International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 11 | Nov -2016 www.irjet.net p-ISSN: 2395-0072

An Effective Approach for Automatic Bug Triage Techniques

Sonali Thaokar[1], Prof. Pragati Patil[2]

[1] *Sonali Thaokar, Dept. of Compuiter Science & Engineering, AGPCE, Maharashtra, India*
[2] *Prof. Pragati Patil, Dept. of Compuiter Science & Engineering, AGPCE, Maharashtra, India*

Abstract - *The process of fixing a bug is called bug triage that goal is assigned to a developer for new coming bug. In a software firm, they spend their time and price to manage the bugs. So to reduce time and price of manual work in software firm they use automatic bug triage. By automatic bug triage, find predicted developer to resolve the bugs. In proposed approach, we used data reduction techniques and machine learning algorithm. To enhance standard of data, we used data reduction techniques, for that feature selection and instance selection techniques are used. We used feature selection and instance selection techniques at the same time to improve the accuracy of automatic bug triage. Also, we used machine learning technique for bug triaging system. We have added a new module here which will describe the status of the bug like whether it assigned to any developer or not and it is rectified or not. In addition, the load between developers based on their experience is re-balanced. The experimental result shows high prediction accuracy by using data reduction techniques and machine learning algorithm.*

Key Words: Bug data reduction, Feature selection technique, Instance selection technique, Machine learning algorithm technique, Bug Triage.

1.INTRODUCTION

In a software firm, bug fixing is very time consuming process. Many open source software projects have an open bug repository that makes it possible for each developer and users to publish defects or issues in the software, suggest possible enhancements, and remark on existing bug reports. In large open source software project have the bug repository that store the details of the bug. For large open source software project, the quantity of every day bugs is so substantial which makes the triaging process more challenging and difficult.There are two challenges associated with bug data that will have an effect on use of bug repositories in software development tasks, specifically the large scale data and low quality data. In a bug repository, a bug is kept up as a bug report, which record in the form of text that reproducing the bug and update as per the status of bug fixing. Manual bug triage is very time consuming for software firm because they spend their time and cost to manages the bug. The process of assigning a proper developer for fixing bug is the bug triage. By using automatic bug triage, software firm manages the bug easily and it save the time and cost of manual work. For automatic bug triage we used machine learning proposed data reduction techniques i.e. feature selection and instance selection techniques. By using these techniques reduce the bug data to save the labor price of developer and enhance the quality of bug data and increase the accuracy of bug triage. Section [2] describes background and Section [3] describes the system architecture of the proposed system. Section [4] describes the data set collection. The details of instance selection, feature selection is given in Section [5] implementation, the snapshot of proposed system given in Section [6] and concluded in Section [7].

2. BACKGROUND

Xuan et al. [1] proposed to reduce the bug data used instance selection and feature selection techniques. Their approach effectively reduced the data scale by using data reduction techniques improved accuracy of bug triage. Anvik et al. [2] they used supervised learning machine algorithm to assignment of bug report to the potential developer. They reached precision levels of 57% and 64% on Eclipse and Firefox respectively.

Alenezi et al. [3] in this approach used term selection method to recognize the good quality of bug report and improve accuracy and used naïve bayes classifier to predict the developer for each new bug report. They result shows that improved F-score.

Anjali et al. [4] proposed Domain Mapping Matrix to predicting the best suited developer to resolve the newly bug reports. They achieved an efficiency of 86% for top-10 and 97% for top-20 developer ranking list.

Cubranic et al. [9] this approach used text categorization dominates the existing bug triage. The first work of bug triage is a supervised text categorization approach using Naive Bayes. Their approach achieved 30% accuracy.

Nhan Minh Phuc [15] proposed To automatically detect duplicate bug reports, used Class-Feature-Centroid (CFC). The recall rate is improved by 10% for 20 predictions for SVN and AgroUML.

3. PROPOSED SYSTEM

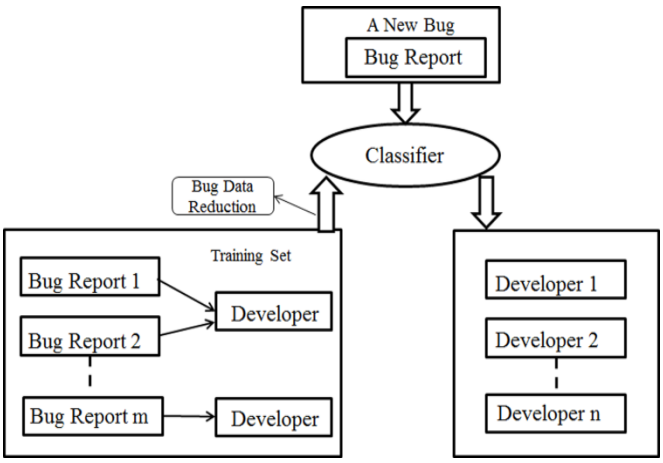

Fig-1: System Architecture

The process of settling the bug is the bug triage, which aim to correctly assign a potential developer to a new bug.

Figure 1 shows the architecture of the system. In the proposed system we collect the data set from eclipse. The input contains the bug data. Each bug data has the bug report and every bug report contain the summary and description. Bug reports are unstructured data which may contain irrelevant words. Therefore, we apply the traditional text processing approach to transform the text data into a meaningful representation.

In this proposed system there are two users one is developer and other is tester. In this proposed system use the data reduction techniques for decrease the scale of data and improve the accuracy of data. Data reduction techniques are applied to the data preparation of bug triage. Data reduction has two techniques, namely feature selection and instance selection. Both techniques are used for the data processing. The instance selection is used for the bug reports in bug data and the feature selection is used for words in the bug data. In the proposed system both techniques are grouping to use.

Artificial neural network classifier is more accurate as compared to naïve Bayes. As there is no limitation to bug data, testers can add huge number of bugs in the system. This is one of the greatest preferences of the proposed framework. Since the entire bug's data is interested in every one of the developers, it takes less time for the developer to take the choice. Developer can rapidly decide to fix the bug.

The purpose of the bug triage is assigning a potential developer to a new bug. Furthermore, in bug repositories, numerous developers have only fixed very few bugs. Such inactive developers may not provide adequate information for predicting correct developers. In our work we improve the more accuracy of the bug triage by using artificial neural network classifier.

4. DATASET COLLECTION

We gather bug reports from Eclipse repository. An eclipse link name is http://bugs.eclipse.org/bugs/show_bug .cg?Id=413120. The bug reports stored in xml file. From every bug report extract attributes like bug id, product name, bug name, description and status (Solved, Unsolved, Reopen, New). The given an example 1 show bug details which are extract from eclipse link and store in xml file. The example is given in xml format.

Example 1 the format of bugs store in xml files

```
<bug>
    <id>413120</id>
    <pname>EclipseLink</pname>
    <bname>Nested Embeddable Null pointer</bname>
    <desc>With weaving enabled for change tracking, a null pointer exception is raised when you create an object which references and embeddable, which in turn references another embeddable and the second embeddable is null. For example, I have a Contact Class which references an Address embeddable which in turn references a ZipCode Embeddable. Creating a new Contact, with a new Address which has a Null Zip Code causes a null pointer exception after persisting The Contact (entitymanager.persist(newContact)) and exiting the ManagerBean.</desc>
    <dev_assign></dev_assign>
    <status>NEW</status>
</bug>
```

In Table 1 show the summary of bugs. The dataset collect from eclipse repository. From eclipse 100 bug details are extracted and for that bug data set 18 developers are taken.

Table -1: Summary of Bugs

Name	Bugs	Developers
Eclipse	100	18

The snapshot of data set collection is given in section [6] figure 2.

5. IMPLEMENTATION

5.1 Feature Selection Technique

Feature selection aims to obtain a subset of relevant features (i.e., words in bug data). It is a preprocessing technique used for selecting a reduced set of features for large scale data sets. The preprocessing technique that we used symbol removal, stop word removal and stemming.

- Symbol Removal: remove unnecessary unwanted symbols

- Stop Words Removal: removing non-informative words which include articles, articles, prepositions, conjunctions and certain high frequency words (verbs, adverbs and adjectives). That does not give the meaning of the documents. These words are treated as stop words. Example for stop words such as the, in, a, an, with, etc.
- Stemming: Stemming is technique to reduce the words to their grammatical roots so that they can be represented with an only term.For example, the words connect, connected, connecting, connections all can be stemmed to the word "connect". The purpose of this method is to remove various suffixes, to reduce the number of words, to have accurately matching stems. We use porter stemmer for stemming the words.

Algorithm1 Pre-processing

Input: Bugs, stop word list, stemming list.
Output**:** Features.

```
S [] = split space by space
If S [] contains any symbol
    S [] = remove all symbols
End if
For each stop words list SLi
    For each of Si
        If SLi contains Si
            Remove Si from S []
        End if
    End for
End for
For each stem list SMi;
    For each Si
        If SMi contains Si
            Replace Si with SM [0]
        End if
    End for
End for
```

Example 2 Feature Extraction

Product Name = WTP Webservices

Bug Name= Axis2: Better error message with Axis2 scenario without installing Axis2 runtime

Bug Description= Since the Web services tools users are not familiar with this requirement to install Axis2 runtime first before going through any Axis2 Web service scenario, they should be shown a meaningful error message to point them exactly to what to do.

Feature Selection = [wtp, webservic, axi, better, error, messag, scenario, instal, runtim, sinc, web, servic, tool, user, ar, not, familiar, thi, requir, befor, ani, thei, shown, meaning, point, exactli]

The snapshot of the feature selection given in section [6] figure 3 and figure 4.

5.2 Instance Selection Technique

In instance selection, reduce the number of instances. By utilizing this technique original data sets are reduced by removing non-representative instances. E.g. out of multiple skills of developers find out proper developer skill.

The snapshot of an instance selection given in section [6] figure 5.

6. SNAPSHOT OF PROPOSED SYSTEM

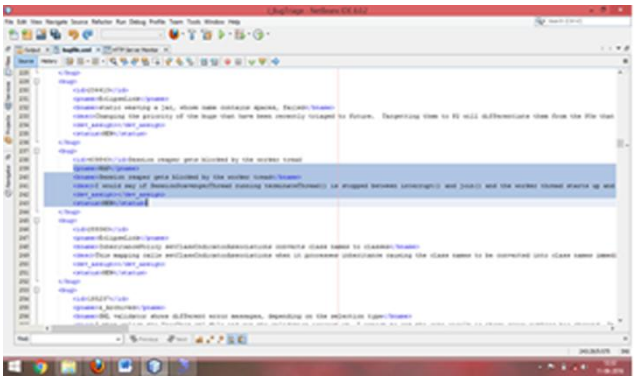

Fig -2: Bug data collection

Fig -3: Registration of bug

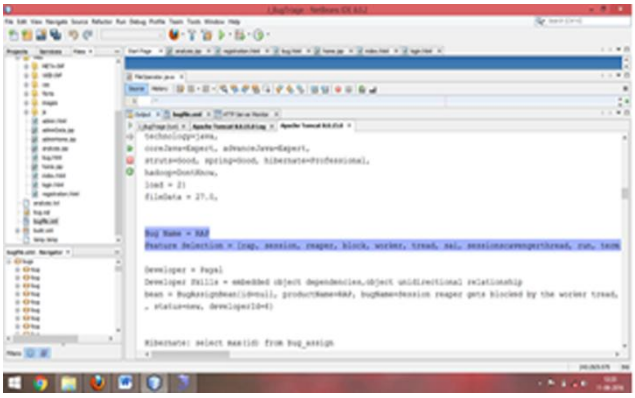

Fig -4: Feature selection

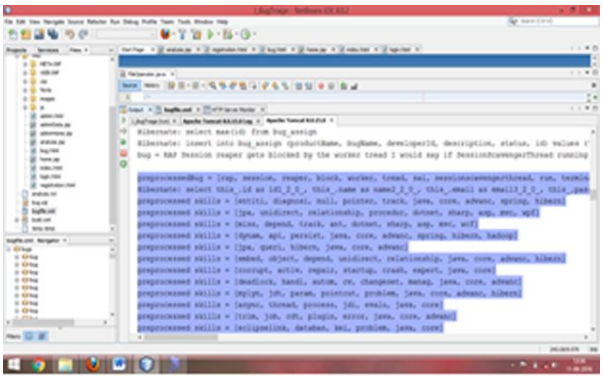

Fig -5: Instance selection

7. CONCLUSION

Bug triaging is crucial part in the various software development companies. But the manually bug triaging is very expensive in labor cost and time cost for software maintenance. So the automatic bug fixing is helpful for the developer to fix the bug.

In this paper, we proposed data reduction techniques for minimize the bug data set. We used combination of two reduction techniques they are feature reduction and instance selection technique. In proposed system, we used large open source project i.e. eclipse for data collection.

The proposed system provides the high quality of data and increase the accuracy of bug triage.

REFERENCE

[1] Jifeng Xuan, He Jiang, "Towards Effective Bug Triage with Software Data Reduction Techniques," IEEE Trans. on Knowledge and Data Engineering, vol. 27, no. 1, Jan. 2015.

[2] J. Anvik, L. Hiew, G. C. Murphy, "Who should fix this bug?," in Proc. 28th Int. Conf. Softw. Eng., May 2006, pp. 361–370.

[3] Alenezi and Kenneth Magel, "Efficient Bug Triaging Using Text Mining" (2013).

[4] Anjali, Sandeep Kumar Singh, "Bug Triaging: Profile Oriented Developer Recommendation" IJIRAE , 2349-2163, Vol. 2, Issue 1 (January 2015)

[5] S. Shivaji, E. J. Whitehead, Jr., R. Akella, and S. Kim, "Reducing features to improve bug prediction," Proc. IEEE/ACM. Intl. Conf. Automated Software Engineering (ASE 09), IEEE, Nov. 2009, pp.600-604.

[6] J. Xuan, H. Jiang, Z. Ren, J. Yan, and Z. Luo, "Automatic bug triage using semi-supervised text classification," in *Proc. Intl. Conf. Software Engineering & Knowledge Engineering*, 2010, pp. 209–214.

[7] 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

[8] Pankaj Gakare, Yogita Dhole, Sara Anjum et al , "Bug Triage with Bug Data Reduction ", IRJET on Computer Science and Engineering, vol. 2, no. 4, July 2015.

[9] D. Cubranic and G. C. Murphy, "Automatic bug triage using text categorization," in *Proc Sixteenth International Conference on Software Engineering*, Citeseer, 2004, pp. 92–97.

[10] Ashwini Jadhav, Komal Jadhav, Anuja Bhalerao, Amol Kharade, "A Survey on Software Data Reduction Techniques for Effective Bug Triage", IJCSIT on Computer Science and Information Technologies, vol. 6 (5), 2015, 4611-4612

[11] S. Kim, E. W. Jr., and Y. Zhang, "Classifying Software Changes: Clean or Buggy?" IEEE Trans. Software Eng., vol. 34, no. 2, pp. 181– 196,2008.

[12] R. Kumar, S. Rai, and J. Trahan, "Neural-Network Techniques for Software-Quality Evaluation," Reliability and maintainability Symposium,1998.

[13] A. Lamkanfi, S. Demeyer, E. Giger, and B. Goethals, "Predicting the severity of a reported bug," in Proc. 7th IEEE Working Conf. Mining Softw. Repositories, May 2010, pp. 1–10.

[14] J. Anvik, L. Hiew G. C. Murphy, "Coping with an Open Bug Repository," OOPSLA Workshop on Ecllipse Technology Exchange, 2005.

[15] Nhan Minh Phuc, "Improving Detection Performance of Duplicate Bug Reports Using Extended Centroid Features," IJARCCE, vol. 3, issue 10, October 2014, 4611-4612

[16] Porter M.F, An algorithm for suffix stripping, Program. 1980; 14, 130-137.

[17] Porter M.F, Snowball: A language for stemming algorithms. 2001.