Human, bot or both? A study on the capabilities of classification models on mixed accounts

Nathan Cassee, Christos Kitsanelis, Eleni Constantinou and Alexander Serebrenik
Eindhoven University of Technology, The Netherlands
n.w.cassee@tue.nl, c.kitsanelis@student.tue.nl, e.constantinou@tue.nl, a.serebrenik@tue.nl

***Abstract*—Several bot detection algorithms have recently been discussed in the literature, as software bots that perform maintenance tasks have become more popular in recent years. State-of-the-art techniques detect bots based on a binary classification, where a GitHub account is either a human or a bot. However, this conceptualisation of bot detection as an account-level binary classification problem fails to account for 'mixed accounts', accounts that are shared between a human and a bot, and that therefore exhibit both bot and human activity. By using binary classification models for bot detection, researchers might hence mischaracterize both human and bot behavior in software maintenance. This calls for conceptualisation of bot detection through a comment-level classification. However, the single such approach solely investigates a small number of mixed account comments. The nature of mixed accounts on GitHub is thus yet unknown, and the absence of appropriate datasets make this a difficult problem to study.**

In this paper, we investigate three comment-level classification models and we evaluate these classifiers on a manually labeled dataset of mixed accounts. We find that the best classifiers based on these classification models achieve a precision and recall between 88% and 96%. However, even the most accurate comment-level classifier cannot accurately detect mixed accounts; rather, we find that textual content alone, or textual content combined with templates used by bots, are very effective features for the detection of both bot and mixed accounts.

Our study calls for more accurate bot detection techniques capable of identifying mixed accounts, and as such supporting more refined insights in software maintenance activities performed by humans and bots on social coding sites.

***Index Terms*—bot identification, classification model, social coding platforms, GitHub, software engineering**

I. Introduction

Social coding platforms such as GitHub or BitBucket have been widely embraced by open-source software (OSS) communities to facilitate collaboration [1]. To streamline the process of external contributions and safeguard the shared codebase, many teams have adopted the pull-based development model [2]. However, in this model developers need to ensure that the community guidelines are followed, new code fragments are reviewed, and approved changes are successfully integrated [3]. To lighten the workload, teams utilize *bots*, tools that automate repetitive tasks and communicate with developers using a conversational interface [4], [5]: *e.g.*, bots triage abandoned issues [6], generate bug patches [7], upgrade out-of-date dependencies [8] and refactor source code [9].

Studies of human aspects of software maintenance require separating human activity from bot activity as falsely reporting human activity as bot activity or vice-versa can threaten internal validity of a study. This is why software engineering studies have often excluded bots and bot activity [10]–[12]. The need to exclude bots led to several bot detection approaches being proposed, often as an account-level binary classification [13], [14] where accounts are classified either as a bot or a human. However, Golzadeh *et al.* have found 'mixed' GitHub accounts [15], accounts shared by a human and a bot, and as such exhibiting both human-like and bot-like behavior: *e.g.*, maintainers might share their private GitHub token with a bot provider, such as the semantic-release bot.[1] The presence of mixed accounts threatens conceptualisation of the bot detection as an account-level binary classification problem and subsequently accuracy of the tools based on this conceptualisation. For instance, studies of an issue or pull-request response time might derive wrong conclusions if mixed accounts are misrecognised as a solely human activity: indeed, some bots immediately respond to issues or code reviews [13].

This observation calls for *comment-level* classification that should support categorisation of accounts into bot, human or mixed. Hence, we pose the following Research Question: *To what extent can comment-level classifiers distinguish human and bot activity in mixed accounts?*. We consider a validated dataset of GitHub accounts and a set of mixed accounts [14], and evaluate three classification models and several classifiers to determine how suited these models and these classifiers are for the detection of mixed accounts.

In this work we find that the most accurate classification models in a binary setting are not able to accurately detect mixed accounts. Moreover, we find that the textual content alone, or combined with information about the use of templates, is an effective feature for the classification of mixed accounts. More accurate classification models for mixed accounts can be used to study and discover mixed accounts, and secondly support a more accurate separation of human and bot activity in software maintenance.

The remainder of this paper is structured as follows. Section II discusses previous work related to the detection of bots. Section III presents the data collection process and the setup of the classification models. In Section IV we present the results and a discussion of the results follows in Section V. Section VI outlines threats to the validity of this research and mentions possible future work, and Section VII concludes.

[1] https://github.com/semantic-release/semantic-release

TABLE I
COMMENTS IN DATASET

	Human comments	Bot comments	Total
Training set	4,351	4,245	8,596
Testing set	4,256	4,283	8,539
Total	**8,607**	**8,528**	**17,135**

II. RELATED WORK

Traditionally bots have been identified manually, *e.g.*, through previous knowledge [16], [17], based on their names [10], or descriptions or templates [4].

Several recent approaches utilize the entire account's activity to classify accounts as bot or human. Dey *et al.* [13] utilised three distinct classification approaches and combined them into an ensemble model to automatically identify and characterize bots through an account's commit activity w.r.t comments and files/projects, and the name/email of the account. Their approach was validated on a custom dataset and it was able to identify 58 out of the 67 bots. This approach has been simplified by Saadat *et al.* [18]. Golzadeh *et al.* [14] proposed a classification model utilising pull request and issue comments. They constructed a ground-truth dataset of bot and human accounts by manually investigating their recent activity, and harnessed it to distinguish the features that would best accommodate a classification model. The final classification model includes such features as the number of comment patterns, the inequality between comments in patterns, and the number of empty comments.

Golzadeh *et al.* [15] proposed a bot detection classifier that classifies individual comments, with the goal of detecting mixed accounts. Golzadeh *et al.* encode comments as vectors using TF-IDF, and find that a multinomial Naive Bayes classifier is most accurate, with an average precision, recall, and F_1 score of 0.88. Moreover, they find that the classifier can correctly classify 80% of the comments of mixed accounts. However, they rely on a small set of 177 and 203 bot and human comments respectively. On the contrary, our work utilizes 14,097 comments from 78 mixed accounts (see Section IV-B). Our study also builds on top of their classifier, by identifying which features can be used to accurately detect human accounts, bot accounts, *and* mixed accounts.

III. METHODOLOGY

Comments made by a mixed account are either produced by a human or by a bot; to detect mixed accounts we classify individual comments as being written by humans or by bots. Optimally, we would train a classifier based on comments of mixed accounts. However, such a strategy poses a great challenge; since mixed accounts are not easily identifiable, it becomes difficult to form a large ground truth dataset. Thus, we would need to train a classifier on a small amount of data and in turn, we expect that such a classifier would perform poorly in unseen data. We therefore opt for a classifier trained on messages from human-only and bot-only accounts.

Dataset: To this end we rely on and extend a manually labeled dataset by Golzadeh *et al.* [14]. The dataset contains a list of 5,000 GitHub accounts, the repository in which they were found, and the label "human" or "bot." We used the GitHub REST API to gather the 500 most recent comments from each repository in which at least one of the accounts was active. From the 374,783 scraped comments we extracted all comments by accounts present in the dataset of Golzadeh *et al.* In total, we gathered 32,870 comments from 393 bot accounts, and 341,913 comments from 4,025 human accounts; 582 accounts did not have any recent activity, so they have been excluded. As this dataset is unbalanced we randomly sample at most 25 comments per account, and undersample the number of human accounts, such that in total, we obtain 8,528 comments from 393 bot accounts, and 8,607 comments from 406 human accounts. We further split these accounts into an equally divided training and a test set, see Table I. Additionally, we measured the overlap in vocabulary between the training and test set, and we found that 68.5% of the vocabulary in the test set does not occur in the training set.

Overview: To classify the comments we preprocessed the data and evaluated three classification models: Model 1 based on TF-IDF representation of the comment akin to Golzadeh *et al.* [15], Model 2 extending Model 1 with templated messages [13], [14] and Model 3 combining Model 2 with the information about activity of the accounts authoring the comments. For each classification model we compare the performance of four classifiers well suited for textual data: multinomial Naive Bayes [19], Decision Trees [20], Random Forest [21], Support Vector Machines [22]. The model performance is evaluated using the accuracy, precision, recall and F_1; while evaluating the performance of the model we also report the performance of the model for individual classes. We do this such that we can be sure that the model can accurately identify both bot and human comments. To optimize the models we use a Grid Search with 5-Fold Cross-Validation to compute the optimum values of hyper-parameters and select the most accurate classifier; for each model, we only discuss the most accurate classifier.

Preprocessing: A manual inspection of the scraped comments has shown occurrence of several non-natural language tokens such as URLs, absolute paths to directories and files, commit SHAs, email addresses, usernames, version numbers and tags to other pull requests or issues. Following Efstathiou and Spinellis [23], we replace non-natural language tokens with dummy strings corresponding to the type of a token, *e.g.*, #URL or #SHA. Next we pre-process the comments by removing punctuation and numbers, converting uppercase characters to lowercase, removing stop words and stemming with Porter's stemmer [24]. The pre-processing is performed using the NLTK Python library.[2]

Classification models: We evaluate the three classification models. Model 1 classifies comments based solely on the

[2]https://www.nltk.org/

textual content of the comment (cf. [15]). We use TF-IDF to create a numerical representation of the comments [25].

Model 2 extends Model 1 based on an observation that bots often use templated messages [13], [14]. These templated messages are often highly structured and similar to each other. Given the high frequency of templated posts used by bots we hypothesize that clustering similar comments could be useful to classify comments. To this end, we take the pre-processed comments and cluster them based on their similarity. We follow Golzadeh *et al.* [14] and use as the dissimilarity measure the average of the Jaccard distance [26] and the normalized Levenshtein [27] distance between the comments

$$\begin{aligned} \textit{Jaccard}(C_1, C_2) &= 1 - \frac{|\textit{tokens}(C_1 \cap C_2)|}{|\textit{tokens}(C_1 \cup C_2)|} \\ \textit{nLevenshtein}(C_1, C_2) &= \frac{\textit{edit}(C_1, C_2)}{\max(|C_1|, |C_2|)} \end{aligned}$$

We start the clustering process with one cluster which contains a single comment (the *original comment* of the cluster). Next we traverse the list of comments and for each comment we compare it to the original comment of each cluster. If the dissimilarity score is lower than .6 we add the comment to the cluster. If the dissimilarity score is higher than .6 we create a new cluster with the comment as the *original comment* of the cluster. A manual evaluation of several values for the dissimilarity score has shown that .6 is the most accurate value, for slightly lower values known bot templates are not clustered, while for slightly higher values a large number of non-templated comments is clustered. Finally, we count the number of comments per cluster, and for each comment we record the cluster size as the number of similar messages for that comment. The number of similar messages per comment is scaled by removing the mean and scaling to unit variance.

Model 3 extends Model 2 with the following features based on the activity of the account that authored the comment: the number of repositories created, the number of gists created, the number of users followed by the account, and the number of users that follows the account. We expect that bot accounts do not use additional features of Github, such as gists, or social features of Github, and that these features help predict whether a comment originated from a bot or not.

Classification of mixed accounts: As explained above, due to the limited amount of mixed accounts we use them only for testing the models. To this end for each of the 78 mixed accounts identified by Golzadeh *et al.* [14], we extract the most recent 100 comments of the most recent 100 issues and pull requests from the GitHub repositories in which the mixed accounts were identified. The final collected dataset includes 14,097 comments from the 78 mixed accounts. We preprocess these comments as discussed above. Finally, for each classification model we select the classifier found to perform best on the previous task and report the percentage of accounts correctly identified as mixed accounts.

TABLE II
VALIDATION OF THE THREE MODELS ON THE TEST SET

			Prediction				
			Bot	Human	F1	Precision	Recall
Actual	Model 1	Bot	3,786	497	0.883	0.881	0.883
		Human	509	3,747	0.882	0.883	0.880
		Total	4,295	4,244	0.882	0.882	0.882
	Model 2	Bot	3,864	419	0.917	0.932	0.902
		Human	280	3,976	0.919	0.904	0.934
		Total	4,144	4,395	0.918	0.918	0.918
	Model 3	Bot	4,153	130	0.963	0.956	0.969
		Human	190	4,066	0.962	0.969	0.955
		Total	4,343	4,196	0.962	0.962	0.962

IV. VALIDATION RESULTS

A. Model Evaluation

For each classification model, we train and test each of the four classifiers and report on their performance. The best performing classifiers for Model 1, Model 2 and Model 3 are the multinomial Naive Bayes, Support Vector Machines and Random Forest, respectively. In most cases the best performing classifier achieves precision and recall values that are very close to the competing algorithms; the differences with the best performing classifier range from less than 1% to 6%. The only exception is Decision Trees of Model 1, with a recall that is 10% lower than the best performing classifier. Due to the small differences we refrain from drawing conclusions about the strengths of each classifier.

Table II presents the evaluation results of each model. Results of Model 1 indicate that 88% of the bot comments and 88% of the human comments are correctly classified. The manual investigation of the misclassified comments indicates three patterns that the model struggles with. First, all comments of 38 (out of 196) bot accounts were incorrectly classified, as they use vocabulary that does not occur in the training set. Second, certain bots post comments that incorporate entire change logs, which significantly alter the TF-IDF score from other comments that have been posted by the same bot. Third, multiple human comments are incorrectly classified because they include tokens prevalent in bot comments such as usernames, version numbers or links to GitHub pages.

The evaluation results for Model 2 show that the addition of the number of similar comments in the feature vectors appears to improve the performance. Identification of human comments is especially improved, as 93.4% of them are correctly classified. A review of the misclassifications indicates that the concerns discussed above, related to differences in vocabulary and presence of change logs, apply to Model 2 as well. However, the template similarity algorithm proves to be an effective measure to mitigate the misclassification of human comments due to shared vocabulary observed for Model 1. This approach is capable of correctly labeling longer human comments, as only 67 out of the 280 human misclassifications involve comments with more than five words. The difficulty to accurately predict the origin of short comments does not

come as a surprise, since these cases can incorrectly appear to stem from a template, as shorter comments are usually highly similar to each other. For instance the commonly occurring pattern "LGTM" (Looks good to me) is often incorrectly identified as being part of template.

Lastly, the results of Model 3 show that the accuracy is significantly improved, with 97% of bot comments and 95.5% of human comments being correctly identified. The addition of account level features, and the difference in GitHub usage between the two classes contributes to the improved performance of Model 3. Nevertheless, the investigation of the misclassifications indicates that these features can also lead to mistakes. For example, the entire activity of two bots was classified as human, presumably because their accounts had created 177 and 2,045 repositories, unusually high numbers for a bot. Simultaneously, on several occasions human comments have been misclassified because they have been posted by authors with activity more akin to bots.

B. Mixed accounts

The goal of this evaluation is determining whether the three models can detect whether both human and bot activity occurred in the comments of an account. Model 1 and Model 2 correctly identify mixed activity in all of the accounts, while Model 3 identifies mixed activity in only 36% of the accounts.

Specifically, Model 3 misclassifies the activity of 48 accounts as entirely human and the activity of 3 accounts as entirely bot. To investigate why the third model is not able to distinguish mixed accounts, we identify which features the model utilises to classify a comment. Figure 1 showcases the ten features with the highest importance scores, feature importance has been determined with the Gini-importance score [28]. Since mixed accounts typically involve human accounts which have given permissions to bots to post comments, the profile information of mixed accounts is more like that of humans. As a result, the incorporation of the profile information leads the vast majority of comments from mixed accounts to be classified as human. Specifically, Model 3 identifies bot activity in only 10% of the mixed accounts.

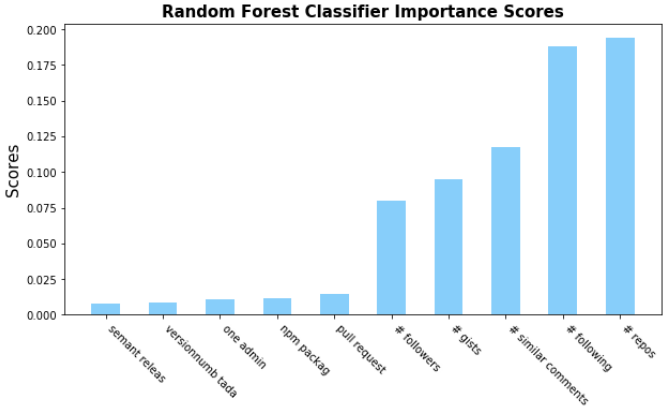

Fig. 1. Most important features in Model 3

V. Discussion

Bots often use templated messages that are highly repetitive, and this property of bots has proven to be very useful for bot detection: Model 2 and Model 3 outperform Model 1. This result confirms and builds on earlier work by Dey *et al.* [13] and Golzadeh *et al.* [14], and further emphasizes the importance of template detection as a feature for bot detection. Moreover, the set of templates used by a bot can be studied to characterize the different actions performed by a bot.

We observe that the most accurate bot detection classifier (Model 3) does not accurately detect mixed accounts. Based on the importance of features used by the classifiers we theorize that account level features such as the number of repositories created by an account can be accurate predictors for a binary classification of comments. However, a classifier that uses these features will fail to identify mixed accounts because account level features of mixed accounts stem from human behavior, making them less useful when identifying mixed accounts. There can also be a mismatch between the account level characteristics of bots that operate as individual accounts and bots that are part of mixed profiles.

Since Model 1 and Model 2 accurately detect all mixed activity we expect that classifiers that use template and text based features can be effective in detecting mixed activity. The extent that templates can improve comment classification of mixed accounts needs to be further investigated on a larger dataset. This is important, as mixed accounts have just recently been reported on by Golzadeh *et al.* [15], but their investigation relies on a small dataset. To further understand the impact of bots on software maintenance activities one should be able to also find mixed accounts. We believe that the classifiers discussed in this work are a next step, allowing researchers to find and study mixed accounts at scale.

Being able to accurately detect bots is becoming increasingly important, as bot adoption is increasing, and more repetitive maintenance activities are being outsourced to bots [4]. A side-effect of this is that an accurate study of behavior on social coding site requires one to separate bot from human activity, as the behavior of bots fundamentally differs from that of humans [4], [13]. The models identified in this work can be used to further improve the bot detection techniques used by researchers to identify or filter out bots from datasets.

Additionally, improved detection techniques could also be used to find more mixed accounts, and study this phenomenon in more detail. From a sample of GitHub accounts identified in the work of Golzadeh *et al.* [29], 78 accounts out of 5,082 accounts are mixed accounts (1.53%) and 632 accounts are bot accounts (12.43%). This indicates that mixed accounts make up a fair share of non-human accounts on GitHub.

VI. Threats to Validity

1) Construct Validity: Threats to construct validity are concerned with the link between the theoretical constructs behind an experiment and the observations. In Model 2 and Model 3 we opted for a computationally less expensive similarity measure as compared to, *e.g.*, the method of Dey *et*

al. [13]. However, this might result in less accurate template identification. A further follow-up is required to evaluate the impact of different similarity measures.

2) Internal Validity: Threats to internal validity are related to the choices made during the study and their impact on the outcome. To mine the comments of the accounts we scraped the 500 most recent comments of the projects in which the accounts are active. However, as a result of this we only scraped recent comments, and our conclusions might be biased towards recent developments on GitHub. A similar issue threatens our study of mixed accounts. To evaluate the accuracy of the three classification approaches we used a train/test split. Therefore, the results we obtained for the experiments might be sensitive to this particular split.

3) External Validity: Threats to external validity are concerned with the possibility to generalize our findings beyond the scope of this study. Our research focuses only on pull request and issue comments in GitHub. However, there is no certainty that the presented models would perform similarly on other software engineering texts created by bots and humans, such as commit messages. Finally, we cannot ascertain whether our approach is generalizable to social coding platforms other than GitHub.

VII. Conclusion

In this paper we study the problem of automatic recognition of mixed accounts based on the comments posted by them on GitHub pull requests and issues. We evaluated three classification models on a curated dataset [14], which includes both bot accounts and mixed accounts. We find that the templates posted by accounts, and account-level features of Github accounts are powerful predictors for bot detection. However, the most accurate model in a binary setting where accounts are either bot or human was not accurate in a setting where mixed accounts were present. By investigating the features used by the model, we conjecture this is due to the fact that the account level features of mixed accounts resemble those of human accounts, and this results in misclassification of bot comments created by a mixed account. Our results are a careful first step in designing more accurate bot detection algorithms that can reliably detect mixed accounts. Potential next steps are studies that attempt to assess the prevalence of mixed accounts on GitHub, and an investigation on how existing bot detection algorithms label mixed accounts. Additionally, the detection of mixed accounts is important for researchers that need to distinguish between human and bot behavior on social coding sites. Algorithms that are able to detect mixed accounts can be used to further identify and study this new phenomenon.

Acknowledgment

We thank Mehdi Golzadeh, Alexandre Decan and Tom Mens for providing the dataset of mixed account names.

References

[1] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in GitHub: Transparency and collaboration in an open software repository," in *CSCW*. ACM, 2012, p. 1277–1286.

[2] G. Gousios, M. Pinzger, and A. v. Deursen, "An exploratory study of the pull-based software development model," in *ICSE*, 2014, pp. 345–355.

[3] G. Gousios, M.-A. Storey, and A. Bacchelli, "Work practices and challenges in pull-based development: The contributor's perspective," in *ICSE*. ACM, 2016, pp. 285—-296.

[4] M. Wessel, B. M. de Souza, I. Steinmacher, I. S. Wiese, I. Polato, A. P. Chaves, and M. A. Gerosa, "The power of bots: Characterizing and understanding bots in OSS projects," *ACM Hum.-Comput. Interact.*, vol. 2, Nov. 2018.

[5] M. Wessel, A. Serebrenik, I. Wiese, I. Steinmacher, and M. A. Gerosa, "Effects of adopting code review bots on pull requests to OSS projects," in *ICSME*, 2020, pp. 1–11.

[6] M. Wessel, I. Steinmacher, I. Wiese, and M. A. Gerosa, "Should I stale or should I close? an analysis of a bot that closes abandoned issues and pull requests," in *BotSE*. IEEE, 2019, p. 38–42.

[7] M. Monperrus, S. Urli, T. Durieux, M. Martinez, B. Baudry, and L. Seinturier, "Repairnator patches programs automatically," *Ubiquity*, pp. 2:1–2:12, Jul. 2019.

[8] S. Mirhosseini and C. Parnin, "Can automated pull requests encourage software developers to upgrade out-of-date dependencies?" in *ASE*. IEEE, 2017, p. 84–94.

[9] M. Wyrich and J. Bogner, "Towards an autonomous bot for automatic source code refactoring," in *BotSE*. IEEE, 2019, p. 24–28.

[10] T. Ahmed, A. Bosu, A. Iqbal, and S. Rahimi, "SentiCR: A customized sentiment analysis tool for code review interactions," in *ASE*, 2017, pp. 106–111.

[11] E. Mirsaeedi and P. C. Rigby, "Mitigating turnover with code review recommendation: Balancing expertise, workload, and knowledge distribution," in *ICSE*, 2020, p. 1183–1195.

[12] Z. Peng, J. Yoo, M. Xia, S. Kim, and X. Ma, "Exploring how software developers work with mention bot in GitHub," in *Chinese CHI*. ACM, 2018, p. 152–155.

[13] T. Dey, S. Mousavi, E. Ponce, T. Fry, B. Vasilescu, A. Filippova, and A. Mockus, "Detecting and characterizing bots that commit code," in *MSR*. ACM, 2020, p. 209–219.

[14] M. Golzadeh, D. Legay, A. Decan, and T. Mens, "Bot or not? detecting bots in GitHub pull request activity based on comment similarity," in *BotSE*. ACM, 2020, p. 31–35.

[15] M. Golzadeh, A. Decan, E. Constantinou, and T. Mens, "Identifying bot activity in github pull request and issue comments," in *BotSE*, 2021.

[16] D. Liu, M. J. Smith, and K. Veeramachaneni, "Understanding user-bot interactions for small-scale automation in open-source development," in *CHI Extended Abstracts*. ACM, 2020, p. 1–8.

[17] G. Robles, J. M. González-Barahona, C. Cervigón, A. Capiluppi, and D. Izquierdo-Cortázar, "Estimating development effort in free/open source software projects by mining software repositories: A case study of openstack," in *MSR*. ACM, 2014, p. 222–231.

[18] S. Saadat, N. Colmenares, and G. Sukthankar, "Do bots modify the workflow of github teams?" in *BotSE*, 2021.

[19] D. D. Lewis, "Naive (bayes) at forty: The independence assumption in information retrieval," in *European Conference on Machine Learning*. Springer, 1998, pp. 4–15.

[20] L. Rokach and O. Maimon, *Data Mining with Decision Trees - Theory and Applications*, 2nd ed., ser. Series in Machine Perception and Artificial Intelligence. WorldScientific, 2014, vol. 81.

[21] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, p. 5–32, 2001.

[22] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, p. 273–297, Sep. 1995.

[23] V. Efstathiou and D. Spinellis, "Code review comments: Language matters," in *ICSE NIER*. ACM, 2018, p. 69–72.

[24] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[25] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*. USA: McGraw-Hill, Inc., 1986.

[26] P. Jaccard, "The distribution of the flora in the alpine zone.1," *New Phytologist*, vol. 11, no. 2, pp. 37–50, 1912.

[27] L. Yujian and L. Bo, "A normalized Levenshtein distance metric," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, pp. 1091–1095, 2007.

[28] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.

[29] M. Golzadeh, A. Decan, D. Legay, and T. Mens, "A ground-truth dataset and classification model for detecting bots in GitHub issue and PR comments," *JSS*, vol. 175, p. 110911, 2021.