Detecting and Characterizing Bots that Commit Code

Tapajit Dey
Sara Mousavi
The University of Tennessee
Knoxville, TN, USA
tdey2@vols.utk.edu
mousavi@vols.utk.edu

Eduardo Ponce
Tanner Fry
The University of Tennessee
Knoxville, TN, USA
eponcemo@utk.edu
tfry2@vols.utk.edu

Bogdan Vasilescu
Carnegie Mellon University
Pittsburgh, PA, USA
vasilescu@cmu.edu

Anna Filippova
Github
San Francisco, CA, USA
annafil@github.com

Audris Mockus
The University of Tennessee
Knoxville, TN, USA
audris@utk.edu

ABSTRACT

Background: Some developer activity traditionally performed manually, such as making code commits, opening, managing, or closing issues is increasingly subject to automation in many OSS projects. Specifically, such activity is often performed by tools that react to events or run at specific times. We refer to such automation tools as bots and, in many software mining scenarios related to developer productivity or code quality it is desirable to identify bots in order to separate their actions from actions of individuals. Aim: Find an automated way of identifying bots and code committed by these bots, and to characterize the types of bots based on their activity patterns. Method and Result: We propose **BIMAN**, a systematic approach to detect bots using author names, commit messages, files modified by the commit, and projects associated with the commits. For our test data, the value for AUC-ROC was 0.9. We also characterized these bots based on the time patterns of their code commits and the types of files modified, and found that they primarily work with documentation files and web pages, and these files are most prevalent in HTML and JavaScript ecosystems. We have compiled a shareable dataset containing detailed information about 461 bots we found (all of whom have more than 1000 commits) and 13,762,430 commits they created.

KEYWORDS

Bots, Code Commits, Bot Characterization, Random Forest, Heuristics

ACM Reference Format:
Tapajit Dey, Sara Mousavi, Eduardo Ponce, Tanner Fry, Bogdan Vasilescu, Anna Filippova, and Audris Mockus. 2020. Detecting and Characterizing Bots that Commit Code. In *Proceedings of Mining Software Repositories (MSR'20).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Bot is a category assigned to software applications that perform automated tasks based on a predefined set of instructions, and either run continuously or are triggered by events associated with environment interactions, time conditions, or manual execution. Examples of applications that can function as bots are configuration scripts, data generation tools, activity loggers, web crawlers, persistent storage tools, and chat bots. Bots are used by a large number of active individual software developers, teams, and software companies to do various, often repetitive, tasks [13, 19], because bots can do those tasks more efficiently than human users.

In social-coding platforms [9], e.g. GitHub and BitBucket, a number of bots regularly create code commits, issues, and pull requests. However, detecting a bot is a challenging task because on the surface there is no apparent difference between the activity of a bot and that of a human. Moreover, the message structure, message content, and linguistic style of a code commit created by a bot might look very similar to a commit created by a human user. While there are a number of well-known and active bots, such as Dependabot[1] and Greenkeeper[2], not all bots are as popular and as easily recognizable, as we reveal in our work.

Our review of the existing literature did not reveal any systematic approach for determining whether a given user in a social-coding platform is a bot. Therefore, in this work, we propose **BIMAN** — *Bot Identification by commit Message, commit Association, and author Name* — a novel technique to detect bots that commit code in revision control systems. **BIMAN** is comprised of three methods that consider independent aspects of a collection of commits pertaining to a particular author: 1)*Commit Message:* Identify commit messages being generated from templates; 2)*Commit Association:* Predict if an author is a bot using a Random Forest model, with features associated with commit files, projects, and activities as predictors, and ; and 3) *Author Name:* Match author's name and email to common bot patterns.

We applied **BIMAN** on the *World of Code* dataset [21], which has a collection of more than 34 million authors who have committed code in git, along with detailed information on over 1.6 billion commits made by these authors. A **dataset** was compiled with

[1]https://dependabot.com/
[2]https://greenkeeper.io/

information about 461 bots, detected by **BIMAN** and manually verified as bots, and each with more than 1000 commits, along with detailed information about 13,762,430 commits made by these bots; which is available at https://zenodo.org/record/3694401 [11] Our results provide an estimate of the prevalence of code-commit bots in open-source, and could be used to estimate the amount of bot generated commits and code on social-coding platforms.

We also aim to characterize the bots found using **BIMAN** based on their activity, i.e. the type of files they modify, the time of the day do they create the commits etc. The activity patterns of these bots can give us insights into what type of work they do, which ecosystems they work with, etc. The result of the characterization include four different categories for the bots based on the pattern of their activity over the 24 hours of a day. In addition, we identified the type of files commonly edited by bots, and the relative prevalence of the bots in different software ecosystems.

In summary, we make the following contributions in this paper: **1) BIMAN**, a generalizable technique comprising three different methods for identifying if a given author is a bot; **2)** Characterization of bots based on their activity patterns; and **3)** A labeled dataset comprised of 461 bots with 13,762,430 commits We hope our efforts will be useful in enabling further research about various bots used in software development.

The rest of the paper is organized as follows: We discuss the motivation for our work and the specific research questions addressed in this paper in Section 2. In Section 3, we discuss the related works in the topic. In Section 4, we discuss the Methodology, focusing on the proposed method for detecting bots and the bot characterization process. In Section 5, we describe the results we found pertaining to our research questions. Finally, we discuss the limitations of the current version of our work and the possible future works in Section 6 and conclude the paper in Section 7.

2 MOTIVATION AND RESEARCH QUESTIONS

The main motivation behind our effort to try to detect bots is twofold: 1) Data cleaning: due to their automated nature, bots can significantly affect the estimates of team size, the amount of activity, and developer productivity, which can threaten the validity of such measures and any decision based on such measures; and 2) Research: enabling further research into bots.

Many software researchers look at the activity of software developers for understanding their culture and behavior [3, 10], estimating the team size [7], productivity [35] etc., and also for studying developer interaction based phenomena like knowledge flow within a software project [18, 22], prediction of build failures [34] etc. While conducting such studies, it is important to account for the bots in the set of developers being studied, because bots typically have a different activity pattern than humans and it may generate physically impossible metrics of activity or productivity, or could at least significantly bias these estimates. Furthermore, the desire to stand out can lead to creation of extreme numbers of files or commits that has to be automated (e.g. Github user *one-million-repo*[3] has 1,102,551 commits, and the repository *biggest-repo-ever* [4] has 9,960,000 commits).

[3]https://github.com/one-million-repo

[4]https://github.com/one-million-repo/biggest-repo-ever

However, the first step in adopting any data cleaning scheme to mitigate the effect of bots in Software Engineering research is finding the bots, and, as mentioned earlier, we found no systematic approach for that. Therefore, the first research question we address is in this paper is:

RQ1: How can we determine if a particular user is a bot?

A logical follow-up to this research question is characterizing the bots found. Earlier studies proposed different methodologies for bot characterization by looking into their design and construction [12] and their intrinsic properties and interaction patterns [20] (discussed in detail in Section 3), while we strive to characterize the bots we found by looking into their activity, i.e. what type of files they modify, at what time of the day do they create the commits etc. In contrast to the existing taxonomy for the bots, which are generated using a theoretical setting, e.g. Erlenhov et. al. used faceted analysis [12], and Lebeuf [20] used the taxonomy generation method proposed by [31], we adopt a data intensive technique to characterize the bots found, since for most of the bots discovered using **BIMAN**, we had little information about them other than what we can observe from their commit activity.

The general assumption about bots is that they primarily take care of the more tedious works, and we wanted to investigate the veracity of that conjecture. In addition, we also want to estimate the prevalence of the bots in different software ecosystems, which can be useful in understanding the differences and similarities between different ecosystems, and can also highlight the areas where there is a scope for bot adoption. Therefore, we present our second research question as:

RQ2: What type of work do bots do and which ecosystems are they active in?

3 RELATED WORKS

The idea of "bots", or software applications that can imitate a human's activity and behavior, dates back to 1950 with Alan Turing asking the question "Can machines think?" [2]. Recent advancements in Artificial Intelligence, especially Natural Language Processing and Machine Learning have led to a proliferation of bots across domains, such as personal assistants (Apple's Siri [33], Google Assistant [28], Amazon Alexa[5], etc.), in education [5, 17], e-commerce [30], customer service [15], and social media platforms [1].

OSS communities, and software engineering in general, use bots primarily to reduce the workload of repetitive tasks. Wessel et. al. [32] studied 351 GitHub projects with more than 2,500 stars and found that 26% of them use bots, and the adoption of bots started rising since 2013. In software engineering, bots support activities like communication and decision making [29], and automate many of the tasks that would have required human interaction in collaborative software development platforms [19]. A number of studies have shown how bots can be used by software developers for tasks like automated communication among the developers [25] and automated deployment and evaluation of software [6].

However, while these studies highlight how bots can be used and how prevalent bot adoption is in various popular OSS projects, they do not present any generalizable way to detect the bots that

[5]https://developer.amazon.com/en-US/alexa

are already present. Wessel et. al. [32] looked into the the GitHub account of a suspected bot and checked if it is tagged as a bot. They also examined the pull request messages, looking for obvious messages like: "This is an automated pull request to...". Erlenhov et. al. [12] analyzed 11 well-known bots, and Lebeuf [20] looked at 3 well-known bots, and neither suggested any method of detecting if a given user is a bot.

In terms of bot characterization, Lebeuf [20] proposed characterizing the bots by analyzing 22 facets organized into 3 dimensions: Environmental (where the bot operates), Intrinsic (internal properties of the bot), and Interaction (how the bot interacts with its environment). Erlenhov et. al. [12] focused on the "DevBots", i.e. the bots that support software development, and proposed a taxonomy comprising 4 facets: Purpose (general vs. specialized), Initiation (triggered by users or system or both), Communication (how the bot communicates with other users), and Intelligence (adaptive or static).

4 METHODOLOGY

In this section, we describe the data used for analysis, present our proposed method for detecting bots, and describe how we characterize the bots found.

4.1 Data

The data used for this study was obtained from the World of Code (WoC) [21] dataset. The version of the data we used (version P) was collected between May 15,2019 and June 5,2019 based on updates/new repositories identified on May 15, 2019 and it contained information about 73,314,320 unique non-forked git repositories, 34,424,362 unique author IDs, and 1,685,985,529 commits.

The author IDs in the dataset were represented by a combination of the authors name and email address, in the following format: `first-name last-name<email-address>`. E.g. for an author whose first name is "John", last name is "Doe", and email address is "myemail@me.com", the corresponding author id in the WoC dataset would be: "`John Doe<myemail@me.com>`".

The data is stored in the form of mappings between various git objects. The mappings we used for our study include the mappings between the commit authors and commits (*a2c*), commits and filenames (including the file path) changed by that commit(*c2f*), commits and the GitHub projects that commit is associated with (*c2p*), and commits and the contents of the commits comprising the commit timestamp, timezone, and commit message (*c2cc*).

Our method of extracting information about the authors comprised the following steps:

(1) Obtaining a list of all authors from the WoC dataset.
(2) Identifying all the commits for the authors using the *a2c* map.
(3) Extracting the list of files modified, the list of projects the commit is associated with, and the commit content for each of the commits for each author using the *c2f, c2p*, and *c2cc* maps respectively.

4.2 Proposed method for identifying bots

BIMAN, our proposed method for identifying bots, consists of three approaches, 1) Bot Identification by Name (**BIN**), 2) Bot Identification by commit Message (**BIM**), and 3) Bot Identification by Commit Association (**BICA**), that rely on three different datasets. Since the data required for each approach is independent of each other, these approaches can also be used independently. An overview of **BIMAN** method is illustrated in Figure 1. We have proposed the three different approaches separately, instead of a single model, because not all of the required data for all three approaches might be available or easily obtainable, and researchers with access to partial data can still use a subset of these approaches.

*4.2.1 Identifying bots by their name (**BIN**):.* We started devising a possible method for detecting bots by first looking for the known bots in our dataset. Erlenhov et.al. [12] studied 11 bots, which we took as a starting point in our investigation. However, since the author names in our dataset consist of name-email combinations, we had to search through the list of authors for identifying the possible author IDs that could be related to one of these 11 bots. We didn't find an entry matching 3 of the 11 bots in our dataset: "First-timers", "Marbot", and "CssRooster", and we found a total of 57 author IDs that could be associated with one of the other bots. We noticed that 25 (37%) of these author IDs had the word "bot" in them. We further looked for other known bots (e.g., Travis CI bots, Jenkins bots) in our dataset and noticed that many of the author IDs we suspected as bots also had the the word "bot" in their names or their email IDs.

Based on these observations, we used regular expressions to identify if an author is a bot by checking if the author name or email matched a known bot pattern. However, to avoid including false positives like "Abbot" or "Botha", we decided to only look for the word "bot" preceded and followed by non-alpha characters. We further excluded author IDs that had the word "bot" only in the domain name of their email addresses (e.g., hr@future-*bot*.ai), since we are not convinced that these are always bots. Although matching regular expressions does not detects all bots, nor it is be able to filter authors trying to disguise themselves as bots, it is a naive approach that doesn't require any other data to run, and can be regarded as a good starting point.

Benchmark Creation: We did not have a *golden dataset* when we started applying our method for training any machine learning model. However, we noted that the name based bot identification approach was very precise, it had few false positives. Therefore, we used this method to find a list of bots to be included in the initial dataset. Two independent raters analyzed author IDs, descriptions, and commit and pull request messages to manually verify the bots previously identified. Cases where the raters were highly sure that an author was not a bot (25 entries) were removed. There were a few ambiguous cases, where the raters were not able to confirm or deny if it was actually a bot, but it was less than 1%, so we were satisfied with this initial dataset. We found a total of 13,150 authors that were suspected to be bots. After we created a set of bot authors, we needed a set of human authors to complete our training dataset. To do so, we randomly selected 13,150 authors from the rest of the authors, and again followed the same process of manually ensuring that we do not have any bots in this list. This was our

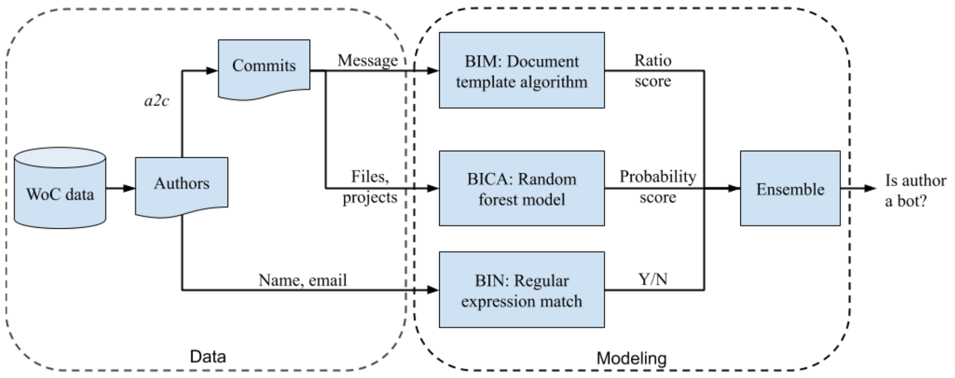

Figure 1: BIMAN workflow: Commit data pertaining to authors is used for message template detection, activity pattern based predictions using a Random Forest model, and name pattern matching. Resultant scores from each approach is used by an ensemble model (another Random Forest model) that classifies the given author as a bot or not bot.

golden dataset, consisting of 26,300 authors, that we used to train the Random Forest model (commit association based bot detection approach), and evaluate the performance of **BIMAN** approach.

Comparing the commit activities of humans and bots: Our initial assumption was that the bots are very active users, and would have a significantly greater number of commits than the humans, therefore, we can detect bots simply by checking how many commits they have made. However, upon investigation of the 13,150 bots in our golden dataset, we found that assumption is incorrect. While the maximum number of commits by a bot was admittedly huge (2, 463, 758), the median number of commits by the bots was just 2, and the first and third quantile values were 1 and 16, respectively. In contrast, the median number of commits by a human author in our initial dataset was 4, and the first and third quantile values were 2 and 17, respectively. So, the number of commits didn't turn out to be very different between the humans and the bots. We hypothesize that the reason behind why a lot of the bots have so few commits could be one or more of the following reasons: (1) since our author id variables consist of a name-email combination, slight variations in either appear as different authors, when they are actually the same author, e.g. a "dotnet-bot" has three variations in name that appear as different authors: `beep boop`, `Beep boop`, and `Beep Boop`, though it has the the same email address— `dotnet-bot@microsoft.com`, and we need to employ anti-aliasing methods [4] to address these issues; (2) some bots might have been implemented as an experiment or part of some coursework, and never used afterwards, e.g. we found a bot named "learn.chat.bot" that likely belongs to this category; (3) some bots might heave been designed for a project, but was never fully adopted due to various issues.

Therefore, we can't identify bots simply by filtering how active a commit author is, we need more complex models for doing that.

4.2.2 *Detecting bots by commit messages (**BIM**):.*

Characteristics of commit messages can be used to identify an author as a bot. Given that bots that commit code are automation tools that generate commit messages, we consider that many bots routinely make use of templetized messages as the starting point for the final commit message. Thus, detection of templetized commit messages can be used to identify such type of bots. Although humans can generate commit messages with similar and consistent patterns (e.g., follow a set of software development guidelines), the key assumption **BIM** follows is that for a large number of commit messages, the variability of messages' content generated by bots is lower than those generated by humans.

BIM utilizes the document template score algorithm presented in Alg. 1. Given a set of documents, the procedure `TemplateScore` compares document pairs and uses a similarity measure to group documents. The `Similarity` procedure represents an arbitrary method that computes a "similarity" measure that is of interest for the application [8, 14, 16, 24]. A group represents documents that are suspected to conform to a similar base document. Each document group has a single *template document* assigned and this is the document always used for comparisons. A new group is created when a document's similarity with any *template document* does not reaches the similarity threshold, k_b, and this document is set as the *template document* for that group. After all documents are compared, the document template score is calculated based on the ratio of the number of *template documents* and the number of documents, $1 - \frac{\|T\|}{\|D\|}$, where T is the set of *template documents* and D is the set of documents.

In **BIM**, commit messages of a particular author correspond to the documents and the percent identity of aligned commit messages serves as the similarity measure. Commit messages were aligned and scored using a combination of global (Needleman-Wunsch [23]) and local (Smith-Waterman [27]) sequence alignment algorithms available via the Python `alignment`[6] library. The similarity threshold, k_b, was set to 40% after testing accuracy of Alg. 1 on the golden data using thresholds of 40, 50, 60, and 70%. The greater the reported score by **BIM**, the more likely the author is a bot that makes use of template-based message generation.

4.2.3 *Bot Identification by Files changed and Projects associated with commits (commit association) (**BICA**):.*

In addition to the commit message, we had access to other metadata about the commits, specifically, the files changed by each commit, the projects that commit is associated with, and the timestamp and timezone of that

[6]https://pypi.org/project/alignment

Algorithm 1 Document template score

```
global: similarity threshold for template comparisons $k_b$
inputs: collection of documents $D$
output: ratio of number of templates and number of documents
1: procedure TemplateScore($D$)
2:     $T \leftarrow \emptyset$                ▷ collection of template documents
3:     $G \leftarrow \{\emptyset\}$  ▷ collection of template groups, $G_i$ is the group associated to template document $i$
4:     for $d \in D$ do
5:         for $t \in T$ and $d \notin G$ do
6:             if Similarity($d, t$) $> k_b$ then
7:                 Add $d$ to $G_t$
8:             end if
9:         end for
10:        if $d \notin G$ then
11:            Add $d$ to $T$
12:            Add $d$ to $G_d$
13:        end if
14:    end for
15:    return $1 - \frac{\|T\|}{\|D\|}$
16: end procedure
```

Table 1: Predictors used in the Random Forest model

Variable Name	Variable Description
Tot.FilesChanged	Total number of files changed by the author in all their commits (including duplicates)
Uniq.File.Exten	Total number of different file extension in all the author's commits
Std.File.pCommit	Std. Deviation of the number of files per commit
Avg.File.pCommit	Mean number of files per Commit
Tot.uniq.Projects	Total number of different project the author's commits have been associated with
Median.Project.pCommit	Median number of project the author's commits have been associated with (with duplicates); we took the median value, because the distribution of projects per commit was very skewed, and the mean was heavily influenced by the maximum value.

commit. We calculated 20 numerical metrics using those measures based on our initial assumptions about how bots and humans might be different, and empirical validation of the assumption by observing the differences in distribution of those variables for bots and humans.

For predicting whether an author is a bot using the numerical features, we tested several modeling approaches: Linear and Logistic Regression, Generalized Additive Models, Support Vector Machines, and Random Forest. The Random Forest (RF) model performed better than the other approaches, so we decided to use that approach. We used the Random Forest implementation available in the "randomForest" package in R, with these 20 variables as predictors, to predict if the author of those commits is a bot. After iteratively selecting and removing predictors based on their importance in the model, and measuring the AUC-ROC every time, we found that a model with only 6 predictors as the optimal model. The list of predictors is given in Table 1, along with the description of each variable. We found that the timestamp of a commit and any time related measure, e.g. how long the bot has been active, at what times of the day they make commits etc. are not important predictors.

In order to tune the RF model, we used the "train" function from the *caret* package in R for performing a grid search (using a 10 fold cross-validation) on the training data to find the optimal values of the model parameters: "ntree" (number of trees to grow) and "mtry" (number of variables randomly sampled as candidates at each split), that gives the highest Accuracy. The optimum value for "ntree" was found to be 100, and that of "mtry" was found to be 2.

4.2.4 Ensemble model: Since each of our approaches looked at very different aspects of the authors and the commits they create, we decided to have an ensemble model, implemented as another Random Forest model, with the outputs of the three different approaches as predictors, to make a final judgement as to whether an author is a bot or not. The output from the name based approach was a binary value (1: bot, 0: human), saying if the author id matches the regular expression we checked against; the output from the commit message based detection approach is (1 - the ratio of number of templates and the total number of commit messages), since otherwise a higher value would mean the author is more likely to be human, which is inconsistent with the other outputs; and the output from the RF model is the probability that one author is a bot.

The results from these approaches were aggregated using a Random Forest model to generate the final output. However, since our golden dataset was generated using the **BIN** approach, the result of which was one of the inputs to the ensemble model, we could not use it for training this model. Instead, we created another training data with the 57 IDs associated with the 8 bots described in [12], along with 10 IDs associated with 3 other known bots, *Scala Steward, codacy-badger,* and *fossabot*,that were not in our golden dataset, as the bot instances (67 in total); and 67 randomly selected and manually validated instances of human authors as the human instances. The final training data for training the ensemble model thus had only 134 observations, however, since we had only 3 predictors, we were reasonably satisfied with this data.

4.3 Characterization of the Bots

Instead of trying to design a taxonomy of classification of the bots, as was done in previous studies [12, 20], we aim to group the bots by their activity patterns, specifically, the files these bots modify and at what time of the day they make commits, because we did not have enough information about most of the bots to use a more rigorous taxonomy. The characterization was applied on the 13,150 bots in our golden dataset. We specifically looked at how the number of commits made by a bot are distributed over the 24 hours of a day. That gives us a good indication of what type of work a bot might be doing. We also looked at which ecosystems different bots are active in, and what type of files they modify.

4.3.1 *Characterization of bots by activity hours.*

By observing the distribution of commits created by a bot over the 24 hours of a day using the time-of-day histograms for number of commits, for a randomly selected sample of 50 bots identified by **BIMAN**, and employing a qualitative analysis technique similar to card sorting, we identified three distinct patterns: 1) Some bots were active almost uniformly over the 24 hours of a day, or in some cases they had no activity over up to 4 hours and almost uniform activity during the rest of the day; 2) Some bots' activity patterns resembled the typical activity patterns of humans very closely, i.e. they were more active during the "working hours", with a few contiguous peak activity hours, and they had limited activity over the rest of the day; 3) Some bots were active only for a few specific hours, and had almost no activity during the rest of the day. We decided to call the three types of bots "Continuous Activity Bots", "Synchronous Activity Bots" (since their activity synchronizes well the typical human users' activity), and "Spike Activity Bots" respectively. There were some bots, however, whose activity pattern didn't follow any of the three patterns described above, so we decided to call them "Other Bots".

We decided to apply this characterization on only the "active" bots, because bots with very few commits would almost always follow the "Spike Activity Bots" or "Other Bots" pattern. Moreover, we were more interested in the "active" bots since they have a greater influence over the projects and ecosystems they are active in, as they have more code contribution than the rest. We designated the bots with more than 1000 commits as "active" bots, and we found 454 (3%) bots out of our dataset of 13,150 bots match that criteria. For the characterization process, we had three independent experts classify the bots based on their commit distribution over the hours of day, and a fourth independent expert compiled the results into a final classification for the bots.

4.3.2 *Characterization of bots by files modified.*

We were also interested to see which ecosystems these bots worked in and what types of files they modified, and we had information on all the commits made by all the 13,150 bots in our golden dataset, and the files modified by each commit. We extracted the file extensions from those files, and used the *linguist* [7] tool to obtain an estimated language classification based on a common open-source model, and used this information to infer which ecosystems a bot worked with.

5 RESULTS

5.1 Qualitative Validation of BIMAN

Before going into the detailed performance evaluation of **BIMAN**, we wanted to test how it performs in detecting a few known bots. As mentioned in Section 4.2.1, we obtained a set of 57 author IDs associated with 8 of the bots described in [12]. In addition, we examined 10 author IDs associated with 3 well-known bots, *Scala Steward, codacy-badger,* and *fossabot*, that were not in our golden dataset. The performance of each of the three approaches of **BIMAN** in identifying these bots is shown in Table 2.

We found that at least one of our approaches were able to identify 60 (87%) out the 69 author IDs as bots. We were able to confirm that 6 out the 9 IDs not recognized as a bot by our three approaches weren't actually associated with the bot, they were either spoofing the name or simply using the same name. The 3 other IDs, 2 of whom were associated with "Deploybot [8] [9]", and the other with "Imgbot [10]", had 1 commit each, and we could not confirm whether they were actually bots or not even with manual investigation.

We did not list the performance of the final ensemble model in Table 2, because that model was trained using this data, as mentioned in Section 4.2.4.

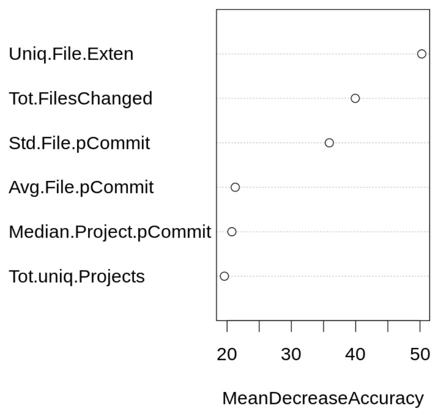

Figure 2: Variable Importance plot for the Random Forest model used to identify bots

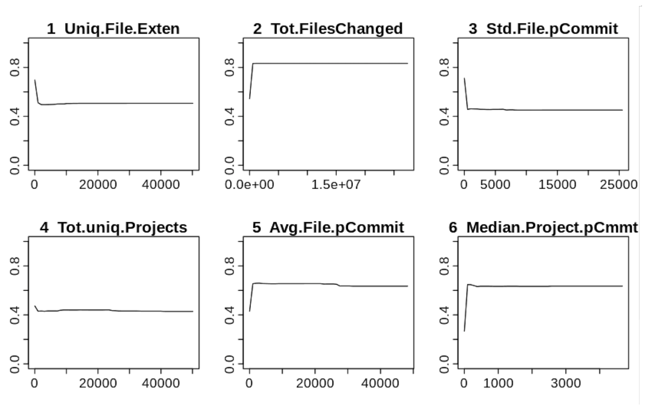

Figure 3: Partial dependence plot for all the predictors in the Random Forest model

5.2 Performance Evaluation of BIMAN

In this section, we discuss the performance of **BIMAN**, our proposed method of bot identification. As mentioned in Section 4.2, **BIMAN** is comprised of three independent approaches, each looking at a different aspect of the commit authors and the commits, and we use an ensemble model for getting the final prediction by combining the results from the three approaches. Since the three approaches are independent, we decided to give a performance evaluation for them separately and discuss what we learnt by using each approach about the bots that commit code.

[7] https://github.com/github/linguist

[8] deploybot-lm <45803032+deploybot-lm@users.noreply.github.com>

[9] DeployBot <deploybot@imqs.co.za>

[10] imgbot<imgbothelp@gmail.com>

Table 2: Performance of the models in detecting 8 known bots from [12] and 3 other known bots outside our golden dataset

Bot	No. of author IDs associated with the bot	No. of IDs identified as bot by BIN approach	No. of IDs identified as bot by BICA approach	No. of IDs identified as bot by BIM approach	No. of IDs not identified as bot by BIMAN
Dependabot	4	4	4	2	0
Greenkeeper	15	10	13	11	2
Spotbot	1	0	1	1	0
Imgbot	5	1	4	3	1
Deploybot	29	9	20	17	6
Repairnator	1	0	0	1	0
Mary-Poppins	1	0	1	1	0
Typot	1	1	0	1	0
Scala Steward	6	0	6	6	0
codacy-badger	2	0	2	2	0
fossabot	2	0	2	2	0
Total	**67**	**25 (37%)**	**53 (79%)**	**47 (70%)**	**9 (13%)**

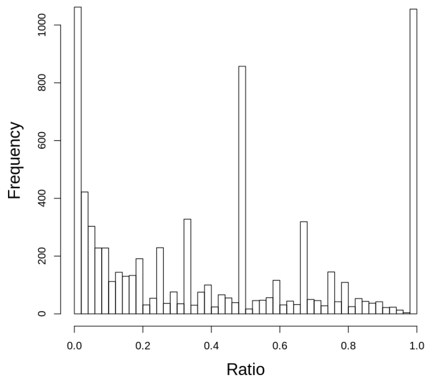

Figure 4: Ratio of number of detected templates and the number of commit messages for the 13,150 bots in our dataset

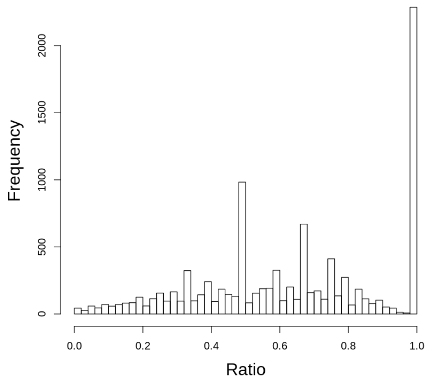

Figure 5: Ratio of number of detected templates and the number of commit messages for the 13,150 humans (non-bots) in our dataset

*5.2.1 Performance of the **BIN** approach:* Since our golden dataset was constructed using the **BIN** approach (see Section 4.2.1), we did not use that dataset to validate the accuracy of this approach. However, as we noticed during creation of the golden dataset, this method has a precision of around 99%, i.e. any author found to a bot using this method has a very high chance of being a bot, which is likely because humans do not generally try to disguise themselves as a bot. However, the recall is not very high, since it can and does miss a lot of cases where the bots do not explicitly have the word "bot" in their name. As mentioned in Section 5.1, we observed an estimated recall of 37% when we applied this method on the set of 69 bot IDs we manually investigated.

*5.2.2 Performance of the **BICA** approach.* As mentioned in Section 4.2.3, the **BICA** approach uses a Random Forest model with the measures listed in Table 1 as predictors for identifying bots. We used the golden dataset generated using the **BIN** method (Section 4.2.1) for training the model and testing its performance. 70% of the data, selected randomly, was used for training the model and the rest 30% was used for testing it, and the procedure was repeated 100 times with different random seeds.

The model showed a good performance, with an AUC-ROC value of 0.89. The variable importance plot (Figure 2) showed that the total number of unique file extensions and the total number of files changed in all the commits made by an author are the most important variables.

To understand what each of the predictors tell us about how the behavior of the bots differ from that of humans, we looked at their partial dependence plots, as shown in Figure 3. The greater values in the Y-axis of each plot correspond to a higher probability of an author being a bot, and the values in the X-axis are the possible values of each of the predictors. These plots give us an empirical understanding about how the behavior of the bots is different from humans. We notice the bots tend to have fewer number of unique file extensions, and their commits are associated with fewer number of different projects, i.e. they tend to stick to one ecosystem. However, their commits tend to be be associated with more number of projects per commit, i.e. the projects they commit to are more popular. The bots typically make larger commits, as we notice that they tend to have more files per commit on an average, and also more number of total files changed. The bots are also more consistent in terms of commit size, since the variation in the number of files per commit is lower. These observations fall in line with our idea of typical bots, who keep updating a consistent set of files, and are typically a part of the more popular projects.

*5.2.3 Performance of the **BIM** method:* Our proposed method of identifying whether an author is a bot by detecting if the commit messages by the author were generated using a template relied solely on the commit messages. We calculated the ratio of the number of templates and the number of commit messages for the bots (shown in Figure 4), and for the humans (shown in Figure 5) in our golden dataset. We can observe that the ratio tends to lower for most of the bots. The reason for a number of bots having a high ratio is that if an author has only 1 commit message, the ratio is bound to be 1, and over 25% of the bots in our golden dataset have only 1 commit.

We wanted to find an optimal threshold for the ratio, so that the authors, for whom the ratio is lower than the threshold, can be regarded as bots (the output from the **BIM** method is (1−this ratio), so for that, a lower value is more likely to be human and vice versa). This information would be useful for researchers who might only use this technique to detect whether a given author is a bot, and this also helps us calculate the performance of this approach. The optimal threshold was found using the "closest.topleft" optimality criterion (calculated as: $min((1 - sensitivities)^2 + (1 - specificities)^2))$ using the *pROC* [26] package in R. The AUC-ROC value using the ratio values as predicted probabilities was 0.70, and the optimal values for the threshold, sensitivity, and specificity were found to be 0.51, 0.67, and 0.63 respectively.

True Positive: The cases where this model could correctly identify bots were cases where the bots actually used templates or repeated the same commit message, e.g. a bot named "Autobuild bot on Travis-CI" used the same commit message "`update html,`" for all of the 739 commits it made, and a bot named "Common Workflow Language project bot" created 1373 commits that used the form: "`$USER-CODE: $SOFTWARE configuration files updated. Change performed by $NAME`". Our method could identify these messages as coming from the same template and could detect these authors as bots.

False Negative: The cases where this model could not correctly identify bots were mostly cases where the bots reviewed code added by humans and created a commit message that added a few words with the commit message written by a human, e.g. a bot named "Auto Roll Bot" created commits messages in the form of: "`$COMMIT-SEQUENCE-NUMBER: $LONG-HUMAN-COMMIT-TEXT $PATTERN`", with one specific example being "`3602: Fix errors in the Newspeak Mac installer genrators. Fix a slip in platforms/Cross/vm/sqCogStackAlignment.h for the ARM's getsp. Eliminate non-spur and stack VMs from the ARM builds (it builds veerry slowly) Include 64-bit and Mac Pharo VMs in archives and uploads.-------------------------------`", with the length of the "$LONG-HUMAN-COMMIT-TEXT" typically ranging between 20 and 50 words. Our method of template detection could not identify this template and misclassified this author as a human.

True Negative: The human authors correctly identified as so had some variation in the text, with the usual descriptions of change, e.g. messages like "`Added a count down controller`" or "`Enabling multiport deployments. By mapping ports a little bit more specific we get all the servers listed in the server browser`" etc.

False Positive: In contrast, humans who were misclassified as bots usually had short commit messages that were not descriptive, and they reused the same commit message multiple times. Example of typical messages are: "`Initial Commit`", "`Added File by Upload`", "`Updated $FILE`" etc.

Our observations while using this method suggest that this technique is useful in identifying the "typical" bots that modify small parts of a message in every commit and "typical" humans who write descriptive commit messages. However, we can also conclude that it is very hard to identify if an author is a bot using just this one signal.

5.2.4 Ensemble Model: We combined the results of **BIN**, **BIM**, and **BICA** using an ensemble model, implemented as a Random Forest model. The dataset used for training and testing the performance of this model had only 134 observations, because of reasons described in Section 4.2.4. We used 80% of the data for training, and 20% for testing, and repeated the process 100 times with different random seeds. Due to the small size of the training data, the performance of this model had some variation. The value of the AUC-ROC measure varied between 0.89 and 0.95, with a median of 0.90.

*To address RQ1, we devised **BIMAN**, a systematic method of detecting bots using information about their names, commit messages, files modified by the commit, and projects associated with the commits.*

5.3 An Estimate of the number of Bots that commit code

While we can easily obtain the number and activity of author IDs that contain the word "bot", it is much more difficult to determine the total number of author IDs that, from their string representation, can not be inferred to be bots. Yet, even a rough gauge on the prevalence of bots and code committed by bots would be helpful to have a handle on the fraction of code commit activity that is automated. To do that we randomly selected a sample of 10,000 authors IDs outside of our datasets used so far (none had the word "bot" in their names). **BIMAN** predicted 1,167 authors IDs to be bots, and we manually investigated randomly sampled 100 authors IDs among the 1,167 IDs. A subjective assessment of these author IDs and associated commits by two authors of this paper discovered at least nine of these author IDs likely to produce mostly automated commits. From this, we can obtain a rough estimate that approximately $11.67\% * 9\% \approx 1\%$ of all authors IDs who commit code are bots. Therefore, the total population of approximately 40 million author IDs in open source git commits results in approximately 400,000 of these IDs to be bots. The nine author IDs that we identified as bots were found to have created between 10 and 1500 times more commits than the remaining author IDs. Such high discrepancy strengthens our concerns described in Section 2 that the empirical analyses relying on measures of developer productivity mat be strongly biased even if the actual bot population represents a modest 1% of all developer IDs.

5.4 Dataset for Sharing

We have compiled the information about 461 bots detected using **BIMAN**, each of whom have created more than 1000 commits, into a single dataset to make it available for researchers interested in conducting studies on such a data. The dataset includes information about $13,762,430$ commits made by these bots. We decided to focus on the more active bots since these bots would have a much larger effect on any estimate of developer productivity, team size etc. and they are the ones that should be accounted for during any data cleaning process.

The data is stored in a csv file (";" as the separator) with the following format in each line: "author_id"; "commit-sha"; "time-of-the-commit"; "timezone"; "files-modified-by-the-commit"; "projects-the-commit-is-associated-with"; "commit-message". In the case of having multiple projects and files for a given commit, they are separated by ','. The data is available on *Zenodo* through the link provided in Section 1. Additional data about other authors, along with the likelihood of each being a bot will be provided upon requests.

5.5 Bot Characterization

We wanted to characterize the bots based on the time of the day they create commits and which ecosystems they work with.

5.5.1 Characterizing the bots based on their activity during the day: As mentioned in Section 4.3, we classified the bots into 3 categories, and an "others" category for bots that do not fall in any of the 3 categories. After going through the categorization process for the 454 bots with more than 1000 commits, as mentioned in Section 4.3, we had 100 "continuous activity pattern bots", 162 "synchronous activity pattern bots", 128 "spike activity pattern bots", and 64 bots classified as "others".

Here we give one typical example from each category, and describe what type of work they do. This would help in understanding why these bots have the observed activity pattern, and would enable inferring what type of the work a bot does if it falls in one of the observed categories. We also show how their commit activity over the 24 hours of a day are distributed using radial activity plots, where we show what is the relative amount of activity of the bot in a given hour over its lifetime. The hours between 8 a.m. and 4 p.m. are highlighted in each plot, since these hours are known to be the typical working hours. The radial activity plots in Figure 6 are all examples of radial activity plots.

1. Continuous Activity Bots: A continuous activity bot shows almost uniform activity over the 24 hours of a day. A representative example of this class is the "Currency bot [11]", which collects up-to-date exchange rate data every hour and distributes it for free. The radial activity plot for this bot is shown in Figure 6a. Since this bot is active throughout the day, it has a very uniform distribution of activity. Other bots in this category also perform tasks that require them to be similarly active throughout the day.

2. Sync Activity Bots: Sync activity bots typically work in response to the activity of a human, which means their activity during the day closely resemble that of a human. A typical example of the Sync bot class is the "Pure bot [12]", which enables automated pull request workflows, reacting to input from web-hooks and performing actions as configured. The radial activity pattern for this bot, as shown in Figure 6b, resembles the typical activity pattern of humans, because it reacts in response to humans creating pull requests.

3. Spike Activity Bots: These bots perform some action at a fixed time of the day, or at regular intervals. The type of the work they do primarily fall into two categories, backup data and update websites at regular intervals. A prime example of such a bot is the "Paper.js Bot [13]", which automatically regenerates the Paper.js website once per day, and its radial activity plot is shown in Figure 6c.

4. Other Bots: There are a number of bots whose activity pattern do not fall in any of the categories described above, like that of the "Nur a Bot [14]", which is responsible for updating the NUR repository based on community updates and NIX repository updates. Nix is a powerful package manager for Linux and other Unix systems that makes package management reliable and reproducible, and NUR is a community repository where anyone can submit software to be added in. Since this bot does two different types of work, its activity doesn't follow a clear pattern, as can be observed by its radial activity plot (Figure 6d).

Since our effort of categorizing the bots shed light on which specific activities they do, it primarily covers the "Initiation" facet described in [12]. Broadly speaking, we can infer that the bots that show continuous activity or spike like activity, are the "active" bots, which are activated by the system running it. In contrast, the "Synchronous Activity Bots" are "reactive", i.e. they work in response to the activity of a human or another bot. We can't infer anything specific about the "Other" bots with the information we have.

5.5.2 What types of files do bots modify? Understanding what types of files these bots work with can give us more insight into what type of work they do and which ecosystems they are active in. Following the characterization steps as described in Section 4.3, we discovered the types of files modified by the 13,150 bots discovered using the **BIN** method. The type of files modified most frequently by the bots are shown in Figure 7. The Frequency values in the Y-axis of the plots represent how many bots have modified a specific type of file. We notice that the bots mostly work with configuration and documentation related files, along with HTML and JavaScript files.

We also tried to investigate which ecosystems most frequently employ bots, so we took the list of ecosystems examined by Wessel et. al. [32] and measured how many of our 13,150 bots have contributed to one of them. The distribution of bots in different ecosystems is shown in Figure 8. We notice that HTML and JavaScript are the ecosystems that use bots most frequently.

Regarding RQ2, we identified four different types of bots based on their commit activity during the day; and found that bots primarily work with different Configuration and Documentation files, and are most prevalent in the HTML and JavaScript ecosystems

6 LIMITATIONS AND FUTURE WORK

Our approach of detecting bots is a first step towards a challenging task, and there are a number of limitations to our approach and possible scope for improvement.

[11]https://github.com/currencybot
[12]https://github.com/syndesisio/pure-bot
[13]https://github.com/paperjs-bot
[14]https://github.com/nur-bot

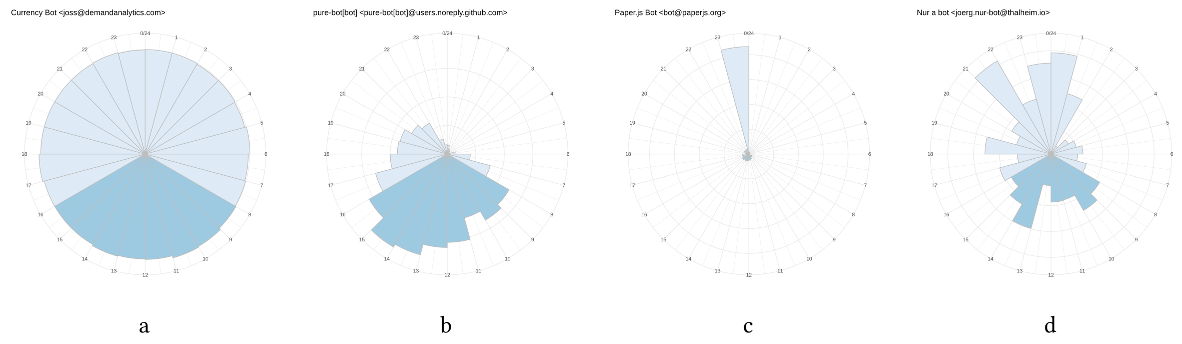

Figure 6: (a): The Currency Bot- A Continuous Activity Bot, (b): The Pure Bot - A Sync Activity Bot, (c):The Paper.js Bot - A Spike Activity Bot, (d):Nur a Bot -An Other Bot

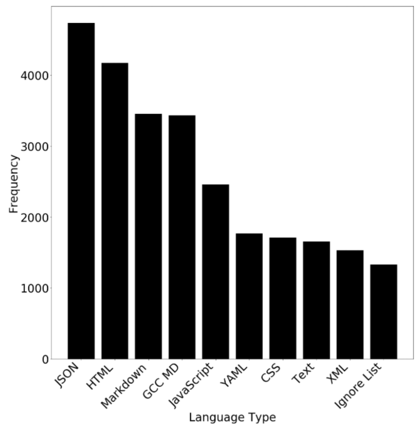

Figure 7: The types of files most frequently modified by bots

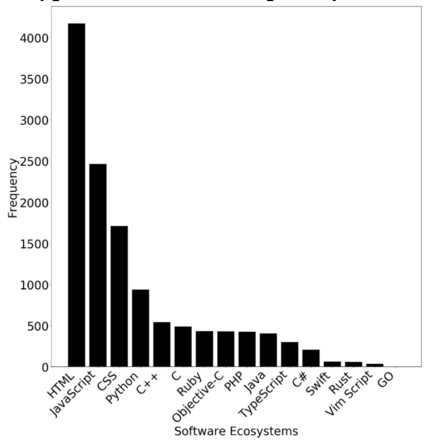

Figure 8: The Languages (Software Ecosystems) Bots contribute to

6.1 Internal Validity

The biggest problem we faced during designing **BIMAN** was the lack of a golden dataset. We only knew about a handful of bots, which is not enough to design an accurate machine learning model. We tackled this problem by creating a dataset with one of the approaches (**BIN**) we proposed and manually validating that. However, the bots found by **BIN** might have a different characteristics than the rest of the bots, which would mean our method might not be able to detect those bots.

Using the dataset generated by **BIN** as a golden dataset also means that we were not able to estimate the recall of **BIN** with it. Instead, we had to use a much smaller dataset to estimate its recall. Similarly, our final ensemble model was also trained with this smaller dataset, which led to some variation in its performance (AUC-ROC value varied between 0.89 and 0.95).

Another threat to the effectiveness of our method is that a number of developers use automated scripts to handle some of their works, which uses their git credentials while making commits. This is a tricky challenge for our method, since the signal from those authors appears mixed; and depending on what fraction of commits made using that author's ID is made by the bots, our method might or might not be able to detect such authors as bots. Similarly, a few organizational IDs are sometimes used by bots as well as humans, and we have a similar issue regarding those IDs as well.

Moreover, since an estimated 1% of the commit authors were found to be bots (Section 5.3), an author detected as a bot by a 90% accurate model has only only 8.3% chance of actually being a bot (using Bayes Theorem), i.e. we are bound to have a lot of False Positives.

6.2 Construct Validity

The construct validity threats primarily apply to the **BIM** approach we used, since it was designed with specific ideas about how a bot might work. **BIM** focuses on identifying bots that authored all the commit messages it is associated with, independent of whether they were generated by a template-based approach or not. However, many developers make use of bots for generating certain commit messages (that uses the the same ID) and this hybrid classification is not addressed in this work. The main factors that give rise to limitations are the content of commit messages, number of commit messages per author, and performance of similarity measure.

The performance of **BIM** depends primarily on the performance of the similarity measure used to compare commit messages. Commit messages tend to be concise, making it difficult to extract content characteristics (structural or semantic) that are useful for text similarity metrics. Humans with consistent message styles become difficult to differentiate from template-based bot messages. Moreover, if there are few commit messages or many unique messages, the document template score will not be effective, thus, this approach works better when enough data is available to almost saturate the document template score. We note that **BIM**'s performance will also vary based on the language of the commit messages (e.g., Spanish and Chinese), and does not support multilingual sets of commit messages.

BIM's performance can be improved by using other more effective similarity measures based on natural language processing, document embeddings, clustering, and machine learning models.

6.3 External Validity

Our goal in this paper was to identify the bots that make commits in social-coding platforms. To that effect, our method of detecting bots might or might not work for detecting other types of bots, e.g. pull-request bots, chat bots, etc.

7 CONCLUSIONS

The automated nature of the bots may inflate the estimates of the amount of team and developer productivity and the total output, and the bot author IDs may inflate the estimates of team size. Such bias may invalidate analyses and decisions based on these measures. Furthermore, bot activity may bias the estimates of social networks linking developers with bots or with other developers with whom they are not in contact. Bots, therefore, should be excluded from studies that focus on modeling the behaviour of human developers in OSS projects. In this work we presented a novel approach, **BIMAN**, to detect bots using the information from code commits. Our approach is comprised of three independent models based on pattern matching, document similarity, and random forest classification.

REFERENCES

[1] Norah Abokhodair, Daisy Yoo, and David W McDonald. 2015. Dissecting a social botnet: Growth, content and influence in Twitter. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. ACM, 839–851.

[2] M Alan. 1950. Turing. *Computing machinery and intelligence. Mind* 59, 236 (1950), 433–460.

[3] Sadika Amreen, Bogdan Bichescu, Randy Bradley, Tapajit Dey, Yuxing Ma, Audris Mockus, Sara Mousavi, and Russell Zaretzki. 2019. A Methodology for Measuring FLOSS Ecosystems. In *Towards Engineering Free/Libre Open Source Software (FLOSS) Ecosystems for Impact and Sustainability*. Springer, Singapore, 1–29.

[4] Sadika Amreen, Audris Mockus, Russell Zaretzki, Christopher Bogart, and Yuxia Zhang. 2019. ALFAA: Active Learning Fingerprint based Anti-Aliasing for correcting developer identity errors in version control systems. *Empirical Software Engineering* (2019), 1–32.

[5] Luciana Benotti, María Cecilia Martínez, and Fernando Schapachnik. 2014. Engaging high school students using chatbots. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*. ACM, 63–68.

[6] Ivan Beschastnikh, Mircea F Lungu, and Yanyan Zhuang. 2017. Accelerating software engineering research adoption with analysis bots. In *2017 IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Technologies Results Track (ICSE-NIER)*. IEEE, 35–38.

[7] Tanmay Bhowmik, Nan Niu, Wentao Wang, Jing-Ru C Cheng, Ling Li, and Xiongfei Cao. 2015. Optimal group size for software change tasks: A social information foraging perspective. *IEEE transactions on cybernetics* 46, 8 (2015), 1784–1795.

[8] David Buttler. 2004. *A short survey of document structure similarity algorithms*. Technical Report. Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States).

[9] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. 2012. Social coding in GitHub: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on computer supported cooperative work*. ACM, 1277–1286.

[10] Tapajit Dey, Yuxing Ma, and Audris Mockus. 2019. Patterns of effort contribution and demand and user classification based on participation patterns in npm ecosystem. In *Proceedings of the Fifteenth International Conference on Predictive Models and Data Analytics in Software Engineering*. ACM, 36–45.

[11] Tapajit Dey, Sara Mousavi, Eduardo Ponce, Tanner Fry, Bogdan Vasilescu, Anna Filippova, and Audris Mockus. 2020. Detecting and Characterizing Bots that Commit Code. (Jan. 2020). https://doi.org/10.5281/zenodo.3694401

[12] Linda Erlenhov, Francisco Gomes de Oliveira Neto, Riccardo Scandariato, and Philipp Leitner. 2019. Current and future bots in software development. In *Proceedings of the 1st International Workshop on Bots in Software Engineering*. IEEE Press, 7–11.

[13] Umer Farooq and Jonathan Grudin. 2016. Human-computer Integration. *interactions* 23, 6 (Oct. 2016), 26–32. https://doi.org/10.1145/3001896

[14] Sven Helmer. 2007. Measuring the structural similarity of semistructured documents using entropy. In *Proceedings of the 33rd international conference on Very large data bases*. VLDB Endowment, 1022–1032.

[15] Mohit Jain, Ramachandra Kota, Pratyush Kumar, and Shwetak N Patel. 2018. Convey: Exploring the use of a context view for chatbots. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 468.

[16] Andreas Karwath and Kristian Kersting. 2006. Relational Sequence Alignment. *MLG 2006* (2006), 149.

[17] Alice Kerry, Richard Ellis, and Susan Bull. 2008. Conversational agents in E-Learning. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*. Springer, 169–182.

[18] Noureddine Kerzazi and Ikram El Asri. 2016. Knowledge flows within open source software projects: A social network perspective. In *International Symposium on Ubiquitous Networking*. Springer, 247–258.

[19] Carlene Lebeuf, Margaret-Anne Storey, and Alexey Zagalsky. 2017. Software bots. *IEEE Software* 35, 1 (2017), 18–23.

[20] Carlene R Lebeuf. 2018. *A taxonomy of software bots: towards a deeper understanding of software bot characteristics*. Ph.D. Dissertation.

[21] Yuxing Ma, Chris Bogart, Sadika Amreen, Russell Zaretzki, and Audris Mockus. 2019. World of Code: An Infrastructure for Mining the Universe of Open Source VCS Data. In *IEEE Working Conference on Mining Software Repositories*. papers/WoC.pdf

[22] Audris Mockus. 2009. Succession: Measuring transfer of code and developer productivity. In *Proceedings of the 31st International Conference on Software Engineering*. IEEE Computer Society, 67–77.

[23] Saul B Needleman and Christian D Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology* 48, 3 (1970), 443–453.

[24] Christian Paul, Achim Rettinger, Aditya Mogadala, Craig A Knoblock, and Pedro Szekely. 2016. Efficient graph-based document similarity. In *European Semantic Web Conference*. Springer, 334–349.

[25] Sara Pérez-Soler, Esther Guerra, Juan de Lara, and Francisco Jurado. 2017. The rise of the (modelling) bots: Towards assisted modelling via social networks. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*. IEEE Press, 723–728.

[26] Xavier Robin, Natacha Turck, Alexandre Hainard, Natalia Tiberti, Frédérique Lisacek, Jean-Charles Sanchez, and Markus Müller. 2011. pROC: an open-source package for R and S+ to analyze and compare ROC curves. *BMC Bioinformatics* 12 (2011), 77.

[27] Temple F Smith, Michael S Waterman, et al. 1981. Identification of common molecular subsequences. *Journal of molecular biology* 147, 1 (1981), 195–197.

[28] Nick Statt. 2016. Why Google's fancy new AI assistant is just called'Google'. *Retrieved March* 21 (2016), 2017.

[29] Margaret-Anne Storey and Alexey Zagalsky. 2016. Disrupting developer productivity one bot at a time. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 928–931.

[30] NT Thomas. 2016. An e-business chatbot using AIML and LSA. In *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2740–2742.

[31] Muhammad Usman, Ricardo Britto, Jürgen Börstler, and Emilia Mendes. 2017. Taxonomies in software engineering: A systematic mapping study and a revised taxonomy development method. *Information and Software Technology* 85 (2017), 43–59.

[32] Mairieli Wessel, Bruno Mendes de Souza, Igor Steinmacher, Igor S Wiese, Ivanilton Polato, Ana Paula Chaves, and Marco A Gerosa. 2018. The power of bots: Characterizing and understanding bots in oss projects. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (2018), 182.

[33] Norman Winarsky, Bill Mark, and Henry Kressel. 2012. The Development of Siri and the SRI Venture Creation Process. *SRI International, Menlo Park, USA, Tech. Rep* (2012).

[34] Timo Wolf, Adrian Schroter, Daniela Damian, and Thanh Nguyen. 2009. Predicting build failures using social network analysis on developer communication. In *Proceedings of the 31st International Conference on Software Engineering*. IEEE Computer Society, 1–11.

[35] Minghui Zhou and Audris Mockus. 2010. Developer fluency: Achieving true mastery in software projects. In *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering*. ACM, 137–146.