Towards Building a Universal Defect Prediction Model

MSR 2014

Joymallya Chakraborty
North Carolina State University
jchakra@ncsu.edu

ABSTRACT

A universal defect prediction model that is built from the entire set of diverse projects would relieve the need for building models for an individual project. A universal model could also be interpreted as a basic relationship between software metrics and defects. But the main challenge here is to handle the variations in the distribution of predictors.
Zhang et al. [1] clustered projects based on the similarity of the distribution of 26 predictors, and derived the rank transformations using quantiles of predictors for a cluster. Their results suggest that a universal defect prediction model may be an achievable goal.

KEYWORDS

Universal Defect Prediction Model, Rank Transformation, Bug

1 INTRODUCTION

In this study, the authors propose a context-aware rank transformation to address the variations in the distribution of predictors before fitting them to the universal defect prediction model. They used 21 code metrics, five process metrics, and six context factors as predictors (i.e., programming language, issue tracking, the total lines of code, the total number of files, the total number of commits, and the total number of developers). The context-aware approach stratifies the entire set of projects by context factors, and clusters the projects with similar distribution of predictors. They applied every tenth quantile of predictors on each cluster to formulate ranking functions. After transformation, the predictors from different projects have exactly the same scales. The universal model was then built based on the transformed predictors.
They applied their approach on 1,398 open source projects hosted on SourceForge and GoogleCode. They examined the generalizability of the universal model by applying it on five external projects also.
They claimed that their main contributions are:

(1) Context-aware rank transformation
(2) Context factors as predictors of the universal model

2 DATA PREPROCESSING

It is very likely that predictors from different projects of various contexts exhibit different distribution [2]. To overcome this challenge towards building a universal defect prediction model, they proposed a context-aware rank transformation approach, as illustrated in Figure 1.

Foundations of Software Science, Fall 2018, Computer Science

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$9999.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

3 CONTEXT FACTORS

They chose six context factors based on their availability to open source projects.

(1) Programming Language (PL)
(2) Issue Tracking (IT)
(3) Total Lines of Code (TLOC)
(4) Total Number of Files (TNF)
(5) Total Number of Commits (TNC)
(6) Total Number of Developers (TND)

They stratified the entire set of projects based on the aforementioned six context factors. They got 5,2,4,4,4, and 4 groups, respectively. In total, They obtained 2560 (i.e.,5 x 2 x 4 x 4 x 4 x 4) non-overlapped groups.

4 CLUSTERING SIMILAR PROJECTS

To derive more accurate quantiles of a particular metric, They grouped the projects with the similar distribution of the metric. Two distributions are similar if their difference is neither statistically significant nor significantly large. For each metric m, the clusters of projects with the similar distribution of metric m are obtained using an algorithm. It has two major steps:-

(1) Comparing the Distribution of Metrics - Mann-Whitney U test was used
(2) Quantifying the Difference between Distributions - Cliff's δ was used

5 OBTAINING RANKING FUNCTIONS

The ranking function transforms the raw metric values to relatively predefined values (i.e., ranging from one to ten). The researchers used the quantiles of metric values to formulate our ranking functions. For example, if every tenth quantile for a metric m1 in cluster Cl is: 11, 22, 33, 44, 55, 66, 77, 88, and 99 respectively. Then the value 27 of metric m1 will be converted to 3 if the corresponding project belongs to cluster Cl. This is because the value 27 is greater than 22 (i.e., the 20% quantile) and less than 33 (i.e., the 30% quantile)

6 BUILDING A UNIVERSAL DEFECT PREDICTION MODEL

They applied Naive Bayes as the modelling technique in their experiments. Before transforming a metric m_i for project p_j , we identify context factors of project p_j and formulate a vector like $< m_i, C++, useIT, moreTLOC, lessTNF, lessTNC, lessTND >$. In order to locate the ranking functions, they compared the vector of project p_j to the vectors of all clusters to determine which cluster project p_j belongs to.

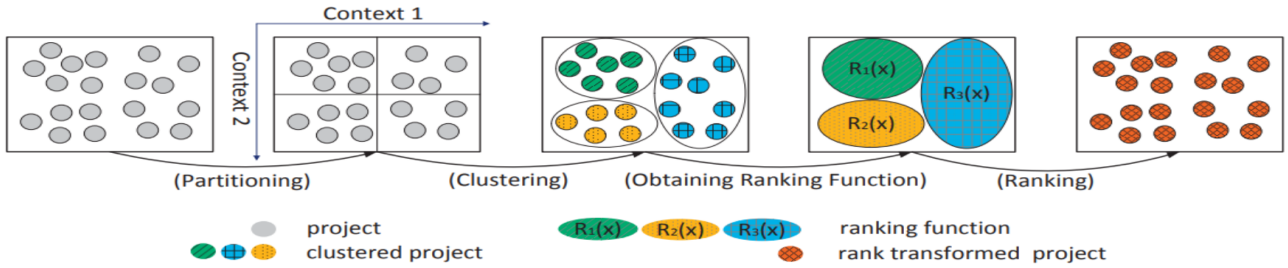

Figure 1: Four-step rank transformation approach

7 CASE STUDY RESULTS

RQ1: Can a context-aware rank transformation provide predictive power comparable to the power of log transformation?

They have proposed a context-aware rank transformation method to eliminate the impact of varied scales of metrics among different projects. The rank transformation converts raw values of all metrics to levels of the same scale. On the other hand, the log transformation uses the logarithm of raw values. Authors compared the performance of defect prediction models built using rank transformations to the models built using log transformations. Table 1. presents the mean values of the six performance measures of both log and rank transformations, and the corresponding p-values of Wilcoxon rank sum test. The results show the difference between the two transformations is small (i.e., less than 0.10). They concluded that rank transformation achieves comparable performance to log transformation. It is reasonable to use the proposed rank transformation method to build universal defect prediction models.

RQ2: What is the performance of the universal defect prediction model?

This research question aims to investigate the best achievable predictive power of the universal model. First, they evaluated if the predictive power of the universal model can be improved by adding context factors as predictors, together with code metrics and process metrics that are commonly used in prior studies for defect prediction. Second, they studied if the universal model can achieve comparable performance as within-project defect prediction models. Table 2. shows that the AUC value keeps increasing when adding more metric sets to the model. The context factors increase the precision, recall, F-measure, g-measure, and the AUC value. Hence, the context factors are good predictors for building a universal defect prediction model.

RQ3: What is the performance of the universal defect prediction model on external projects?

They tried to test the performance of universal defect prediction model fpr external projects. So, they chose five external projects and compared the performance of universal model with within project model for these five projects. The universal model achieves higher recall and better AUC values but has a higher false positive rate. Considering the five projects might conduct different development strategies than SourceForge or GoogleCode projects, there is a high chance to apply the universal model on more external projects with acceptable predictive power.

Table 1: Wilcoxon rank sum tests and mean values of six performance measures

Measures	LogTran	RankTran	p-value	Cohen's d
prec	0.48	0.48	0.71	-0.01
pd	0.57	0.58	0.31	0.03
fpr	0.36	0.35	0.43	0.04
F-measure	0.49	0.50	0.33	-0.03
g-measure	0.53	0.54	$4.7e{-}^{03}$	-0.07*
AUC	0.61	0.62	0.14	-0.03

Table 2: The performance measures for the universal models built using CM , CPM and CPMC

Measures	CM	CPM	CPMC
prec	0.36	0.38	0.40
pd	0.91	0.83	0.86
fpr	0.87	0.76	0.70
F-measure	0.51	0.51	0.55
g-measure	0.23	0.36	0.42
AUC	0.58	0.60	0.65

8 CONCLUSION

In future, their plan is to evaluate the feasibility of the universal model for commercial projects. They have also thought about making a plugin for a version control system or IDE.

REFERENCES

[1] Feng Zhang, Audris Mockus, Iman Keivanloo, and Ying Zou. 2014. Towards building a universal defect prediction model. In *Proceedings of the 11th Working Conference on Mining Software Repositories*. ACM, 182–191.

[2] Feng Zhang, Audris Mockus, Ying Zou, Foutse Khomh, and Ahmed E Hassan. 2013. How does context affect the distribution of software maintainability metrics?. In *Software Maintenance (ICSM), 2013 29th IEEE International Conference on*. IEEE, 350–359.