EasyChair Preprint

№ 2992

# Cross-project Reopened Pull Request Prediction in GitHub

Abdillah Mohamed, Li Zhang and Jing Jiang



March 19, 2020

# Cross-project Reopened Pull Request Prediction in GitHub

Abdillah Mohamed, Li Zhang, Jing Jiang*
State Key Laboratory of Software Development Environment, Beihang University, Beijing, China
Email:{abdillah,lily,jiangjing}@buaa.edu.cn

*Abstract*—In GitHub, pull requests may get reopened again for further modification and code review. Prediction of within-project reopened pull requests work well if there is enough amount of training data to build the training model. However, for new projects that have a limited amount of pull requests, using training data from other projects can help to predict the reopened pull requests. Therefore, it is important to study cross-project reopened pull request prediction and help integrators in new projects.

In this paper, we propose a cross-project approach that consists of building a decision tree training model based on an external project as a source project to predict the reopened pull requests in another project. We evaluate the effectiveness of cross-project prediction on 7 open source projects containing 100,622 pull requests. Experiment results show that the cross-project prediction achieves accuracy from 78.76% to 96.52%, and F1-measure from 53.34% to 90.58% across 7 projects. We examine the feature importance using the decision tree predictor and find that the number of commits is the most important feature in the majority of projects.

*Keywords*—Reopened pull request prediction, Cross project, GitHub.

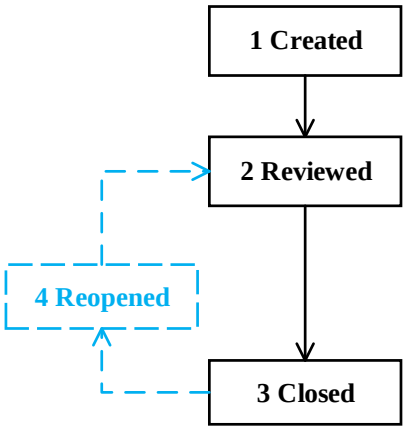


Fig. 1. Reopened pull request evaluation process

## I. Introduction

GitHub[1] is popular among a large number of software developers around the word [1]. GitHub provides support for pull-based development, and allows developers to make contributions flexibly and efficiently [2]. Fig. 1 shows the life cycle of pull requests in GitHub: When a set of changes is ready, contributors create and submit pull requests to the main repository in GitHub. Second, integrators inspect the submitted code changes, identify issues, and make accept or reject decisions. Third, integrators close pull requests. Fourth, some pull requests may be opened again for further modification and code review, and these pull requests are called reopened pull requests.

To identify whether or not a pull request will be reopened, we proposed in our prior work a within-project predictor that consists of splitting the entire dataset of a project into a training set and a testing set to predict whether or not a closed pull request would be reopened [3]. Prediction of within-project reopened pull requests works well if there is enough amount of training data to build the training model.

However, for new projects that have a limited amount of pull requests, using training data from other projects can help to predict the reopened pull requests. It is important to study cross-project reopened pull request prediction, and help integrators in new projects. If pull requests are reopened a long time after their close, they may cause conflicts with new submitted pull requests, add software maintenance cost, and increase burden for already busy developers. Several researchers studied the cross-project defect prediction [4]–[11]. To the best of our knowledge, the cross-project reopened pull request prediction has not been explored yet.

In this paper, we proposes a cross-project approach that consists of building a decision tree training model based on an external project as source project to predict the reopened pull requests in another project. This approach first extracts code features of modified changes, review features during evaluation, and developer feature of contributors from a source project. Then it uses decision tree classifier to make prediction for pull requests in a target project.

In order to explore the performances of this approach, we collect datasets of 7 open-source projects and 100,622 pull requests. Results show that the cross-project reopened pull request prediction achieves accuracy of 78.76%, 95.11%, 94.12%, 89.95%, 93.06%, 96.52%, 94.87%, and F1-measure of 53.34%, 86.52%, 83.72%, 73.54%, 81.54%, 90.58%, 85.72% for the target projects *bootstrap, cocos2d-x, symfony, homebrew-cask, zendframework, rails*, and *angular.js* respectively. We explore feature importance, and find that in the majority of projects, number of commits is the most important in the prediction of reopened pull requests.

*Corresponding author
[1]http://github.com

The main contributions of this paper are as follow:

- We build a cross-project approach based on a source project to predict the reopened pull requests in a target project. Results that cross-project approach performs well in predicting reopened pull requests.
- We find that the number of commits is the most important feature in the cross-project reopened pull request prediction in most of the projects.

The remainder of the work is structured as follows. Section II presents the background and data collection. In Section III, we present the approach of the cross-project reopened pull requests. Section IV presents the experimental settings. Section V presents the experimental results of our approach. In section VI, we present threats to validity. Section VII presents the related work. Finally, section VIII presents summarise our findings.

## II. Background and data collection

In this section, we provide background information about reopened pull requests and describe how our datasets were selected for our study.

### A. Background

GitHub allows developers to work effectively in a distributed software open projects enabled by Git [12]. Unlike control version system such as subversion, with Git there is no canonical copy of the code base. All copies are working copies, and developers can commit local changes on a working copy without needing to be connected to a centralized server [13].

When a set of changes is ready, contributors create and submit pull requests to the main repository in GitHub. Integrators inspect submitted code changes, identify issues, make accept or reject decision, and close the pull requests. Nevertheless, in some cases, pull requests may be opened again for further modification and code review, and these pull requests are called reopened pull requests. We illustrated an example of the reopened pull requests in our previous works [3], [14].

### B. Data collection

We use the same dataset as our previous work [3]. We choose 7 popular projects with more than 5,000 stars, because they receive many pull requests and provide datasets for our research. We describe these 7 projects as follow:

- **rails**[2] is a web application development framework that includes everything needed to create database-backed web applications according to the Model-View-Controller (MVC) pattern.
- **cocos2d-x**[3] is a multi-platform framework for building 2d games, interactive books, demos and other graphical applications. It is an open-source game framework written in C++, with a thin platform dependent layer. It is widely used to build games, apps and other cross platform GUI based interactive programs.
- **symfony**[4] is a PHP framework for web applications and a set of reusable PHP components. It was originally conceived by the interactive agency SensioLabs for the development of web sites for its own customers;
- **homebrew-cask**[5] is a command line interface workflow for the administration of Mac applications distributed as binaries.
- **zendframework**[6] is a collection of professional PHP packages used to develop web applications and services using PHP.
- **angular.js**[7] is an open source JavaScript tool set for building the framework of web application.
- **bootstrap**[8] is an open source framework for developing responsive, mobile first projects on the web with HTML, CSS, and JavaScript.

Table I shows the basic statistics of 7 projects. The table represents the percentage of reopened pull requests. In the fifth column, the value before the slash is the number of reopened pull requests, and the value after the slash is its percentage. 2.97% and 3.78% of pull requests are reopened in projects *angular.js* and *zendframework* respectively. In projects *rails, symfony, homebrew-cask* and *bootstrap*, more than 1% of pull requests are reopened. Reopened pull requests exist in all projects.

## III. Approach

In this section, we describe the cross-project reopened pull request prediction. Figure 2 presents the overall framework of the cross-project reopened pull requests prediction model that has two phases: a model-building phase and a prediction phase. In the model-building phase, our goal is to build a cross-project reopened pull-request prediction model that is learning from a source project. In the prediction phase, the model is used to predict reopened pull requests in a target project.

### A. Model-building phase

As shown in Figure 2, our framework takes as input instances (pull requests) from source project (step 1) with a known class (i.e., reopened or non-reopened). We collect code features, review features and developer feature. Next, it extracts various metrics from the source project to build the cross-project model (step 2). More specifically, we compute code features, review features and developer feature for each pull request in training dataset from a source project. Then we use a weighted vector to represent each pull request, and each element in this vector corresponds to the value of a feature. For pull requests in the training set, we know whether they are reopened or not. We run training dataset and build a decision tree classifier. A decision tree classifier is a machine learning algorithm that uses a tree-like model of decisions to help identify a strategy most likely to reach a goal (e.g., to predict whether or not a pull request will be reopened). We describe details of features as follow:

[2]https://github.com/rails/rails/
[3]https://github.com/cocos2d/cocos2d-x/
[4]https://github.com/symfony/symfony/
[5]https://github.com/caskroom/homebrew-cask/
[6]https://github.com/zendframework/zendframework/
[7]https://github.com/angular/angular.js/
[8]https://github.com/twbs/bootstrap/

TABLE I
BASIC INFORMATION OF PROJECTS.

| Project owner | Repository | Language | #Pull requests | #Reopened pull requests | #Stars |
|---|---|---|---|---|---|
| rails | rails | Ruby | 19,190 | 467/2.43% | 36,253 |
| cocos2d | cocos2d-x | C++ | 14,134 | 113/0.80% | 10,514 |
| symfony | symfony | PHP | 14,569 | 220/1.37% | 14,800 |
| caskroom | homebrew-cask | Ruby | 31,980 | 331/1.04% | 11,229 |
| zendframework | zendframework | PHP | 5,631 | 213/3.78% | 5,522 |
| angular | angular.js | JavaScript | 7,504 | 223/2.97% | 56,359 |
| twbs | bootstrap | JavaScript | 7,614 | 136/1.79% | 112,425 |

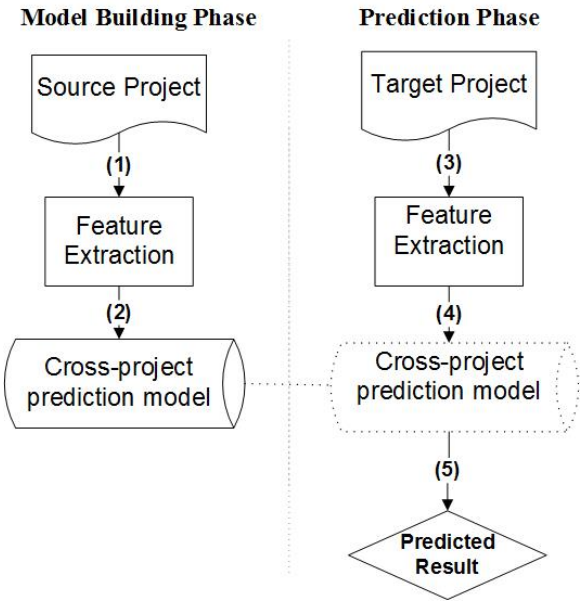


Fig. 2. Overall framework of the cross-project predictor

**Code feature.** In previous work [15], code features were having already been used to predict whether pull requests would be accepted. We also use code features in cross-project reopened pull requests prediction. We only consider pull requests features at the first close. Some pull requests are reopened and closed several times, and they may have further modification and updated values of features. We take in count four features to measure modified codes, including *number of commits*, *number of changed files*, *number of added lines* and *number of deleted lines* in a pull request.

**Review feature.** Previous work [16] found that pull requests with a lot of comments are much less likely to be accepted. The evaluation process is related to the code review decisions, and it may more affect reopened pull requests. Integrators inspect the submitted code changes, identify issues, and make accept or reject decisions. Integrators' attitude in code review towards pull requests may also have an impact on reopening pull requests. Therefore, we consider review features, including *number of comments*, *evaluation time* and *closed status*. *Evaluation time* is the time difference between the pull request's submission and first close. *Closed status* assess whether a pull request is accepted or rejected at its first close.

**Developer feature.** An initial study [17] noticed that certain people were more productive at either fixing bugs or reassigning bugs to others who fix them. Their bugs are less likely to be reopened after they are closed. Previous work [15] appraises contributors' historical accept ratios in predicting whether pull requests would be accepted. Developers' performance records are important in predicting reopened bugs or accepted pull requests. We also apply developer feature which quantifies the reputation of contributors who submit pull requests. Pull requests submitted by contributors with hight experience may be less likely to be reopened. To compute the developer's *reputation*, we collect contributors who submitted the pull requests, the creation time and statuses (merged or rejected) for pull requests in each project. For pull requests submitted by the same contributor, we sort them by their creation time. For each pull request, we compute the number of accepted and rejected pull requests submitted by the same contributor before its creation time. Briefly, the reputation is the proportion of previous pull requests which are submitted by the contributor and get accepted.

### B. Prediction phase

In the prediction phase, the same cross-project prediction model built in step 2 is applied to predict whether a closed pull request would be reopened in the target project. For a pull request in a target project, we first extract code features, review features and developer feature as those extracted the model-building phase (step 3). We then input the values of these metrics into the cross-project model (Step 4). It outputs the pull request prediction result about whether it will be either reopened or non-reopened (Step 5).

## IV. EXPERIMENTAL SETTINGS

In this section, we aim at presenting the experimental setting to evaluate the performance of our approach. The main goal of this work is twofold. (i) We build trained model based one source project to train a model and use it to predict the reopening of a pull request of another project. (ii) We study feature importance in predicting reopened pull requests.

### A. Evaluation process and metrics

As shown in Table I, our datasets include 7 projects and 100,622 pull requests. For each project, its dataset is used as a testing dataset of a target project, and other project is used as a training dataset of a source project. We use a training dataset to build a cross-project prediction model, and use the testing dataset to evaluate performance.

In evaluation, we use precision, recall and f1-measure. The accuracy measures the number of correctly classified reopened

pull requests (both non-reopened and reopened) over the total number of pull requests. Precision is the ratio of correctly predicted reopened pull requests over all the pull requests predicted as reopened. Recall is the ratio of correctly predicted reopened pull requests over all actually reopened pull requests. F1-measure is the weighted harmonic mean of precision and recall. These metrics are commonly used in the software engineering literature [18], [19].

### B. Research Questions

We are interested to answer following research questions:

**RQ1: How does the cross-project prediction perform?**

**Motivation**. Zimmermann et al. found that when no or a little data was available, developers used data from another project to successfully make defect prediction for another one project [9]. In this research question, we aim at building a cross-project predictor based on one project as a source project to predict the pull request reopening in a data of another project. We wonder how the cross-project prediction perform.

**Approach**. To solve this research question, we aim at building decision tree training models based on one projects as a source project and persist them by crossing the seven projects between them. For each of the 6 source projects used separately to predict the reopened pull requests in one and only target project, we select the results of the source project that achieves high f1-measure.

**RQ2: Which features are important in cross-project reopened pull request prediction?**

**Motivation**. Different features may have various weights in cross-project reopened pull request prediction. We wonder which features are more important than other.

**Approach**. In order to answer this question, we use decision tree classifier to compute feature importance in the prediction of reopened pull requests. Feature importance is calculated as the decrease in node impurity weighted by the probability of reaching that node. The node probability can be calculated by the number of reopened pull request that reach the node, divided by the total number of pull requests. The higher the value is, and the more important the feature is.

## V. Experimental results

In this section, we study the results of our study aiming at answering above research questions.

### A. RQ1: Performance of cross-project prediction

In order to answer RQ1, we study results based on different combination of source projects and target project. We first analyze the project *rails* as an example. Table II shows results when the project *rails* is the target project. In each row, we predict reopened pull requests in the project *rails* as target projects by crossing the projects *symfony, cocos2d-x, angular.js, zendframework, homebrew-cask* and *bootstrap* respectively as source projects. The best results are in bold. Results show that the combination *cocos2d-x =>rails* achieves the best performance by achieving an accuracy of 96.52% and f1-measure of 90.58%.

TABLE II
Predicting the reopened pull request based on the project rails as the target project

| Source =>Target projects | Accuracy | Precision | Recall | F1-measure |
|---|---|---|---|---|
| symfony =>rails | 96.47% | **98.07%** | 83.92% | 90.45% |
| cocos2d-x =>rails | **96.52%** | 96.60% | 85.20% | **90.58%** |
| angular.js =>rails | 96.02% | 95.29% | 85.00% | 89.85% |
| zendframework =>rails | 96.42% | 96.61% | 84.75% | 90.29% |
| homebrew-cask =>rails | 92.24% | 78.51% | 83.92% | 81.13% |
| bootstrap =>rails | 94.83% | 82.77% | **92.97%** | 87.57% |

Table III shows the performances of the cross-projects reopened pull requests prediction across 7 projects. The projects on top of the table are used as a target for single source cross-projects, while the projects on the left side of the table are used as source projects. We use the source project to train the decision tree model, and the target project is used as a class project to predict the reopened pull requests. Results in green color represent the highest performance predictions of the cross-project prediction of each target across 6 target projects. For example, in the third column, we use the project *angular.js* as a target project to predict the reopened pull requests in the source projects *rails, cocos2d-x, symfony, homebrew-cask, zendframework* and *bootsrap* respectively. Results show that when predicting reopened pull requests in the target project *angular.js*, the source project *symfony* is more suitable comparing to the other source projects. In the same way, we compared the performances of the other source projects, and find the source project which achieves the highest F1-measure in predicting reopened pull requests for a specific target project.

The Table IV presents the combinations of the cross-project that carry out the best results across 42 combinations from the Table III. Each target project is used separately with each of the six remaining projects as source projects to predict the reopened pull requests and select the combination (i.e., the prediction results of the crossed projects) that achieves the best results. For instance, the second row presents the best result when combining the project homebrew-cask as a source project to predict the reopened pull request in the project *twbs* as a target. In the same way, we processed to select the best combination of crossed projects (sources and targets) that has good performances. This we notice that the single source cross-project reopened pull requests prediction achieves good performances in most of the projects.

> **RQ1:** Across the 7 projects, the single source cross-project reopened pull requests prediction achieves good performances in most of the projects.

### B. RQ2: Important features for predicting reopened pull requests.

We use decision tree classifier to predict whether pull requests will be reopened or not. Decision tree classifier also computes the importance of each feature in the prediction of reopened pull requests, and we plot the results in the Table V. Feature importance may be different in various projects. For example, in source project *rails* and the target

TABLE III
F1-MEASURE COMPARISON BETWEEN THE CROSS-PROJECTS REOPENED PULL REQUESTS PREDICTION

| Source/Target | rails | angular.js | cocos2d-x | Symfony | homebrew-cask | zendframework | bootstrap |
|---|---|---|---|---|---|---|---|
| rails | / | 83.61% | 86.22% | 82.55% | 61.82% | 81.54% | 24.81% |
| angular.js | 89.85% | / | 84.06% | 80.65% | 59.58% | 77.73% | 24.25% |
| cocos2d-x | 90.58% | 84.26% | / | 67.61% | 73.54% | 80.36% | 35.74% |
| symfony | 90.45% | 85.72% | 84.18% | / | 61.79% | 79.57% | 20.59% |
| homebrew-cask | 81.13% | 81.62% | 83.75% | 66.15% | / | 79.40% | 53.34% |
| zendframework | 90.29% | 84.87% | 86.52% | 83.72% | 67.34% | / | 33.68% |
| bootstrap | 87.57% | 76.33% | 84.68% | 69.84% | 73.24% | 74.43% | / |

TABLE IV
PERFORMANCES OF CROSS-PROJECT REOPENED PULL REQUESTS PREDICTOR

| Source=>Target projects | Accuracy | Precision | Recall | F1-measure |
|---|---|---|---|---|
| homebrew-cask =>bootstrap | 78.76% | 48.12% | 59.83% | 53.34% |
| zendframework =>cocos2d-x | 95.11% | 97.36% | 77.86% | 86.52% |
| zendframework =>symfony | 94.12% | 93.72% | 75.64% | 83.72% |
| cocos2d-x =>homebrew-cask | 89.95% | 78.51% | 69.16% | 73.54% |
| rails =>zendframework | 93.06% | 90.18% | 74.41% | 81.54% |
| cocos2d-x =>rails | 96.52% | 96.60% | 85.20% | 90.58% |
| symfony =>angular.js | 94.87% | 97.91% | 76.24% | 85.72% |

*zendframework*, the three most important features are the number of commits, number of changed files, and the number of added lines. In source project *homebrew-cask* and the target project *bootstrap*, the three most important features include a number of commits, closed status, and number of added lines. In the majority of projects, the number of commits is the most important in the prediction of reopened pull requests. Some pull requests have many commits, and they may be difficult for integrators to make a complete evaluation. Therefore, pull requests with many commits are likely to be reopened, and the number of commits is the most important feature.

**RQ2:** In the majority of projects, the number of commits is the most important in the cross-project reopened pull request prediction.

## VI. THREATS TO VALIDITY

In this section, we introduce threats to the validity of our study.

Threats to external validity relate to the generalization of our research. Firstly, our experimental results are limited to 7 projects in GitHub. We cannot claim that other projects will achieve the same results. In the future, we plan to use more projects to better generalize the results of our method. We will conduct broader experiments to validate whether the singe source cross-project prediction performs well. Secondly, we analyze open-source software projects in GitHub, and it is unknown whether other platforms have similar results. In the future, we plan to study other platforms and compare their results with our findings in GitHub.

Threats to construct validity refer to the degree to which the construct being studied is affected by experiment settings. We use accuracy, precision, recall, and F1-measure. These evaluation metrics are also used by various automated software engineering techniques [18], [19]. As a results, there is little threat to construct validity.

## VII. RELATED WORKS

In this section, we mainly discuss related works, including reopened pull requests and cross-project prediction.

### A. Reopened pull requests

In GitHub, there are several works which are focusing on pull requests evaluation and prediction [3], [14]. We conducted a case study to understand reopened pull requests [14]. We proposed an approach DTPre which was an automatic predictor of reopened pull requests based on decision tree classifier [3]. Previous work [3] designed a within-project reopened pull request prediction, while this paper explores the cross-project reopened pull request prediction.

### B. Cross-project prediction

The cross-project prediction has been the main area of researches in different aspects by reusing training data from other projects to make a prediction in a new project. Several authors discussed the cross-project defect prediction [4]–[11]. Rahman et al. [4] compared the cross-project defect prediction with the prediction within a project, and they found that cross-project prediction performance was no worse than within-project performance and considerably better than random prediction. Xin et al. [20] showed that cross-project prediction worked well if there was a sufficient amount of training data to build the model. Xin et al. further proposed a HYbrid model reconstruction approach for cross-project defect prediction [7]. Canfora et al. [6] conducted a study for cross-project defect prediction, based on a multi-objective logistic regression model built using a genetic algorithm. Turhan et al. [10] proposed a practical defect prediction method for companies that did not track defect related data to investigate the applicability of cross-company (CC) data for building localized defect predictors using static code features.

Unlike the above researches, we address a different problem, namely cross-project reopened pull request prediction.

## VIII. CONCLUSION

Cross-project reopened pull requests are important for the projects that do not have enough historical data to build prediction models. In this paper, we propose a cross-project approach for predicting reopened pull requests in GitHub. This study brings new insight into the performances of the

TABLE V
FEATURE IMPORTANCE FOR CROSS-PROJECT REOPENED PULL REQUESTS PREDICTION

| Features | homebrew-cask =>bootstrap | zendframework =>cocos2d-x | zendframework =>symfony | cocos2d-x =>homebrew-cask | rails =>zend-framework | cocos2d-x =>rails | symfony =>Angular.js | Average |
|---|---|---|---|---|---|---|---|---|
| Number of commits | 0.327 | 0.275 | 0.275 | 0.611 | 0.476 | 0.611 | 0.463 | 0.434 |
| Number of changed file | 0.038 | 0.411 | 0.411 | 0.040 | 0.361 | 0.040 | 0.274 | 0.225 |
| Number of added lines | 0.128 | 0.000 | 0.000 | 0.000 | 0.045 | 0.000 | 0.000 | 0.025 |
| Number of deleted lines | 0.003 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| Number of comments | 0.019 | 0.033 | 0.034 | 0.002 | 0.017 | 0.002 | 0.015 | 0.017 |
| Evaluation time | 0.079 | 0.169 | 0.169 | 0.083 | 0.041 | 0.084 | 0.116 | 0.106 |
| Closed status | 0.322 | 0.038 | 0.038 | 0.234 | 0.040 | 0.234 | 0.025 | 0.133 |
| Reputation | 0.084 | 0.074 | 0.073 | 0.029 | 0.021 | 0.029 | 0.107 | 0.060 |

cross-project using a decision tree classifier. Based on 100,622 pull requests from 7 open-source projects, experimental results show that the cross-project reopened pull request prediction achieves an f1-measure of 53.34%, 86.52%, 83.72%, 73.54%, 81.54%, 90.58%, and 85.72% for the target projects *bootstrap, cocos2d-x, symfony, homebrew-cask, zendframework, rails*, and *angular.js* respectively. We use decision tree to compute feature importance, and find that number of commits is the most important feature in the majority of projects.

In the future, we plan to use more projects from different open-source projects to explore whether our approaches would have similar results in the prediction of reopened pull requests.

## ACKNOWLEDGMENT

This work is supported by the National Key Research and Development Program of China No. 2018AAA0102301, the National Natural Science Foundation of China under Grant No. 61672078, the State Key Laboratory of Software Development Environment under Grant No.SKLSDE-2019ZX-05, and the Massiwa Technology of Comoros under Grant No.9108-B-19.

## REFERENCES

[1] A Lima, L Rossi, and M Musolesi. Coding together at scale: Github asa collaborative social network. In *Proceedings of 8th AAAI International Conference on Weblogs and Social Media*, 2014.

[2] Georgios Gousios, Andy Zaidman, Margaret-Anne Storey, and Arie Van Deursen. Work practices and challenges in pull-based development: the integrator's perspective. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 1, pages 358–368. IEEE, 2015.

[3] Abdillah Mohamed, Li Zhang, Jing Jiang, and Ahmed Ktob. Predicting which pull requests will get reopened in github. In *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*, pages 375–385. IEEE, 2018.

[4] Foyzur Rahman, Daryl Posnett, and Premkumar Devanbu. Recalling the" imprecision" of cross-project defect prediction. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, pages 1–11, 2012.

[5] Feng Zhang, Quan Zheng, Ying Zou, and Ahmed E Hassan. Cross-project defect prediction using a connectivity-based unsupervised classifier. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, pages 309–320. IEEE, 2016.

[6] Gerardo Canfora, Andrea De Lucia, Massimiliano Di Penta, Rocco Oliveto, Annibale Panichella, and Sebastiano Panichella. Multi-objective cross-project defect prediction. In *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation*, pages 252–261. IEEE, 2013.

[7] Xin Xia, David Lo, Sinno Jialin Pan, Nachiappan Nagappan, and Xinyu Wang. Hydra: Massively compositional model for cross-project defect prediction. *IEEE Transactions on software Engineering*, 42(10):977–998, 2016.

[8] Annibale Panichella, Rocco Oliveto, and Andrea De Lucia. Cross-project defect prediction models: L'union fait la force. In *2014 Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*, pages 164–173. IEEE, 2014.

[9] Thomas Zimmermann, Nachiappan Nagappan, Harald Gall, Emanuel Giger, and Brendan Murphy. Cross-project defect prediction: a large scale experiment on data vs. domain vs. process. In *Proceedings of the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 91–100, 2009.

[10] Burak Turhan, Tim Menzies, Ayşe B Bener, and Justin Di Stefano. On the relative value of cross-company and within-company data for defect prediction. *Empirical Software Engineering*, 14(5):540–578, 2009.

[11] Tim Menzies, Andrew Butcher, Andrian Marcus, Thomas Zimmermann, and David Cok. Local vs. global models for effort estimation and defect prediction. In *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*, pages 343–351. IEEE, 2011.

[12] Chandra Maddila, Chetan Bansal, and Nachiappan Nagappan. Predicting pull request completion time: a case study on large scale cloud services. In *Proceedings of the Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 874–882, 2019.

[13] Yue Yu, Huaimin Wang, Gang Yin, and Tao Wang. Reviewer recommendation for pull-requests in github: What can we learn from code review and bug assignment? *Information and Software Technology*, 74:204–218, 2016.

[14] Jing Jiang, Abdillah Mohamed, and Li Zhang. What are the characteristics of reopened pull requests? a case study on open source projects in github. *IEEE Access*, 7:102751–102761, 2019.

[15] Georgios Gousios, Martin Pinzger, and Arie van Deursen. An exploratory study of the pull-based software development model. In *Proceedings of the 36th International Conference on Software Engineering*, pages 345–355. ACM, 2014.

[16] Jason Tsay, Laura Dabbish, and James Herbsleb. Influence of social and technical factors for evaluating contribution in github. In *Proceedings of the 36th international conference on Software engineering*, pages 356–366. ACM, 2014.

[17] Thomas Zimmermann, Nachiappan Nagappan, Philip J Guo, and Brendan Murphy. Characterizing and predicting which bugs get reopened. In *Proceedings of the 34th International Conference on Software Engineering*, pages 1074–1083. IEEE Press, 2012.

[18] Jing Jiang, Yun Yang, Jiahuan He, Xavier Blanc, and Li Zhang. Who should comment on this pull request? analyzing attributes for more accurate commenter recommendation in pull-based development. *Information and Software Technology*, 84:48–62, 2017.

[19] Emad Shihab, Akinori Ihara, Yasutaka Kamei, Walid M Ibrahim, Masao Ohira, Bram Adams, Ahmed E Hassan, and Ken-ichi Matsumoto. Predicting re-opened bugs: A case study on the eclipse project. In *Reverse Engineering (WCRE), 2010 17th Working Conference on*, pages 249–258. IEEE, 2010.

[20] Xin Xia, David Lo, Shane McIntosh, Emad Shihab, and Ahmed E Hassan. Cross-project build co-change prediction. In *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pages 311–320. IEEE, 2015.