ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

High Data Reduction Techniques to Maintain Bug Triage

Mr.Podomoni Suresh Kumar
MCA,
CMR College of Engineering & Technology, Hyderabad.

CH.Dayakar Reddy
CMR College of Engineering & Technology, Hyderabad.

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.

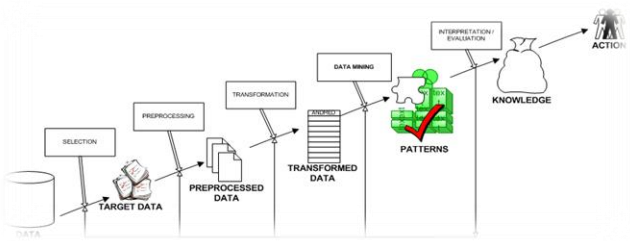
To determine the order of applying instance selection and feature selection, we extract attributes from historical bug data sets and build a predictive model for a new bug data set.

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.

INTRODUCTION

What is Data Mining?



Structure of Data Mining:

Generally, data mining (sometimes called data or knowledge discovery) is the process of analyzing data from different perspectives and summarizing it into useful information - information that can be used to increase revenue, cuts costs, or both. Data mining software is one of a number of analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases.

How Data Mining Works?

While large-scale information technology has been evolving separate transaction and analytical systems, data mining provides the link between the two. Data mining software analyzes relationships and patterns in stored transaction data based on open-ended user queries. Several types of analytical software are available: statistical, machine learning, and neural networks.

Generally, any of four types of relationships are sought:

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Classes: Stored data is used to locate data in predetermined groups. For example, a restaurant chain could mine customer purchase data to determine when customers visit and what they typically order. This information could be used to increase traffic by having daily specials.

lusters: Data items are grouped according to logical relationships or consumer preferences. For example, data can be mined to identify market segments or consumer affinities.

Associations: Data can be mined to identify associations. The beer-diaper example is an example of associative mining.

Sequential patterns: Data is mined to anticipate behavior patterns and trends. For example, an outdoor equipment retailer could predict the likelihood of a backpack being purchased based on a consumer's purchase of sleeping bags and hiking shoes

EXISTING SYSTEM:

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 network to examine the collaboration among developers based on the bug data in Mozilla project. This developer social network 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.

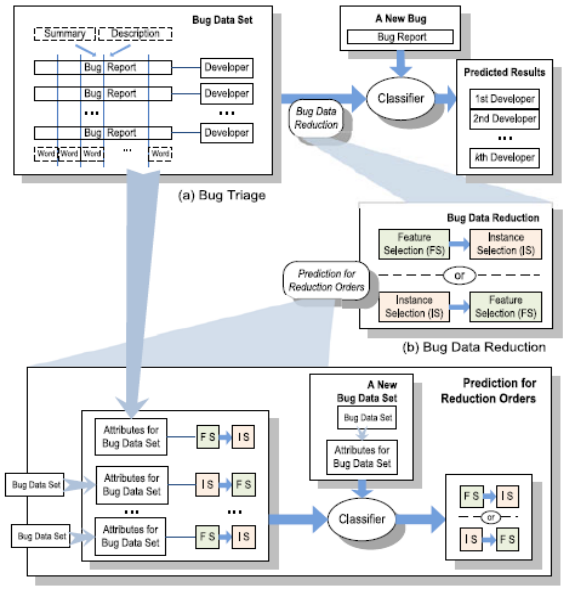
PROPOSED SYSTEM:

In this paper, 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. 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 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.

We evaluate the reduced bug data according to two criteria: the scale of a data set and the accuracy of bug triage. In this paper, we propose a predictive model to determine the order of applying instance selection and feature selection. We refer to such determination as prediction for reduction orders. Drawn on the experiences in software metrics,1 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.

SYSTEM ARCHITECTURE:

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal



IMPLEMENTATION

MODULES:

- Dataset Collection
- Preprocessing Method
- Feature Selection/ Instance Selection
- Bug Data Reduction
- Performance Evaluation

MODULES DESCSRIPTION:

Dataset Collection:

To collect and/or retrieve data about activities, results, context and other factors. It is important to consider the type of information it want to gather from your participants and the ways you will analyze that information. The data set corresponds to the contents of a single database table, or a single statistical data matrix, where every column of the table represents a particular variable. After collecting the data to store the Database.

Preprocessing Method:

Data preprocessing or Data cleaning, Data is cleansed through processes such as filling in missing values, smoothing the noisy data, or resolving the inconsistencies in the data. And also used to removing the unwanted data.

Commonly used as a preliminary data mining practice, data preprocessing transforms the data into a format that will be more easily and effectively processed for the purpose of the user.

Feature Selection/ Instance Selection:

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 is a technique to reduce the number of instances by removing noisy and redundant instances. By removing uninformative words, feature selection improves the accuracy of bug triage. It recovers the accuracy loss by instance selection.

Bug Data Reduction:

The data set can reduce bug reports but the accuracy of bug triage may be decreased. It improves the accuracy of bug triage. It tends to remove these words to reduce the computation for bug triage. The bug data reduction to reduce the scale and to improve the quality of data in bug repositories. It reducing duplicate and noisy bug reports to decrease the number of historical bugs.

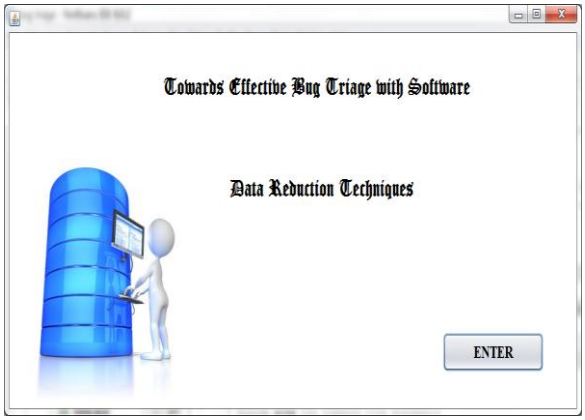
Performance Evaluation:

In this Performance evaluation, algorithm can provide a reduced data set by removing non-representative instances. The quality of bug triage can be measured with the accuracy of bug triage. to reduce noise and redundancy in bug data sets.

Screenshots:

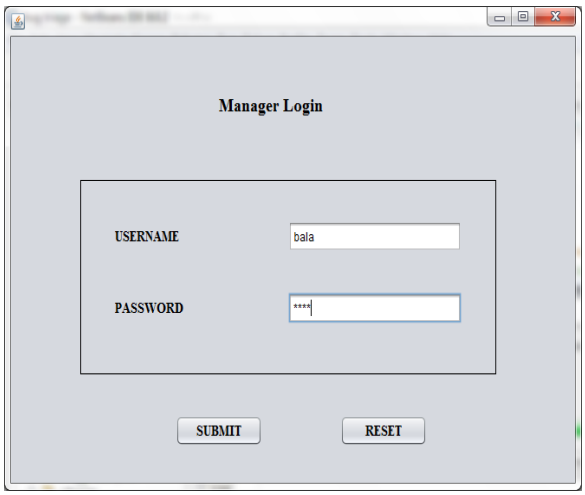
Admin: Towards effective bug triage eith software data reduction techniques

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

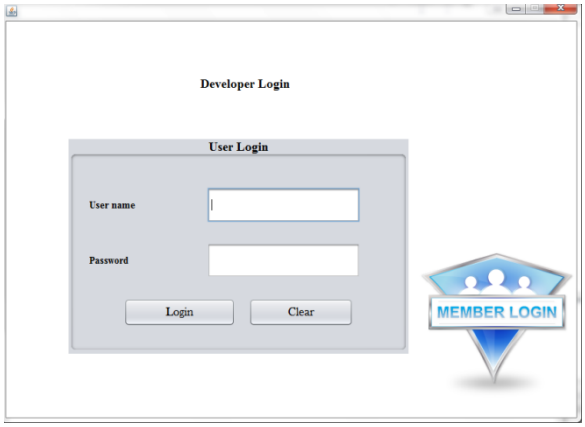


Login: Registered user has credential to login

Manager Login: Manager have credential to manage all the user accounts



Developer login: Developer login is Testing login where they will use admin/admin



CONCLUSION:

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 determine 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 highquality 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.

REFERENCES:

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.

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.

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.

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

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

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.

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.

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.

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.

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.

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.

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.

Eclipse. (2014). [Online]. Available: http://eclipse.org/

B. Fitzgerald, "The transformation of open source software," MIS Quart., vol. 30, no. 3, pp. 587–598, Sep. 2006.

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.