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

Towards Effective Bug Triage with Software Data Reduction Techniques Using Instance Selection

Kunal Rawade[1], Kishor Satpute[2], Amar Chadchankar[3]
[1,2]*Student, Dept. of Information Technology and Engg., Zeal College Of Engineering and Research, Pune, India*
[3]*Asst. Prof., Dept. of Information Tech. and Engg., Zeal College Of Engineering and Research, Pune, India*

***Abstract*—A software bug is a problem which causes a computer program or system to crash or produce invalid output or to behave unintended way. Software bugs are unavoidable. Many software companies have to face large number of software bugs. Bug Triage consumes more time for handling software bugs. It is the process of assigning a new bug to the correct potential developer. In this paper, we deal with the software bugs where large Software Company spent lot many of their cost in the same. The step of fixing the bug is called as bug triage where we correctly assign a developer to a new bug. Here, we address the problem of data reduction for bug triage. The problem of data reduction deal with how to reduce the scale and improve the quality. Hence, we combine instance selection with feature selection both simultaneously to reduce bug dimension and word dimension. We also extract the historical bug data set and predictive model to build new data set. This work provides leveraging techniques on data processing for high quality bug data in the software development.**

***Index Terms*— Bug triage, bug repositories, bug data reduction, feature selection, instance selection, machine learning techniques.**

I. Introduction

Software companies spend over 45 percent of cost infixing bugs. Due to the daily-reported bugs, a large number of new bugs are stored in bug repositories. 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. On one hand, due to the daily-reported bugs, a large number of new bugs are stored in bug repositories. On the other hand, software techniques suffer from the low quality of bug data. Two typical characteristics of low-quality bugs are noise and redundancy. Noisy bugs may mislead related developers while redundant bugs waste the limited time of bug handling.

II. Literature Survey

To avoid the expensive cost of manual bug triage, an automatic bug triage approach was proposed, which applies text classification techniques to predict developers for bug reports. In this approach, a bug report is mapped to a document and a related developer is mapped to the label of the document. Then, bug triage is converted into a problem of text classification and is automatically solved with mature text classification techniques, e.g., Naive Bayes. Based on the results of text classification, a human triage assigns new bugs by incorporating his/her expertise. Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then next steps are to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support.

III. Survey on Research Paper

In this paper [1], Semi-automated approach uses a supervised machine learning algorithm to suggest developers who may be qualified to resolve the bug. It has provided help triage to assigning bugs more efficient. If company has little knowledge then new triage can work on it. Bug triage aims to allocate an appropriate developer to fix a new bug that is to determine who should fix a bug. Author first proposes the problem of automatic bug triage to reduce the cost of manual bug triage. When a new report arrives, the classifier produced by the supervised machine learning technique offered a small number of developers suitable to resolve the report. The process only can work on two projects i.e., Mozilla and Firefox. In this Paper [2], for the text representation and processing a concept of distance graphs is proposed. This paper able to introduce the idea of distance graph representations of text statistics. Such representations preserve facts approximately the relative ordering and distance between the words inside the graphs and offer a far richer illustration in phrases of sentence shape of the underlying facts. This technique permits knowledge discovery from textual content which isn't always viable with using a natural vector area representation, because it loses an awful lot much less records approximately the ordering of the underlying phrases. The detail study of the problems of similarity search, plagiarism detection, and its applications wasn’t specified. In This paper [3], it presents a dynamic test generation technique for the domain of dynamic Web applications. The technique utilizes both combined concrete and symbolic execution and explicit-state model checking. The technique generates tests automatically, runs the tests capturing logical constraints on inputs, and minimizes the conditions on the inputs to failing tests so that the resulting bug reports are small and useful in finding and fixing the underlying faults.

It detects run time error and use HTML validate as on oracle. They perform automated analysis to minimize the size of failure inducing input. In This Paper [4], here proposed an approach where profile is created for each developer based on his previous work and is mapped to a domain mapping matrix which indicates the expertise of each developer in their corresponding area. A key collaborative hub for many software improvement projects is the bug file repository. Although its use can enhance the software program improvement process in some of methods, reports introduced to the repository want to be triaged. A triage determines if a record is meaningful. Significant reviews are then organized for integration into the assignment's improvement system. To assist triages with their work, this article offers a device getting to know method to create recommenders that assist with a ramification of selections aimed at streamlining the improvement method. This paper created using this method have a precision among 70% and ninety eight% over five open source projects. The software developer improves the process of finding the solution in number of way on particular error. It utilizes the expertise profile of developers maintained in Domain Mapping Matrix (DMM).

In This Paper [5], They Mentioned that the bug can be automatically assign to the potential developer for evaluating all the bug report carefully which saves resources used in bug triage or bug assigning task. Bug triage, deciding what to do with an incoming bug report, is taking up increasing amount of developer resources in large open-source projects. In this paper, propose to apply machine learning techniques to assist in bug triage by using text categorization to predict the developer that should work on the bug based on the bug's description. We demonstrate our approach on a collection of 15,859 bug reports from a large open-source project. Our evaluation shows that our prototype, using supervised Bayesian learning, can correctly predict 30 % of the report assignments to developers. The two problems with this approach is sometime the developer who fix the bug is not the one to whom it was officially assigned, second the algorithm does not proved to be as efficient as it was thought to be.

A. *Existing System*

In traditional software development, new bugs are manually triaged by an expert developer, i.e., a human triage. Due to the large number of daily bugs and the lack of expertise of all the bugs, manual bug triage is expensive in time cost and low in accuracy. Conventional software analysis is not fully suitable for the large-scale and complex data in bug repositories. The existing system working is:

1. In traditional software development, bugs were triaged by human, the new bugs were triages by him manually. Triaging the huge number of bugs manually takes much more time and cost for them. To overcome the problem, automatic bug triage system is introduced in the existing system. The automatic bug triage system uses the text classification technique, in which each the each reported bug is assigned to the developer. Developer is mapped to the label of the document containing bugs that are to be resolved. Bug triage is then converted into the problem of text classification and bugs are automatically solved with text classification techniques. For.eg. Naive Bayes. From the results of text classification, a bug triage assigns new bug by incorporating his/her expertise.

B. *Proposed System*

In this paper, we propose a predictive model to determine the order of applying instance selection and feature selection. We address the problem of data reduction for bug triage, i.e., how to reduce the bug data to save the labor cost of developers and improve the quality to facilitate the process of bug triage. We evaluate the reduced bug data according to two criteria: the scale of a data set and the accuracy of bug triage. To avoid the bias of a single algorithm, we empirically examine the results of four instance selection algorithms and four feature selection algorithms. In our work, we combine existing techniques of instance selection and feature selection to simultaneously reduce the bug dimension and the word dimension. The reduced bug data contain fewer bug reports and fewer words than the original bug data and provide similar information over the original bug data.

1. 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.
2. Given an instance selection algorithm and a feature selection algorithm, the order of applying these two algorithms may affect the results of bug triage.
3. Drawn on the experiences in software metrics, we extract the attributes from historical bug data sets. Then, we train a binary classifier on bug data sets with extracted attributes and predict the order of applying instance selection and feature selection for a new bug data set.
4. In the experiments, we evaluate the data reduction for bug triage on bug reports of two large open source projects, namely Eclipse and Mozilla. Experimental results show that applying the instance selection technique to the data set can reduce 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.
5. Meanwhile, combining both techniques can increase the accuracy, as well as reduce bug reports and words. For example, when 50% bug reports and 70% words

are removed, the accuracy of Naive Bayes on Eclipse improves by 2% to 12% and the accuracy on Mozilla improves by 1% to 6%. Based on the attributes from historical bug data sets, our predictive model can provide the accuracy of 71.8% for predicting the reduction order.

C. *Motivation*

Real-world data always include noise and redundancy. Noisy data may mislead the data analysis techniques while redundant data may increase the cost of data processing. In bug repositories, all the bug reports are filled by developers in natural languages. The low-quality bugs accumulate in bug repositories with the growth in scale. Such large-scale and low-quality bug data may deteriorate the effectiveness of fixing bug.

D. *Architecture View*

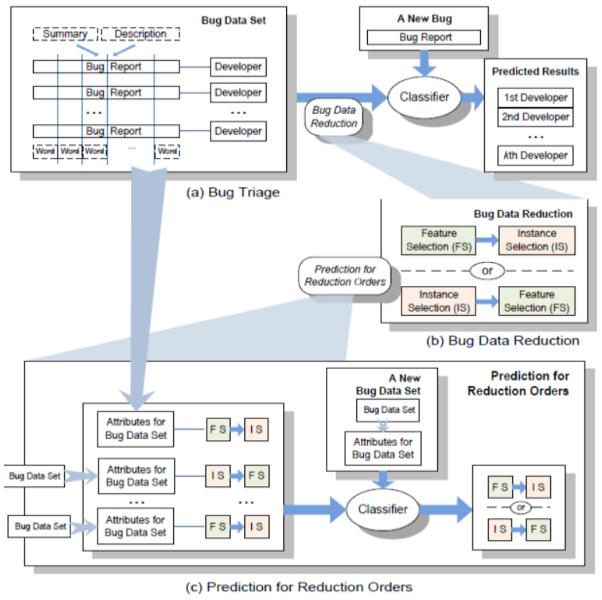

Fig. 1. Architecture view

E. *System Architecture*

Illustration of reducing bug data for bug triage. Sub-figure (a) presents the framework of existing work on bug triage. Before training a classifier with a bug data set, we add a phase of data reduction, in (b), which combines the techniques of instance selection and feature selection to reduce the scale of bug data. In bug data reduction, a problem is how to determine the order of two reduction techniques. In (c), based on the attributes of historical bug data sets, we propose a binary classification method to predict reduction orders.

F. *Problem Definition*

For reducing the affluent cost of manual bug triage we used automatic bug triage method. To build a predictive model for a new bug data sets that present the problem of data reduction for bug triage.

IV. CONCLUSION

This paper presented an all-inclusive survey on the data reduction technique for bug triage. The main features, the advantages and disadvantages of each technique are described. As Bug triage is a vital step of software maintenance in both labor cost and time cost. The goal is to correctly assign a developer to a new bug for further handling. Many software companies spend their most of cost in dealing with these bugs. The motivation of this work is to reduce the large scale of the training set and to remove the noisy and redundant bug reports for bug triage. As per survey, there is strong need to focus on reducing bug data set in order to have less scale of data and quality data. Propose the improved feature selection method by using kruskal model for addressing the problem of data reduction.

REFERENCES

[1] J. Anvik, L. Hiew, and G. C. Murphy, “Who should fix this bug?” in Proc. 28th Int. Conf. Softw. Eng., May 2006, pp. 361–370.
[2] S. Artzi, A. Kie_zun, J. Dolby, F. Tip, D. Dig, A. Paradkar, and M. D. Ernst, “Finding bugs in web applications using dynamic test generation and explicit-state model checking,” IEEE Softw., vol. 36, no. 4, pp. 474–494, Jul./Aug. 2010.
[3] J. Anvik and G. C. Murphy, “Reducing the effort of bug report triage: Recommenders for development-oriented decisions,” ACM Trans. Soft.
[4] Eng. Methodol., vol. 20, no. 3, article 10, Aug. 2011.
[5] C. C. Aggarwal and P. Zhao, “Towards graphical models for text processing,” Knowl. Inform. Syst., vol. 36, no. 1, pp. 1–21, 2013.
[6] Bugzilla, (2014). [Online].Avaialble: http://bugzilla.org/
[7] K. Balog, L. Azzopardi, and M. de Rijke, “Formal models for expert finding in enterprise corpora,” in Proc. 29th Annu.Int. ACM SIGIR Conf.Res. Develop. Inform. Retrieval, Aug. 2006, pp. 43–50.
[8] P. S. Bishnu and V. Bhattacherjee, “Software fault prediction using quad tree-based k-means clustering algorithm,” IEEE Trans. Knowl. Data
[9] Eng., vol. 24, no. 6, pp. 1146–1150, Jun. 2012.
[10] 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.
[11] S. Breu, R. Premraj, J. Sillito, and T. Zimmermann, “Information needs in bug reports: Improving cooperation between developers and users,” inProc. ACM Conf. Comput. Supported Cooperative Work, Feb. 2010, pp. 301–310.
[12] V. Bolon-Canedo, N. S anchez-Marono, and A. Alonso-Betanzos, “A review of feature selection methods on synthetic data,” Knowl. Inform.Syst., vol. 34, no. 3, pp. 483–519, 2013.
[13] Y. Fu, X. Zhu, and B. Li, “A survey on instance selection for active learning,” Knowl. Inform. Syst., vol. 35, no. 2, pp. 249–283, 2013.
[14] I. Guyon and A. Elisseeff, “An introduction to variable and feature selection,” J. Mach. Learn. Res., vol. 3, pp. 1157–1182, 2003.