International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 04 Issue 07
June 2017

Data Reduction and Effective Bug Triage in Software

V. SURESH KUMAR[1] & MR. M. DHARANI KUMAR[2]

[1]M.TECH, DEPT. OF CSE P.V.K.K INSTITUTE OF TECHNOLOGY, ANANTHAPURAMU, AFFILIATED TO JNTUA INDIA.

[2]ASSISTANT PROFESSOR, DEPT. OF CSE P.V.K.K INSTITUTE OF TECHNOLOGY, ANANTHAPURAMU, AFFILIATED TO JNTUA INDIA.

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. In this paper, we address the problem of data reduction for bug triage, i.e., how to reduce the scale and improve the quality of bug data. We 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.

Key words: - Bug Triage; Data Reduction; Bug Repositories; Prediction for Reduction Order; Feature Selection; Instance Selection; Word Dimension; Bug Dimension.

1. INTRODUCTION

MINING software repositories is an interdisciplinary domain, which aims to employ data mining to deal with software engineering quandaries. In modern software development, software repositories are immensely colossal-scale databases for storing the output of software development, e.g., source code, bugs, emails, and designations. Traditional software analysis is not thoroughly opportune for the sizable voluminous-scale and intricate data in

software repositories. Data mining has emerged as a promising denotes to handle software data. By leveraging data mining techniques, mining software repositories can denude fascinating information in software repositories and solve real-world software quandaries. A bug repository (a typical software repository, for storing details of bugs), plays a paramount role in managing software bugs. Software bugs are ineluctably foreordained and fine-tuning bugs is sumptuous in software development. Software companies spend over 45 percent of cost in fine-tuning bugs. Astronomically immense software projects deploy bug repositories (additionally called bug tracking systems) to fortify information amassment and to avail developers to handle bugs. 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 fine-tuning. A bug repository provides a data platform to fortify many types of tasks on bugs, e.g., fault presage, bug localization and reopened bug analysis. In this paper, bug reports in a bug repository are called bug data.

2. RELEGATED WORK

2.1Existing System

A time-consuming step of handling software bugs is bug triage, which aims to assign a correct developer to fine-tune an incipient bug. In traditional software development, incipient bugs are manually triaged by an expert developer, i.e., a human triage. Due to the astronomically immense number of daily bugs and the lack of expertise of all the bugs, manual bug triage is sumptuous in time cost and low in precision. In manual bug triage in Eclipse, percent of bugs are assigned by mistake while the time cost between opening one bug and its first triaging is 19.3 days on average. To evade the extravagant cost of manual bug triage, subsisting work has proposed an automatic bug triage approach, which applies text relegation techniques to prognosticate developers for bug reports. In this approach, a bug report is mapped to a document and a cognate developer is mapped to the label of the document. Then, bug triage is converted into a quandary of text relegation and is automatically solved with mature text relegation techniques, e.g., Verdant Bayes. Predicated on the results of text relegation, a human triager assigns incipient bugs by incorporating his/her expertise. To ameliorate the precision of text relegation

techniques for bug triage, some further techniques are investigated, e.g., a tossing graph approach and a collaborative filtering approach. However, sizably voluminous-scale and low-quality bug data in bug repositories block the techniques of automatic bug triage. .Since software bug data are a kind of free-form text data (engendered by developers), it is compulsory to engender well-processed bug data to facilitate the application

In the subsisting system, we address the quandary of data reduction for bug triage, i.e., how to reduce the bug data to preserve the labor cost of developers and amend the quality to facilitate the process of bug triage. Data reduction for bug triage aims to build a minuscule-scale and high-quality set of bug data by abstracting bug reports and words, which are redundant or non-informative. In our work, we amalgamate subsisting techniques of instance cull and feature cull to simultaneously reduce the bug dimension and the word dimension. The reduced bug data contain fewer bug reports and fewer words than the pristine bug data and provide homogeneous information over the pristine bug data. We evaluate the reduced bug data according to two criteria: the scale of a data set and the precision of bug triage. To eschew the partialness of a single algorithm, we empirically examine the results of four instance cull algorithms and four feature cull algorithm.

2.2Proposed System

In the proposed system, a bug report is mapped to a document and a cognate developer is mapped to the label of the document. Then, bug triage is converted into a quandary of text relegation and is automatically solved with mature text relegation techniques, e.g., Ingenuous Bayes. Predicated on the results of text relegation, a human triager assigns incipient bugs by incorporating his/her expertise. To amend the precision of text relegation techniques for bug triage, some further techniques are investigated, e.g., a tossing graph approach and a collaborative filtering approach. However, astronomically immense-scale and low-quality bug data in bug repositories block the techniques of automatic bug triage. Since software bug data are a kind of free-form text data (engendered by developers), it is indispensable to engender well-processed bug data to facilitate the application.

Since bug triage aims to presage the developers who can fine-tune the bugs, we follow the subsisting work to abstract unfixed bug reports, e.g., the incipient bug reports or will-not-fine-tune bug reports. Thus, we only cull bug reports, which are fine-tuned and duplicate (predicated on the items status of bug reports). Moreover, in bug repositories, several developers have only fine-tuned very few bugs. Such dormant developers may not provide ample information for soothsaying correct developers. In our work, we abstract the developers, who have fine-tuned less than 10 bugs.

3. IMPLEMENTATION

3.1 Admin:

In this module, the Admin has to login by using valid user name and password. After admin login successful he can do some operations such as add domain, add projects, assign projects, view all bugs, list all projects, list all assigned projects, list all users, view searched history.

3.1.1Add domain:

In this module, the admin can add the domain. If the admin want add the domain, he will enter domain name and click on submit button. The details will be stored in the database.

3.1.2 Add projects:

In this module, when the admin wants to add projects, he clicks on add projects and enter project name, project description, domain name, start date, end date and project image and click on submit button the details will be stored in the data base.

3.1.3 Assign project:

In this module, when the admin wants to assign projects, he clicks on assign projects and select project and select developer and click on submit button, then he enters the developer name, project name, project description, domain start time and end time the corresponding details will be assigned successfully.

3.2User

In this module, there are n numbers of users are present. User should register before doing some operations. After registration successful he has to login by using authorized user name and password. Login successful he will do some operations like view my details, view project assigned, view send bug report, view all bugs, list search other bugs, list my searched history. If user clicks on my details button, then the server

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 04 Issue 07
June 2017

will give response to the user with their tags such as UID, developer name, project name, project description, domain, start date, end date, project image..

3.2.1 Send bug reports:

In this module when user clicks on send bug report button, he will select the project and clicks on submit button, then he will enter developer name, project name, project description, domain, start date, end date, select bug type, enter bug description and click on assign button, the corresponding details will be stored.

3.2.2 Feature selection method and instance selection method and view bugs:

In this module when admin click on feature selection method he will get different defect bug type like UI defects, boundary related defects, error handling defects, calculation defects, control flow defect, interpreting data defect, race condition defect and load balancing defect on the basis of details as follows bug ID, developer name, project name, project description, domain, start date, end date, bug type, bug description and status. Click on solution provider select bug ID and enter solution, click on submit button and fix the bug.

4. EXPERIMENTAL RESULTS

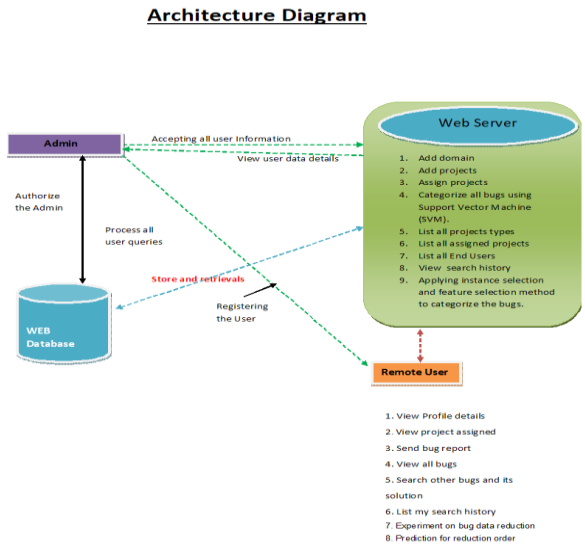

Fig 1 Architecture Diagram

Fig 2 Home Page

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 04 Issue 07
June 2017

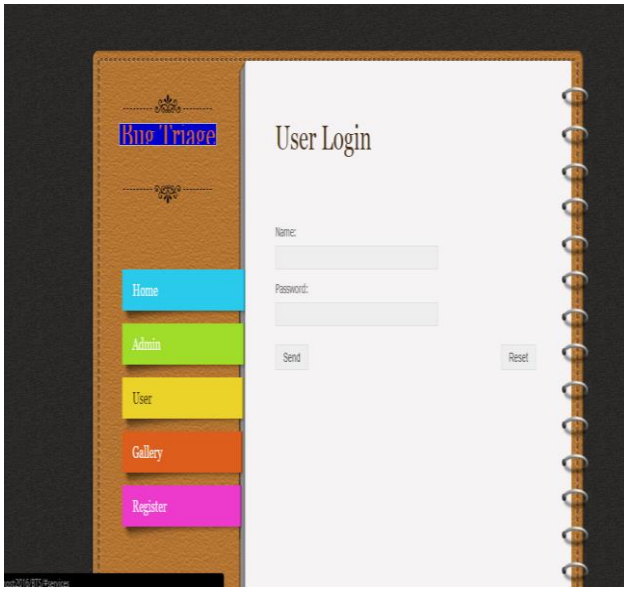

Fig 3 Login Page

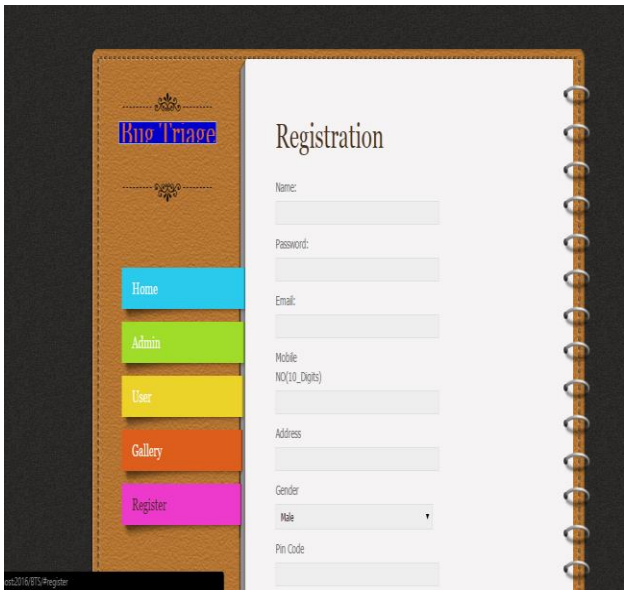

Fig 4 Registration Page

Fig 5 Projects Page

Fig 6 Project Assign Page

5. CONCLUSION

Bug triage is a sumptuous step of software maintenance in both labor cost and time cost. In this paper, we cumulate feature cull

with instance cull to reduce the scale of bug data sets as well as ameliorate the data quality. To determine the order of applying instance cull and feature cull for an incipient bug data set, we extract attributes of each bug data set and train a predictive model predicated on historical data sets. We empirically investigate the data reduction for bug triage in bug repositories of two immensely colossal open source projects, namely Eclipse and Mozilla. Our work provides an approach to leveraging techniques on data processing to compose reduced and high-quality bug data in software development and maintenance. In future work, we orchestrate on ameliorating the results of data reduction in bug triage to explore how to prepare a high quality bug data set and tackle a domain-concrete software task. For soothsaying reduction orders, we orchestrate to pay efforts to ascertain the potential relationship between the attributes of bug data sets and the reduction orders.

6. REFERENCE

[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. Eng. Methodol., vol. 20, no. 3, article 10, Aug. 2011.

[4] C. C. Aggarwal and P. Zhao, "Towards graphical models for text processing," Knowl. Inform. Syst., vol. 36, no. 1, pp. 1–21, 2013.

[5] Bugzilla, (2014). [Online]. Avaialble: http://bugzilla.org/

[6] 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.

[7] P. S. Bishnu and V. Bhattacherjee, "Software fault prediction using quad tree-based k-means clustering algorithm," IEEE Trans. Knowl. Data Eng., vol. 24, no. 6, pp. 1146–1150, Jun. 2012.

[8] 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.

[9] S. Breu, R. Premraj, J. Sillito, and T. Zimmermann, "Information needs in bug reports: Improving cooperation between developers and users," in Proc. ACM Conf. Comput. Supported Cooperative Work, Feb. 2010, pp. 301–310.

[10] V. Bol_on-Canedo, N. S_anchez-Maro~no, and A. Alonso-Betanzos, "A review of feature selection methods on synthetic data," Knowl. Inform. Syst., vol. 34, no. 3, pp. 483–519, 2013.