Asian Journal of Information Technology 15 (8): 1334-1339, 2016
ISSN: 1682-3915

Auto-Bug Triager for Assisting Manual Bug Triage

[1]S. Kirubakaran, [2]K. Maheswari and [1]K. Reshma Revathi
[1]Info Institute of Engineering,
[2]SNS College of Technology, Coimbatore, Tamil Nadu, India

Abstract: Bug triage is an unescapable process in every software organization. A separate team in every software companies takes care of this process. The complete process occurs in manual which increases the production time and cost. One of the time taking tasks in bug triage process is assigning an appropriate developer to fix the new coming bugs than fixing that bug. Automation is the key solution for this problem. Since the bug reports are in free form of data, it is efficient to use data mining techniques to handle. In recent research, different techniques were applied by different authors trying to automate the bug triage process. But, up till now only 64% of triaging accuracy is achieved. Our survey shows that the decrease in accuracy is caused by one of the major problem called data reduction. In this study, we proposed a new framework called Auto-BugTriager which focuses on the elimination of data reduction problem to greater extent. Auto-BugTriager consist of three phases namely InfoZie, DataReduction and NBClassifier which works together to predict the recommendation list of expert developer for fixing the new bug. We have done the complete theoretical and experimental analysis of the proposed framework. Our analysis shows, that the proposed framework eliminated the problem of data reduction to greater extent, thus the data quality and accuracy of bug triage is increased.

Key words: Bug Triage process, manual triaging, data mining techniques, data reduction, India

INTRODUCTION

Software companies are almost deals with the flow of bugs in all kinds of projects scenario. In open source environment, almost 350 bugs are encountered every day whereas in domain specific environment, almost 50-150 bugs are encountered each day and the growth in bug depends up on the dimension of the projects. This is considerable for programmers to handle by themselves as one may encounter various number of defects/bugs during the build time. Also, the bug repository contains numerous bug reports of all kinds, mostly which are duplicate of one another. Therefore, each bug report in the repository must be validated for duplication and then needs to be triaged. Practically, assigning an appropriate developer to fix a new bug is time-consuming process than fixing a new bug.

In traditional software development, triaging is performed manually by an expert developer, i.e., a human triage/manual triage. Due to the huge number of daily bugs and the lack of knowledge of all the bugs, manual bug triage is more expensive both in terms of time and cost. In human triage when a new bug acts, a proficient developer is assigned, who will try to fix this bug. Most of the time the prediction of expert is not 100% accurate. Manual triage is error-prone due to the enormous number of daily bugs and the lack of knowledge about all bugs by the developers. Human bug triage outcomes in expensive time loss, high cost and low accuracy. Reassigning the new bug for different developers to fix lasts for months while fixing that bug takes only two to three days. Therefore, it is essential to automate the bug triage process in every software companies in the motivation of improving their production quality.

Current research employs data mining concepts to deal with the software engineering glitches. There are various number of mining techniques such as text classification, fuzzy logic, text mining, extraction methods etc., are being implemented to automate the bug triage process. Bug reports are free-form of data which has two foremost challenges. First challenge is the duplicate reports are available in the bug repository. Mining large scale data will only results in low accuracy. The second challenge is the lack of data quality i.e the presence of uninformative stop words in every bug reports. Both these challenges are together are called as data reduction problem which degrades the accuracy of bug triage.

In this study, a new framework called Auto- BugTriager is proposed which focus on eliminating the problem of data reduction and improving accuracy in

Corresponding Author: S. Kirubakaran, Info Institute of Engineering, Coimbatore, Tamil Nadu, India

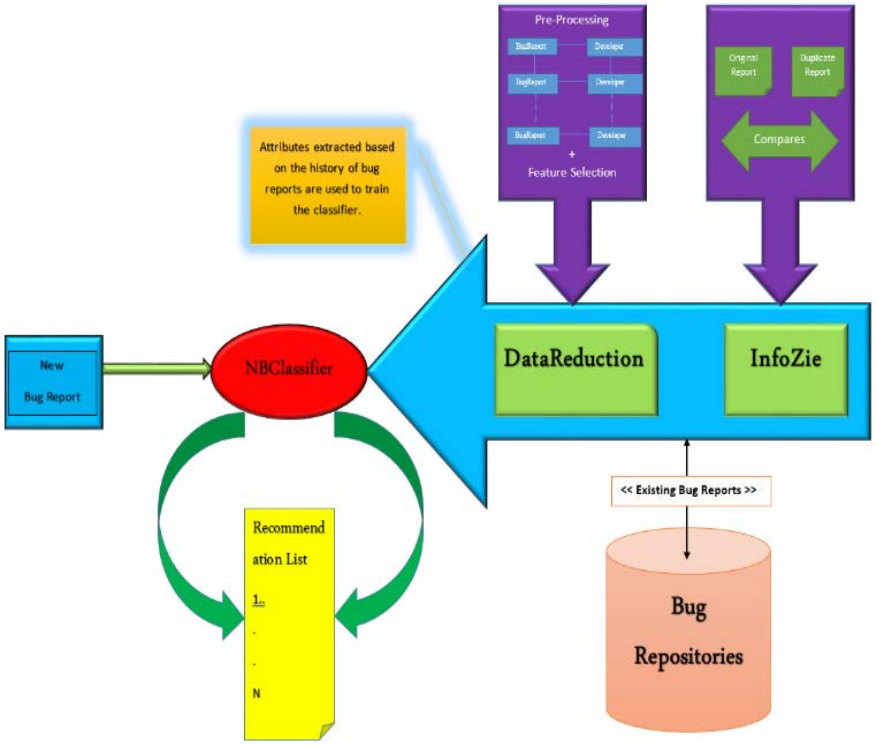

Fig. 1: The proposed framework called Auto-BugTriager

automating bug triage process. This proposed framework consist of three phases namely InfoZie, DataReduction and NBPredictor which process together for the prediction of recommendation list consisting of expert developers for fixing the new bug. In order to spread our idea for the practical use, we have proposed to implement this new framework in a domain specific environment. Also, complete experimental study has been done by employing the Auto-BugTriager at the backend process of a software system.

The complete analysis report of the performance analysis of Auto-BugTriager is provided in the end. It is therefore seen that the prediction of appropriate developer is measured to be almost 98.5% accurate. As bug triage occurring to be more serious problem, the corporate world faces the loss both in terms of cost and time. Therefore, it is significant to solve this problem for both software productivity and quality.

Problem identification

Aim of the study: The main objective of our project is to study and analyze the data reduction problem in automating the process of bug triage. We have also designed a new framework called Auto-BugTriager which focus on the elimination of data reduction problem to the greater extent in bug triaging. To improve the accuracy of automatic bug triage. To encompass our idea for real-time use and assist manual bug triage.

Issues: Manual bug triage is expensive both in terms of cost and time. Therefore, automatic bug triage methodologies are being proposed using data mining techniques such as text classification (Anvik and Murphy, 2004). The accuracy of these automatic approaches doesn't provide accuracy of > 60-65%. The cause for the low accuracy is the major problems of data reduction (Xuan *et al*., 2015), i.e., to reduce the bug dimension (Redundant data) and the word dimension (Data quality). Eliminating the duplicate bug reports entirely from the bug repository reduces the accuracy of triaging due to the loss of valuable information's in the duplicate reports. Also, all noise data's are not detached, where the data quality is not enriched and leads to less accuracy. Recently, all organization uses manual bug triage mechanism and not any industry has automated the process yet which diminishes the software productivity and quality.

MATERIALS AND METHODS

Proposed system: The newly designed framework entitled Auto-BugTriager is shown in Fig. 1. This proposed system targets to eliminate the problem of data reduction in bug triage. The framework (Revathi and Kirubakaran,

2016) consist of three levels, viz. InfoZie, DataReduction and NBClassifier which works organized to predict the recommendation list of proficient developers who can able to fix the new bug. The functioning of each level is described below.

InfoZie: The first level of Auto-BugTriager is entitled as InfoZie which perform as a validating tool. Duplicate bug reports frequently comprises with valuable superfluous information (Bettenburg *et al*., 2008a,b). Consequently the duplicate reports are validated for superfluous information using this InfoZie tool. At this point, the master reports and duplicate reports are compared using a modified Diff based algorithm (Revathi and Kirubakaran, 2016) Diff algorithm is a UNIX based procedure that compares two files of similar version for superfluous information. The InfoZie tool is based on the model of diff algorithm. If duplicate report comprises superfluous valuable information, it is merged with the original master report otherwise skipped. This level of the framework improves the bug dimension as well as the accuracy of bug triage.

Data reduction: Data Reduction is the next level of the proposed framework which implements two subsequent procedures.

Pre-processing: The text classification technique is used to convert the abridged bug reports into text matrix where each row signposts one bug report and each column signposts one word.

Noise reduction: This part of the framework makes use of feature extraction technique (Bolon Canedo *et al*., 2013) to diminish the word dimension, i.e., eliminates the uninformative stop words from the bug reports.

NB Classifier: The final level of Auto-BugTriager framework which predicts the recommendation list for the new bug report i.e. the NBClassifier which is created using Naïve Bayes classifier. The classifier is trained with the history attributes extracted from existing bug reports. Attributes are extracted both in terms of technical aspect and history of developer details who fixed the existing bugs. Here, the new bug report is the test set which fed as input to the Auto-BugTriager and compared with the trained set of NBClassifier in order to predict the recommendation list. Auto-BugTriager helps the software testing team in administration of bug reports by the accumulation of added auxiliary features (Breu *et al*., 2010).

Data mining concepts plays a vital part in automating the bug triaging process in effective way. Diff based algorithm, Pre-Processing, Feature selection algorithm and Naive Bayes classifiers are the crucial methods used in our newly proposed framework. A pure and profound theoretical study of each methods are explained below.

Data pre-processing: The principal of handling the bug reports using the Auto-BugTriager framework is the data pre-processing. As the real world data's are incomplete and inconsistent, mining such raw data is tiresome and error prone process which is one of the causes for the fall of accuracy while automating the triaging process. Therefore, transforming those uncooked data into a reasonable format will make the mining process easy and consequently accuracy can be attained. The bug reports have a uniform format where the information are divided into reasonable classes. Therefore, the proposed framework uses relational database format which provides a convenient way for transforming each bug reports into number of rows and columns.

Feature selection: Word dimension and bug dimension are the two Data Reduction problems encountered in automating the bug triage process. Word dimension means data quality. By nature, bug reports are defined in natural language, containing noisy information, i.e., uninformative words. The presence of noisy data drops the quality of the data and thus resulting inaccuracy of triaging. So, eliminating such noise information from the bug data is an essential process for increasing the correctness of triaging results. Feature selection algorithm is a suitable technique that offers attribute selection in a best way to improve the quality of data hence making the triaging process more accurate. The proposed framework applies the feature selection aka attribute selection in two stages of the proposed system.

Primarily, the attribute selection (Khoshgoftaar *et al*., 2010) is implemented based on the selection of stop words from the bug reports. Stop words are the words that exist in the bug reports as noisy data that makes the mining process more challenging and erroneous. Hence, removing such stop words is inevitable. This will progress the data quality and ease the mining process. Thus the accuracy of triaging is improved. Also, additional set of attribute selection were done established on the technical terms which are used to train the Naive Bayes classifier by means of prediction algorithm. These selected attributes are second-handed to the third level called NB classifier. Feature extraction method

benefits the Auto-BugTriager in improving the quality of bug data by removing all the noise information present in the bug reports.

InfoZie using Diff algorithm: An additional leading problem in automating the bug triage process is the duplication of information in the bug repository due to the presents of duplicate bug reports that resides in it. Removals of these duplicate reports are imperative, but every duplicate reports in the bug repositories are not always a duplicate of the original. Mostly the duplicate reports are resubmitted purposely by the developers/programmer with some improvements or added information to the bug report. As a result, there are 90% of probabilities for the existence of added valuable information in the duplicate bug reports. So, eliminating such valued information entirely from the database will decrease the precision of bug triage.

For that reason, it is important to validate every duplicate bug reports existing in the bug repository. This validation is be accomplished using a particular tool as in called InfoZilla (Anvik and Murphy, 2011) which authenticates the duplicate bug reports over the mined structural facts from every bug reports. In our proposed framework, we generated a tool so-called InfoZie which is established on the basis of UNIX based diff algorithm ("diff" is a data comparison tool used to show the changes between two versions of the same file in UNIX environment) that matches the technical features extracted from both duplicate and master bug reports. When duplicate report holds additional information, the duplicate report is merged with the master bug report if not merging is skipped. This level of InfoZie progresses the bug dimension and as well the accuracy of bug triage is increased.

The modified diff algorithm used at InfoZie level is specified further down, this procedure the whole thing as per the following assumptions. Considering two data sets named NewBugReports and BugRepository where the first holds the new incoming bug reports and the last holds the existing bug reports along with its history of developer who have fixed it.

Algorithm 1; Modified Diff Algorithm for InfoZie:

For each Bugdata in NewBugrepository
Compare NewData with each MasterData in BugRepository
If there is a match sequence
If (mirrored data found)
Merge the result
Else//Good best match found
ShowReport, ShowOptions (Merge, Skip)
Else
Skip//No matches found
End If
End For

The modified Diff Algorithm.1 helps in completely removing the duplicate bug reports from the repositories without any loss of valuable information hence diminishing the bug dimension i.e. the redundant data. This part of Auto-BugTriager is the heart of our proposed framework that helps in growing both the quality of data and accuracy of triaging.

Naive Bayes Classifier: Auto-BugTriager framework practices the Naive Bayes classifier at its final level of the system for predicting the expert developer recommendation list. Naive Bayes classifier is easy to implement and predominantly suitable for very large set of data. Consequently, this is a fitting classifier for triaging the bug reports which upturns the corporate world.

NBClassifier is a simple and known predictor that outperform even on very much difficult data. Figure 2

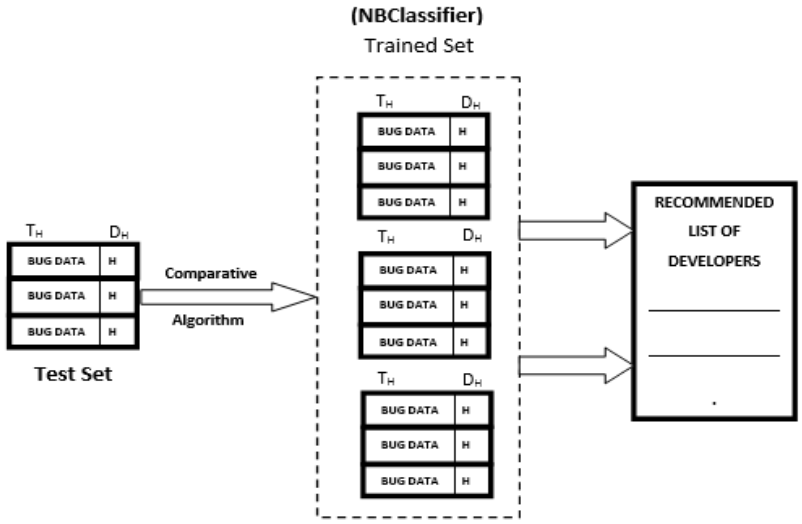

Fig. 2: Working analysis of NB classifier

demonstrations the working analysis of NBClassifier. Here, T_H and D_H are the twofold attributes viz. Technical History and Developer History. The output of the Auto-BugTriager is the list of recommended developers (Anvik and Morphy, 2011) who can fix new similar bug.

RESULTS AND DISCUSSION

Here, the experimental analysis of the Auto-BugTriager has been done thoroughly. The monitored analysis report from the implemented system has been provided in detail here. The experiment is done for different set of datasets for analysing the quality and improvement of accuracy. The performance analysis of the implemented Auto-BugTriager has also been given in detail for reference. The data reduction problem in automating the bug triage process has been eliminated almost. Figure 3 shows the detection rate of duplicate information present in examined sets of different sample bug reports. Approximately 98% of the duplicate reports has been well identified without any loss of valuable information, with the support of InfoZie.

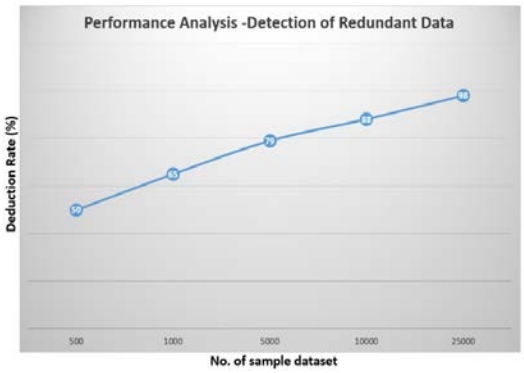

Fig. 3: Graph showing detection in the bug reports experimented for different sets of sample dataset.

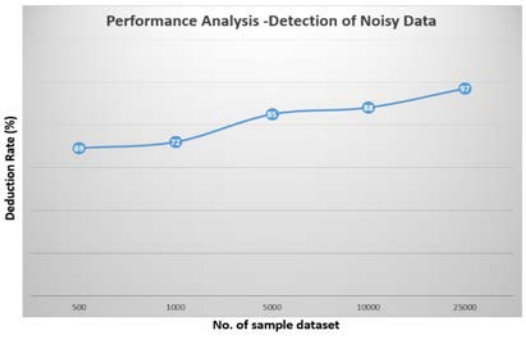

Fig. 4: Detection rate of noisy information in the bug reports experimented for different sets of sample dataset

Similarly, an experiment for different sets of sample bug reports starting from 5000 to 25000 has been done for the analysis of noisy information in the bug reports. And the detection rate is found to be 97% which is shown in Fig. 4. Also, the analysis of data reduction problem has been experimented and the report is given in Fig. 5. This report compares the performance rate for both before and after the removal of data reduction problem. Therefore, this clearly explains the increase in data quality in Auto-BugTriager. The performance rate reaches almost 98.7% of data quality. When data quality is high, the triaging process will be accurate with its result. Finally, the yearly comparison of data reduction problem, data quality analysis and accuracy of automation in bug triage process is experimented for the same set of different datasets. The analysis report is shown in Fig. 6 which compares the result of Auto-BugTriager with the Automatic Bug triage system developed between the years 2004-2015. The analysis report clearly shows that proposed framework called Auto-BugTriager has achieved almost 94% in elimination of data reduction problem, 96% of data quality and automation accuracy 98.5%.

Fig. 5: Examination of data quality individually before and after the removal of data reduction problem

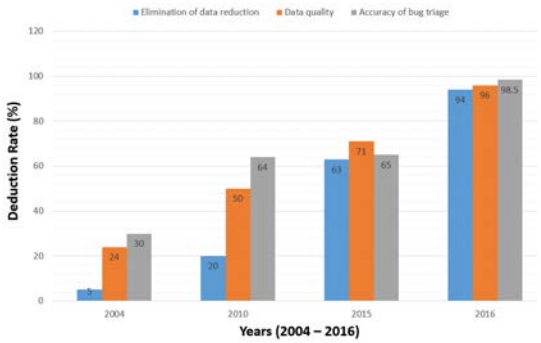

Fig. 6: Evaluation of data reduction problem, data quality and accuracy of bug triage process between the years 2004 to 2016. This includes overall project performance

CONCLUSION

In this study a new framework called AutoBugTriager has proposed which aims to assist the manual bug triage in every software companies. Data reduction is found to be a major problem in automating the bug triage process using data mining techniques. The proposed frameworks works better when implemented for domain specific environment as the attribute extractions are in terms of technical aspects. A complete theoretical and experimental analysis of the proposed framework is reported. The analysis shows, that the quality of data and accuracy of bug triage has been improved rapidly.

RECOMMENDATIONS

Our future work focuses on the complete automation of bug triage without human help to create a new infrastructure for the software testing team. It might help them handle bugs in effectively. Also, we tend to implement this model with real time industrial data.

REFERENCES

Anvik, J. and G.C. Murphy, 2011. Reducing the effort of bug report triage: Recommenders for development-oriented decisions. ACM. Trans. Software Eng. Method., Vol. 20, 10.1145/2000 791.2000794

Bettenburg, N., R. Premraj, T. Zimmermann and S. Kim, 2008a. Duplicate bug reports considered harmful really?. Proceeding of the IEEE International Conference on Software Maintenance, September 28-October 4, 2008, IEEE, Beijing, China, ISBN: 978-1-4244-2613-3, pp: 337-345.

Bettenburg, N., R. Premraj, T. Zimmermann and S. Kim, 2008b. Extracting structural information from bug reports. Proceedings of the 2008 International Working Conference on Mining Software Repositories, May 10-18, 2008, ACM, Leipzig, Germany, ISBN: 978-1-60558-024-1, pp: 27-30.

Bolon Canedo, V., N. Sanchez-Marono and A. Alonso-Betanzos, 2013. A review of feature selection methods on synthetic data. Knowl. Inf. Syst., 34: 483-519.

Breu, S., R. Premraj, J. Sillito and T. Zimmermann, 2010. February Information needs in bug reports: Improving cooperation between developers and users. Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work, February 06-10, 2010, ACM, New York, USA., ISBN: 978-1-60558-795-0, pp: 301-310.

Khoshgoftaar, T.M., K. Gao and N. Seliya, 2010. Attribute selection and imbalanced data: Problems in software defect prediction. Proceedig of the 22nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI), October 27-29, 2010, Arras, France, ISBN: 978-1-4244-8817-9, pp: 137-144.

Revathi, K.R. and S. Kirubakaran, 2016. A Survey on automatic bug triage using data mining concepts. Intl. J. Sci. Res., 5: 184-186.

Xuan, J., H. Jiang, Y. Hu, Z. Ren and W. Zou *et al*., 2015. Towards effective bug triage with software data reduction techniques. Knowl. Data Eng. IEEE. Trans., 27: 264-280.