2013 17th European Conference on Software Maintenance and Reengineering

A Comparative Study of Supervised Learning Algorithms for Re-opened Bug Prediction

Xin Xia[1]*, David Lo†, Xinyu Wang*, Xiaohu Yang*, Shanping Li*
and Jianling Sun*
*College of Computer Science and Technology, Zhejiang University
Email: {xxkidd,wangxinyu,yangxh,shan,sunjl}@zju.edu.cn
†School of Information Systems, Singapore Management University
Email:davidlo@smu.edu.sg

***Abstract*—Bug fixing is a time-consuming and costly job which is performed in the whole life cycle of software development and maintenance. For many systems, bugs are managed in bug management systems such as Bugzilla. Generally, the status of a typical bug report in Bugzilla changes from new to assigned, verified and closed. However, some bugs have to be reopened. Reopened bugs increase the software development and maintenance cost, increase the workload of bug fixers, and might even delay the future delivery of a software.**

Only a few studies investigate the phenomenon of reopened bug reports. In this paper, we evaluate the effectiveness of various supervised learning algorithms to predict if a bug report would be reopened. We choose 7 state-of-the-art classical supervised learning algorithm in machine learning literature, i.e., kNN, SVM, SimpleLogistic, Bayesian Network, Decision Table, CART and LWL, and 3 ensemble learning algorithms, i.e., AdaBoost, Bagging and Random Forest, and evaluate their performance in predicting reopened bug reports. The experiment results show that among the 10 algorithms, Bagging and Decision Table (IDTM) achieve the best performance. They achieve accuracy scores of 92.91% and 92.80%, respectively, and reopened bug reports F-Measure scores of 0.735 and 0.732, respectively. These results improve the reopened bug reports F-Measure of the state-of-the-art approaches proposed by Shihab et al. by up to 23.53%.

***Index Terms*—bug reports; reopened reports; classification; supervised learning algorithms; comparative study**

I. Introduction

In software development and maintenance process, bug fixing is one of the key activities. Most of the open source software communities and commercial companies have their own bug tracking systems, such as Bugzilla. Various research studies have proposed automated techniques to manage bug repositories, such as bug triaging [1], duplicate bug report detection [2], etc. Generally speaking, a typical bug fixing process can be summarized as the following 4 steps (we take Bugzilla as an example):

1) The tester detected a bug in the software, and published a bug report to describe the bug in Bugzilla. The current bug status is "new";
2) The bug triager assigned this bug to the most appropriate developer. The current bug status is "assigned";
3) The developer read the bug report, and verified whether it is a bug, and tried to fix it. The current bug status is "verified";
4) Finally, after the developer had fixed the bug or verified that it is not a bug, the bug triager closed this bug. The current bug status is "closed";

However, in some cases, a bug has to be reopened. There are various reasons for reopening a bug report [3]; for example, the tester did not provide sufficient information for the bug, the developer misunderstood the root cause of the bug, the bug reappeared in the current version of system although it was fixed in the previous system (i.e., a regression bug). Reopened bugs take much longer time to resolve, and the developers need to re-understand the context of the bugs description and discussion, which increases the work for these already-busy developers [5]. Detecting these reopened bug reports early can reduce maintenance cost and improve reliability of systems.

To our best knowledge, there are a few studies that investigate reopened bug reports; the most related studies are [4], [5], and [3]. Shihab et al. propose the problem of predicting reopened bug reports [4], [5]. Zimmermann et al. comprehensively investigate factors that cause bug reports to be reopened in Microsoft [3]. In this paper, we comprehensively investigate the effectiveness of many supervised learning algorithms for the task of predicting reopened bugs.

In this study, we categorize the supervised learning algorithms into two families: traditional supervised learning algorithms and ensemble supervised learning algorithms. Ensemble learning algorithms combined multiple instances of traditional supervised learning algorithms. For traditional supervised learning algorithms, we evaluate kNN, SVM, SimpleLogistic, Bayesian Network, Decision Table, CART, and LWL; for ensemble supervised learning algorithms, we evaluate AdaBoost, Bagging, and Random Forest.

The main contributions of this paper are as follows:

1) We investigate the effectiveness of 10 state-of-the-art supervised learning algorithms in machine learning literature in predicting reopened bug reports;
2) We demonstrate that Decision Table and Bagging achieve the best performance compared to all other algorithms. The best performing algorithms outperform the result achieved by the algorithm used in [4], [5].

[1]The work was done while the author was visiting Singapore Management University.

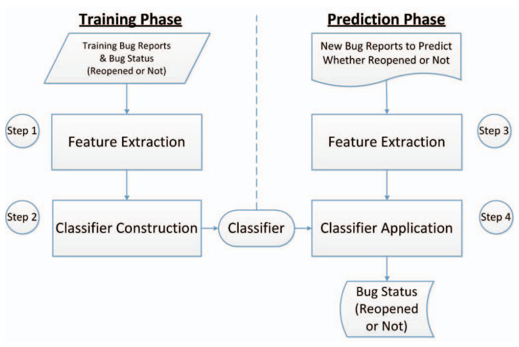

Fig. 1. Proposed Reopened Bug Prediction Framework

TABLE I
FEATURES EXTRACTED FROM BUG REPORTS

Types	Features	Description
TIME	Time	Hour the bug is closed
	Weekday	Day of the week the bug is closed
	Month day	Day of the month the bug is closed
	Month	Month the bug is closed
BF	Time days	Time to solve the bug
	Last status	Last status of the bug
	No. of source files	Number of source code files related to the bug
HUMAN	Reporter Name	The bug reporter name
	Fixer Name	The bug fixer name
	Reporter Experience	Number of bug reports reported by this reporter
	Fixer Experience	Number of bug reports fix*ed* by this fixer
BR	Component	Component affected by the bug
	Platform	Platform affected by the bug
	Severity	Severity of the bug report
	Priority	Priority of the bug report
	CC list Number	Number of people in CC list
	Description size	Number of words in description text
	Description text	Description text content
	No. of comments	Number of comments
	Comment size	Number of words in comment text
	Comment text	Comment text content
	Priority changed	Whether the priority was changed

The remainder of the paper is organized as follows: In Section II, we present a high-level view of how to use learning algorithms to predict reopened bug reports, what features can be extracted to discriminate bug reports that are reopened from those that are not, and what algorithms we investigate to predict re-opened bugs. In Section III, we report the results of our experiment which compares those algorithms for predicting reopened bugs; In Section IV, we present related studies; In Section V, we conclude and mention future work.

II. OVERALL FRAMEWORK

Figure 1 shows our reopened bug report prediction framework. The whole framework includes two phases: training phase and prediction phase. In the training phase, our goal is to build a classifier from the historical bug reports which have known status (i.e., reopened or not). In the prediction phase, this classifier would be used to predict if an unknown bug report would be reopened or not.

Our framework first extracts features from the set of training bug reports (i.e., bug reports with known status) (Step 1). Features are various quantifiable characteristics of bug reports that could potentially differentiate reports that are reopened from those that are not. In this paper, we consider 4 types of features as described in [4]: time related (TIME), bug fix related (BF), human related (HUMAN) and bug report related (BR) features. These features are briefly described in Table I.

Next, our framework constructs a classifier based on features of the training bug reports (Step 2). A classifier is a machine learning model which assigns labels (in our case: reopened or not) to a data point (in our case: a bug report) based on its features. The classifier construction phase would compare and contrast the features of bug reports that are reopened, and those of bug reports that are not. Various thresholds or rules would then be learned, and these are stored in the constructed classifier.

TABLE II
PARAMETER SETTING FOR DIFFERENT SUPERVISED LEARNING ALGORITHMS

Algorithm	Parameter Setting
kNN	Set number of neighbors (i.e., k) as 5, use Euclidean distance
SVM	Use sequential minimal optimization algorithm (SMO), set the kernel as normalized polynomial kernel
SLR	Use the algorithm proposed in [12]
BN	Use TAN algorithm [13] to generate the learning Bayesian network
IDTM	Set the search strategy as BestFirst
CART	Use CART-pruned strategy
LWL	Consider all neighbors, use C4.5 as base classifier
AdaBoost	Use C4.5 as base classifier, set number of iterations as 10, use re-sample strategy
Bagging	Use C4.5 as base classifier, set number of iterations as 10
RF	Set number of trees to 10, use C4.5 as the base decision tree algorithm

There are various supervised learning algorithms that could be used for this step. The goal of our study is to investigate the effectiveness of various algorithms to predict if bug reports get reopened. The algorithms grouped into two families: traditional and ensemble algorithms, respectively. The traditional classification algorithms include K-Nearest Neighbor (kNN) [6], Support Vector Machine (SVM) [6], SimpleLogistic (SLR) [7], Bayesian Network (BN) [6], Decision Table (IDTM) [6], CART [6], and Locally-Weighted Learning algorithm (LWL) [8]. The ensemble classification algorithms include Adaptive Boosting (AdaBoost) [9], Bootstrap Aggregating (Bagging) [10], and Random Forest [11].

In the prediction phase it is then used to predict whether a bug report with unknown label would be reopened or not. For each such bug report, we first extract features from it (Step 3). We then input the features to the classifier in the classifier application step (Step 4). This step would output the prediction result which is one of the following labels: reopened or not reopened.

III. EXPERIMENTS AND RESULTS

In this section, we evaluate the 10 supervised learning algorithms to predict if reopened bug reports. The experimental environment is an Intel(R) Core(TM) i5 3.20 GHz CPU, 4GB RAM desktop running Windows 7 (32-bit).

A. Experiment Setup

We use the dataset collected by Shihab et al. [4], [5]. The whole dataset contains 1,530 bug reports, and 246 bug reports were reopened, while 1,284 were not. We use this dataset to evaluate the effectiveness of the algorithms.

Ten times stratified 10-fold cross validation is used to evaluate the supervised learning algorithms [5], i.e., we repeat 10-fold cross validation 10 times. For each stratified 10-fold cross validation, we randomly divide the dataset into 10 folds, and 9 folds are use to train the classifier, while the remaining 1 fold is used to evaluate the performance, and the class distribution in training and test set is the same as the original dataset to simulate the actual usage of the algorithm. For the evaluation metric, we choose the 7 metrics following the study by Shihab et al. [4], [5]: accuracy, reopened precision (Precision(re)), reopened recall (Recall(re)), reopened F-Measure ((F-Measure(re)), not reopened precision (Precision(nre)), not reopened recall (Recall(nre)), and not reopened F-measure (F-Measure(nre)). Accuracy corresponds to the proportion of bug reports that are correctly labeled. Reopened (not reopened) precision refers to the proportion of bug reports that are correctly labeled among those labeled as reopened (not reopened). Reopened (not reopened) recall refers to the proportion of reopened (not reopened) bug reports that are correctly labeled. F-Measure is the harmonic mean of precision and recall. It could be viewed as a summary measure that combines both precision and recall – it evaluates if an increase in precision (recall) outweighs a reduction in recall (precision).

We use the implementation of the 10 learning algorithms in Weka [14]. Each of the 10 algorithms might take one or more parameters. Table III-A presents the parameter setting for the 10 learning algorithms. For ensemble learning algorithms, such as AdaBoost and Bagging, they combine multiple instances of a supervised learning algorithm. In this paper, unless otherwise stated, we just use C4.5 as the base classifier for AdaBoost and Bagging, although other algorithms can also be used.

We would like to answer the following research questions:

1) What are the performance of the different supervised learning algorithms to predict reopened bug reports?
2) Do ensemble learning methods show better performance than the non ensemble learning methods?

The first research question is the most important one. The answer would shed light on the effectiveness of different supervised learning algorithms to predict reopened bug reports.

B. RQ1: Performance of Different Algorithms

Table III presents the experimental results for different supervised learning algorithms in predicting reopened bug reports. Since reopened bug reports are the minority (c.f., [15]), and the goal of the work is to predict re-opened bugs in the dataset, we focus more on the performance of the algorithms in predicting reopened bug reports, i.e., precision(re), recall(re), and F-measure(re).

Among the 10 algorithms, we notice LWL and kNN have the worst performance – their accuracy scores are 89.90% and 90.52% respectively, and F-Measure(re) scores are 0.673 and 0.600 respectively. Moreover, Random Forest does not show good performance either, its F-Measure for reopened bug reports is only 0.628, which is the second lowest among the 10 algorithms.

On the other hand, Bagging and Decision Table (IDTM) show the best performance, their accuracy scores are 92.91% and 92.80% respectively, and F-Measure scores for reopened bug reports are 0.735 and 0.732 respectively. After Bagging and IDTM, SLR also shows good results, its accuracy score is 92.21%, and F-Measure score for reopened bug reports is 0.714.

Table IV presents the experiments results of the algorithms proposed by Shihab et al. [5]. Four algorithms: Zero, Naive Bayes (NB), Logistic Regression (LR), and C4.5 are extended (by adding a re-sampling/re-weighting strategy) and used. We re-implemented their approach. We notice that the experiment results are little different that what are reported in [5]. This is the case as the ten times 10-folds considered in our experiments when doing multiple cross-validations are randomly created and might be different from the ten 10-folds used in Shihab et al.'s experiments. We use the same ten 10-folds for all the algorithms (Shihab et al.'s and the 10 additional algorithms considered in this paper). It is interesting to notice that our proposed supervised learning algorithms show much better performance. The best F-Measure for re-opened bug prediction in Table III is 0.735, while the best F-Measure is only 0.595 in Table IV (23.53% improvement).

C. RQ2: Performance of Ensemble Learning Algorithms

In the previous sub-section, we choose C4.5 as the base classifier for AdaBoost and Bagging. In this section, we would choose more base classifier for AdaBoost and Bagging to evaluate the performance of those ensemble learning algorithms for predicting reopened bug reports. We choose Naive Bayes, Bayesian Network, C4.5 and Decision Tree (IDTM) as the base classifier for AdaBoost and Bagging. Table V presents the experimental result for AdaBoost and Bagging with different base classifiers.

Considering Table III and V, we notice that the AdaBoost does not achieve better performance than its base classifier for the reopened bug reports, and Bagging slightly improves its base classifier except IDTM.

IV. Related Work

To our best knowledge, there are only three papers analyzing reopened bug reports [4], [3], [5]. Shihab et al. propose the problem of identifying reopened bug reports using machine learning algorithms [4], [5]. In this work, we extend the work of Shihab et al. by evaluating the effectiveness of many supervised learning algorithms for predicting reopened bug reports. Our best algorithms outperform the result reported by Shihab et al. on the same dataset that they used in their paper. We also show that ensemble approach could gain additional performance as compared to the base supervised learning algorithm.

Zimmermann et al. analyze Microsoft Windows operating system project to characterize when bug reports would be reopened [3]. Their analysis follows a mixed-method approach. After they have completed a survey of 358 Microsoft employees, they categorized the primary reasons for reopened bug reports. Then they reinforce the survey result with a large-scale study of Windows bug reports. Finally, logistic regression was used to analyze the relationships of various metrics on bug reports getting reopened. Considering the framework described in Section II, Zimmermann et al. analyze the effectiveness of various feature extractor component (Step 2). On the other hand, in this work, we investigate the effectiveness of various classifier construction techniques (Step 3). We do not use the features proposed by Zimmermann et al. as many of them are

TABLE III
EXPERIMENT RESULTS FOR DIFFERENT SUPERVISED LEARNING ALGORITHMS. PREC = PRECISION. BEST RESULTS IN BOLD.

Algorithm	Accuracy	Prec(re)	Recall(re)	F-Measure(re)	Prec(nre)	Recall(nre)	F-Measure(nre)
kNN	90.52%	0.932	0.443	0.600	0.903	0.994	0.946
SVM	91.26%	0.814	0.591	0.685	0.926	0.974	0.949
SLR	92.21%	0.871	0.606	**0.714**	0.929	0.983	0.955
BN	91.50%	0.785	0.650	0.711	0.935	0.966	0.950
IDTM	92.80%	0.911	0.611	**0.732**	0.930	0.989	0.958
CART	92.15%	0.890	0.584	0.705	0.921	0.986	0.954
LWL	89.90%	0.702	0.647	0.673	0.933	0.947	0.940
AdaBoost	91.61%	0.822	0.612	0.701	0.929	0.974	0.951
Bagging	92.91%	0.922	0.611	**0.735**	0.930	0.990	0.959
RF	91.10%	0.882	0.515	0.650	0.914	0.987	0.949

TABLE IV
EXPERIMENT RESULTS OF APPROACHES PROPOSED BY SHIHAB ET AL. [5]. PREC = PRECISION.

Algorithm	Accuracy	Prec(re)	Recall(re)	F-Measure(re)	Prec(nre)	Recall(nre)	F-Measure(nre)
ZeroR	83.92%	0	0	0	0.839	1	0.913
NB	83.90%	0.499	0.735	0.595	0.944	0.859	0.899
LR	82.00%	0.458	0.668	0.543	0.932	0.849	0.888
C4.5	83.80%	0.499	0.700	0.582	0.937	0.865	0.900

TABLE V
EXPERIMENT RESULTS FOR ADABOOST AND BAGGING WITH DIFFERENT BASE CLASSIFIERS. PREC = PRECISION.

Algorithm	Accuracy	Prec(re)	Recall(re)	F-Measure(re)	Prec(nre)	Recall(nre)	F-Measure(nre)
AdaBoost+NB	86.32%	0.570	0.610	0.589	0.924	0.912	0.918
AdaBoost+BN	90.48%	0.756	0.603	0.671	0.927	0.963	0.944
AdaBoost+C4.5	91.61%	0.822	0.612	0.701	0.929	0.974	0.951
AdaBoost+IDTM	91.77%	0.819	0.627	0.710	0.931	0.973	0.952
Bagging+NB	87.59%	0.600	0.687	0.640	0.938	0.912	0.925
Bagging+BN	91.75%	0.807	0.639	0.714	0.934	0.971	0.952
Bagging+C4.5	92.91%	0.922	0.611	0.735	0.930	0.990	0.959
Bagging+IDTM	92.90%	0.914	0.617	0.736	0.931	0.989	0.958

only available within Microsoft – they are not computed or stored in many Bugzillas of open source programs.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we investigate the effectiveness of 10 supervised learning algorithms in predicting reopened bug reports. The algorithms could be divided into 2 families: traditional algorithms and ensemble algorithms. Experimental results show that among 10 algorithms, Bagging and Decision Table (IDTM) achieve the best performance; Their accuracy scores are 92.91% and 92.80% respectively, and F-measure scores for reopened bug reports are 0.735 and 0.732, respectively. These scores outperform the best result known so far as reported by Shihab et al. [4], [5] by a relative improvement of up to 23.53%. We also show that Bagging, an ensemble algorithm, could improve the performance of its base line traditional learning algorithm when C4.5, Naive Bayes, Bayesian Network are used as base line algorithms.

In this paper, we investigate the performance of supervised learning algorithms using one parameter setting. In the future, we plan to investigate the performance of supervised learning algorithms with various parameter settings and evaluate the performance of more supervised learning algorithms with more reopened bug reports. We also plan to improve reopened bug report prediction further, in particular, to increase the F-Measure(re) score. Moreover, we would also explore other evaluation metrics such as AUC-ROC and P-Opt [16].

Acknowledgment. This work is supported by the Ministry of Industry and Information Technology of China (No. 2010ZX01042-002-003-001). We would like to thank Emad Shihab for sharing us his datasets and for his inputs and advice on how to compare our approaches with the approaches considered in his past work.

REFERENCES

[1] J. Anvik, L. Hiew, and G. Murphy, "Who should fix this bug?" in *ICSE*. ACM, 2006, pp. 361–370.
[2] C. Sun, D. Lo, X. Wang, J. Jiang, and S. Khoo, "A discriminative model approach for accurate duplicate bug report retrieval," in *ICSE*. ACM, 2010, pp. 45–54.
[3] T. Zimmermann, N. Nagappan, P. Guo, and B. Murphy, "Characterizing and predicting which bugs get reopened," in *ICSE*, 2012.
[4] E. Shihab, A. Ihara, Y. Kamei, W. Ibrahim, M. Ohira, B. Adams, A. Hassan, and K. Matsumoto, "Predicting re-opened bugs: A case study on the eclipse project," in *WCRE*. Citeseer, 2010, pp. 249–258.
[5] E. Shihab, A. Ihara, Y. Kamei, W. M. Ibrahim, M. Ohira, B. Adams, A. E. Hassan, and K. ichi Matsumoto, "Studying re-opened bugs in open source software," *Empirical Software Engineering*, September 2012.
[6] J. Han and M. Kamber, *Data mining: concepts and techniques*. Morgan Kaufmann, 2006.
[7] M. Sumner, E. Frank, and M. Hall, "Speeding up logistic model tree induction," *Knowledge Discovery in Databases: PKDD 2005*, pp. 675–683, 2005.
[8] C. Atkeson, A. Moore, and S. Schaal, "Locally weighted learning," *Artificial intelligence review*, vol. 11, no. 1, pp. 11–73, 1997.
[9] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p. 1612, 1999.
[10] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.
[11] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
[12] N. Landwehr, M. Hall, and E. Frank, "Logistic model trees," *Machine Learning*, vol. 59, no. 1, pp. 161–205, 2005.
[13] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine learning*, vol. 29, no. 2, pp. 131–163, 1997.
[14] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten, "The weka data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.
[15] W. Liu and S. Chawla, "Class confidence weighted knn algorithms for imbalanced data sets," *Advances in Knowledge Discovery and Data Mining*, pp. 345–356, 2011.
[16] T. Mende and R. Koschke, "Revisiting the evaluation of defect prediction models," in *Proceedings of the 5th International Conference on Predictor Models in Software Engineering*. ACM, 2009, p. 7.