International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 01 | Jan 2020 www.irjet.net p-ISSN: 2395-0072

Data Reduction in Bug Triage using Supervised Machine Learning

Sagarsingh Chauhan[1], Manoj Katre[2], Prof. Tejasvi Jawalkar[3]

[1,2]UG Scholar, Computer Engineering, Dhole Patil College of Engineering, Pune, Maharashtra, India
[3]Professor, Computer Engineering, Dhole Patil College of Engineering, Pune, Maharashtra, India

---

Abstract - The rate at which new bug reports appear in the bug repository, software companies are getting burdened. Every company faces huge amount of time and cost consumption to fix this bugs. The present technique gives solution which reduces this time cost in manual work. Text classification techniques are applied to conduct automatic bug triage, which use huge bug repositories due to which there is tremendous time consumption in the process. To eliminate this time consumption proposed system suggest data reduction for bug triage to create bug repository with small scale and quality set of bug data by removing bug reports and words which are not informative and redundant. Data reduction can be implemented by using techniques such as instance selection and feature selection. The resultant bug repository arrives and the classifier produced by the machine learning technique suggests a developer suitable to resolve the specific bug. Classification can be done by using supervised and unsupervised learning; this will be resulting in high prediction accuracy while reducing training and prediction time

Key Words: (Bug trigging, text classification, data reduction, mining software repositories, machine learning algorithm.

I. 1. INTRODUCTION

Many software companies spend a lot of money repairing project errors. Large software projects have a well-managed error repository that contains all error information. The initial step in the error repository is to handle software errors. Each software error in the error repository includes a detailed report called error data. The error report includes text error information and updates based on the status of the error correction. Traditional software analysis is not fully adapted to complex and large-scale data in software repositories.

Numerous organizations spend overwhelming sum in fixing the bugs of the undertaking. Huge programming ventures have an all around kept up bug store that holds all the data identified with bugs. Beginning advance in bug archive is to oversee programming bugs. Every product bug in bug storehouse has a point by point report called as bug information. The bug report comprises of literary data of the bug and the reports based on the status of bug settling. Customary programming investigation isn't completely suitable for the expansive scale and complex information in programming software.

For the correction of bugs, the classification of bugs is an important step; delivers errors to an important developer to solve it. Once a developer is assigned to the error report, he tries to resolve the error. For open source programming software, many errors are created day after day, making the triage process extremely complicated.

A. Feature Selection

The primary target of highlight determination calculation is to expel the unimportant and excess words from the chose data set. An element choice calculation by and large comprises of steps, for example, subset age, subset assessment, halting basis, and result approval. Highlight choice method is utilized to expel the words in bug reports which are repetitive and non informative.

B. Instance Selection

The goal of the instance selection algorithm is to eliminate noisy data from a data set while maintaining the integrity of the actual data set. The instance selection technique is used to minimize the repeated number of instances by eliminating reports of noisy and collection errors.

2. LITERATURE SURVEY

1. Jifeng Xuan, He Jiang, Yan Hu, Zhilei Ren, Weiqin Zou, Zhongxuan Luo, and Xindongn Wo(2005), "Towards Effective Bug Triage with Software Data Reduction Techniques", In this paper define the benefits of the data reduction techniques. They introduce algorithms which are used to word dimension and bug dimension. The main drawback of this paper is, not all the noise and redundancy are removed as well as the data quality ispoor.0

2. S. Artzi, A. Kie_zun, J. Dolby, F. Tip, D. Dig, A. Paradkar, and M.D. Ernst(2010)"Finding bugs in web applications using dynamic test generation and explicit-state model checking", they introduces a dynamic test generation technique for the dynamic Web applications. The cinque uses both combined concrete and symbolic execution and explicit-state model checking. This technique generates tests automatically, by capturing logical constraints on inputs it runs the tests, and minimizes the conditions on the inputs so that resulting bug reports are small and useful in finding and fixing the underlyingfaults.

3. John Anvik, Lyndon Hiew and Gail c. Murphy "Who should fix This Bug?" In this paper, author represent a semi-automated approach propose to simplicity one part of this process, the task of reports to a developer. The approach uses a machine learning algorithm to the open bug repository to study the kinds of reports each developer

resolves. When a new report arrives, the machine learning technique suggests developers suitable to resolve the bug.

4. S. Breu, R. Premraj, J. Sillito, and T. Zimmermann(2010) "Information needs in bug reports: Improving cooperation between developers and users," In this base paper we learns the relationship between developer and user we have quantitatively and qualitatively examine the questions fromthe MOZILLA and ECLIPSE projects. We categorized the questions and examine response and time by category and project. In this paper results show that the role of users goes more than simply reporting bugs: their active and ongoing participation is important for making progress on the bugs they report.

5.G. Jeong, S. Kim, and T. Zimmermann (2009) "Improving bug triage with tossing graphs," author propose a graph based model Markov chains, which captures bug tossing history. This model has several pleasing qualities. First, it displays developer networks which can be used to find out team structures and to find suitable experts for a new task. Second, it helps to assign expert developers to bug reports. In our experiments with 445,000 bug reports, our model reduced tossing events, by up to 72%.In addition the model increased the prediction accuracy by up to 23 percentage points compared to traditional bug triaging approaches.

6. D. Carbamic and G. C. Murphy(2004) "Automatic bug triage using text categorization," author propose to apply machine learning techniques to assist in bug triage by using text categorization to predict the developer that should work on the bug based on the bug's description. Our approach demonstrates on a collection of 15,859 bug reports from a large open-source project. Our evaluation shows that using supervised Bayesian learning, can correctly predict 30% of the report assignments to developers.

7. C. Sun, D. Lo, S. C. Khoo, and J. Jiang(2011) "Towards more accurate retrieval of duplicate bug reports," author deals with the bug tracking system, different testers submit multiple reports on the same bugs that they encounter, referred to as duplicates, which may cost more efforts in triaging and fixing bugs. In order to detect duplicates accurately, we propose a retrieval function (REP) to measure the similarity between two bug reports. It uses the information available in a bug report including the similarity of textual content in summary and description fields, also similarity of non-textual fields such as product, version, and component.

3. ARCHITECTURE FOR BUG TRIAGE SYSTEM

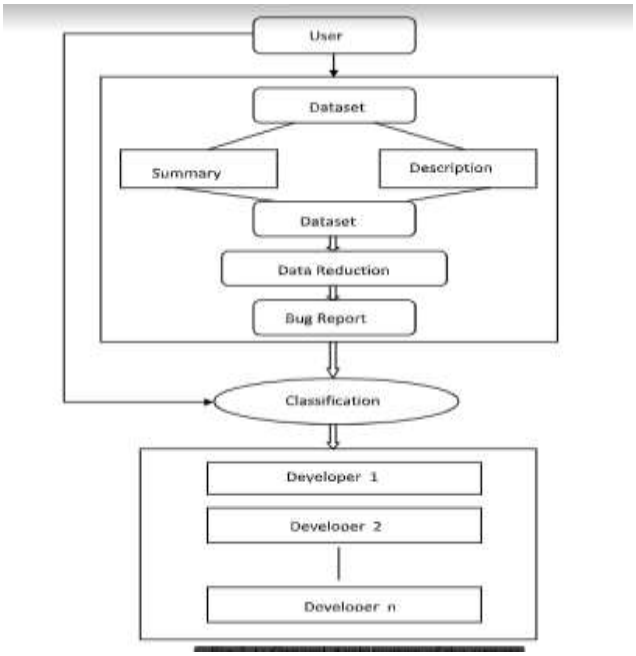

Fig-1 ARCHITECTURE FOR BUG TRIAGE

Reducing data for bug triage plans to build lower and better bug information by eliminating bug reports and unnecessary, repetitive words. To reduce the word dimension and the bug dimension, existing techniques are used such as instance selection and feature selection. The reduced bug data contains fewer bug reports and fewer words than the main bug data and gives virtually identical information about the main bug data. The proposed system evaluates the diminished bug data, which is the size of an illuminating buildup and the accuracy of bug triage.

In the proposed framework the info is as bug informational index. The bug informational Index comprises of bug report and the subtle elements of the designer who have chipped away at the particular bug. The bug report is for the most part isolated in two sections: Summary and Description In bug measurement we diminish loud or copy bug answer to diminish the quantity of verifiable bug. In word measurement to diminish boisterous and copy words. In the framework bug triage use to anticipate the right engineer who can settle the bugs. In framework bug stores, a few designers just settled not very many bugs. Such inert engineer does not give adequate data to task revise designers. In proposed framework we dispense with the engineers, who have settled under 10 bugs.

MODULES

The contribution of the proposed framework is as bug informational collection. The bug informational index comprises of bug report of huge open source venture. We likewise get the all subtle elements of the engineer who have taken a shot at the individual bug. The bug report is principally isolated in two sections: I. Summary and II. Description. The framework is gives anticipated outcomes as yield when decreased informational index.

The flow of the architecture, as given below:

Dataset:

Dataset used for this project is a Bug dataset collected from two large open source projects, namely Eclipse and Mozilla. Eclipse is multi-language software development environment, including an Iterated Development Environment (IDE) and an extensible plug-in system; Mozilla is an Internet application suit. Up to December 31, 2011, 366,443 bug reports over 10 years have been recorded to Mozilla. In this project 37,745 number of Bug reports are collected in continues manner. For each Bug report attributes are considered such as Bug ID, Summary, Description, and Category of each Bug.

Data Reduction based Algorithm:

To perform data reduction combined approach of instance selection and feature selection algorithm is used. In instance selection algorithm it reduced the dimensionality of vocabulary but also used to irreverent terms out of that huge data set. It also reduced the data by removing noisy and irreverent data and provides us with good quality of data. .To select the reduced set of data a preprocessing technique or algorithm feature selection algorithm are used on such large scale data. Feature selection reduces the dimensionality and thus contributes to accuracy and efficient results.

To remove irrelevant and duplicate words in a dataset uses feature selection algorithm. Removing words which is not informative accuracy can be increased of Bug triage. The algorithmic steps are as follow: First select a minimum set of features. Eliminating # of patterns in the data which are easy in understanding. Create new attribute. The attributes catch the essential data. Use the littlest portrayal which is sufficient to settle the task. The estimation of feature weight updates midpoints of the contribution of all the hits and all the misses,

4.RESULTS AND DISSCUSION

4.1 Dataset Loading

In this experiment Dataset is being collected from the Eclipse bug data site named 'Bugzilla' in csv format which is being loaded in the sql database in order to make it appropriate for applying the preprocessing steps to obtain the desired textual data.

4.2 Preprocessing of Dataset

In this Experiment those Bug reports are chosen, which are fixed (based on the items status of bug reports). Moreover, in bug repositories, several developers shave only fixed very few bugs. Since bug triage aims to predict the developers who can fix the bugs, it follows the existing work to remove unfixed bug reports, e.g. the new bug reports or will-not-fix bug reports.

4.3 Data Reduction

In this experiment, data is being fetched from the database and been applied the pre-processing steps, that is taken placed. Now the data is prepared for feature extraction i.e. word dimension reduction is applied on the data in order to reduce the words attributes from the description.

4.4 Detection of Bug category and Assignment of Developer

In this experiment wordsets are formed and labelled according to the name of category of bugs. When new bug is inserted to detect its category it is taken as string and after tokenization of each word is compared with the wordset and Term frequency is calculated. Term frequency is shown in front of each category. After calculating Term frequency, Majority score is calculated among all categories and category of Bug is defined by Majority score.

4.5 Data analysis using K-Nearest Neighbor

Developers history data is fetched from database and according to Analysis Criteria developers are suggested. Clustering level defined number of developers which are sorted using KNN. Euclidian distance is calculated by considering two parameters number of year experience and rating of developer. Then resulted developers are sorted in ascending order, the lowest distance is considered first and according to cluster level N, first N developers are suggested from sorted developers.

5. CONCLUSIONS

Existing systems have disadvantages that fresh bugs are manually triaged by an expert developer. Due to the large number of regular bugs and the lack of expertise of all the bugs, manual bug triage is expensive in time charge and low in accuracy. On the other pointer, software techniques in those systems suffer from the little eminence of bug data.

This Dissertation work combines feature selection with instance selection which is used to reduce the scale of bug data sets as well as recover the data quality. Firstly, bug data set scale is reduced and the classifier produced by the machine learning technique suggests a developer suitable to resolve the specific bug. This dissertation results in assigning new bug to appropriate developer automatically and automatic bug triaging is achieved which eliminate cost of manual bug assignment.

6. 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, January2015

2. S. Artzi, A. Kie_zun, J. Dolby, F. Tip, D. Dig, A. Parrikar, 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. John Anvik, Lyndon Hiew and Gail C. Murphy "Who Should Fix This Bug? "Department of Computer Science University of British Columbia @cs.ubc.ca

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

5. G. Jeong, S. Kim, and T. Zimmermann, "Improving bug triage with tossing graphs," in Proc. Joint Meeting 12 th Eur. Software Engineer Conference 17th ACM SIGSOFT Symp. Found. Software Engineer, Aug. 2009, pp.111–120.

6. D. Cubrani. c and G. C. Murphy, "Automatic bug triage using text categorization," inProc. 16th Int. Conf. Softw. Engineer Knowl. Engineer, Jun. 2004, pp. 92–97.

7. C. Sun, D. Lo, S. C. Khoo, and J. Jiang, "Towards more accurate retrieval of duplicate bug reports," in Proc. 26 th IEEE/ACM Int. Conf. Automated Softw. Engineer, 2011, pp.253–262.