IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 25, Issue 6, Ser. 1 (Nov. – Dec. 2023), PP 62-64
www.iosrjournals.org

An Approach To Recommend Clones For Refactoring Using Machine Learning And Feature Selection

Manpreet Kaur[1*], Dhavleesh Rattan[2] and Madan Lal[3]
[1,2,3]*Department of Computer Science and Engineering, Punjabi University, Patiala, Punjab, India.*
Email [1*]*manpreet.kaur09@gmail.com,* [2] *dhavleesh.ce@pbi.ac.in,* [3]*madanlal@pbi.ac.in*

ABSTRACT
This paper presents a technique to recommend clones for refactoring. Clones are duplicate code fragments that exist in the source code of software. Clones increase the maintenance cost of the software and therefore, are required to manage through refactoring. Our technique is based on feature selection and machine learning. The proposed technique helps to improve the accuracy of recommending clones for refactoring through feature selection.
Keywords: *Software clones; Clone management; Clone recommendation, Clone refactoring*

I. INTRODUCTION

Reusing existing code by copying and pasting is called software cloning. The duplicate code fragments that get generated using software cloning are called software clones. The existence of software clones in a source code can lead to the following problems:

- Increases the probability of bug propagation[1].
- Increases maintenance cost.
- Source code can become error-prone due to inconsistent updation of all cloned fragments.
- Programs become difficult to understand.
- The size of the source code increases which increases memory requirements.

To minimize problems generated by software clones, many tools were developed to find software clones[2]. These clone detection tools work on different algorithms and granularity[3], [4]. Each clone detection tool generates output in a different format. So, analysis of software clone detection results is a time-consuming process. On the other hand, management of clones is required so that their harmful effects can be minimized. Management of clones can be done by two methods. One method is clone tracking and the other is clone refactoring[5], [6].

Clone tracking[7] is the process of monitoring changes that take place in all clone instances of a clone group during maintenance. If any clone instance experiences a change, then the same change needs to be propagated to other instances of the same clone. Clone tracking software helps to achieve this operation.

Clone refactoring[8] helps to remove clones through various existing software refactoring methods. However, the major issue is that all clones are not suitable to perform refactoring. So there is a need for some approaches that can filter important clones for refactoring.

Current work aims to use machine learning algorithms and feature selection to automate the clone recommendation for refactoring. We propose to analyze the performance of various machine learning and feature selection methods to recommend clones for refactoring.

II. RELATED WORK

Higo et al. [9] [10][11] prepared a tool, aries that uses CCShaper to find refactoring-oriented clones. They used clone metrics to find clones suitable for refactoring.

Schulze et al. [12] used clone location and the type of statement of code clones to give their refactoring proposal. Choi et al. [13] proposed a metric-based method to find suitable code clones for refactoring. They discussed that a combination of clone metrics are better to recommend clones for refactoring instead of individual clone metric.

Mondal et al. [14] defined SPCP clones that can be important for refactoring. The SPCP (Similarity Preserving Change Pattern) clones are defined as two or more clone fragments from the same clone class that evolve by receiving similarity-preserving changes. These clones having lower change couplings than other classes are good candidates for refactoring.

The studies [15]–[18] used machine learning to predict clones for refactoring. Wang and Godfrey [15] used a decision tree for automating recommending clones for refactoring. They achieved precision from 77.3% to 87.9%.

Fanqi [19] used clustering to find refactorable clones. They trained the SOM model using metrics of selected refactorable clones like LEN (length of code clone in terms of token), POP (number of clones in a clone group), NIF (number of files in which clones of a clone group are distributed), etc. The trained model was used to classify unknown code clones.

Yue et al. [18] used various features of code clones to build a clone recommendation tool for refactoring. They used AdaBoost, Random Forest, C4.5, SMO, and Naïve Bayes to recommend clones for refactoring. They concluded that AdaBoost suggested clones for refactoring with higher accuracy in both with-in-project and cross-project testing.

Rongrong et al.[16] used Naïve Bayes, C4.5, and Bayesian networks to build machine learning-based models for recommending clones for refactoring. They used seven open-source projects written in C for evaluation. Sheneamer [17] used AST features and four machine learning algorithms (K-nearest neighbor, Forest PA, Bagging, and Random Forest) for recommending clones for refactoring. The approach identified that Random Forest achieved better outcomes among all classifiers.

Our current work proposed an approach to recommend clones for refactoring. The approach is based on feature selection and machine learning. Existing approaches have not used feature selection to improve the accuracy of machine learning algorithms.

III. METHODOLOGY

This section discusses the steps of our proposed technique. The major steps of the technique are 1) Preparation of clone refactoring dataset 2) Train and evaluate the performance of machine learning algorithms 3) Apply feature selection 4) Evaluate and compare the performance of machine learning algorithms after feature selection

A. Preparation of clone refactoring dataset

To create a machine learning-based model for recommending clones for refactoring, we need a labeled dataset of clone refactoring. The major issue is the unavailability of open-source clone refactoring datasets. To prepare the clone refactoring dataset, the following method is proposed:

1. Detect clones from open-source software using clone detection tools like NICAD, Deckard, etc.
2. Track changes of detected clones in various versions of open-source software that the clones had undergone during clone evolution.
3. If any clone was refactored using the extract method, pull-up method, extract class, and extract superclass, save that clone in a database of refactored clones in history.
4. Collect clones as described in step 3 as many as possible to build a large dataset.
5. Collect clones that did not experience any refactoring during clone evolution. This step helps to build a balanced clone refactoring dataset.
6. Extract features of clones to prepare a dataset of refactored and not refactored clones.

B. Train and evaluate the performance of machine learning algorithms

We proposed to train various machine learning algorithms like Random forest, Random tree, Adaboost, Decision tree, Bagging, LogitBoost, etc. for the classification of refactorable and non-refactorable clones. After training the models, evaluate the performance of these machine learning-based models using evaluation metrics like precision, recall, and F-measure.

C. *Apply feature selection*

Feature selection methods help to choose optimal features from the dataset. We proposed to use a filter-based feature selection method that is based on Correlation, ReliefF, and Infogain. After applying the filter-based feature selection method, the optimal subset of features is selected from the dataset. The classifiers are trained using optimal features.

D. Evaluate and compare the performance of machine learning algorithms after feature selection

The last step evaluates and compares the performance of filter-based feature selection methods. The selection of optimal features will help to improve the accuracy and prediction time of machine learning models for recommending clones for refactoring.

IV. DISCUSSION

We proposed a filter-based feature selection technique to enhance the performance of different classifiers for recommending clones for refactoring. We can apply the proposed technique to clone refactoring datasets. We proposed to analyze the performance of a filter-based feature selection method that is based on Correlation, Infogain, and ReliefF on different classifiers. First, we can analyze the performance of all classifiers

on a complete dataset without feature selection. Then, we can apply the proposed feature selection method to evaluate the performance of different classifiers.

V. CONCLUSION

Our work proposed a technique to improve the performance of various machine learning algorithms to recommend suitable clones for refactoring. The technique is based on filter-based feature selection that uses Correlation, Infogain, and ReliefF. The proposed technique can improve the accuracy of clone recommendations for refactoring.

REFERENCES

[1]. M. Mondal, C. K. Roy, and K. A. Schneider, "Bug propagation through code cloning: An empirical study," in Proceedings - 2017 IEEE International Conference on Software Maintenance and Evolution, ICSME 2017, 2017, pp. 227–237, doi: 10.1109/ICSME.2017.33.

[2]. C. K. Roy and J. R. Cordy, "Benchmarks for software clone detection: A ten-year retrospective," 25th IEEE Int. Conf. Softw. Anal. Evol. Reengineering, SANER 2018 - Proc., vol. 2018-March, pp. 26–37, 2018, doi: 10.1109/SANER.2018.8330194.

[3]. C. K. Roy and J. R. Cordy, "A survey on software clone detection research," Queen's Sch. Comput. TR, vol. 541, no. 115, pp. 64–68, 2007.

[4]. D. Rattan, R. Bhatia, and M. Singh, "Software clone detection: A systematic review," Inf. Softw. Technol., vol. 55, no. 7, pp. 1165–1199, 2013.

[5]. M. Mondal, C. K. Roy, and K. A. Schneider, "A survey on clone refactoring and tracking," J. Syst. Softw., vol. 159, p. 110429, 2020.

[6]. E. Duala-Ekoko and M. P. Robillard, "Tracking code clones in evolving software," in Proceedings - International Conference on Software Engineering, 2007, pp. 158–167, doi: 10.1109/ICSE.2007.90.

[7]. E. Duala-Ekoko and M. P. Robillard, "Clone tracker: Tool support for code clone management," in Proceedings - International Conference on Software Engineering, 2008, pp. 843–846, doi: 10.1145/1368088.1368218.

[8]. Z. Chen, Y. W. Kwon, and M. Song, "Clone refactoring inspection by summarizing clone refactorings and detecting inconsistent changes during software evolution," J. Softw. Evol. Process, vol. 30, no. 10, pp. 1–24, 2018, doi: 10.1002/smr.1951.

[9]. Y. Higo, T. Kamiya, S. Kusumoto, and K. Inoue, "Refactoring Support Based on Code Clone Analysis," in Product Focused Software Process Improvement: 5th International Conference, PROFES 2004, Kansai Science City, Japan, April 5-8, 2004. Proceedings 5, 2004, pp. 220–233, doi: 10.1007/978-3-540-24659-6_16.

[10]. Y. Higo, T. Kamiya, S. Kusumoto, and K. Inoue, "ARIES: Refactoring support tool for code clone," Proc. - Int. Conf. Softw. Eng., pp. 53–56, 2005, doi: 10.1145/1083292.1083306.

[11]. Y. Higo, S. Kusumoto, and K. Inoue, "A metric-based approach to identifying refactoring opportunities for merging code clones in a Java software system," J. Softw. Maint. Evol. Res. Pract., vol. 20, pp. 435–461, 2008, doi: 10.1002/smr.394.

[12]. S. Schulze, M. Kuhlemann, and M. Rosenmüller, "Towards a refactoring guideline using code clone classification," pp. 1–4, 2009, doi: 10.1145/1636642.1636648.

[13]. E. Choi, N. Yoshida, T. Ishio, K. Inoue, and T. Sano, "Extracting code clones for refactoring using combinations of clone metrics," in IWCS, 2011, pp. 7–13, doi: 10.1145/1985404.1985407.

[14]. M. Mondal, C. K. Roy, and K. A. Schneider, "Automatic identification of important clones for refactoring and tracking," in Proceedings - 2014 14th IEEE International Working Conference on Source Code Analysis and Manipulation, SCAM 2014, 2014, pp. 11–20, doi: 10.1109/SCAM.2014.11.

[15]. W. Wang and M. W. Godfrey, "Recommending clones for refactoring using design, context, and history," in 2014 IEEE International Conference on Software Maintenance and Evolution, 2014, pp. 331–340.

[16]. S. Rongrong, Z. Liping, and Z. Fengrong, "A Method for Identifying and Recommending Reconstructed Clones," in Proceedings of the 2019 3rd International Conference on Management Engineering, Software Engineering and Service Sciences, 2019, pp. 39–44.

[17]. A. M. Sheneamer, "An Automatic Advisor for Refactoring Software Clones Based on Machine Learning," IEEE Access, vol. 8, pp. 124978–124988, 2020, doi: 10.1109/ACCESS.2020.3006178.

[18]. R. Yue, Z. Gao, N. Meng, Y. Xiong, X. Wang, and J. D. Morgenthaler, "Automatic clone recommendation for refactoring based on the present and the past," in Proceedings - 2018 IEEE International Conference on Software Maintenance and Evolution, ICSME 2018, 2018, no. 1, pp. 115–126, doi: 10.1109/ICSME.2018.00021.

[19]. M. Fanqi, "Using self organized mapping to seek refactorable code clone," Proc. - 2014 4th Int. Conf. Commun. Syst. Netw. Technol. CSNT 2014, pp. 851–855, 2014, doi: 10.1109/CSNT.2014.177.