Efficient Software Bug Triaging Using Software Data Reduction Techniques

Ankita Godse[1], Subhash Pingale[2], Ravindra A. Takalikar[3]
[1] ME- CSE, *SKNSCOE*, [2]Assistant Professor, *SKNSCOE*, [3]Assistant Professor, *SKNSCOE*
[1]*ankita.godse@gmail.com*, [2]*sub.pingale83@gmail.com*, [3]*ravi.takalikar@gmail.com*

Abstract— An inevitable step of fixing bugs is bug triage, which aims to correctly assign a developer to a new bug. To decrease the time and cost of bug triaging, work presents an automatic approach to predict a developer with relevant experience to solve the new coming report. The work combines keyword selection with instance selection to simultaneously reduce data scale on the word dimension and the bug dimension. For applying keyword selection and instance selection, we extracts attributes from existing bug data sets and build a predictive model for a new bug data set. . In addition, work re-balances the load between developers based on their experience; also the new bug report will be assigned with ranking to the predicted list of developers. The work also completes the software testing life cycle.

Keywords— Bug triage, data reduction, Instance selection, Keyword selection, Data Mining, Mining software repositories.

I. INTRODUCTION

A bug plays a vital task in handling software bugs. Various open source software projects attempt an open bug database that allows both developers and users to handle defects or problems in the software, providing possible improvements, and comment on existing bug reports.

The number of regular occurring bugs for open source large-scale software projects is so much large that makes the triaging process very difficult and challenging .For fixing software bugs most of software companies pays a lot . The large scale and the low quality are main two challenges which are related with bug data that may affect the effective use of bug repositories in software development tasks. Bug is maintained as a bug report in a bug repository that records the reproducing bug in textual form and updates according to the status of bug fixing.

A time-consuming step of handling software bugs is bug triage, which aims to assign a correct developer to fix a new bug. In traditional software development, new bugs are manually triaged by an expert developer, i.e., a human triager. 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.

The work addresses 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. 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. The work has the two goals of data reduction as: Reducing the Data Scale & Improving the Accuracy.

II. LITERATURE SURVEY

A semi-automated approach uses a supervised machine learning algorithm to suggest developers who may be qualified to resolve the bug. This approach applies text classification techniques to predict developers for bug reports. Based on the results of text classification, a human triager assigns new bugs by incorporating his/her expertise. [2]

In Developer Prioritization approach a predicted list of developers were predicted by a classifier and those developers were ranked by the priorities after bug triage. A task-based developer prioritization was not applied in bug repositories to improve a specified task with the developer rankings. [3]

While in Profile Oriented Developer Recommendation an approach where profile is created for each developer based on his previous work. This profile is mapped to a domain mapping matrix which indicates the expertise of each developer in their corresponding area. It utilizes the expertise profile of developers maintained in Domain Mapping Matrix (DMM). [4]

To avoid low-quality bug reports in bug triage, a semi-supervised classifier were trained by combining unlabeled bug reports with labeled ones that improves the classification accuracy with both the labeled and unlabeled bug reports. For large bug repositories this technique was little tedious. [5]

III. PROPOSED SYSTEM

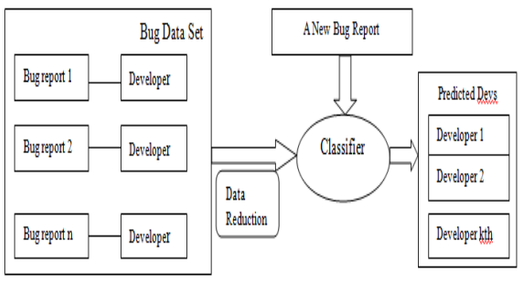


Fig 1: System Architecture

Fig. 1 illustrates the basic framework of bug triage based on text classification. A bug data set contains bug reports with respective developers. On this bug data set the bug data reduction is applied as a phase in data preparation of bug triage. Work combines existing techniques of

instance selection and keyword selection to remove certain bug reports and words.

In bug triage, a bug data set is converted into a text matrix with two dimensions, namely the bug dimension and the word dimension. Work leverages the combination of instance selection and keyword selection to generate a reduced bug data set. Replace the original data set with the reduced data set for bug triage. For a given data set in a certain application, instance selection is to obtain a subset of relevant instances (i.e., bug reports in bug data) while keyword selection aims to obtain a subset of relevant words in bug data. To distinguish the orders of applying instance selection and keyword selection, we give the following denotation. Given an instance selection algorithm as IS and a keyword selection algorithm KS, we use KS!IS to denote the bug data reduction, which first applies KS and then IS.

Algorithm 1: Data reduction based on KS ->IS

Input: training set T with n words and m bug reports,
reduction order KS -> IS
final number nK of words,
final number mI of bug reports,

Output: reduced data set TKI for bug triage

1) Apply KS to n words of T and calculate objective values for all the words;

2) Select the top nK words of T and generate a training set TK;

3) Apply IS to mI bug reports of TK;

4) Terminate IS when the number of bug reports is equal to or less than mI and generate the final training set TKI.

IV. BUG TRIAGE

Work presents 3 modules:

- Manager
- Developer
- Tester

Manager has authorities like to add employee and assign them permissions such as tester/developer, add project details on which he is working and to view all the bugs present in that project so that he can assign an unassigned bug to predicted developer or he can add comments in specific bug.

Developer has authorities like report new bug, check fixed bug for the fixation, view all the bugs he has filled until now also accept the bug after prediction.

Tester has authorities like view all the bugs he has assigned until now, to report new bug, he has authority to edit the bug for fixation or addition of any comment.

WORKING:

1. Manager files details about project such as project name, start date, end date, technology, no. of employees etc.. He adds employee details and assigns permissions such as tester/developer.

2. He can view the prediction order of developers after bug triaging.

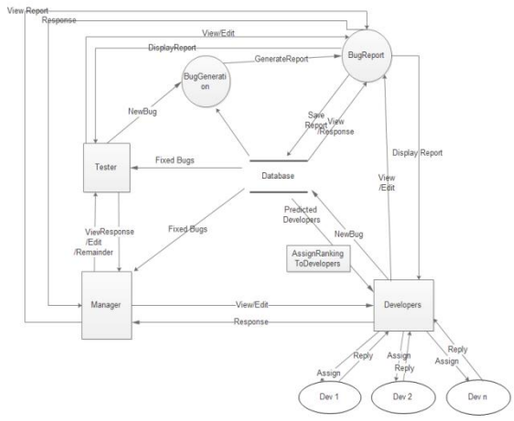


Fig 2: Work Flow of Bug Triage

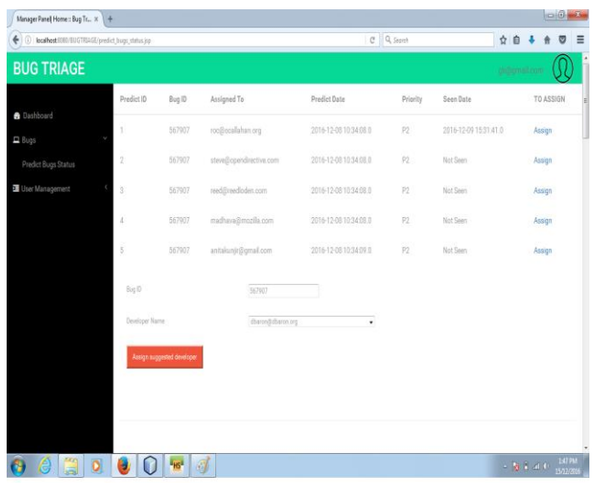


Fig 3: Manager Predict Bug List

3. Testers report new bug and saves in database and displays predicted list of developers on the basis of keyword selection, instance selection and the priority of bug from the history of related bugs. On the basis of this developers have assigned ranking.

For example there is new bug related to memory: 1st it will be checked for the duplicates, rejected bug list using feature and instance selection algos, after that from the reduced list of bugs, history of that bugs will be checked for the priority purpose so that developers could be predicted and assign ranking on that basis.

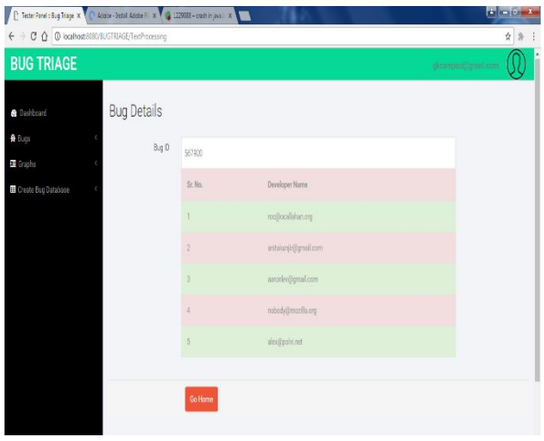


Fig 4: Prediction Result

4. After that top most developer can see his name in prediction order. He checks details of the bug and accept the bug for fixation.

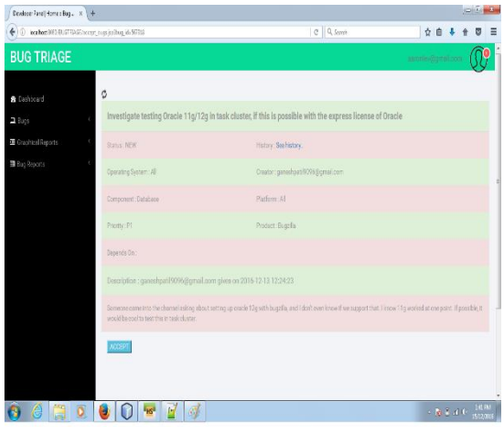


Fig 5: Accept Bug by Developer

V. RESULTS AND DISCUSSIONS

The performance of bug data set can be measured by using both training and test bug data set. In this attributes of each training and test bug data set can be calculated. The attributes are named as bug dataset details as B1 to B10 and developer details as D1 to D8. The pre-processing techniques for data reduction i.e., feature selection and instance selection is applied to in training bug data set.

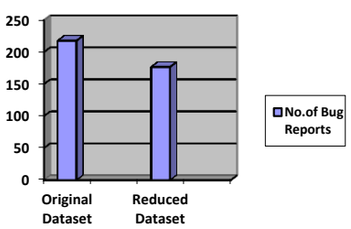
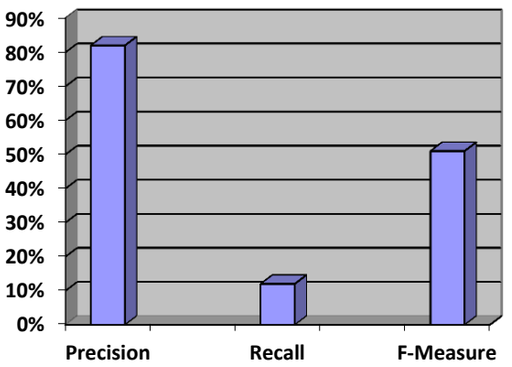


Fig.6: Comparison result between original and reduced bug data set (Training bug data Set)

The training data set contains 218 bug records which give complete information about the bug data stored in large database i.e., Bugzilla. Fig.6 illustrates the comparison between original bug data set and reduced bug data set.



Performance Measure

Fig 7: Comparison Graph for Precision, Recall and F-measure

The classifier i.e., Naive Bayes is trained by training data set with their data reduction order. Then, the classifier is used to predict the correct order to test data set and reduce the labor cost. By this, the bug triage approach is upgraded by their performance. Fig.7 illustrates the precision, recall, and F measure values of Mozilla bug data set are 82%, 12% and 51%. The accuracy is measured as 81%.

VI. CONCLUSIONS

The work presents an approach to automatically assign bug reports to developers with the appropriate expertise. This work combines keyword selection with instance selection to reduce the scale of bug data sets as well as improves the data quality. Also provides an approach for leveraging techniques on data processing to form reduced and high-quality bug data in software development and maintenance.

REFERENCES

[1] J. Xuan, H. Jiang, Z. Ren, J. Yan, and Z. Luo, "Towards Effective Bug Triage with Software Data Reduction Techniques," IEEE Transactions on Knowledge & Data Engineering, Vol. 27, No. 1, JAN 2015.

[2] 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.

[3] J. Xuan, H. Jiang, Z. Ren, and W. Zou, "Developer prioritization in bug repositories," in Proc. 34th Int. Conf. Softw. Eng.(ICSE), 2012, pp. 25–35.

[4] Anjali Sandeep Kumar Singh, "Bug Triaging: Profile Oriented Developer Recommendation," (IJIRAE) ISSN: 2349-2163 Volume 2 Issue 1 (January 2015).

[5] 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.

[6] C. C. Aggarwal and P. Zhao, "Towards graphical models for text processing," Knowl. Inform. Syst., vol. 36, no. 1, pp. 1–21, July 2012.

[7] G. Canfora and L. Cerulo. How software repositories can help in resolving a new change request, in Workshop on Empirical Studies in Reverse Engineering, 2005.

[8] J. Anvik, "Automating bug report assignment," in Proc 28th International Conference on Software Engineering. ACM, 2006, pp. 937–940.

[9] W. Zou, Y. Hu, J. Xuan, and H. Jiang, "Towards training set reduction for bug triage," in Proc. IEEE 35th Annual Computer Software and Applications Conference, Washington, DC, USA: IEEE Computer Society, 2011, pp. 576–581.

[10] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter, and C. Weiss, "What makes a good bug report?" IEEE Trans. Softw. Eng., vol. 36, no. 5, pp. 618–643, Oct. 2010