Imperial Journal of Interdisciplinary Research (IJIR)
Vol-2, Issue-5, 2016
ISSN: 2454-1362, http://www.onlinejournal.in

Bug Triage Using Data Reduction Technique

Prof. Amruta Gadekar, Ms. Nikita Waghmare, Ms. Pranjali Taralkar & Mr. Rahul Dapke

[1] *Professor, Dr. D.Y. Patil Institute of engineering and technology, pune, Maharashtra, India*
[2,3,4] *Student, Dr. D. Y. Patil Institute of engineering and technology, pune, Maharashtra, India*

Abstract: *Recently software repositories are large-scale databases for storing the output of software development e.g source code, bugs, emails, and specifications. In this paper address the problem of data reduction for bug triage. Bug triage is a fixing bug which aims correctly assign a developer to a new bug. This paper combine instance selection with feature selection to simultaneously reduce data scale. Instance selection reduces the bug dimension and feature selection reduces the word dimension. Historical bug data set is used to determine the order of applying instance selection and feature selection and build a predictive model for a new bug data set. The results show that data reduction can effectively improve the accuracy of bug triage by reducing the data scale. In software development and maintenance this project provides advanced techniques on data processing to form reduced and high-quality data. Data reduction for bug triage aims to built small scale and high quality set of bug data by removing bud reports and words, which are redundant and non informative[1].*

Key Words: bug repositories, bug triage, bug data reduction, feature selection, instance selection

1. Introduction

In traditional software analysis large scale and complex software repositories is not sutaible. A bug depository plays an important role in managing software bugs. In bug depository bug is maintained as bug report. Bug reports are in a form of textual description of bugs and updates according to status of bug finding[1]. This paper address the problem of data reduction for bug triage, it means how to reduce the bug data to save the labor cost of developers and improve the quality for the process of bug triage. Large scale and the low quality are two challenges related to bug data [1],[3]. This paper combine instance section and features selection to reduce the large data set. 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. The order of applying instance section and features selection may affect the results of bug triage. This paper propose a predictive model to determine order of applying instance selection and features selection. On one hand, the daily-reported bugs, a huge number of new bugs are stored in bug repositories. For open source project like Eclipse, as an example, an average of new 30 bugs are reported in bug repositories for every day in 2007; It is a challenge to by hand examine such large-scale bug data in software development. Due to the large amount of daily bugs manual bug triage is luxurious in time cost and low in accuracy. We propose a combination approach to addressing the problem of data reduction. Bug reports are imperative for any software development [2] .To avoid the bugs of a software, we empirically observe the results of instance selection algorithms and feature selection algorithm.

2. Background

A bug repository is a typical software repository, for storing details of bugs, It plays an important role. Bug depositories are widely used for maintaining software bugs, e.g., a popular and open source bug repository, Bugzilla [5] A recorded bug is called a bug information, Once a bug report is produced, a human triager assigns this bug to a developer, who will attempt to fix this bug. A developer, who is assigned to a new bug report, starts to fix the bug based on the data of historical bug fixing [6], [4]. Bug triage is a practice of passing the fixing bugs to exact developer. A developer, who is assigned to a inventive bug information, starts to fix the bug based on the information of previous bug fixing. Typically, the developer pays efforts to recognize the new bug report and traditionally fixed bugs as a reference (e.g., searching for similar bugs and applying accessible solutions to the new bug). status of a bug report is changed according to the recent result of handling this bug until the bug is completely fixed. presented work employs the approaches based on text disposition to assist bug triage, e.g.,[7] In such a way, the summary of a bug report are extract as the textual content while the developer who can attach this bug is as the label for classification. It gives low accuracy[1].

3. System Architecture

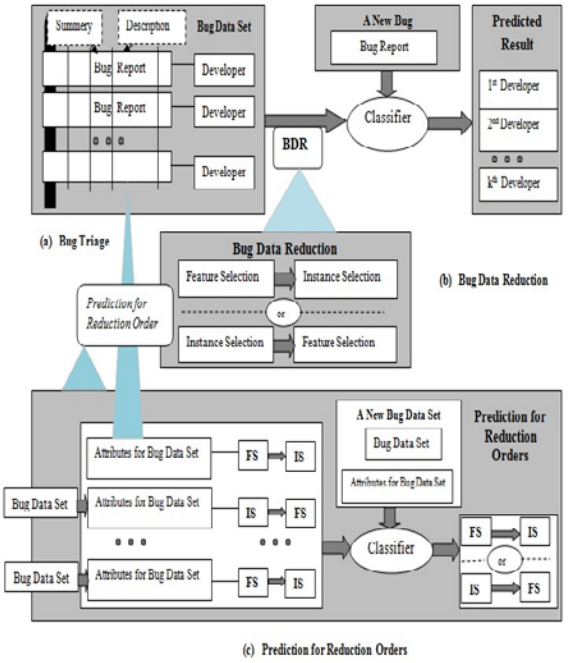

fig. Bug triage with bug data reduction

fig. Illustrate the bug data reduction for bug triage. sub figure (a)present framework for existing work on bug triage.(b)represent combine the instance selection and features selection to reduce the scale of data..(c) describe historical bug data set.

Noise and redundancy include in real world data which may mislead the data analysis techniques and increase the cost of data processing. This design represents a original shift from ordinary selection aid structures. Rather than absolutely deliver records to the give up consumer through question and remedy software, the superior evaluation Server applies customers' enterprise fashions at once to the warehouse and returns a proactive evaluation of the most applicable records .

A. Bug triage

The goal of bug triage is to assign a bug to the correct potential developer. In bug triage, a bug data set is converted into a text matrix with two proportions, namely the bug dimensions and word dimension. Bug reports are in the form of summary and description.

B. Bug Data reduction for bug Triaging

Bug data decrease to compress the scale and to improve the quality of data in bug repositories which is applied as a stage in data preparation of bug triage. existing techniques of instance selection and feature selection are used to remove definite bug reports and words. Classifiers are also used to fix new bug report. It also used to predicate developer to fix bug. It 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.

C. Prediction for Reduction Orders

Due to the large number of daily bugs and the lack of expertise of all the bugs, manual bug triage is expensive in time cost and low in accuracy. To avoid the expensive cost of manual bug triage, existing work has proposed an automatic bug triage approach, which applies text classification techniques to predict developers for bug reports. In this approach, a bug report is mapped to a document and are late developer is mapped to the label of the document. Then, bug triage is converted into a problem of text classification and is automatically solved with mature text classification techniques.

4. Implementation

Bug data decrease to reduce the scale and to improve the quality of data in bug repositories. It is applied as a phase in data preparation of bug triage. obtainable techniques of instance selection and feature selection to eliminate certain bug reports and words. A trouble for reducing the bug data is to determine the order of applying instance selection and feature selection, which is denoted as the prediction of decrease orders. In this section, primary how to concern instance selection and feature selection to bug data, i.e data decrease for bug triage. Then, we list the benefit of the data decrease. The details of the prophecy for reduction orders.

Algorithm 1. Data reduction based on FS → IS

Input: training set $\mathcal{T}$ with n words and m bug reports,
reduction order FS→IS
final number n_F of words,
final number m_I of bug reports,
Output: reduced data set $\mathcal{T}_{FI}$ for bug triage
1) apply FS to n words of $\mathcal{T}$ and calculate objective values for all the words;
2) select the top n_F words of $\mathcal{T}$ and generate a training set $\mathcal{T}_F$;
3) apply IS to m_I bug reports of $\mathcal{T}_F$;
4) terminate IS when the number of bug reports is equal to or less than m_I and generate the final training set $\mathcal{T}_{FI}$.

4.1 Text Representation and Feature Selection

Each document is represent as a vector of words, as is naturally for information retrieval (Salton & McGill, 1983). For most text recovery applications, the entries in the vector are weighted to reflect the frequency of terms in documents and the circulation of terms across the collection as a whole. A popular weighting system is: $w_{ij} = tf_{ij}*idf_i$, where tf_{ij} is the frequency with word i occurs in document j, and idf_i is the contrary document frequency. The tf*idf weight is sometimes used for text categorization (Joachims, 1998), but we have used much simpler binary feature values with good success.

For reasons of both effectiveness and efficiency, feature selection is extensively used when applying machine learning methods to text classification. To reduce the large number of features, we first remove features based on in general frequency counts, and then select a small number of features based on their fit to categories. We used the common information, $MI(X_i, C)$, between each feature, X_i, and the category, C, to select features. $MI(X_i, C)$ is distinct as:

$$MI(X_i, C) = \sum_{x_i = X_i, c = C} P(x_i, c) \log \frac{P(x_i, c)}{P(x_i) P(c)}$$

We select the k features for which reciprocal information is largest for each category. These features are used as input to the SVM learning algorithms.

4.2 Learning Support Vector Machines (SVMs)

We used simple linear SVMs because they give good simplification accuracy and because they are closer to learn. Joachims (1998) has explored two classes of non-linear SVMs, polynomial classifiers and radial origin functions, and has observed only small remuneration compared to linear models. We worn Platt's Sequential Minimal Optimization (SMO) method (1998; this feature) to learn the vector of feature weights, $\vec{w}$. Once the weights are learned, new items are confidential by computing $\vec{w} \cdot \vec{x}$ where $\vec{w}$ is the vector of learned weights, and $\vec{x}$ is the binary vector instead of the new document to classify. We also learned two parameters of a sigmoid function to renovate the output of the SVM to probabilities.

4.3 Data reduction For Bug Triage

We propose bug data diminution to reduce the scale and to get better the quality of data in bug repositories. Which is applied as a stage in data training of bug triage.We mingle existing techniques of instance selection and feature selection to eliminate certain bug information and words. A crisis 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. In this section, we first nearby how to apply instance selection and feature selection to bug data, i.e data shrinking for bug triage. Then, we list the benefit of the data reduction. The details of the prophecy for reduction orders.

4.4. Instance and Feature Selection

In bug triage, a bug data set is transformed into a text matrix with two proportions, namely the bug element and the word dimension. In our work, we leverage the amalgamation of instance selection and feature selection to generate a reduced bug data set. We replace the inventive data set with the reduced data set for bug triage. Instance selection and feature selection are widely used techniques in data dispensation. 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 significant features In our project, we utilize the combination of instance selection and feature selection. To distinguish the orders of applying instance selection and feature selection.

4.5 Bug Triage

Bug triage aims to assign an proper developer to fix a new bug, i.e to resolve who should fix a bug. Cubranicand Murphy first propose the problem of automatic bug triage to decrease the cost of manual bug triage. They apply text categorization techniques to expect related developers. Anvik et al. examine multiple techniques on bug triage, together with data training and typical classifiers. Anvik and Murphy extend above work to decrease the effort of bug triage 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 decrease reassignment in bug triage. To avoid inferior 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 propose a mutual filtering approach to reducing the bug putting in place time.

For bug data, several other responsibilities exist once bugs are triaged. For example, severity identification aims to detect the significance of bug reports for further preparation in bug handling; time prophecy 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 circumvent delaying the software release.

In data mining, the crisis of bug triage relates to the troubles of expert finding and ticket routing. In contrast to the broad domains inexpert finding or ticket steering, bug triage only focuses on assign developers for bug reports. Moreover, bug reports in bug triage are transferred into documents (not keywords inexpert result) and bug triage is a kind of content-based categorization[8].

4.6 Data Quality in Defect Prediction

In this system, we tackle the problem of data reduction for bug triage. To our awareness, no presented work has examine the bug data sets for bug triage. In a related trouble, defect prediction, some work has focused on the data value of software defects. In contrast to numerous classification in bug triage, defect prediction is a binary classification trouble, which aims to predict whether a software artifact (e.g a source code file, a class, or a component) contains faults according to the extracted features of the artifact. In software engineering, defect prediction is a kind of exertion on software metrics. To improve the data quality and to inspect the techniques on feature selection to handle imbalanced fault data. present how to measure the noise confrontation in defect prediction and how to detect noise data. In this paper, in dissimilarity to the above work, we tackle the problem of data decrease for bug triage. Our work can be viewed as an expansion of software metrics. In our work, we predict a value for a set of software artifacts while obtainable work in software metrics predict a value for an individual software artifact.

5. Conclusion

In this paper we have paying attention on minimizing bug data set in order to have less scale of data and superiority data. Our work supply an approach to leveraging technique to form compact and high excellence bug data in software improvement and preserves. Our new results showed that this data diminution technique will give quality data as well as it will decrease the data scale.
In future work, we plan on humanizing the results of data diminution in bug triage to explore how to arrange a high quality bug data set and deal with a domain-specific software task and we want to examine effect of other term selection methods.

6. Acknowledgement

We would like to thanks Dr. D. Y. Patil Institute of Engineering and Technology, Pune gratefully acknowledge Computer department of their helpful support and guidance for writing this paper.

REFERENCES

[1] Jifeng Xuan, He Jiang,Member,IEEE,Yan hu "*Towards Effective Bug Triage with Software Data Reduction Techniques*" in Vol.27 JAN 2015.

[2]. Mamdouh Alenezi and Kenneth Magel, Shadi Banitaa "*Efficient Bug Triaging Using Text Mining" 2013 academy publisher*

[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].J. Xuan, H. Jiang, Z. Ren, and W. Zou, "Developer prioritization in bug repositories," in Proc. 34th Int.Conf. Softw. Eng., 2012, pp. 25

[5]. 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.

[6]. 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.

[7].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.

[8]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.