ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

Effective Bug Triage Scheme with Data Reduction

Sujata Birajdar[1], **H.B.Torvi** [2], **S.P.Deshmukh**[3]

Department Of Computer Science and Engineering, V.V.P. Institute of Engineering and Technology, Solapur, India
Assistant Professor , Department of Computer Science and Engineering, V.V.P. Institute of Engineering and Technology, Solapur, India
Assistant Professor, Department of Computer Science and Engineering, V.V.P. Institute of Engineering and Technology, Solapur, India

ABSTRACT: The major need is essential for all software companies like how to solve the bugs and they spend lots of amount to handle these scenarios. In bug triage process, when the new Bug was logged by tester then every bug should assigned to developer. Manual triage process increases cost and time in the software development life cycle. This system, propose the solution to the problem of information reduction for bug triage, i.e., to reduce the scale and improve the nature of bug data. Combine example choice with highlight determination to simultaneously reduce information scale on the bug dimension and the word dimension. To decide the request of applying case determination and highlight choice, we extricate qualities from verifiable bug informational indexes and fabricate a prescient model for another bug informational collection. We experimentally research the execution of information lessening on absolutely 600,000 bug reports of two huge open source ventures, to be specific Eclipse and Mozilla. The outcomes demonstrate that our information diminish can successfully lessen the information scale and enhance the accuracy of bug triage.

KEYWORDS: Bug data reduction, feature selection, instance selection, bug triage, prediction for reduction orders.

I.INTRODUCTION

Mining programming storehouses is an interdisciplinary area, which means to utilize information mining to manage programming designing issues. In present day programming improvement, programming storehouses are extensive scale databases for putting away the yield of programming advancement, e.g., source code, bugs, messages, and particulars. Customary programming examination is not totally reasonable for the huge scale and complex information in programming archives.

Information mining has risen as a promising intends to deal with programming information. By utilizing information mining systems, mining programming archives can reveal fascinating data in programming storehouses and tackle real world programming issues. A bug storehouse [a common programming vault, for putting away points of interest of bugs], assumes an imperative part in overseeing programming bugs. Programming bugs are inescapable and settling bugs is costly in programming improvement. Programming organizations spend more than 45 percent of cost in settling bugs. Expansive programming ventures send bug storehouses [likewise called bug following frameworks] to bolster data accumulation and to help designers to deal with bugs.

In a bug store, a bug is kept up as a bug report, which records the literary depiction of duplicating the bug and updates as indicated by the status of bug settling. A bug archive gives an information stage to bolster many sorts of undertakings on bugs, e.g., blame forecast, bug confinement, and reopened bug investigation. In this paper, bug reports in a bug vault are called bug information. There are two difficulties identified with bug information that may influence the powerful

utilization of bug stores in programming improvement undertakings, in particular the expansive scale and the low quality. On one hand, because of the every day revealed bugs, countless bugs are put away in bug archives.
Taking an open source extend, Eclipse, for instance, a normal of 30 new bugs are accounted for to bug storehouses every day in 2007; from 2001 to 2010, 333,371 bugs have been accounted for to Eclipse by more than 34,917 designers and clients. It is a test to physically inspect such expansive scale bug information in programming advancement. Then again, programming strategies experience the ill effects of the low nature of bug information. Two ordinary attributes of low-quality bugs are clamor and repetition. Boisterous bugs may deceive related designers while excess bugs squander the restricted time of bug dealing with. A tedious stride of taking care of programming bugs is bug triage, which intends to allocate a right designer to settle another bug.
In customary programming advancement, new bugs are physically triaged by a specialist engineer, i.e., a human triager. Because of the substantial number of day by day bugs and the absence of aptitude of the considerable number of bugs, manual bug triage is costly in time cost and low in precision. In manual bug triage in Eclipse, 44 percent of bugs are assign by misstep while the time cost between opening one bug and its first triaging is 19.3 days by and large. To avoid the costly cost of manual bug triage, existing work has proposed a programmed bug triage approach, which applies content order methods to anticipate designers for bug reports. In this approach, a bug report is mapped to a record and a related designer is mapped to the mark of the archive. At that point, bug triage is changed over into an issue of content grouping and is naturally understood with develop content characterization procedures, e.g., Naive Bayes.
In light of the aftereffects of content characterization, a human triager assigns new bugs by consolidating his/her skill. To enhance the precision of content grouping procedures for bug triage, some further strategies are examined, e.g., a hurling chart approach and a synergistic separating approach. In any case, extensive scale and low-quality bug information in bug stores obstruct the strategies of programmed bug triage. Since programming bug information are a sort of freestyle content information [produced by engineers], it is important to create all around prepared bug information to encourage the application.

In this framework, we address the issue of information decrease for bug triage, i.e., how to lessen the bug information to spare the work cost of engineers and enhance the quality to encourage the procedure of bug triage. Information lessening for bug triage means to construct a little scale and top notch set of bug information by expelling bug reports and words, which are excess or non-educational. In our work, we consolidate existing systems of case choice and highlight choice to all the while lessen the bug measurement and the word measurement. The lessened bug information contain less bug reports and less words than the first bug information and give comparable data over the first bug information. We assess the lessened bug information as indicated by two criteria: the size of an informational index and the exactness of bug triage.

II.LITERATURE SURVEY

In the year of 2010, the authors "J. A. Olvera-L opez, J. A.Carrasco-Ochoa, J. F. Mart ınez-Trinidad, and J. Kittler" proposed a paper titled "A review of instance selection methods" such as in supervised learning, a training set provided that previously known information is used to classify new instances. Commonly, several instances are stored in the training set but some of them are not functional for classifying therefore it is possible to get acceptable classification rates ignoring non useful cases; this process is known as instance selection.

Through instance selection the training set is compact which allows reducing runtimes in the classification and/or training stages of classifiers. This work is focused on presenting a survey of the main instance selection methods reported in the literature.

In the year of 2010, the authors "E. Shihab, A. Ihara, Y. Kamei, W. M. Ibrahim, M. Ohira, B. Adams, A. E. Hassan, and K. Matsumoto" proposed a paper titled "Predicting re-opened bugs: A case study on the eclipse project" such as Bug fixing is one of the most time-consuming and costly activities of the software development life cycle. In general, bugs are reported

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

in a bug tracking system, validated by a triage team, assigned for someone to fix, and finally verified and closed. However, in some cases bugs have to be reopened.
Reopened bugs increase software maintenance cost, because of rework for already busy developers and in some cases even delays the future delivery of a software release. Therefore, a few recent studies paying attention on studying reopened bugs. However, these prior studies did not achieve high performance (in terms of precision and recall), required manual intervention, and used very basic techniques when dealing with this textual data, which leads us to believe that further improvements are possible. In this paper, we propose Reopen Predictor, which is an automatic, high accuracy predictor of reopened bugs. Reopen Predictor uses a number of features, including textual features, to achieve high accuracy prediction of reopened bugs. As part of Reopen Predictor, we propose two algorithms that are used to automatically estimate various thresholds to exploit the prediction performance.
To examine the benefits of Reopen Predictor, we perform experiments on three large open source projects--namely Eclipse, Apache HTTP and Open Office. Our results show that Reopen Predictor outperforms prior work, achieving a reopened F-measure of 0.744, 0.770, and 0.860 for Eclipse, Apache HTTP and Open Office, respectively. These results correspond to an enhancement in the reopened F-measure of the method proposed in the prior work by Shihab et al. by 33.33, 12.57 and 3.12 % for Eclipse, Apache HTTP and OpenOffice, respectively.

In the year of 2011, the authors "C. Sun, D. Lo, S. C. Khoo, and J. Jiang" described into their paper titled "Towards more accurate retrieval of duplicate bug reports" such as 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 preservation 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 comparison between two bug reports. It fully utilizes the information available in a bug report including not only the likeness of textual content in summary and description fields, but also likeness of non-textual fields such as product, component, version, etc. For more accurate measurement of textual similarity, we extend BM25F - an effective similarity formula in information retrieval community, especially for duplicate report retrieval.

Lastly we use a two-round stochastic gradient descent to automatically optimize REP for specific bug repositories in a supervised learning manner. We have validated our technique on three large software bug repositories from Mozilla, Eclipse and OpenOffice. The experiments show 10-27% comparative improvement in recall rate@k and 17-23% relative improvement in mean average precision over our previous model.

III.PROPOSED STUDY

To keep away from the inclination of a solitary calculation, we observationally look at the consequences of four occurrence choice calculations and four component choice calculations. Given an example determination calculation and an element choice calculation, the request of applying these two calculations may influence the aftereffects of bug triage. In this paper, we propose a prescient model to decide the request of applying occasion determination and highlight choice. We allude to such assurance as expectation for lessening orders.

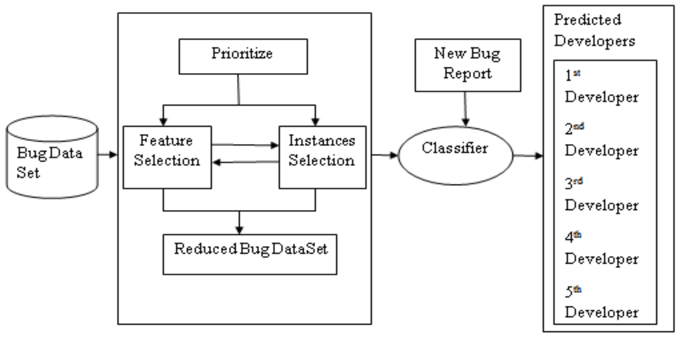

Fig. 1. Proposed System Architecture

Drawn on the encounters in programming metrics,1 we separate the qualities from verifiable bug informational collections. At that point, we prepare a twofold classifier on bug informational collections with separated traits and foresee the request of applying occurrence determination and highlight choice for another bug informational collection. In the trials, we assess the information diminishment for bug triage on bug reports of two expansive open source ventures, to be specific Eclipse and Mozilla. Test comes about demonstrate that applying the occasion choice system to the informational collection can diminish bug reports yet the exactness of bug triage might be diminished; applying the element determination strategy can evoke words in the bug information and the precision can be expanded.

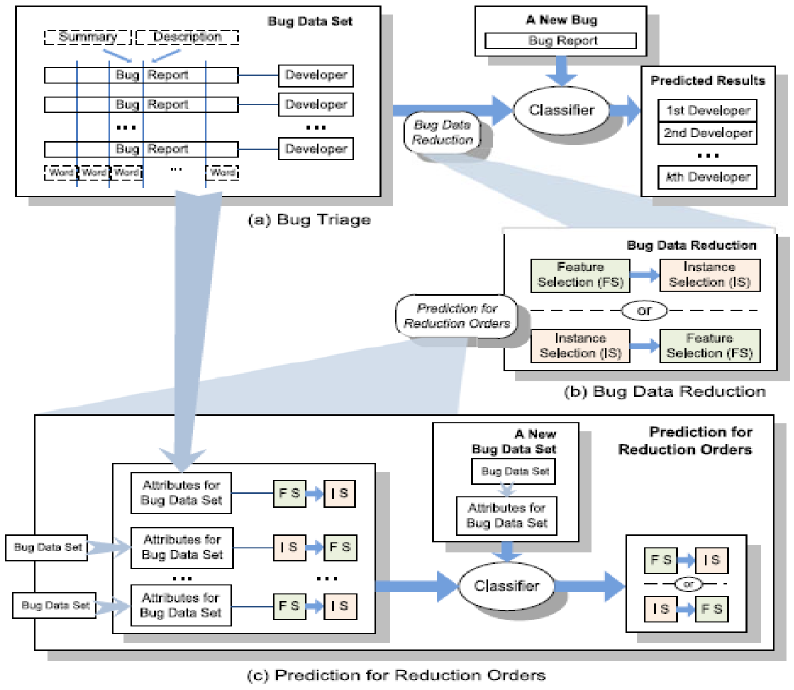

Fig. 2. Illustration of reducing bug data for bug triage

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

Then, joining both systems can expand the exactness, and in addition decrease bug reports and words. For instance, when 50 percent of bugs and 70 percent of words are expelled, the precision of Naive Bayes on Eclipse enhances by 2 to 12 percent and the exactness on Mozilla enhances by 1 to 6 percent. In view of the traits from chronicled bug informational indexes, our prescient model can give the precision of 71.8 percent for foreseeing the diminishment arrange. In view of top hub examination of the traits, comes about demonstrate that no individual characteristic can decide the lessening request and each ascribe is useful to the forecast.

TABLE 1
Part of History of Bug 284541 in Eclipse

Triager	Date	Action
Kaloyan Raev	2009-08-12	Assigned to the developer Kiril Mitov
Kaloyan Raev	2010-01-14	Assigned to the developer Kaloyan Raev
Kaloyan Raev	2010-03-30	Assigned to the developer Dimitar Giormov
Dimitar Giormov	2010-04-12	Changed status to assigned
Dimitar Giormov	2010-04-14	Changed status to resolved Changed resolution to fixed

Real Contributions of the System

The essential commitments of this paper are as per the following:

(a) We present the issue of information decrease for bug triage. This issue expects to increase the informational index of bug triage in two angles, specifically a) to at the same time diminish the sizes of the bug measurement and the word measurement and b) to enhance the precision of bug triage.

(b) We propose a mix way to deal with tending to the issue of information decrease. This can be seen as a use of occurrence determination and highlight choice in bug archives.

(c) We construct a double classifier to anticipate the request of applying occasion determination and highlight choice. As far as anyone is concerned, the request of applying example choice and highlight choice has not been researched in related spaces.

Methodologies

After examine the result of data reduction various algorithms in instances selection, bug triage and classification are applied. The details can be found as follows.

a. Model for Prediction

Build a binary classifier to predict the order of applying instance selection and feature selection. Add new attributes extracted from bug data sets, prediction for reduction orders, and experiments on instance selection algorithms, feature selection algorithms, and their combinations. These features are used for training of classification algorithm which is then used for classification of bug reports. The classification algorithm used in proposed system is multinomial Naïve Bayes. Naive Bayes is used to predict correct developer to solve and fix the bug reports in order to shrink the manual triager cost.

b. Instances Selection

For a given data set in a certain application, instances selection is to obtain a subset of relevant instances (bug report in bug data). After Examine result of instances selection algorithm, cosine similarity algorithm based clustering approach helps to increase the rate of fault detection in distributed environment. The techniques for finding the similarity between two documents, time required for cluster generation by using Cosine Similarity measure takes less amount of time as compare to other algorithm because of using mathematical formula for calculating the similarity measure between the documents.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

c. Feature Selection

Feature selection aims to obtain a subset of relevant features (word in bug data). x 2 statistic (CH) is a typical feature selection algorithm , which measure the dependence between word and developers. By removing uninformative word, feature selection improves the accuracy of bug triage. This can recover the accuracy loss by instances selection.

d. Bug Triage

Text-Based bug triage approaches and domain-based approaches are effective for bug triage. In contrast to focusing on textual content in bug reports in text-based approaches, domain-based approach uses bug re-assignment as domain knowledge.

In the implementation of DP (Developer Prioritization), the text-based classifier is set as Naïve Bayes and the domain knowledge is extracted from developer priorities based on products and components. In this system present the problem of data reduction for bug triage. This problem aims to augment the data set of bug triage in two aspects a) to simultaneously reduce the scales of the bug dimension and the word dimension b) Improve the Accuracy of Bug Triage. Build a binary classifier to predict the order of applying instances selection and feature selection.

IV.EXPERIMENTAL RESULTS

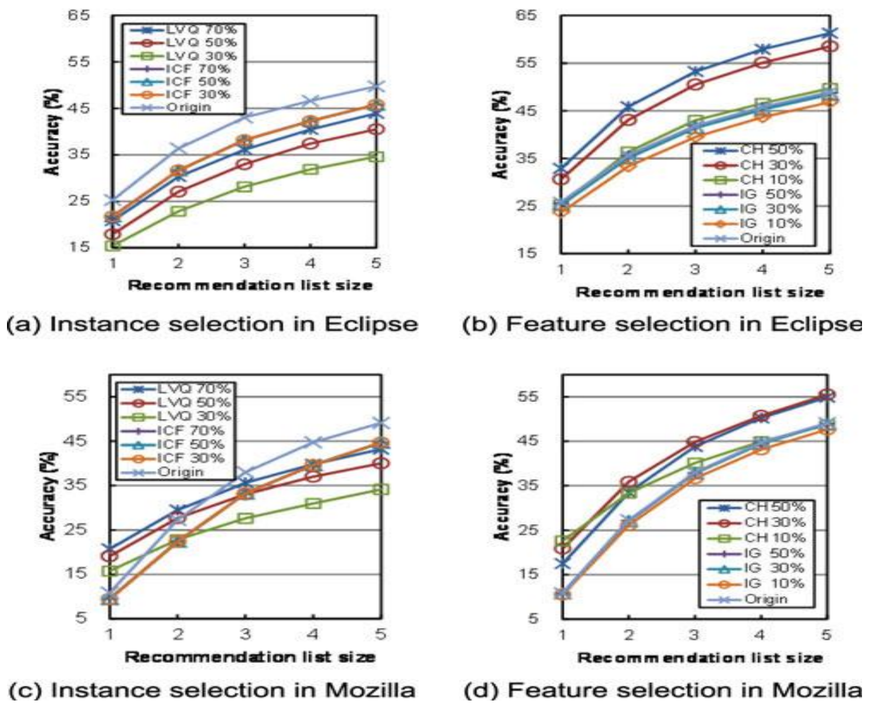

Fig.3. Accuracy for instance selection or feature selection on Eclipse (DS-E1) and Mozilla (DS-M1). For instance selection, 30, 50, and 70 percent of bug reports are selected while for feature selection, 10, 30, and 50 percent of words are selected. The origin denotes the results of Naive Bayes without instance selection or feature selection. Note that some curves of ICF may be overlapped since ICF cannot precisely set the rate of final instances.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

V.CONCLUSION

Bug triage is a costly step of programming maintenance in both work cost and time cost. In this paper, we consolidate highlight choice with case choice to decrease the size of bug informational collections and also enhance the information quality. To decide the request of applying example choice and highlight choice for another bug informational index, we extricate properties of each bug informational index and prepare a prescient model in light of recorded informational collections. We observationally examine the information decrease for bug triage in bug archives of two substantial open source ventures, in particular Eclipse and Mozilla. Our work gives a way to deal with utilizing methods on information handling to frame decreased and top notch bug information in programming advancement and upkeep. In future work, we anticipate enhancing the consequences of information lessening in bug triage to investigate how to set up an amazing bug informational collection and handle a space particular programming assignment. For anticipating decrease orders, we plan to pay endeavors to discover the potential relationship between the characteristics of bug informational collections and the lessening orders.

REFERENCES

[1] Jifeng Xuan ,He jiang ,Zhilei Ren ,weiqin Zou ,Zhongxuan Lou ,and xindong wu, “ Towards Effective Bug Traige With Software Data Reduction Techniques”IEEE Transaction ,Volume 27,No 1 Jan 2015.
[2] 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.
[3] 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.
[4] 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.
[5] C. C. Aggarwal and P. Zhao, “Towards graphical models for text processing,” Knowl. Inform. Syst., vol. 36, no. 1, pp. 1–21, 2013.
[6] Bugzilla, (2014). [Online]. Avaialble: http://bugzilla.org/
[7] 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.
[8] 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.
[9] 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.
[10] 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.
[11] 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.
[12] 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.
[13] 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.
[14] Eclipse. (2014). [Online]. Available: http://eclipse.org/