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
[‡]University Institute of Technology, University of Comoros, Comoros
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 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. 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.

I. Introduction

GitHub is popular among a large number of software developers around the word [1].

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 [2]. 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. Several researchers studied the cross-project defect prediction [3]–[5]. 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.

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 show 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 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. Data collection

We use the same dataset as our previous work [2]. We choose 7 popular projects such as rails, cocos2d-x, symfony, homebrew-cask, zendframework, angular.js, and bootstrap with more than 5,000 stars, because they receive many pull requests and provide datasets for our research.

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. Reopened pull requests exist in all projects.

III. approach

In this section, we describe the cross-project reopened pull request prediction.

[*]Corresponding author

DOI reference number: 10.18293/SEKE2020-072

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

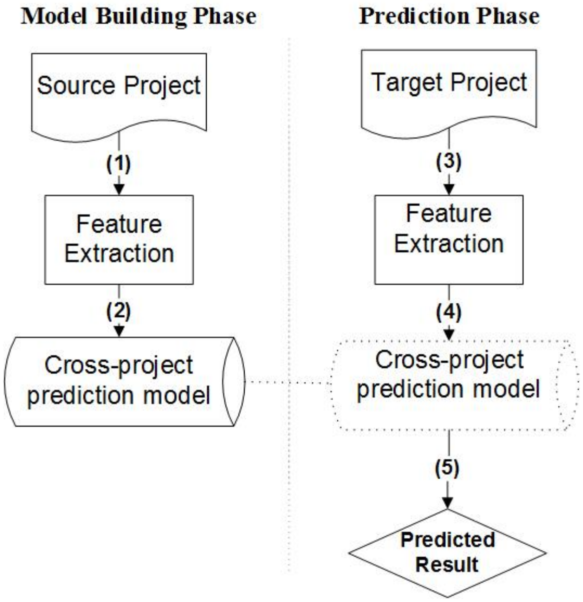

Fig. 1. Overall framework of the cross-project predictor

A. Model-building phase

As shown in Figure 1, 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). Then we use a weighted vector to represent each pull request, and each element in this vector We describe details of features as follow:

Code feature. We use code features in cross-project reopened pull requests prediction at the first close. 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. 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. We apply developer feature which quantifies the reputation of contributors who submit pull requests. 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

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

In evaluation, we use accuracy, precision, recall and f1-measure. The accuracy measures the number of correctly classified reopened pull requests (both non-reopened and re-opened) 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.

B. Research Questions

We are interested to answer following research questions:

RQ1: How does the cross-project prediction perform?

Motivation. 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.

Approach. To solve this research question, we aim at building decision tree training models based on one projects as a source project 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. 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 that achieves the best results. In the same way, we processed to select the best combination of crossed projects (sources and targets) that has good performances. Thus, we notice that the single source cross-project reopened pull requests prediction achieves good performances in most of the projects.

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%

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.*

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. 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. In the future, we plan to use more projects to better generalize the results of our method. Secondly, we analyze open-source software projects in GitHub. 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. As a results, there is little threat to construct validity.

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 V

FEATURE IMPORTANCE FOR CROSS-PROJECT REOPENED PULL REQUESTS PREDICTION

Features	homebrew-cask =>bootstrap	zendframework =>cocos2d-x	zendframework =>symfony	cocos2d-x =>homebrew-cask	rails =>zendframework	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

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 [2], [6]. We conducted a case study to understand reopened pull requests [6]. Previous work [2] 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 [3]–[5]. Rahman et al. [3] 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.

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 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.

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.

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] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.