International Journal of Research in Engineering, Science and Management
Volume-1, Issue-10, October-2018
www.ijresm.com | ISSN (Online): 2581-5782

Bug Triage with Data Reduction Techniques Using Instance Selection Algorithm

Akshay A Uppin[1], Mira V Bhosale[2], A. S. Chadchankar[3]
[1,2]*Student, Department of Information Technology, ZCOER, Pune, India*
[3]*Assistant Professor, Department of Information Technology, ZCOER, Pune, India*

***Abstract*—Bugs are one of the critical issues for any software firm. Most of the software firms pay near about 45 percent value to manage these bugs. An inevitable step of managing bugs is bug triage, which involves assigning new incoming bug to an expert person who can solve this bug. To handle these bugs manually is very time consuming process. To decrease the time in manual work, text classification techniques are applied to conduct automatic bug triage. This paper address the matter of reduction for bug triage, i.e., the way to reduced scale and improve the standard of bug information. We combine instance selection method with feature selection method to reduced information scale on the bug dimension and the word dimension. Then we propose a binary classification method to predict the order of instance selection and feature selection method based on the attributes of historical bug datasets. This method of information reduction will effectively reduce the scale of dataset and improve the accuracy of bug triage technique. Then the bugs are distributed to bug solving experts.**

***Index Terms*—Bug data reduction, feature selection, instance selection, bug triage, prediction for reduction orders**

I. Introduction

Many of the software companies need to deal with large amount of software bugs daily. Software bugs are unavoidable and fixing these software bugs is a very expensive task. In fact, many of the Software organization spend lots of their resources in managing these bugs. For handling this bugs, bug repositories are used. Bugs which are reported by users are stored in the bug repository. In a bug repository, a bug is maintained as a bug report, which records the textual description of reproducing the bug and updates according to the status of bug fixing. Many open source software projects have an open bug repository that permits both users and developers to submit faults or issues in the software and suggest possible improvements. For open source large scale software projects, the number of daily bugs is so large which makes the triaging process very difficult and challenging. There are two challenges related to bug data that may affect the effective use of bug repositories in software development tasks, namely the large scale and the low quality.

When a bug report is formed, a human triage is used to provide this bug to a developer, who will try to fix these bugs. Bug assignment to the developer is also recorded in a bug report. The method of assigning a correct developer for fixing the bug is known as bug triage. Bug triage basically is one of the most time consuming step in managing of bugs in software projects. Manual bug triage is very time consuming process. In traditional bug repositories system, all the large amount of bugs were manually triaged by some specialized developers i.e human triager. Manual bug triage is expensive in time cost and low in accuracy because of large number of daily bugs. To avoid this problem existing system has proposed automatic bug triage approach [1]. This approach applies text classification techniques to predict the developer for bug report. Based on the result of text classification bug is assigned to specialized developer. To improve the result of text.

Classification techniques for bug triage, some further techniques are investigated, e.g., a tossing graph approach [10] and a collaborative filtering approach [13]. Large-scale and low-quality bug data in bug repositories block the techniques of automatic bug triage. This paper proposes data reduction technique for bug triage to reduce the size of bug dataset and improve the quality of bug dataset. Data reduction techniques is used to remove some bug reports and words which are redundant and non-informative. For that we used two methods i.e. instance selection (IS) and feature selection (FS).The order of applying instance selection and feature selection algorithms may also affect the bug triage process. So to decide order, we build a binary classifier based on the historical data set. This process gives us reduced data set that can be used for assigning developer to an incoming bug.

II. Literature Survey

Bug triage aims to assign an appropriate developer to fix anew bug, i.e., to determine who should fix a bug. Cubrani cand Murphy [5] first proposed the problem of automatic bug triage to reduce the cost of manual bug triage. They applied text classification techniques to predict related developers for new incoming bug. In their work a bug report is mapped to a document and an assigned developer is mapped to the label of the document. Then, bug triage is converted into a problem of text classification and then it is automatically solved with the help of any mature text classification techniques, for e.g., Naive Bayes [5]. Based on the results a human triager assigns new bugs by incorporating his/her expertise. Anvik and Murphy [2] extended above work to reduce the effort of bug triage by

creating development-oriented recommenders. Jeong et al. [10] found that over 37 percent of bug reports had been reassigned in manual bug triage. They introduced a graph model, which captures bug tossing history. This graph model reveals developer networks which can be used to discover team structures and to find suitable experts for a new task. Then it helps to assign developers to bug reports, correctly. To avoid low-quality bug reports in bug triage, Xuan et al. [19] trained a semi-supervised classifier by combining unlabeled bug reports with labeled ones. This new approach combines naive Bayes classifier and expectation maximization to take advantage of both labeled and unlabeled bug reports. This approach trains a classifier with a fraction of labeled bug reports. Then the approach iteratively labels numerous unlabeled bug reports and trains a new classifier with labels of all the bug reports. They also employed a weighted recommendation list to boost the performance by imposing the weights of multiple developers in training the classifier. Park et al. [13] converted bug triage into an optimization approach [13]. Large-scale and low-quality bug data in bug repositories block the techniques of automatic bug triage. Problem and propose collaborative filtering approach to reduce the bug fixing time. In a general recommendation problem, content based recommendation (CBR) is well known to suffer from over specialization recommending only the types of bugs that each developer has solved before. This problem is critical in practice, as some experienced developers could be overloaded, and this would slow the bug fixing process. Then they took two directions to address this problem: First, they reformulated the problem as an optimization problem of both accuracy and cost. Second, they adopted a content-boosted collaborative filtering (CBCF), combining an existing CBR with a collaborative filtering recommender (CF), which enhances the recommendation quality of either approach alone.

III. System Architecture

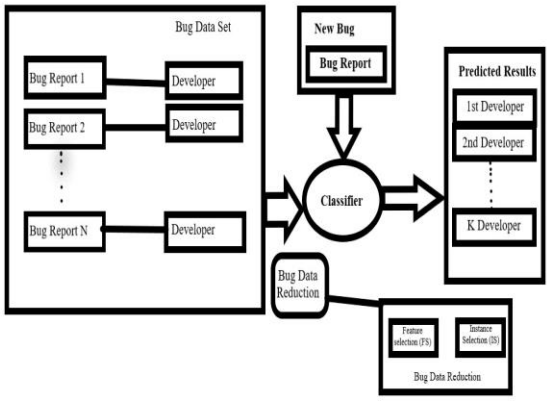

Fig. 1. System Architecture

In the data reduction, removal of bug reports and words are done which are unnecessary and noisy. Here, bug dimension and word dimension are removed simultaneously. For that in proposed system, existing techniques for instance selection (IS) and feature selection (FS) are used. Applying only instance selection gives the reduced bug reports but correctness of bug triage gets decrease. And applying only feature selection gives reduced words of bug data and correctness gets increase. Hence, proposed system combines both techniques which can increase accuracy; also reduce bug reports and words of bug reports. In proposed system, firstly, attributes from historical data set gets extracted. Then, existing algorithm of instance selection gets applied on the data set. It gives the bug reports which have more instances. On the other hand, Feature selection gets applied on the data set. In Feature selection, initially objective value of words of bug reports gets calculated. Then Feature selection selects the features with more objective value. So feature selection creates subset of important Feature. In Proposed system, we are merging the results of these two algorithms. In this merging process, important feature subset from feature selection algorithms gets applied on bug reports of instance selection algorithm. So, we get bug reports which contains important feature. Therefore finally we get reduced bug data set. Then for new bug, we create new bug report. Then Naive Bayes classifier is used and new bug reports gets compared with existing dataset, from this we get bug reports with respect to newly arrived bug. Then, we are considering this data to find top k solution to fix the bug. Therefore Ranking technique is used for that final dataset, to predict top k results. Here, Top k pruning algorithm is used. Bug Solutions will get rank, based on how many times that solution gets updated. As per ranking, we get descending order of top results, to fix bug. Finally, proposed system will predict the top k results for fixing new bug.

Algorithm-1: Data reduction based on FS→IS

The algorithm gives how to decrease the bug data based on FS→IS. Output of data reduction algorithm is a new and decreased data set. Here, two techniques i.e. Feature selection and Instance selection are applied sequentially.

1. Input: training set T with p words and q bug reports,
2. Reduction order FS→IS
3. Final number pF of words,
4. Final number qi of bug reports,

Output: reduced data set TFI for bug triage

Steps:

1. Apply FS to p words of T and calculate objective values for all the words;
2. Select the top pF words of T and generate a training set TF
3. Apply IS to qI bug reports of TF;
4. Terminate IS when the number of bug reports is equal to or less than qI and generate the final training set TFI.

IV. Proposed Work

The proposed framework is categorized into five phases.
It is listed as follows:

A. Data Set Collection

The data is collected from bug repository that contains activities, results, context and other factors. The data collection Plays a very important role to evaluate the classification. The data is stored in the form of statistical matrix where each column represents a specific variable.

1) Preprocessing methods

Data preprocessing is known as data cleaning process. It is processed by resolving missing values, smoothing the noisy data and resolving the inconsistencies. The irrelevant data is also eliminated from the dataset.

2) Feature selection and instance selection

The integration of instance selection and feature selection is employed to reduce the scale of the data. The task of instance selection is to reduce the number of instances by eliminating the noise and redundant sets. By disposing the high dimensional data, an effective bug triage is obtained and Feature selection is used for increase the accuracy.

3) Bug data reduction

The task is to reduce the level of bug data. It also used for improving the accuracy of the bug triage. Concurrently it also reduces the scale and quality of the data.

4) Performance evaluation

The algorithm is used for reduced the high dimensional spaces using some non-representative instances. The quality of bug triage can be measured with the exactness of bug triage to reduce noise and redundancy in bug data sets.

5) Keyword selection

Key words in this areas in order to optimize your site for your keyword list. It has potential to break sites revenue thus key word selection plays extremely important part in SEO process. It is a first step in implementing a SEO strategy.

1. Make the list of keywords for that purpose use keyword resources tool
2. Refine your keyword list
3. Determine how competitive your keyword phrases are.

V. Proposed Work of Project Topic

We address the problem of data reduction for bug triage, i.e., how to decrease the bug data to save the labor cost of developers and improve the quality to facilitate the process of bug triage. Data reduction for bug triage aims to build a small-scale and high-quality set of bug data by removing bug reports and words, which are redundant or non-informative. In our work, we combine techniques of instance selection and feature selection to simultaneously reduce the bug dimension and the word dimension. The reduced bug data contain little bug reports and little words than the original bug data and provide similar information over the original bug data. We evaluate the reduced bug data according to two criteria: the scale of a data set and the accuracy of bug triage. Given an instance selection algorithm and a feature selection algorithm, the order of applying these two algorithms may affect the results of bug triage. we propose a predictive model to determine the order of applying instance selection and feature selection. We refer to such resolution as prediction for reduction orders. Then, we train a binary classifier on bug data report with extracted attributes and predict the order of applying instance selection and feature selection for a new bug data set. The instance selection technique to the data set can decrease bug reports but the accuracy of bug triage may be decreased; applying the feature selection technique can reduce words in the bug data and the accuracy can be increased. Meanwhile, combining both techniques can improve the accuracy, as well as reduce bug reports and words

The primary contributions of our project are as follows:

1. We present the problem of data reduction for bug triage. This issue aims to augment the data set of bug triage in two aspects, namely
   a) To simultaneously decrease the scales of the bug dimension and the word dimension and
   b) To improve the accuracy of bug triage.
2. We propose a combination approach to addressing the issue of data reduction. This can be viewed as an application of instance selection and feature selection in bug repositories.
3. We build a binary classifier to predict the order of applying instance selection and feature selection. To our skill, the order of applying instance selection and feature selection has not been investigated in related domains.

VI. Conclusion

Bug fixing is a cultivated part of software organization. One of the major challenges in process of bug triage is to allocate a skilled developer to fix a new bug. This paper proposes the complete abstraction of an automatic bug triage approach and a framework that removes the issue of bug data to a huge extent. Feature selection and Instance selection techniques are combined to obtain better quality of bug data. Use of NB Classifier is proposed for suggesting a list of expert developers for fixing the bug.

References

[1] J. Xuan, H. Jiang, Y Hu, Z Ren, W. Zou, Z. Luo, andvX. Wu, “Towards Effective Bug Triage with Software Data Reduction Techniques, IEEE transactions on Knowledgeand Data Engineering, 2015.
[2] Mrunal V. Wankhede, Shrutika P. Ingole and Snehal V. Alone,” Bug Triage with Bug Data Reduction” ICRTEST 2017, Volume: 5 Issue: 1.
[3] M. Sri Ramya, G. Malini Devi,” Software Data Reduction Techniques for Effective Bug Triage”, IJETSR,Volume 4, Issue 6, June 2017
[4] Snehal Chopade, Prof. Pournima More,” Effective bug triage with Prim's algorithm for Feature Selection”, ICSPC’17, July 2017
[5] J. Anvik, L. Hiew, and G. C. Murphy, “Who should fix this bug?”inProc. 28th Int. Conf. Softw. Eng., May 2006, pp. 361–370.
[6] J. Anvik and G.C.Murphy,“Reducing the effort of bug report triage:Recommenders for development-oriented decisions,” ACMTrans. Soft. Eng. Methodol., vol. 20, no. 3, article 10, Aug. 2011.
[7] H. Brighton and C. Mellish, “Advances in instance selection for instance-based learning algorithms,” Data Mining Knowl. Discovery,vol. 6, no. 2, pp. 153–172, Apr. 2002.

[8] V. Cerver_on and F. J. Ferri, “Another move toward the minimum consistent subset: A tabu search approach to the condensed nearest neighbor rule,” IEEE Trans. Syst., Man, Cybern.,Part B, Cybern., vol. 31, no. 3, pp. 408–413, Jun. 2001.
[9] D.Cubrani_c and G.C.Murphy,“Automatic bug triage using text categorization,” in Proc. 16th Int. Conf. Softw. Eng. Knowl. Eng., Jun. 2004, pp. 92–97.
[10] A. K. Farahat, A. Ghodsi, M. S. Kamel, “Efficient greedy feature selection for unsupervised learning,” Knowl. Inform. Syst., vol. 35, no. 2, pp. 285–310, May 2013.
[11] J. Han, M. Kamber, and J. Pei, Data Mining: Concepts and Techniques, 3rd ed. Burlington, MA, USA: Morgan Kaufmann, 2011.
[12] G. Jeong, S. Kim, and T. Zimmermann, “Improving bug triage with tossing graphs,” in Proc. Joint Meeting 12th Eur. Softw. Eng. Conf. 17th ACM SIGSOFT Symp. Found. Softw. Eng., Aug. 2009, pp. 111–120.
[13] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen, and K. Torkkola, “LVQ_PAK: The learning vector quantization program package,” Helsinki Univ. Technol., Esbo, Finland, Tech. Rep. A30, 1996.
[14] J. A. Olvera-L_opez, J. F. Mart_ınez-Trinidad, and J. A. Carrasco- Ochoa, “Restricted sequential floating search applied to object selection,” in Proc. Int. Conf. Mach. Learn. Data Mining Pattern Recognit. 2007, pp. 694–702.
[15] Bug Triage with Data Reduction Techniques 35 Pvg’s Coet-Information Technology - 2017-2018.
[16] J. W. Park, M. W. Lee, J. Kim, S. W. Hwang, and S. Kim,“Costriage: A cost-aware triage algorithm for bug reporting systems,” inProc. 25th Conf. Artif. Intell., Aug. 2011, pp. 139–144.
[17] J. C. Riquelme, J. S. Aguilar-Ru_ız, and M. Toro, “Finding representative patterns with ordered projections,” Pattern Recognit., vol. 36, pp. 1009–1018, 2003.
[18] M. Robnik-_Sikonja and I. Kononenko, “Theoretical and empirical analysis of relieff and rrelieff,” Mach. Learn., vol. 53, no. 1/2, pp. 23–69, Oct. 2003.
[19] I. H. Witten, E. Frank, and M. A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, 3rd ed. Burlington, MA, USA: Morgan Kaufmann, 2011.
[20] J. Xuan, H. Jiang, Z. Ren, J. Yan, and Z. Luo, “Automatic bug triage using semi-supervised text classification,” in Proc. 22nd Int. Conf. Softw. Eng. Knowl. Eng., Jul. 2010, pp. 209–214.
[21] Y. Yang and J. Pedersen, “A comparative study on feature selection in text categorization,” in Proc. Int. Conf. Mach. Learn., 1997, pp. 412–420.
[22] D. R. Wilson and T. R. Mart_ınez, “Reduction techniques for instance-based learning algorithms,” Mach. Learn., vol. 38, pp. 257–286, 2000.
[23] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schr€oter, and C. Weiss, “What makes a good bug report?” IEEE Trans. Softw. Eng., vol. 36, no. 5, pp. 618–643, Oct. 2010.