Effective Bug Triage Using Cold-Start Recommendation System

Rajusekhar Alle, Aditya Aranya, Abhishek Thakur, Mr. Anilkumar J Kadam

[#]Department of Computer Engineering, Savitribai Phule Pune University
AISSMS COE, Pune, India
ajkadam@aissmscoe.com
sekhar.alle@gmail.com
Abshekt60@gmail.com
adityaaranya6@gmail.com

ABSTRACT

Over 40-50% cost is spent in dealing with software bugs in Software companies. An inevitable step of fixing bugs is bug triage, which aims to correctly assign a developer to a new bug. To decrease the time cost in manual work, text classification techniques are applied to implement automatic bug triage. A system can be constructed which will address the problem of data reduction for bug triage, i.e., how to reduce the scale and improve the quality of bug data. A combination of instance selection and feature selection can be used simultaneously to reduce data scale on the bug dimension and the word dimension. And this system will solve the Cold-Start Problem encountered in the current system at the starting phase when there is no trained dataset and this system will use designation-matching.

ARTICLE INFO

Keywords—Instance Selection, Feature Selection, Naive Bayesian Classifier, Bug triage, Prediction for reduction orders.

I. INTRODUCTION

WHAT IS BUG TRIAGE?

Bug Triage is the process of assigning bugs to developers as per the history of the particular developer. Mining software repositories is an interdisciplinary domain , which aims to employ data mining to deal with software engineering problems [11]. In modern software development ,software repositories are large-scale databases for storing the output of software development , e.g. , source code, bugs, emails, and specifications. Traditional software analysis is not completely suitable for the large-scale and complex data in software repositories[12]. Data mining has emerged as a promising means to handle software data. By leveraging data mining techniques, mining software repositories can uncover interesting information in software repositories and solve real world software problems. A bug repository plays an important role in managing software bugs.

Software bugs are inevitable and fixing bugs is expensive in software development. Large software projects deploy bug repositories to support information collection and to assist developers to handle bugs. In a bug repository, a bug is maintained as a bug report, which records the textual description of reproducing the bug and updates according to the status of bug fixing[16]. A bug repository provides a data platform to support many types of tasks on bugs, e.g., fault prediction, reopened bug analysis. In this paper, bug reports in a bug repository are called bug data. In traditional software development, new bugs are manually triaged by an expert developer , i.e., a human triager assigns bugs to the developers.

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 a related 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 , e.g. , Naive Bayes[13].Based on the results of text classification, human triager assigns new bugs by incorporating his/her expertise. To improve the accuracy of text classification techniques for bug triage, some further techniques are investigated, e.g., a tossing graph approach [14] and a collaborative filtering approach [15].

However, large-scale and low-quality bug data in bug repositories block the techniques of automatic bug triage. Since software bug data are a kind of free-form text data, it is necessary to generate well-processed bug data to facilitate the application . So the sample bug report given below shows an example of a bug report. The bug report is mainly divided into summary and description. A

developer reads through the description and undertakes that particular bug and changes the status of the bug whether it is resolved, assigned or not assigned.

Bug 123-**Tabs Reappearing After Closing** ← **Summary**
Status: Resolved fixed **Reported:** 2015-11-12 03:50 IST by Manish Paul **Modified:** 2015-12-27 (History) **CC List:** 4 users (show)
Product: Chrome **Component:** jst.ejb **Version:** 10.5 **Platform:** Android **Assigned To:** Oliver Queen ← **Assigned-to Developer**
← **Description** Build ID: I20090611-1540 Steps to solve problem: 1.Go to setting and turn off tabs options 2.Now tabs will not be displayed as different applications 3.And this reappearing problem is due to some development going on this topic. 4.So it will be done in new update and for now turn off tabs.

Fig 1. Example of a bug report.

II. ARCHITECTURAL OVERVIEW

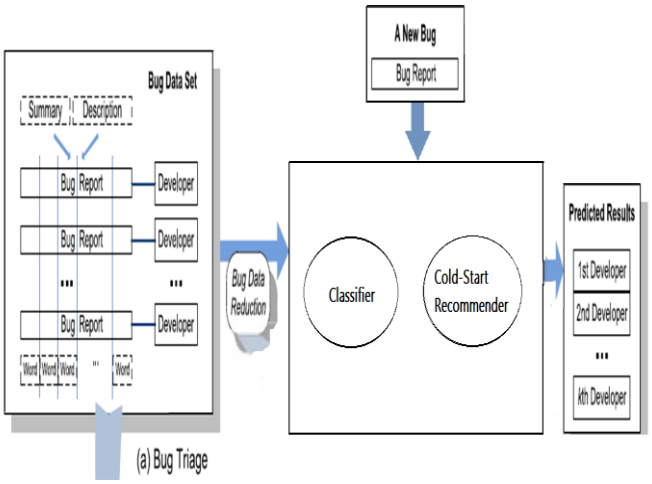

Fig 2. Architecture

An architectural overview is illustrated in Fig. 1,a tester adds a new bug to the system then the new bug is sent to the data reduction module where all the redundant words and duplicate records are eliminated. Here we use two data reduction techniques i.e., Feature Selection[17] and Instance Selection[9].After the bug is reduced by word and bug dimension then it is sent to the classifier where it is compared to the history of the developers. Then the matched developer is assigned to the bug. If there is no history or the trained dataset then the problem called Cold-Start problem can occur. So the system is designed in such a way that till every developer has no history we will use recommendation system using designation-match. So in the early stages we will use designation of the developers and assign the bugs as per their designations. And after the history for every developer is developed we will classify it using Naive Bayesian Classifier as usual.

So the recommendation system we will use will be called as Cold-Start Recommender. Also we will use combination of Instance Selection and Feature Selection which is a good strategy for achieving good accuracy[18]. The order whether to use Instance Selection first and then Feature Selection or vice-versa can be selected using prediction for reduction orders so as per this we will test some trained examples and find out accuracy and use that algorithm.

III. MODULES

A. Tester Login Module

In this module, we take username and password from tester to authenticate login to application.

B. Developer Login Module

In this module, we take username and password from developer and authenticate login to the application.

C. Bug Data Reduction Module

This module has two sub models:

- Instance Selection:

  This module is used to select the instances of bug reports and eliminate all the duplicate bugs or bug reports.

- Feature Selection:

  This module is used to eliminate stop words, noisy and redundant words from the bug reports. For Ex: the, are, version codes , etc. These all words are of no use as we are using plain text classification and they have no importance.

D. Classifier Module

In this module we classify the incoming bug reports by comparing them to each of the developers history and as per the number of words matched we will assign a developer.

We will use Naive Bayesian Classifier to classify as our application is a text classification and Naive Bayesian Classifier works good for text.

E. Adding Bugs

Tester adds new bugs to the system through this module.

F. Cold-Start Recommender Module

In this module we check whether there is any history of the developers in the trained data set for incoming bugs and if history is not present we will use this module and assign bugs as per the designation.

G. Prediction for Reduction Orders Module

This module includes selection of the order for reduction i.e., first Instance Selection then Feature Selection or vice-versa. So the system can be trained with some examples and check which yields good accuracy and choose that order.

H. Admin Module

The role of the admin is to login to the server and maintain databases of bugs. The administrator will be able to see the no. of bugs assigned and also how many are not assigned and also he will be able to see the graph of the developers.

I. Update Bug Status Module

This module includes all activities that take care of updating status of bug .

IV. ALGORITHMS

A. Data Reduction:

The system that will be constructed will use Instance and Feature Selection as data reduction algorithms. Combination of both these algorithms will be used as the combination of both these algorithms provides high accuracy rather than using them as separate algorithms.

A problem for reducing the bug data is to determine order of applying instance selection and feature selection which is denoted as the prediction of reduction orders. In this section, we first present how to apply instance selection and feature selection to bug data , i.e. , data reduction for bug triage.

ALGORITHM: Data reduction based on FS->IS

Input: training set 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 T_{FI} for bug triage
1) apply FS to n words of T and calculate objective values for all the words;
2) select the top n_F words of T and generate a training set T_F;
3) apply IS to m_I bug reports of 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 T_{FI}.

The same can be used for IS->FS vice-versa just have to interchange the steps. And the selection of the reduction order will depend on the results they provide for some trained examples.

For a given data set in a certain application, instance selection is to obtain a subset of relevant instances (i.e., bug reports in bug data) while feature selection aims to obtain a subset of relevant features (i.e.,words in bug data). In our work, we will employ the combination of instance selection and feature selection. To distinguish the orders of applying instance selection and feature selection, we give the following denotation. Given an instance selection algorithm IS and a feature selection algorithm FS, we use FS->IS to denote the bug data reduction, which first applies FS and then IS; on the other hand, IS->FS denotes first applying IS and then FS.

B. Naive Bayesian Classifier:

Naive Bayesian Classifier is a group of many classification algorithms. The bug triaging problem is converted into a text classification problem and then the Naive Bayesian Classifier which is a binary text classifier is applied. Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes theorem with the "naive" assumption of independence between every pair of features. Given a class variable y and a dependent feature vector $x_1$through x_n, Bayes' theorem states the following relationship:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)}$$

Using the naive independence assumption that,

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y),$$

for all i, this relationship is simplified to

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

Since $P(x_1, \ldots, x_n)$ is constant given the input, we can use the following classification rule:

$$P(y \mid x_1, \ldots, x_n) \propto P(y)\prod_{i=1}^{n} P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_y P(y)\prod_{i=1}^{n} P(x_i \mid y),$$

and we can use Maximum A Posteriori (MAP) estimation to estimate P(y) and P(x|y); the former is then the relative frequency of class y in the training set.The different Naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of P(x|y).In spite of their apparently over-simplified assumptions, Naive Bayes classifiers have worked quite well in many real-world situations , famously document classification and spam

filtering. They require a small amount of training data to estimate the necessary parameters. (For theoretical reasons why Naive Bayes works well, and on which types of data it does, see the references below.) Naive Bayes learners and classifiers can be extremely fast compared to more hard or sophisticated methods.

V. CONCLUSION

Our application, Effective Bug Triage Using Cold-Start Recommendation system improves the accuracy of bug triage by combining both Instance and Feature Selection algorithm[18]. Also, as this is a recommendation system the current problem of Cold-Start can be solved. Thus, our application can be used for effective assigning of bugs in software companies.

ACKNOWLEDGEMENT

Apart from our own, the success of this paper depends largely on the encouragement and guidelines of many others. We are especially grateful to our guide Prof A.J. Kadam and Prof D. P. Gaikwad, Head of Computer Engineering Department, AISSMSCOE who has provided guidance, expertise and encouragement. We are thankful to the staff of Computer Engineering Department for their cooperation and support .We would like to put forward our heartfelt acknowledgement to all our classmates, friends and all those who have directly or indirectly provided their over whelming support during this project work and the development of this paper.

REFERENCES

[1] C. C. Aggarwal and P. Zhao,"Towards graphical models for text processing,"Published by Knowl. Inform. Syst., vol. 36, no. 1, pp. 1- 21, 2013.
[2] 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. 11461150, Jun. 2012.
[3] D. Matter, A. Kuhn, and O. Nierstrasz," Assigning bug reports using a vocabulary-based expertise model of developers , "Published by Proc. 6th Int. Working Conf. Mining Softw. Repositories , May 2009, pp. 131140.
[4] E. Murphy-Hill , T. Zimmermann , C. Bird , and N. Nagappan,"The design of bug fixes,"Published by Proc. Int. Conf. Softw. Eng., 2013, pp. 332341.
[5] A . Lamkan , S . Demeyer , E .Giger , and B. Goethals , " Predicting the severity of a reported bug , " Published by IEEE 978-1-4799-3083 8/14/31.00,2014.
[6] J. Anvik, L. Hiew, and G. C. Murphy,"Who should fix this bug?," Published by Proc. 28th Int. Conf. Softw. Eng., May 2006, pp 361370.
[7] Eclipse. (2014). [Online]. Available: http://eclipse.org/
[8] Bugzilla, (2014). [Online]. Avaialble: http://bugzilla.org/
[9] D . R . Wilson and T . R . Martınez , " Reduction techniques for instance-based learning algorithms," Mach. Learn., vol. 38, pp. 257–286, 2000.
[10] C . Sun , D . Lo , S . C . Khoo , and J . Jiang , " Towards more accurate retrieval of duplicate bug reports," in Proc. 26th IEEE/ACM Int. Conf. Automated Softw. Eng., 2011, pp. 253–262.
[11] A. E. Hassan, " The road ahead for mining software repositories ," in Proc. Front. Softw. Maintenance, Sep. 2008, pp. 48–57.
[12] T. Xie, S. Thummalapenta, D. Lo, and C. Liu, "Data mining for software engineering," Comput., vol. 42, no. 8, pp. 55–62, Aug. 2009.
[13] D. Cubranic 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] 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.
[15] J. W. Park, M. W. Lee, J. Kim, S. W. Hwang, and S. Kim, " Costriage: A cost-aware triage algorithm for bug reporting systems," in Proc. 25th Conf. Artif. Intell., Aug. 2011, pp. 139–144.
[16] T. Zimmermann, R. Premraj, N. Bettenburg, S. Just, A. Schroter, and C. Weiss, "What makes a good bug report?" IEEE Trans. Softw. Eng., vol. 36, no. 5, pp. 618–643, Oct. 2010.
[17] M. Rogati and Y. Yang, " High-performing feature selection for text classification," in Proc. 11th Int. Conf. Inform. Knowl. Manag., Nov. 2002, pp. 659–661.
[18] Suvarna Kale and Ajay Kumar Gupta, "A Technique to Combine Feature Selection with Instance Selection for Effective Bug Triage.