International Journal of Engineering Research-Online
A Peer Reviewed International Journal
Email:editorijoer@gmail.com http://www.ijoer.in

Vol.4., Issue.2., 2016
(Mar-Apr)

RESEARCH ARTICLE

ISSN: 2321-7758

BUG TRIAGE WITH SOFTWARE DATA REDUCTION TECHNIQUES

SHENBAGA DEVI R[1], KRISHNA MOORTHY M[2]

[1]Department of MCA, Panimalar Engineering College, Chennai

[2]Professor, Department of MCA, Panimalar Engineering College, Chennai

ABSTRACT

Software companies spend over 45 percent of cost in dealing with software bugs. An inevitable step of fixing bugs is bug triage, which aims to correctly assign a developer to a new bug. To decrease the time cost in manual work, text classification techniques are applied to conduct automatic bug triage. The address of problem in data reduction for bug triage, i.e., how to reduce the scale and improve the quality of bug data. Combine instance selection with feature selection to simultaneously reduce data scale on the bug dimension and the word dimension. To determine the order of applying instance selection and feature selection, we extract attributes from historical bug data sets and build a predictive model for a new bug data set. We empirically investigate the performance of data reduction on totally 600,000 bug reports of two large open source projects, namely Eclipse and Mozilla. The results show that our data reduction can effectively reduce the data scale and improve the accuracy of bug triage. Our work provides an approach to leveraging techniques on data processing to form reduced and high-quality bug data in software development and maintenance.

Keywords— Bug triage, Bug data reduction, Data management in bug repositories, Data preprocessing, Feature selection, Instance selection.

I. INTRODUCTION

Many software companies spend most of the money in fixing the bugs. Large software projects have bug repository that collects all the information related to bugs. In bug repository, each software bug has a bug report. The bug report consists of textual information regarding the bug and updates related to status of bug fixing. Once a bug report is formed, a human triager assigns this bug to a developer, who will try to fix this bug. This developer is recorded in an item assigned-to. The assigned to will change to another developer if the previously assigned developer cannot fix this bug.

The process of assigning a correct developer for fixing the bug is called bug triage. Bug triage is one of the most time consuming step in handling of bugs in software projects. Manual bug triage by a human triager is time consuming and error-prone since the number of daily bugs is large and lack of knowledge in developers about all bugs. Because of all these things, bug triage results in expensive time loss, high cost and low accuracy.

The information stored in bug reports has two main challenges. Firstly the large scale data and secondly low quality of data. Due to large number of daily reported bugs, the number of bug reports is scaling up in the repository. Noisy and redundant bugs are degrading the quality of bug reports. In this paper an effective bug triage system is proposed which will reduce the bug data to save the labor cost of developers. It also aims to build a high quality set of bug data by removing the redundant and non-informative bug reports.

II. EXISTING SYSTEM

A.EXISTING CONCEPT

Modeling Bug Data to investigate the relationships in bug data, form a bug report network to examine the dependency among bug reports. This developer social network is helpful to understand the developer community and the project evolution. By mapping bug priorities to developers, identify the developer prioritization in open source bug repositories. The developer prioritization can distinguish developers and assist tasks in software maintenance. Bug triage aims to assign an appropriate developer to fix a new bug .

The problem of automatic bug triage is to reduce the cost of manual bug triage.

They apply text classification techniques to predict related developers. Examine multiple techniques on bug triage, including data preparation and typical classifiers.

B.DRAWBACKS

- ✓ low quality bug report in bug triage.
- ✓ noise and redundancy

III. PROPOSED SYSTEM

A.PROPOSED CONCEPT

This main aim is to simultaneously reduce the scales of the bug dimension and the word dimension and to improve the accuracy of bug triage.

We build a binary classifier to predict the order of applying instance selection and feature selection.

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 extension, we add new attributes extracted from bug data sets, prediction for reduction orders, and experiments on four instance selection algorithms, four feature selection algorithms, and their combinations.

We evaluate the reduced bug data according to two criteria: the scale of a data set and the accuracy of bug triage.

B.ADVANTAGES

- ✓ To reduce the scale of bug dimension and word dimension.
- ✓ To improve the accuracy of bug triage.
- ✓ It reduces the time cost in manual work.

IV. SYSTEM ARCHITECTURE

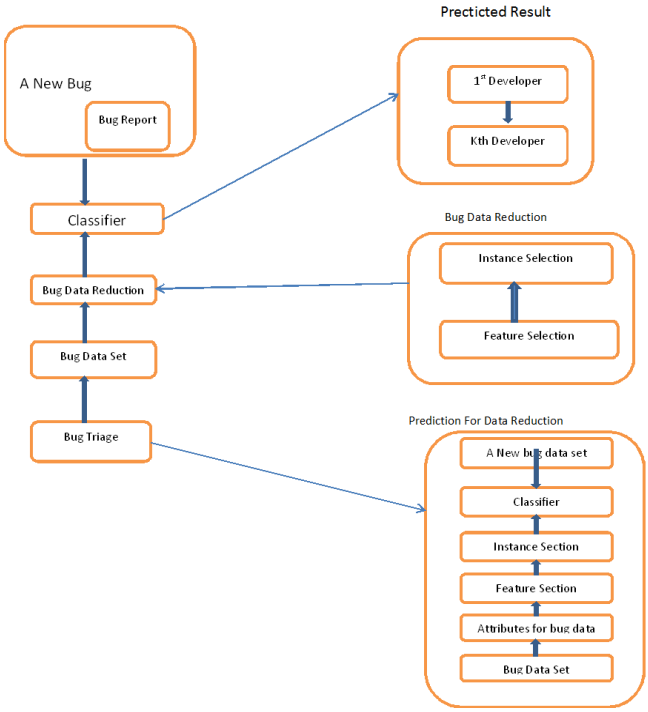

Fig.1

V. MODULE DESCRIPTION

A. DATA REDUCTION

Data reduction is the transformation of numerical or alphabetical digital information derived empirically or experimentally into a corrected, ordered, and simplified form. When the data are already in digital form the 'reduction' of the data typically involves some editing, scaling, coding, sorting, collating, and producing tabular summaries. When the observations are discrete but the underlying phenomenon is continuous then

smoothing and interpolation are often needed. Often the data reduction is undertaken in the presence of reading or measurement errors.

B. DATA REDUCTION FOR BUG TRIAGE

We propose bug data reduction to reduce the scale and to improve the quality of data in bug repositories. We combine existing techniques of instance selection and feature selection to remove certain bug reports and words.

A problem for reducing the bug data is to determine the order of applying instance selection and feature selection, which is denoted as the prediction of reduction orders.

C. APPLYING INSTANCE SELECTION AND FEATURE SELECTION ALGORITHM

The combination of instance selection and feature selection to generate a reduced bug data set. We replace the original data set with the reduced data set for bug triage.

Instance selection and feature selection are widely used techniques in data processing. For a given data set in a certain application, instance selection is to obtain a subset of relevant instances while feature selection aims to obtain a subset of relevant features.

D. REDUCTION ORDER

To avoid the time cost of manually checking both reduction orders, we consider predicting the reduction order for a new bug data set based on historical data sets. We convert the problem of prediction for reduction orders into a binary classification problem.

A bug data set is mapped to an instance and the associated reduction order is mapped to the label of a class of instances.

Note that a classifier can be trained only once when facing many new bug data sets. That is, training such a classifier once can predict the reduction orders for all the new data sets without checking both reduction orders.

VI. OTHER RELATED WORK

A. Modeling Bug Data

To investigate the relationships in bug data, Sandusky et al. Form a bug report network to examine the dependency among bug reports. Besides studying relationships among bug reports, Hong et al.Build a developer social net-work to examine the collaboration among developers based on the bug data in Mozilla project. This developer social net-work is helpful to understand the developer community and the project evolution. By mapping bug priorities to developers, Xuan et al. identify the developer prioritization in open source bug repositories. The developer prioritization can distinguish developers and assist tasks in software maintenance.

To investigate the quality of bug data, Zimmermann et al. design questionnaires to developers and users in three open source projects. Based on the analysis of questionnaires, they characterize what makes a good bug report and train a classifier to identify whether the quality of a bug report should be improved. Duplicate bug reports weaken the quality of bug data by delaying the cost of handling bugs. To detect duplicate bug reports, Wang et al. design a natural language processing approach by matching the execution information; Sun et al. propose a duplicate bug detection approach by optimizing a retrieval function on multiple features.

To improve the quality of bug reports, Breu et al. [9] have manually analyzed 600 bug reports in open source projects to seek for ignored information in bug data. Based on the comparative analysis on the quality between bugs and requirements, Xuan et al. transfer bug data to requirements databases to supplement the lack of open data in requirements engineering.

In this paper, we also focus on the quality of bug data. In contrast to existing work on studying the characteristics of data quality (e.g., [9],) or focusing on duplicate bug reports, our work can be utilized as a preprocessing technique for bug triage, which both improves data quality and reduces data scale.

B. Bug Triage

Bug triage aims to assign an appropriate developer to fix a new bug, i.e., to determine who should fix a bug. Cubranic and Murphy [12] first propose the problem of automatic bug triage to reduce the cost of manual bug triage. They apply text classification techniques to predict related

developers. Anvik et al. [1] examine multiple techniques on bug triage, including data preparation and typical classifiers. Anvik and Murphy [3] extend above work to reduce the effort of bug tri-age by creating development-oriented recommenders.

Jeong et al. find out that over 37 percent of bug reports have been reassigned in manual bug triage. They propose a tossing graph method to reduce reassignment in bug triage. To avoid low-quality bug reports in bug triage, Xuan et al. train a semi-supervised classifier by combining unlabeled bug reports with labeled ones. Park et al. convert bug triage into an optimization problem and pro-pose a collaborative filtering approach to reducing the bug-fixing time.

For bug data, several other tasks exist once bugs are triaged. For example, severity identification aims to detect the importance of bug reports for further scheduling in bug handling; time prediction of bugs models the time cost of bug fixing and predicts the time cost of given bug reports; reopened-bug analysis, identifies the incorrectly fixed bug reports to avoid delaying the software release.

In data mining, the problem of bug triage relates to the problems of expert finding (e.g., [6]) and ticket routing. In contrast to the broad domains in expert finding or ticket routing, bug triage only focuses on assign developers for bug reports. Moreover, bug reports in bug triage are transferred into documents (not keywords in expert finding) and bug triage is a kind of content-based classification (not sequence-based in ticket routing).

C. Data Quality in Defect Prediction

In our work, we address the problem of data reduction for bug triage. To our knowledge, no existing work has investigated the bug data sets for bug triage. In a related problem, defect prediction, some work has focused on the data quality of software defects. In contrast to multiple-class classification in bug triage, defect prediction is a binary-class classification problem, which aims to predict whether a software artifact (e.g., a source code file, a class, or a module) contains faults according to the extracted features of the artifact.

In software engineering, defect prediction is a kind of work on software metrics. To improve the data quality, Khoshgoftaar et al. and Gao et al. [21] examine the techniques on feature selection to handle imbalanced defect data. Shivaji et al. proposes a framework to examine multiple feature selection algorithms and remove noise features in classification-based defect pre-diction. Besides feature selection in defect prediction, Kim et al. present how to measure the noise resistance in defect prediction and how to detect noise data. Moreover, Bishnu and Bhattacherjee [7] process the defect data with quad tree based k-means clustering to assist defect prediction.

In this paper, in contrast to the above work, we address the problem of data reduction for bug triage. Our work can be viewed as an extension of software metrics. In our work, we predict a value for a set of software artifacts while existing work in software metrics predict a value for an individual software artifact.

VII. EXPERIMENTS AND RESULTS

A. Data Preparation

In this part, we present the data preparation for applying the bug data reduction. We evaluate the bug data reduction on bug repositories of two large open source projects, namely Eclipse and Mozilla.

Eclipse [13] is a multi-language software development environment, including an Integrated Development Environment (IDE) and an extensible plug-in system; Mozilla is an Internet application suite, including some classic products, such as the Firefox browser and the Thunderbird email client. Up to December 31, 2011, 366,443 bug reports over 10 years have been recorded to Eclipse while 643,615 bug reports over 12 years have been recorded to Mozilla.

In our work, we collect continuous 300,000 bug reports for each project of Eclipse and Mozilla, i.e., bugs 1-300000 in Eclipse and bugs 300001-600000 in Mozilla.

Actually, 298,785 bug reports in Eclipse and 281,180 bug reports in Mozilla are collected since some of bug reports are removed from bug repositories (e.g., bug 5315 in Eclipse) or not allowed anonymous access (e.g., bug 40020 in

Mozilla). For each bug report, we download web-pages from bug repositories and extract the details of bug reports for experiments.

Since bug triage aims to predict the developers who can fix the bugs, we follow the existing work [1],to remove unfixed bug reports, e.g., the new bug reports or will-not-fix bug reports. Thus, we only choose bug reports, which are fixed and duplicate (based on the items status of bug reports). Moreover, in bug repositories, several developers have only fixed very few bugs. Such inactive developers may not provide sufficient information for predicting correct developers.

In our work, we remove the developers, who have fixed less than 10 bugs.

TABLE:1

	Name	DS-E1	DS-E2	DS-E3	DS-E4	DS-E5
Eclipse	Range of Bug IDs	200001 - 220000	220001 - 240000	240001 - 260000	260001 - 280000	280001 - 300000
	# Bug reports	11,313	11,788	11,495	11,401	10,404
	# Words	38,650	39,495	38,743	38,772	39,333
	# Developers	266	266	286	260	256
	Name	DS-M1	DS-M2	DS-M3	DS-M4	DS-M5
Mozilla	Range of Bug IDs	400001 - 440000	440001 - 480000	480001 - 520000	520001 - 560000	560001 - 600000
	# Bug reports	14,659	14,746	16,479	15,483	17,501
	# Words	39,749	39,113	39,610	40,148	41,577
	# Developers	202	211	239	242	273

To conduct text classification, we extract the summary and the description of each bug report to denote the con-tent of the bug. For a newly reported bug, the summary and the description are the most representative items, which are also used in manual bug triage [1]. As the input of classifiers, the summary and the description are converted into the vector space model [4]. We employ two steps to form the word vector space, namely tokenization and stop word removal. First, we tokenize the summary and the description of bug reports into word vectors. Each word in a bug report is associated with its word frequency, i.e., the times that this word appears in the bug. Non-alphabetic words are removed to avoid the noisy words, e.g., memory address like 0x0902f00 in bug 200220 of Eclipse. Second, we remove the stop words, which are in high frequency and provide no helpful information for bug triage, e.g., the word “the” or “about”. The list of stop words in our work is according to SMART information retrieval system. We do not use the stemming technique in our work since existing work [1], [12] has examined that the stemming technique is not helpful to bug triage. Hence, the bug reports are converted into vector space model for further experiments.

VIII. CONCLUSION AND FUTURE WORK

Bug triage is an expensive step of software maintenance in both labor cost and time cost. In this paper, we combine feature selection with instance selection to reduce the scale of bug data sets as well as improve the data quality. To deter-mine the order of applying instance selection and feature selection for a new bug data set, we extract attributes of each bug data set and train a predictive model based on historical data sets. We empirically investigate the data reduction for bug triage in bug repositories of two large open source projects, namely Eclipse and Mozilla. Our work provides an approach to leveraging techniques on data processing to form reduced and high-quality bug data in software development and maintenance.

In future work, we plan on improving the results of data reduction in bug triage to explore how to prepare a high-quality bug data set and tackle a domain-specific software task. For predicting reduction orders, we plan to pay efforts to find out the potential relationship between the attributes of bug data sets and the reduction orders.

IX. REFERENCES

[1]. Jifeng Xuan, He Jiang, Yan Hu, Zhilei Ren, Weiqin Zou, Zhongxuan Luo, and Xindong Wu, “Towards Effective Bug Triage with Software Data Reduction Techniques”, IEEE transactions on knowledge and data engineering, vol. 27, no. 1, january 2015

[2]. D. Cubrani and G. C. Murphy, “Automatic bug triage using text categorization,” in Proc. 16th Int. Conf. Softw. Eng. Know l. Eng., Jun. 2004, pp. 92–97.

[3]. 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.

[4]. 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.

[5]. K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using EM," Machine Learning. Hingham, MA, vol. 39, pp. 103-134, May 2000.

[6]. M. Alenezi, K. Magel, S. Banitaan "Efficient Bug Triaging Using Text Mining" Academy Publisher, 2013.

[7]. 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.

[8]. A. Tamrawi, T. Nguyen, J. Al-Kofahi, and T. Nguyen, "Fuzzy set and cache-based approach for bug triaging," in Proc. 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, 2011, pp. 365–375.

[9]. 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.

[10]. Q. Shao, Y. Chen, S. Tao, X. Yan, and N. Anerousis, "Efficient ticket routing by resolution sequence mining," in Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2008, pp. 605–613.

[11]. M. D'Ambros, M. Lanza, and M. Pinzger. "A Bug's Life" Visualizing a Bug Database. In Proceedings of IEEE International Workshop on Visualizing Software for Understanding and Analysis (VisSoft 2007), pages 113–120, Banff, Alberta, Canada, 2007. IEEE Computer Society.

[12]. C. A. Halverson, J. B. Ellis, C. Danis, and W. A. Kellogg. "Designing task visualizations to support the coordination of work in software development." In CSCW '06: Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work, pages 39–48, 2006.

[13]. P. Bhattacharya P. and Neamtiu I.: Fine-grained incremental learning and multi-feature tossing graphs to improve bug triaging, in Proc. of ICSM'10, pp.1-10 (2010).

[14]. P. Bhattacharya, L. Neamtiu, C. R. Shelton "Automated, highly-accurate, bug assignment using machine learning and tossing graphs" , 2012.

[15]. V.Akila, Dr.G.Zayaraz, Dr.V.Govindasamy "Effective Bug Triage – A Framework", International Conference on Intelligent Computing, Communication & Convergence, 114 – 120, 2015

[16]. John Karsten Anvik "Assisting Bug Report Triage through Recommendation" The university of British columbia November, 2007

[17]. D. Matter, A. Kuhn, and O. Nierstrasz, "Assigning bug reports using a vocabulary-based expertise model of developers," in Proc. 6th Int. Working Conf. Mining Softw. Repositories, May 2009, pp. 131–140

[18]. O. B. Michael and G. C. Robin, "A Bug You Like: A Framework for Automated Assignment of Bugs.," IEEE 17th international conference, 2009.

[19]. L. Chen and P. Pu. "Survey of preference elicitation methods." Technical report, Swiss Federal Institute of Technology in Lausanne (EPFL), 2004.

[20]. J. W. Park, M. W. Lee, J. Kim, S. W. Hwang, and S. Kim, "Costriage: A cost-aware triage algorithm for bug reporting systems," in Proc. 25th Conf. Artif. Intell., Aug. 2011, pp. 139–144

[21]. A. Tamrawi, T. Nguyen, J. Al-Kofahi, and T. Nguyen, "Fuzzy sat based automatic bug triaging" 2011

[22]. T. Zhang, G. Yang, B. Lee, I. Shin "Role Analysis-based Automatic Bug Triage Algorithm", 2012

[23]. http://ieeeprojects.agplivenews.com/2015-2016-cse-be-btech-in-chennai/java-data-mining/ajjdm1512-towards-effective-bug-triage-with-software/