our ground-truth dataset. We found 169 humans out of 4,473 (3.8%) containing the string "bot" in their account name, either at the end (46 cases) or in the middle (123 cases). Out of the 527 bots we have in the dataset, 394 of them (i.e., 74.7%) actually contained "bot" in their account name, usually at the end of the name (378 cases). Although this may seem high for such a simple heuristic, it still implies that more than one out of four bots is missed with this method, and about one out of 25 humans is mistakenly considered a bot. For comparison, around only one out of 25 (3.8%) bots have been misclassified as humans by our model, and around only one out of 100 humans (1.1%).

8. Threats to Validity

Based on the structure recommended by Wohlin et al. [74] we discuss the threats that might call into question the validity of our findings, their potential impact and how we have tried to mitigate them.

Construct validity examines the relationship between the theory behind the experiments performed and the observations found. This threat is mainly related to correctness of the dataset used in the experiments. The results of our study are strongly dependent on the correctness of the ground-truth dataset. We are confident that the ground truth contains very few errors, since we achieved an *almost perfect* agreement ($\kappa = 0.96$) based on an iterative rating process involving all authors of this paper. One of the most likely threats is the existence of "mixed" commenters in the dataset. Such commenters are difficult to classify, even by human raters, since they combine both bot-like and human-like behaviour. Mixed commenters constitutes a very small proportion of our dataset (78 cases, corresponding to 1.5% of all considered accounts). We excluded all these cases from the dataset since we could not agree on them. However, it is possible that the dataset still contains such cases that were not iden-

[11]See https://docs.github.com/en/github/building-a-strong-community/about-issue-and-pull-request-templates

tified by the raters. Given the very low ratio of such mixed accounts, it is however unlikely to affect our findings.

Internal validity concerns choices and parameters of the experimental setup that could affect the results of the observations. Given that our classification method is fully based on features computed from comments, we required each commenter included in the dataset to have contributed at least 10 (possible empty) comments. This threshold is based on previous experiences and findings [32]. As such, we cannot claim that our model applies on commenters who made fewer than 10 comments. Similarly, we considered at most 100 comments for each commenter but, as explained in Section 5.2, this upper limit on the number of comments is unlikely to have biased our results, since we already achieved high $F1$-score starting from 10 non-empty comments.

Conclusion validity concerns whether the conclusions derived from the analysis are reasonable. Our conclusions are based on the evaluation and application of the classification model on the test set. Given that we properly followed a standard grid-search cross-validation method to identify the best classifier, and that we evaluated the model on the test set (i.e., examples that have not been used to train or select the classifier), the results we obtained and conclusions we reached are unlikely to be affected.

External validity concerns the degree to which the conclusions we derived are generalisable outside the scope of this study. The main threat to external validity is related to the construction of the ground-truth dataset. To avoid any potential bias, we randomly selected a large collection of GitHub repositories related to software development and corresponding to actual packages being officially distributed, following the guidelines of Kalliamvakou et al. [42]. While this dataset can be regarded as representative of bots contributing to GitHub repositories through PR and issue comments, we do not make any claim about its generalisability to other activities (e.g., commit messages) or other social coding platforms (e.g., BitBucket or GitLab). Nevertheless, the underlying approach could be made applicable to such activities or platforms.

9. Future work

As future work, we intend to use our classification model in socio-technical empirical analyses of collaborative software development, by studying the effect of the presence of bots on various development-related activities, such as the productivity and quality of handling issues, bugs and pull requests, code reviewing, intra- and inter-repository collaboration, developer onboarding, and so on.

With the emergence of more advanced AI, machine learning and natural language processing techniques, we can expect future bots to behave more and more like humans. These new technological advances may make our classification model less capable of distinguishing bots from humans. An example of such technique is *Generative Pretrained Transformer 3* (GPT-3), an autoregressive language model developed by OpenAI and integrating 175 billion parameters. GPT-3 generates text of sufficiently high quality that it is difficult to distinguish them from that written by humans [9]. To cope with this, we will explore more advanced machine learning methods that could take into account the semantics of the comments. In particular, we will consider techniques relying on natural language processing and deep neural networks to develop classification models that are more resilient to human-like bots, as well as "mixed accounts" corresponding to bots that copy, transfer or translate human comments.

In this study, we provided a binary definition of commenters being either bots or humans. The aforementioned study by Erlenhov et al. [24] revealed that a more fine-grained definition would be needed, and they came up with three DevBot *personas* based on their autonomy, chat interface, and smartness. Such a more fine-grained classification could be used to refine our ground-truth dataset, and to provide more advanced classification models, possibly even computing the probability that an account belongs to each of the considered *personas*.

Because of the growing use of bots during collaborative development activities [22], we can expect to see a proliferation of bots to automate software development in GitHub repositories. For instance, GitHub introduced in November 2019 GitHub Actions[12], a feature providing automated workflows for repository maintainers. These actions, fully integrated with GitHub, allow the automation of tasks based on a various set of triggers (e.g., commits, pull request, issue, comments, etc.). Since they are easily shareable from one repository to another one (through the GitHub Marketplace), we expect their use to become more widespread, even in smaller repositories and, as a result, to see more "bots" and their comments in GitHub repositories. However, tasks triggered through GitHub actions are automatically labelled as such by the GitHub API, eliminating the need to create a model to identify these "bots". Recently, GitHub action variants of many well-known bots (e.g., Coveralls, Codecov, Snyk) have been published to the GitHub Marketplace, and these actions are rapidly increasing in popularity. Consequently, we expect their GitHub action variant to replace progressively the bots currently being used in GitHub repositories.

We aim to extend our study, including the ground truth dataset, classification model and tool to accommodate other social coding platforms such as GitLab and BitBucket. This will generalise our approach, and allow us to study to which extent the selected platform affects the way in which bots are being used as part of the development process. We also aim to evaluate our model on other types of activities (e.g., git commit messages). This will allow us to understand to what extent our model can be used to identify bots based on commit activities.

[12] `https://github.com/features/actions`

10. Conclusion

In this paper, we proposed a novel approach to distinguish between bots and humans in collaborative software development repositories on GitHub, based on the comments they made in issues and PRs.

Our first contribution is the creation of a ground-truth containing 5,000 GitHub accounts including 527 bots (10.5%), based on a manual rating process with very high inter-rater agreement ($\kappa = 0.96$).

Using this ground-truth dataset, we developed a classification model to identify bots based on four features: the total number of comments of a commenter; its number of non-empty comments; its number of comment patterns; and the inequality between the numbers of comments in each pattern. The chosen features align with behavioural differences we observed between bots and humans. Indeed, we found that most human commenters tend to have diverse sets of comments with little repetition, while bots tend to frequently use a limited set of comment patterns.

Following a standard grid-search 10-fold cross validation process, we evaluated and compared five families of classifiers (random forest, k-nearest neighbours, decision trees, logistic regression and support vector machines) on a training set including 60% of all data. We performed hyper-parameter tuning to select the best parameters of each classifier family based on their precision, recall and $F1$-score. We selected the random forest classifier since it achieved the highest $F1$-score (98.4%).

We evaluated the selected classifier on new data, and found that it achieves high precision and recall. Based on a manual assessment and categorisation of bots and humans that were misclassified, we identified why the classification model had difficulties with detecting them, and we provided suggestions for further improvements to the classification model.

We implemented the classification model into a Python command-line tool, called BoDeGHa. This open source tool is made freely available to practitioners and researchers to allow them to analyse GitHub repositories and to identify which accounts correspond to bots and which correspond to humans.

References

[1] Abokhodair, N., Yoo, D., McDonald, D.W., 2015. Dissecting a social Botnet: Growth, content and influence in Twitter, in: ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW), ACM. pp. 839–851. doi:10.1145/2675133.2675208.

[2] Aha, D.W., Kibler, D., Albert, M.K., 1991. Instance-based learning algorithms. Machine Learning 6, 37–66. doi:10.1007/BF00153759.

[3] Amleshwaram, A.A., Reddy, N., Yadav, S., Gu, G., Yang, C., 2013. CATS: Characterizing automation of Twitter spammers. International Conference on Communication Systems and Networks (COMSNETS) doi:10.1109/COMSNETS.2013.6465541.

[4] Balachandran, V., 2013. Reducing human effort and improving quality in peer code reviews using automatic static analysis and reviewer recommendation, in: International Conference on Software Engineering (ICSE), IEEE. pp. 931–940. doi:10.1109/ICSE.2013.6606642.

[5] Ben Mimoun, M.S., Poncin, I., Garnier, M., 2017. Animated conversational agents and e-consumer productivity: The roles of agents and individual characteristics. Information & Management 54, 545–559. doi:10.1016/j.im.2016.11.008.

[6] Benotti, L., Martínez, M.C., Schapachnik, F., 2014. Engaging high school students using chatbots, in: Conference on Innovation and Technology in Computer Science Education (ITiCSE), ACM. pp. 63–68. doi:10.1145/2591708.2591728.

[7] Beschastnikh, I., Lungu, M.F., Zhuang, Y., 2017. Accelerating software engineering research adoption with analysis bots. International Conference on Software Engineering: New Ideas and Emerging Results Track , 35–38doi:10.1109/ICSE-NIER.2017.17.

[8] Breiman, L., 2001. Random forests. Machine learning 45, 5–32. doi:10.1023/A:1010933404324.

[9] Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al., 2020. Language models are few-shot learners. arXiv preprint arXiv:2005.14165 .

[10] Burridge, J., 1991. Journal of the Royal Statistical Society. Series A (Statistics in Society) 154, 361–364. URL: http://www.jstor.org/stable/2983054.

[11] Campbell, J.L., Quincy, C., Osserman, J., Pedersen, O.K., 2013. Coding in-depth semistructured interviews: Problems of unitization and intercoder reliability and agreement. Sociological Methods & Research 42, 294–320. doi:10.1177/0049124113500475.

[12] Cliff, N., 1993. Dominance statistics: Ordinal analyses to answer ordinal questions. Psychological Bulletin 114, 494–509. doi:10.1037/0033-2909.114.3.494.

[13] Cosley, D., Frankowski, D., Terveen, L., Riedl, J., 2007. SuggestBot: Using intelligent task routing to help people find work in Wikipedia, in: International Conference on Intelligent User Interfaces (IUI), ACM. pp. 32–41. doi:10.1145/1216295.1216309.

[14] Dabbish, L., Stuart, C., Tsay, J., Herbsleb, J., 2012. Social coding in GitHub: Transparency and collaboration in an open software repository, in: ACM Conference on Computer Supported Cooperative Work (CSCW), ACM. pp. 1277–1286. doi:10.1145/2145204.2145396.

[15] Dale, R., 2016. The return of the chatbots. Natural Language Engineering 22, 811–817. doi:10.1017/S1351324916000243.

[16] Dey, T., Ma, Y., Mockus, A., 2019. Patterns of effort contribution and demand and user classification based on participation patterns in NPM ecosystem, in: International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE), ACM. pp. 36–45. doi:10.1145/3345629.3345634.

[17] Dey, T., Mousavi, S., Ponce, E., Fry, T., Vasilescu, B., Filipova, A., Mockus, A., 2020. Detecting and characterizing bots that commit code, in: International Conference on Mining Software Repositories (MSR). doi:10.1145/3379597.3387478.

[18] Dominic, J., Houser, J., Steinmacher, I., Ritter, C., Rodeghero, P., 2020. Conversational bot for newcomers onboarding to open source projects doi:https://doi.org/10.1145/3387940.3391534.

[19] Dorfman, R., 1979. A formula for the gini coefficient. The Review of Economics and Statistics 61, 146–149. doi:10.2307/1924845.

[20] Efthimion, P.G., Payne, S., Nicholas, P., 2018. Supervised machine learning bot detection techniques to identify social twitter bots. SMU Data Science Review 1.

[21] Elkan, C., 2001. The foundations of cost-sensitive learning, in: International Joint Conference on Artificial Intelligence (IJCAI), Morgan Kaufmann. pp. 973–978. doi:10.5555/1642194.1642224.

[22] Erlenhov, L., Gomes de Oliveira Neto, F., Scandariato, R., Leitner, P., 2019. Current and future bots in software development, in: International Workshop on Bots in Software Engineering (BotSE), pp. 7–11. doi:10.1109/BotSE.2019.00009.

[23] Erlenhov, L., Neto, F.G.d.O., Chukaleski, M., Daknache, S., 2020a. Challenges and guidelines on designing test cases for test bots, in: International Workshop on Bots in Software Engineering (BotSE). doi:10.1145/3387940.3391535.

[24] Erlenhov, L., Neto, F.G.d.O., Leitner, P., 2020b. An empirical study of bots in software development: Characteristics and challenges from a practitioner's perspective, in: ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foun-

dations of Software Engineering (ESEC/FSE), ACM. pp. 445–455. doi:10.1145/3368089.3409680.

[25] Ester, M., Kriegel, H.P., Sander, J., Xu, X., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise, in: International Conference on Knowledge Discovery and Data Mining (KDD), AAAI Press. pp. 226—231. doi:10.5555/3001460.3001507.

[26] Farooq, U., Grudin, J., 2016. Human-computer integration. Interactions 23, 26–32. doi:10.1145/3001896.

[27] Frank, E., Hall, M., 2001. A simple approach to ordinal classification, in: De Raedt, L., Flach, P. (Eds.), European Confernece on Machine Learning (ECML), Springer. pp. 145–156. doi:10.1007/3-540-44795-4_13.

[28] Fryer, L.K., Ainley, M., Thompson, A., Gibson, A., Sherlock, Z., 2017. Stimulating and sustaining interest in a language course: An experimental comparison of chatbot and human task partners. Computers in Human Behavior 75, 461 – 468. doi:https://doi.org/10.1016/j.chb.2017.05.045.

[29] Geiger, R.S., 2013. Are computers merely "supporting" cooperative work: Towards an ethnography of bot development, in: International Conference on Computer Supported Cooperative Work (CSCW), ACM. pp. 51–56. doi:10.1145/2441955.2441970.

[30] Gnewuch, U., Morana, S., Mädche, A., 2017. Towards designing cooperative and social conversational agents for customer service, in: Kim, Y.J., Agarwal, R., Lee, J.K. (Eds.), International Conference on Information Systems (ICIS), Association for Information Systems. URL: http://aisel.aisnet.org/icis2017/HCI/Presentations/1.

[31] Golzadeh, M., Decan, A., Mens, T., 2019. On the effect of discussions on pull request decisions, in: The 18th Belgium-Netherlands Software Evolution Workshop, CEUR Workshop Proceedings. URL: http://ceur-ws.org/Vol-2605/16.pdf.

[32] Golzadeh, M., Legay, D., Decan, A., Mens, T., 2020. Bot or not? detecting bots in GitHub pull request activity based on comment similarity. International Workshop on Bots in Software Engineering (BotSE) , 31–35doi:10.1145/3387940.3391503.

[33] Gousios, G., Pinzger, M., Deursen, A.v., 2014. An exploratory study of the pull-based software development model, in: International Conference on Software Engineering (ICSE), ACM. pp. 345–355. doi:10.1145/2568225.2568260.

[34] Gousios, G., Storey, M.A., Bacchelli, A., 2016. Work practices and challenges in pull-based development: The contributor's perspective, in: International Conference on Software Engineering (ICSE), ACM. pp. 285–296. doi:10.1145/2884781.2884826.

[35] Grbac, T.G., Mausa, G., Basic, B.D., 2013. Stability of software defect prediction in relation to levels of data imbalance, in: Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications (SQAMIA). URL: http://ceur-ws.org/Vol-1053.

[36] Gunn, S., 1998. Support Vector Machines for Classification and Regression. Project Report. doi:10.1039/B918972F.

[37] He, H., Garcia, E.A., 2009. Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering 21, 1263–1284. doi:10.1109/TKDE.2008.239.

[38] Hofmann, H., Kafadar, K., Wickham, H., 2011. Letter-value plots: Boxplots for large data. Technical Report. had.co.nz. doi:10.1080/10618600.2017.1305277.

[39] Hu, Z., Gehringer, E.F., 2019. Improving feedback on GitHub pull requests: A bots approach, in: Frontiers in Education Conference (FIE), IEEE. pp. 1–9. doi:10.1109/FIE43999.2019.9028685.

[40] Jaccard, P., 1912. The distribution of the flora in the alpine zone. New Phytologist 11, 37–50. doi:https://doi.org/10.1111/j.1469-8137.1912.tb05611.x.

[41] Jain, M., Kota, R., Kumar, P., Patel, S.N., 2018. Convey: Exploring the use of a context view for chatbots, in: Conference on Human Factors in Computing Systems (CHI), ACM. pp. 1–6. doi:10.1145/3173574.3174042.

[42] Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D.M., Damian, D., 2014. The promises and perils of mining GitHub, in: International Conference on Mining Software Repositories (MSR), ACM. pp. 92–101. doi:10.1145/2597073.2597074.

[43] Katz, J., 2020. Libraries.io open source repository and dependency metadata (version 1.6.0). URL: https://doi.org/10.5281/zenodo.3626071, doi:10.5281/zenodo.2536573.

[44] Kerry, A., Ellis, R., Bull, S., 2009. Conversational agents in e-learning, in: Allen, T., Ellis, R., Petridis, M. (Eds.), Applications and Innovations in Intelligent Systems XVI, Springer. pp. 169–182. doi:10.1007/978-1-84882-215-3_13.

[45] Landis, J.R., Koch, G.G., 1977. The measurement of observer agreement for categorical data. Biometrics 33, 159–174. URL: http://www.jstor.org/stable/2529310.

[46] Lebeuf, C., Storey, M., Zagalsky, A., 2018. Software bots. IEEE Software 35, 18–23. doi:10.1109/MS.2017.4541027.

[47] Lebeuf, C., Storey, M.A., Zagalsky, A., 2017a. How software developers mitigate collaboration friction with chatbots URL: http://arxiv.org/abs/1702.07011, arXiv:1702.07011.

[48] Lebeuf, C., Storey, M.A., Zagalsky, A., 2017b. Software Bots. IEEE Software 35, 18–23. doi:10.1109/MS.2017.4541027.

[49] Lebeuf, C.R., 2018. A taxonomy of software bots: towards a deeper understanding of software bot characteristics. Ph.D. thesis.

[50] Levenshtein, V., 1966. Binary codes capable of correcting deletions insertions and reversals. Soviet Physics Doklady 10, 707–710.

[51] Lin, B., Zagalsky, A., Storey, M.A., Serebrenik, A., 2016. Why developers are slacking off: Understanding how software teams use slack. ACM Conference on Computer Supported Cooperative Work (CSCW) , 333–336doi:10.1145/2818052.2869117.

[52] Lin, C.t., Ma, S.P., Huang, Y.W., 2020. MSABot : A chatbot framework for assisting in the development and operation of microservice-based systems , 36–40doi:10.1145/3387940.3391501.

[53] Liu, S., Gao, C., Chen, S., Nie, L.Y., Liu, Y., 2019. ATOM: Commit message generation based on abstract syntax tree and hybrid ranking 14, 1–14. URL: http://arxiv.org/abs/1912.02972, arXiv:1912.02972.

[54] Mann, H.B., Whitney, D.R., 1947. On a test of whether one of two random variables is stochastically larger than the other. The annals of mathematical statistics , 50–60doi:10.2307/2236101.

[55] McHugh, M.L., 2012. Interrater reliability: the kappa statistic. Biochemia medica 22, 276–282. URL: https://pubmed.ncbi.nlm.nih.gov/23092060https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3900052/.

[56] Minnich, A., Chavoshi, N., Koutra, D., Mueen, A., 2017. Botwalk: Efficient adaptive exploration of twitter bot networks, in: IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 467–474.

[57] Mirhosseini, S., Parnin, C., 2017. Can automated pull requests encourage software developers to upgrade out-of-date dependencies?, in: International Conference on Automated Software Engineering ({ASE}), pp. 84–94. doi:10.1109/ASE.2017.8115621.

[58] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12, 2825–2830.

[59] Perez-Soler, S., Guerra, E., De Lara, J., Jurado, F., 2017. The rise of the (modelling) bots: Towards assisted modelling via social networks. IEEE/ACM International Conference on Automated Software Engineering (ASE) , 723–728doi:10.1109/ASE.2017.8115683.

[60] Rebai, S., Ben Sghaier, O., Alizadeh, V., Kessentini, M., Chater, M., 2019. Interactive refactoring documentation bot, in: International Working Conference on Source Code Analysis and Manipulation (SCAM), IEEE. pp. 152–162. doi:10.1109/SCAM.2019.00026.

[61] Rodríguez-Ruiz, J., Mata-Sánchez, J.I., Monroy, R., Loyola-González, O., López-Cuevas, A., 2020. A one-class classification approach for bot detection on Twitter. Computers & Security 91, 101715. doi:10.1016/j.cose.2020.101715.

[62] Romano, J., Kromrey, J.D., Coraggio, J., Skowronek, J., Devine, L., 2006. Exploring methods for evaluating group differences on the NSSE and other surveys: Are the t-test and Cohen's d indices the most appropriate choices?, in: Annual Meeting of the Southern Association for Institutional Research.

[63] Romero, R., Parra, E., 2020. Experiences building an answer bot for Gitter. International Workshop on Bots in Software Engineering (BotSE) , 66–70doi:10.1145/3387940.3391505.

[64] Safavian, S., Landgrebe, D., 1991. A survey of decision tree classifier methodology. IEEE Transactions on Systems, Man and Cybernetics 21, 660–674. doi:10.1109/21.97458.

[65] Storey, M.A., Zagalsky, A., 2016. Disrupting developer productivity one Bot at a Time. ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE) , 928–931doi:10.1145/2950290.2983989.

[66] Thomas, N.T., 2016. An e-business chatbot using AIML and LSA, in: International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 2740–2742.

[67] Tsay, J., Dabbish, L., Herbsleb, J., 2014a. Influence of social and technical factors for evaluating contribution in GitHub, in: International Conference on Software Engineering (ICSE), ACM. pp. 356–366. doi:10.1145/2568225.2568315.

[68] Tsay, J., Dabbish, L., Herbsleb, J., 2014b. Let's talk about it: Evaluating contributions through discussion in GitHub, in: ACM SIGSOFT Symposium on Foundations of Software Engineering (FSE), ACM. pp. 144–154. doi:10.1145/2635868.2635882.

[69] Turing, A.M., 1950. Computing machinery and intelligence. Mind LIX, 433–460. doi:10.1093/mind/LIX.236.433.

[70] Urli, S., Yu, Z., Seinturier, L., Monperrus, M., 2018. How to design a program repair bot?: Insights from the repairnator project. International Conference on Software Engineering (ICSE) , 95–104doi:10.1145/3183519.3183540.

[71] Wessel, M., De Souza, B.M., Steinmacher, I., Wiese, I.S., Polato, I., Chaves, A.P., Gerosa, M.A., 2018. The power of bots: Understanding bots in OSS projects. The ACM International Conference on Human-Computer Interaction doi:10.1145/3274451.

[72] Wessel, M., Steinmacher, I., 2020. The inconvenient side of software bots on pull requests, in: International Workshop on Bots in Software Engineering (BotSE). doi:10.1145/3387940.3391504.

[73] Witten, I.H., Frank, E., Hall, M.A., 2011. Data Mining: Practical Machine Learning Tools and Techniques. 3rd ed., Morgan Kaufmann Publishers Inc.

[74] Wohlin, C., Runeson, P., Hst, M., Ohlsson, M.C., Regnell, B., Wessln, A., 2012. Experimentation in Software Engineering. Springer.