International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 23 Issue 9 – DECEMBER 2016.

BUG RECTIFICATION WITH SOFTWARE DATA REDUCTION USING BUG TRIAGE

Dr. G. P. Saradhi Varma [#1] and Alluri Surya Charan [*2]

[#] *Professor, Head of IT Department, SRKR Engineering College, Bhimavaram. India*

[*] *MTech Student, Department of IT, SRKR Engineering College, Bhimavaram. India*

***Abstract*— We primarily focus the bug reduction system in this project with an assumption that the communication channel between developer and the bug reduction is maintained. We have to prevent redundant bug in the repository. We introduce novel alternative that provides significantly-improved bug report. Users dislike the redundancy of same bug frequently in the bug data, and assign appropriate developer to resolve bug issues. The second approach allows the associated developer to resolve them according to bug classification. This is a tedious assumption, since private data can be exposed by either software bugs or configuration errors at the trusted servers or by malicious administrators. Finally, relying on heavy-weight mechanisms to obtain provable redundant bug report.**

***Index Terms*— Mining software repositories, prediction and reduction technique, application of data preprocessing, data management in bug repositories, bug data reduction, feature selection, instance selection, bug triage.**

I. INTRODUCTION

Mining software repositories is an interdisciplinary domain, which aims to employ data mining to deal with software engineering problems [6]. In modern soft-ware development, software repositories are large-scale databases for storing the output of software development, e.g., source code, bugs, emails, and specifications. Traditional software analysis is not completely suitable for the large-scale and complex data in software repositories [5]. Data mining has emerged as a promising means to handle software data. By leveraging data mining techniques, mining software repositories can uncover interesting information in software repositories and solve real world software problems. A bug repository (a typical software repository, for storing details of bugs), plays an important role in managing soft-ware bugs. Software bugs are inevitable and fixing bugs are expensive in software development. Software companies spend over percent of cost in fixing bugs [3]. Large soft-ware projects deploy bug repositories (also called bug tracking systems) to support information collection and to assist 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 fixing [5]. A bug repository provides a data platform to support many types of tasks on bugs, e.g., fault prediction [7], [4], bug localization [2], and reopened-bug analysis [3]. In this paper, bug reports in a bug repository are called bug data.

II. OBJECTIVES

1. Study of multiple feature selection techniques to classify bugs in software code changes.
2. Use of triage algorithm to utilize accuracy and cost of bug prediction
3. We have focused on reducing bug data set in order to have less scale of data and quality data.
4. Data reduction more in bug triage to explore how to prepare a high quality bug data set.

III. AIM

We are doing data reduction on bug data set which will reduce the scale of the data as well as increase the quality of the data. We are using instance selection and feature selection simultaneously with historical bug data.

IV. RELATED WORK

A. *A survey on instance selection for active learning [1]*

From this paper we Refer-

Active learning aims to train an accurate prediction model with minimum cost by labelling most informative instances. In this paper, we survey existing works on active learning from an instance-selection perspective and classify them into two categories with a progressive relationship: (1) active learning merely based on uncertainty of independent and identically distributed (IID) instances, and (2) active learning by further taking into account instance correlations. Using the above categorization, we summarize major approaches in the field, along with their technical strengths/weaknesses, followed by a simple runtime performance comparison, and discussion about emerging active learning applications and instance-selection challenges therein. This survey intends to provide a high-level summarization for active learning and motivates interested readers to consider instance-selection approaches for designing effective active learning solutions.

B. Towards More Accurate Retrieval of Duplicate Bug Reports [2]

From this paper we Refer-

In a bug tracking system, different testers or users may submit multiple reports on the same bugs, referred to as duplicates, which may cost extra maintenance efforts in triaging and fixing bugs. In order to identify such duplicates accurately, in this paper we propose a retrieval function (REP) to measure the similarity between two bug reports. It fully utilizes the information available in a bug report including not only the similarity of textual content in *summary* and *description* fields, but also similarity of non-textual fields such as *product*, *component*, *version*, etc.

C. Experiments with a New Boosting Algorithm [3]

From this paper we Refer-

In an earlier paper, we introduced a new ―boosting ―algorithm called **AdaBoost** which, theoretically, can be used to significantly reduce the error of any learning algorithm that consistently generates classifiers whose performance is a little better than random guessing. We also introduced the related notion of a ―pseudo-loss‖ which is a method for forcing a learning algorithm of multi-label concepts to concentrate on the labels that are hardest to discriminate. In this paper, we describe experiments we carried out to assess how well **AdaBoost** with and without pseudo-loss, performs on real learning problems. We performed two sets of experiments. The first set compared boosting to Breiman's ―bagging‖ method when used to aggregate various classifiers (including decision trees and single attribute value tests).

We compared the performance of the two methods on a collection of machine-learning benchmarks. In the second set of experiments, we studied in more detail the performance of boosting using a nearest-neighbour classifier on an OCR problem.

D. Finding Bugs in Web Applications Using Dynamic Test Generation and Explicit-State Model Checking [4]

From this paper we Refer-

Web script crashes and malformed dynamically generated web pages are common errors, and they seriously impact the usability of Web applications. Current tools for webpage validation cannot handle the dynamically generated pages that are ubiquitous on today's Internet. We present 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. Our tool Apollo implements the technique for the PHP programming language. Apollo generates test inputs for a Web application, monitors the application for crashes, and validates that the output conforms to the HTML specification. This paper presents Apollo's algorithms and implementation, and an experimental evaluation that revealed 673 faults in six PHP Web applications

E. Information Needs in Bug Reports: Improving Cooperation between Developers and Users [5]

From this paper we Refer-

For many software projects, bug tracking systems play a central role in supporting collaboration between the developers and the users of the software. To better understand this collaboration and how tool support can be improved, we have quantitatively and qualitatively analysed the questions asked in a sample of 600 bug reports from the MOZILLA and ECLIPSE projects. We categorized the questions and analysed response rates and times by category and project. Our results show that the role of users goes beyond simply reporting bugs: their active and ongoing participation is important for making progress on the bugs they report. Based on the results, we suggest four ways in which bug tracking systems can be improved.

F. Efficient Greedy Feature Selection for Unsupervised Learning [6]

From this paper we Refer-

Reducing the dimensionality of the data has been a challenging task in data mining and machine learning applications. In these applications, the existence of irrelevant and redundant features negatively the efficiency and effectiveness of different learning algorithms. Feature selection is one of the dimension reduction techniques which has been used to allow a better understanding of data and improve the performance of other learning tasks. Although the selection of relevant features has been extensively studied in supervised learning, feature selection with the absence of class labels is still challenging task. This paper proposes a novel method for unsupervised feature selection, which efficiently selects features in a greedy manner. The paper is an effective criterion for unsupervised feature selection which measures the reconstruction error of the data matrix based on the selected subset of features. The paper then presents a novel algorithm for greedily minimizing the reconstruction error based on the features selected so far.

G. Another Move Toward the Minimum Consistent Subset: A Tabu Search Approach to the Condensed Nearest Neighbour Rule [7]

From this paper we Refer-

This paper presents a new approach to the selection of prototypes for the nearest neighbour rule which aims at obtaining an optimal or close-to-optimal solution. The problem is stated as a constrained optimization problem using the concept of consistency. In this context, the proposed method uses tabu search in the space of all possible subsets. Comparative experiments have been carried out using both synthetic and real data in which the algorithm has demonstrated its superiority over alternative approaches. The results obtained suggest that the tabu search condensing algorithm offers a very good trade-off between computational burden and the optimality of the prototypes selected.

H. Towards Graphical Models for Text Processing [8]

From this paper we Refer-

The rapid proliferation of the World Wide Web has increased the importance and prevalence of text as a medium for dissemination of information. A variety of text mining and management algorithms have been developed in recent years such as clustering, classification, and indexing and similarity search. Almost all these applications use the well-known *vector-space model* for text representation and analysis. While the vector-space model has proven itself to be an effective and efficient representation for mining purposes, it does not preserve information about the ordering of the words in the representation. In this paper, we will introduce the concept of *distance graph representations* of text data. Such representations preserve information about the relative ordering and distance between the words in the graphs, and provide a much richer representation in terms of sentence structure of the underlying data. Recent advances in graph mining and hardware capabilities of modern computers enable us to process more complex representations of text. We will see that such an approach has clear advantages from a qualitative perspective. This approach enables knowledge discovery from text which is not possible with the use of a pure vector-space representation, because it loses much less information about the ordering of the underlying words.

I. The Design of Bug Fixes [9]

From this paper we Refer-

When software engineers fix bugs, they may have several options as to how to fix those bugs. Which fix they choose has many implications, both for practitioners and researchers?

What is the risk of introducing other bugs during the fix? Is the bug fix in the same code that caused the bug? Is the change fixing the cause or just covering a symptom? In this paper, we investigate alternative fixes to bugs and present an empirical study of how engineers make design choices about how to fix bugs. Based on qualitative interviews with 40 engineers working on variety of products, data from 6 bug triage meetings, and a survey filled out by 326 engineers, we found a number of factors, many of them non-technical, that influence how bugs are fixed, such as how close to release the software is. We also discuss several implications for research and practice, including ways to make bug prediction and localization more accurate.

V. PROPOSED SYSTEM ARCHITECTURE

The primary contributions of this paper are as follows:

1) We present the problem of data reduction for bug triage. This problem aims to augment the data set of bug triage in two aspects, namely a) to simultaneously reduce the scales of the bug dimension and the word dimension and b) to improve the accuracy of bug triage.

2) We propose a combination approach to addressing the problem of data reduction. This can be viewed as an application of instance selection and feature selection in bug repositories.

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

To our knowledge, the order of applying instance selection and feature selection has not been investigated in related domains. This paper is an extension of our previous work. In this extension, we add new attributes extracted from bug data sets, prediction for reduction orders, and experiment son four instance selection algorithms, four feature selection algorithms, and their combinations 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.

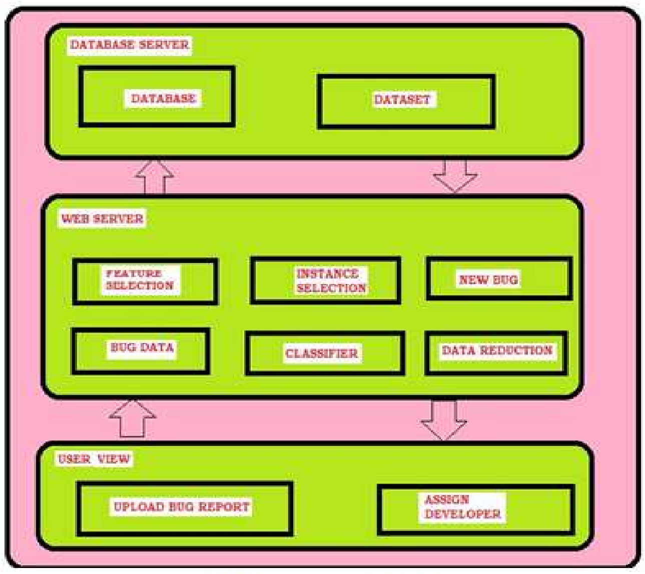

Fig1:Proposed System Architecture.

VI. RESULT ANALYSIS

This module show's four part's as follow:

1) Firstly it will show how many bugs are not assigned to any developer. It will give complete status about the bugs to the admin so that he will come to know which bugs are not assigned yet.

2) Secondly it will show how many bugs are not assigned to any developer. It will give complete status about the bugs to the admin so that he will come to know which bugs are assigned.

3) Thirdly it will show how many bugs are rectified by the developer's. It will give complete status about the bugs to the admin so that he will come to know which bugs are rectified completely.

4) Fourthly it will show how many bugs are not rectified by the developer's. It will give complete status about the bugs to the admin so that he will come to know which bugs are not rectified yet.

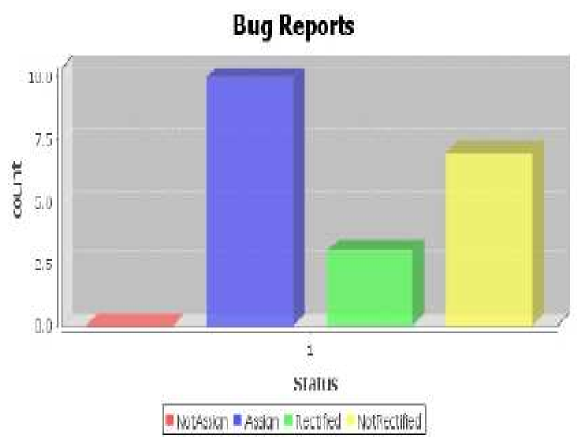

Figure 2: Bug Report

VII. CONCLUSION

From the consideration of all above points we conclude that Bug triage is an expensive step of software maintenance in both labour 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 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.

REFERENCES

[1] Yifan fu · xingquan zhu · bin li‖ a survey on instance selection for active learning‖ 17 march 2012 © springer-verlag london limited 2012
[2] Chengnian sun‖ towards more accurate retrieval of duplicate bug reports‖ http://www.bugzilla.org/
[3] Yoav freund —experiments with a new boosting algorithm‖ machine learning: proceedings of the thirteenth international conference, 1996
[4] Shay artzi, adam kie _zun —finding bugs in web applications using dynamic test generation and explicit-state model checking‖ ieee transactions on software engineering, vol. 36, no. 4, july/august 2010
[5] Silvia breu —information needs in bug reports: improving cooperation between developers and users‖ cscw 2010, february 6–10, 2010, savannah, georgia, usa.
[6] Ahmed k. Farahat —e_cient greedy feature selection for unsupervised learning‖ department of electrical and computer engineering university of waterloo waterloo, ontario, canada n2l 3g1
[7] Vicente cerverón and francesc j. Ferri —another move toward the minimum consistent subset: a tabu search approach to the condensed nearest neighbor rule‖ ieee transactions on systems, man, and cybernetics—part b: cybernetics, vol. 31, no. 3, june 2001
[8] Charu c. Aggarwal1 and peixiang zhao2 —towards graphical models for text processing‖ under consideration for publication in knowledge and information systems emerson murphy-hill —the design of bug fixes‖ 978-1-4673-3076-3/13 c 2013 ieee.