Volume 5, Issue 6, June 2015

ISSN: 2277 128X



A Review of Bug Serverity Prediction Techniques Using Data Mining

Pankaj Rana M.Tech, Computer Science & Engg., Sri Sai University, Palampur, Himachal Pradesh, India Asst. Prof. Saurabh Sharma Computer Science & Engineering Sri Sai University, Palampur, Himachal Pradesh, India

Abstract: A crucial item of a bug report is that the alleged "severity", i.e. the impact the bug has on the victorious execution of the laptop computer code. The severity of a according bug might even be a major take into consideration deciding but quickly it/'s to be mounted. Bug fixing accounts for Associate in Nursing oversized amount of the package maintenance resources. Generally, bugs unit according, fixed, verified and closed. among the past analysis work, several text mining approaches unit of measurement projected to predict the severity pattern advanced learning models. Once the Bug is according on Bug chase System, their attributes unit analyzed and later allotted to varied fixers for his or her resolution.

Bug severity, associate attribute of a computer code package bug report is that the degree of impact that a defect has on the event or operation of a part or system. throughout this paper we\'ve a bent to analyze whether or not or not we have a tendency to ar able to accurately predict the severity of a reportable bug by analyzing its matter description exploitation text mining algorithms.

Prioritization of bugs decides the bug fix sequence. Incorrect prioritization of bugs ends up in delay of breakdown the required bugs, that leads delay in unhitch of the computer code package. Re-opened bugs increase maintenance costs, degrade the user-perceived quality of the computer code package and cause unneeded work on by busy practitioners.

Keywords: Bug Severity, Bug repositories, Bug priority, Severity Prediction, Natural language processing, machine learning

I. INTRODUCTION

Bug coverage and fixing is a vital section of package development, refinement and maintenance for each open and shut supply comes. varied bug coverage systems are developed for submission or coverage of a bug and

trailing their progress of fix. Bug coverage and trailing systems give a platform to record the problems/failures roundfaced by the shopper or user of the package. It helps in fixing future unharness in addition as measurement the standard of package by applying totally different mathematical and applied math models. it's important that the crucial bug ought to be mounted on priority basis. The urgency of the bug needs a parameter that must be provided throughout bug coverage method. The criticality of the bug is such by the severity of the bug that conjointly must be stuffed up throughout the bug coverage method. Bug severity is that the degree of impact that a defect has on the event or operation of a part system. Bug severity will be classified into totally different levels supported its impact on the system.

Bug severity has fully completely different vary of levels supported the appliance of the package. to boot to the priority, a package development team to boot keeps track of the questionable severity: the impact the bug has on the victorious execution of the package package. whereas the priority of a bug can be a relative assessment looking on the alternative reportable bugs and additionally the time until succeeding unleash, severity is associate absolute classification. Ideally, fully completely different persons news an identical bug have to be compelled to assign it an identical severity. Consequently, package comes typically have clear recommendations on the thanks to assign a severity to a bug. High severity typically represent fatal errors and crashes whereas low severity in the main represent cosmetic issues. looking on the project, several intermediate categories exist additionally. Despite their differing objectives, the severity can be a essential place confidence in deciding the priority of a bug. and since the number of reportable bugs is usually quite high1, tool support to help a development team in confirming the severity of a bug is fascinating. Since bug reports typically accompany matter descriptions, text mining algorithms ar likely candidates for providing such support.

Large code systems are getting more and more vital within the daily lives of the many folks. an oversized portion of the price of those code systems is attributed to their maintenance. Previous work treated all bugs equally, meaning, they didn't differentiate between re-opened and new bugs. Re-opened bugs square measure bugs that were closed by developers, however re-opened at a later time. Bugs will be re-opened for a range of reasons.

Debugging could be a major concern in computer code maintenance as a result of it always prices a lot of time and human resource to form corrections. For large-scale computer code comes with overwhelming numbers of defect reports, temporal arrangement demand is even a lot of tighter owing to an enormous quantity of debugging price. In most defect pursuit systems, the urgency of fixing the according defects is expressed in a very priority field.

Rana, et al., International Journal of Advanced Research in Computer Science and Software Engineering 5(6), June- 2015, pp. 1363-1366

With the frequent changes within the code ASCII text file to satisfy the dynamical and large necessities of the users, an oversized range of bugs ar being reported on each day. A bug could also be reported by a user, a developer, or by any staffer. we have a tendency to assign priority to a bug so a vital bug shouldn't be left untreated for an extended time. Correct prioritization is once more a haul. Triager (a one who analyzes, expands and overall refines the bugs) together with his information and skill assign the priority to the bug. Manually doing this can be a cumbersome task.

II. LITERATURE SURVEY

Attributes of a bug is wont to predict the priority, severity, fixer and standing of the bug. Canfora ANd Cerulo (2006) have projected a study on however modification requests are allotted to developers concerned in an open supply project (Mozilla and KDE) and a technique to counsel the set of best candidate developers to resolve a replacement modification request. Tamrawi et al. (2011) projected a fuzzy set-based approach for automatic assignment of developers to bug reports. Lamkanfi et al. (2010) investigated whether or not we are able to accurately predict the severity of a reported bug by analyzing its matter description victimization text mining algorithms. Authors finished that it's attainable to predict the severity with an inexpensive accuracy (both preciseness and recall vary between zero.65–0.75 with Mozilla and Eclipse; zero.70–0.85 within the case of GNOME). Chaturvedi and Singh (2012) have created an endeavor to demonstrate the pertinence of machine learning algorithms specifically NB, K-Nearest Neighbor, NB Multinomial, SVM, J48 and manslayer in determinative the severity category of bug report information of NASA from PROMISE repository.

In the analysis areas of computer code maintenance, severity prediction is Associate in Nursing rising issue since computer code repository mining grants several attentions recently. In 2008, Menzies Associate in Nursingd Marcus planned an automatic severity prediction mechanism by initial choosing the foremost vital words in step with their tf-idf (term frequency-inverse document In the analysis areas of computer code maintenance, severity prediction is AN rising issue since computer code repository mining grants several attentions recently. In 2008, Menzies ANd Marcus planned an automatic severity prediction mechanism by 1st choosing the foremost vital words in line with their tf-idf (term frequency-inverse document frequency) scores, then reranking the foremost informative terms in line with their data Gain (IG) measures, and eventually employing a rule-based machine learning tool to find out the classification rules. In 2010, Lamkanfi et al. explored the severity prediction drawback additional to answer additional queries of mistreatment text mining techniques . In their study, Naive Thomas Bayes (NB) classifiers were employed in 3 open supply comes, Mozilla, Eclipse, and GNOME. The NB classifiers give severity predictions in line with chance of the looks of words within the short outline descriptions. the most objective of their study is to debate four vital analysis problems in applying text mining techniques to the severity prediction task. First, they found that some terms square measure smart indicators for the severity prediction, particularly in their case-study, severe indicators tend to possess high specificity across completely different computer code elements, however non-severe indicators square measure abundant heterogenous. Second, they found that considering the longer full descriptions of defect reports impairs the prediction performance owing to misidentifying several severe defect reports as non-severe reports. The doable reason is that longer descriptions could contain several inapplicable words like stack traces and segments of ASCII text file specified the informative words can not be effectively extracted. In distinction, the words in brief outline descriptions square measure usually additional elliptical to elucidate the defects.

In 2011, Lamkanfi et al. additional created a comparison study to debate the prediction effectiveness of the NB classifiers, the Multinomial NB classifiers, 1-NN classifiers, and Support Vector Machines (SVM). Their experimental results show that Multinomial NB classifiers are able to do the very best performance within the space underneath Curve (AUC) measures mistreatment the Receiver operational Characteristic (ROC) analysis. According

to the studies in , we have a tendency to discuss the effectiveness of feature choice strategies supported the Multinomial NB classifiers. Since the elements studied during this paper have problem-specific characteristics as indicated in we have a tendency to don't discuss the cross-component prediction during this study.

In this study, we tend to have confidence the belief that reporters use doubtless important terms in their descriptions of bugs identifying non-severe from severe bugs. for instance, once a newsmanexpressly states that the appliance crashes all the time or that there's a literal within the application, then we tend to assume that we tend to ar managing severally a severe and a non-severe bug. The bug reports we tend to studied originated from Bugzilla bug chase systems wherever the severity varies between trivial, minor, normal, major, vital and blocker. There exist way clear tips on a to assign the severity of a bug. Bugzilla additionally permits users request options exploitation the overage mechanism within the type of a report with "severity" improvement.

Antoniol et al. additionally used text mining techniques on the descriptions of reportable bugs to predict whether or not a report is either a true bug or letter of invitation for associate improvement. They used techniques like call trees, provision regression and additionally a Na[°]ive theorem classifier for this purpose. The performance of this approach on 3 cases (Mozilla, Eclipse associated JBoss) indicated that reports is foretold to be a bug or an improvement with between seventy seven and eighty two correct selections.

Other current analysis regarding bug characterization and prediction principally apply text mining techniques on the descriptions of bug reports. This work will be divided in 2 groups: mechanically assignment fresh rumored bugs to associate degree applicable developer supported his or her experience and detection duplicate bug reports.

The main contribution of our work is that it with success addresses a very important drawback, which has been for the most part unheeded by the analysis community (i.e., automatic defect severity assessment from terribly loosely structured text). abundant previous work has used customary knowledge miners, to find out software system defect recognizers from historical records of static code options. Those data processing ways assume that the computer file is very

Rana, et al., International Journal of Advanced Research in Computer Science and Software Engineering 5(6), June- 2015, pp. 1363-1366

structured, that isn't the case, during this analysis, we tend to generate severity predictors from an information supply that's thus unstructured that it might defeat the antecedently explored data processing ways. the most finding of this work is that the success and potency of the answer stemming from the simplicity of its elements (i.e., the mix of ordinary text mining and rule learning methods), that additionally build it straightforward to use and pliant to different knowledge sets.

Problem Formulation:

The thesis starts with the assumption that automatic bug triaging is in its preliminary stage. It is generally manual task and depends on knowledge of expert. Thus it proves to be quite costly in terms of labor expenditure. Automation of triaging process can help triager in assigning particular bug to relevant developer and in deciding which bug needs to be fixed earlier. Bug severity classification helps in making automatic bug triaging process. Classification of impact of bug is important for various reasons.

It helps to decide which bug need immediate fixation among all reported bugs. It helps to decide which bug could be delayed and could be fixed in next release. Automation of classification of bug also helps in automating the bug triaging work. Thus the problem considered in this thesis work is "Automation of bug severity classification". This thesis presents an approach of bug severity classification which can improve the performance of automatic process of bug severity classification. The approach includes creating dictionary of critical terms that help in deciding severity level automatically. These critical terms are gathered after preprocessing of bug reports and using feature selection methods. This is accomplished by:

1) Extraction of datasets of bug reports of open source software from bug repositories.

2) Preprocessing of the datasets by using text mining approach.

3) Extraction of terms specifying severity levels using feature selection methods.

4) Creation of dictionary of critical terms specifying severity levels.

5) Use TF-IDF score for creation of a TDM matrix and using information gain and chi-square for dimensionality reduction.

6) Classification of the bugs by training the dataset using KNN and Naïve Bayes Classifier and testing the efficacy of the proposed algorithm in terms of accuracy, precision and recall.

Methodology/ Planning of work

The whole process of severity classification which is proposed in this thesis can be summarized as follows:

Datasets Acquisition . Preprocessing. Feature Selection. Creation of dictionary. Training and testing.

Bug reports are extracted from respective bug repository. Then preprocessing on textual information of bug reports is applied to obtain more reliable information. Term-document matrix (TDM) is created and by using feature selection method dictionary of critical terms is created. Then reduced TDM obtained by using critical dictionary terms is fed to classifier for classification of severe and non severe bug report.

Dataset Acquisition- Bug report instances are downloaded from bug repository. These instances were reported in Bugzilla (Bug Tracking System). Bug reports of Eclipse are considered in experiment. These reports were reported about four components of Eclipse. These components are UI, SWT, debug and core. Bug reports have severity of type block, critical, Enhancement, major, minor, normal, trivial. The reports that have normal and enhancement severity type are not considered in experiment. Bug reports with severity blocker, critical and major are considered as severe while bug reports with severity minor and trivial are considered as non severe.

Pre-processing- Pre-processing steps on textual description of bug reports are performed. It includes tokenization, stop word removal and stemming. Tokenization divides textual description into tokens by removing punctuation marks. Then stop words are performed that remove unnecessary information (conjunctions, interjections and articles) from datasets. Stemming on reduced datasets are performed to reduced terms into their root terms. Porter's stemming algorithm are used to perform stemming.

Term -Document matrix- The Term- Document Matrix (TDM) is created. Each column in matrix represents the terms occurring in documents and row represents id of each bug report. The cells of matrix are filled with TF-IDF score. If term is not present in the particular bug reports then cell is filled with zero.

Feature Selection- Feature selection methods are used to retrieve the most informative terms from corpus of datasets. In our research, we have used two feature selection methods info gain and CHI square methods. These methods are applied on TDM matrix to reduce matrix.

Creating dictionary of terms- The terms obtained after applying feature selection are sorted in descending order according to their weights. The top m- terms are used for creating dictionary. The dictionary contains the terms that help in specifying the severity levels of each bug report. Training and testing of Machine learning algorithms- The TDM matrix of dictionary terms are created and given to each of the selected Machine learning algorithms namely KNN and Naive Bayes for its training. The performance of classifier is analyzed and compared on basis of accuracy, precision and recall. Performance matrices Performance of selected classifier for our work is compared from the given metric described below:

Rana, et al., International Journal of Advanced Research in Computer Science and Software Engineering 5(6), June- 2015, pp. 1363-1366

i.**Accuracy:** Accuracy is the whole correctness of the model. It is calculated as the sum of correct classifications of class x divided by the total number of classifications of class x. It is defined as:

ii. **Precision:** Precision is the proportion of the examples which truly have class x among all those which are classified as class x. This is defined as:

iii. **Recall:** It is a measure of the ability of a prediction model to select instances of a certain class from a data set also called as sensitivity and corresponds to the true positive rate. It is defined as:

IV. CONCLUSION

In package maintenance, severity prediction on defect reports is associate degree rising issue for mitigating the extensive triaging price. within the past analysis work, many text mining approaches are projected to get correct predictions. Although these approaches demonstrate the effectiveness of predicting the severity, they are doing not discuss the matter of how to realize the indications in smart quality. A important item of a bug report is that the alleged "severity", and consequently tool support for the person reportage the bug within the sort of a recommender or verification system is fascinating. we have a tendency to identified that for the cases under investigation (two open supply systems: Eclipse and GNOME), Naive Thomas Bayes Multinomial is that the classifier with the best accuracy as measured by the Receiver operational Characteristic. Moreover, Na re Thomas Bayes Multinomial is additionally the quickest of the four and it needs the littlest coaching set. so we have a tendency to conclude that Na re Thomas Bayes Multinomial is best suited to the aim of classifying bug reports. This paper shows that it's doable to predict the severity based mostly on alternative data contained during a bug report, specially the matter data describing the bug. we have a tendency to evaluated the performance of predictions supported 3 cases drawn from the ASCII text file community (Mozilla, Eclipse and GNOME). We conclude that it's doable to predict the severity of a according bug given spare coaching information (five hundred bug reports for every severity) with an inexpensive performance where each exactness and recall vary between zero.65-0.75 for selected parts of Mozilla and Eclipse.).

REFERENCES

- [1] T. Zimmermann, R. Premraj, and A. Zeller, "Predicting defects for eclipse," in PROMISE '07: Proceedings of the Third International Workshop on Predictor Models in Software Engineering, 2007, p. 9.
- [2] M. Cataldo, A. Mockus, J. A. Roberts, and J. D. Herbsleb, "Software dependencies, work dependencies, and their impact on failures," IEEE Transactions on Software Engineering, vol. 99, no. 6, pp. 864–878, 2009.
- [3] M. D'Ambros, M. Lanza, and R. Robbes, "On the relationship between change coupling and software defects," Reverse Engineering,
- [4] C. Bird, N. Nagappan, P. Devanbu, H. Gall, and B. Murphy, "Does Distributed Development Affect Software Quality? An Empirical Case Study of Windows Vista," ICSE '09: Proceedings of the 31st International Conference on Software Engineering, pp. 518–528, 2009.
- [5] A. Lamkanfi, S. Demeyer, E. Giger, and B. Goethals, "Predicting the Severity of a Reported Bug," in *Proceedings of the 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*, 2010, pp. 1–10.
- [6] A. Lamkanfi, S. Demeyer, Q. D. Soetens, and T. Verdonck, "Comparing Mining Algorithms for Predicting the Severity of a Reported Bug," in *Proceedings of the 15th European Conference on Software Maintenance and Re-engineering (CSMR 2011)*, 2011, pp. 249–258.
- [7] T. Menzies and A. Marcus. Automated severity assessment of software defect reports. In *IEEE International Conference on Software Maintenance*, Oct. 2008, pp. 346–355.
- [8] T. Menzies and A. Marcus, "Automated severity assessment of software defect reports," in IEEE International Conference on Software Maintenance, 28 2008-Oct. 4 2008, pp. 346–355.
- [9] A. E. Hassan and K. Zhang, "Using decision trees to predict the certification result of a build," in ASE '06: Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering, 2006, pp.
- [10] "Bugzilla," http://www.bugzilla.org/.
- [11] "Bugzilla a bug's life cycle," <u>https://bugs.eclipse.org/bugs</u>.
- [12] S. Shivaji, E. J. Whitehead, Jr., R. Akella, and S. Kim, "Reducing Features to Improve Bug Prediction," in *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering (ASE '09)*,2009, pp. 600–604.
- [13] W. Shang, Z. M. Jiang, B. Adams, and A. E. Hassan, "Mapreduce as a general framework to support research in mining software repositories (MSR)," in MSR '09: Proceedings of the Fourth International Workshop on Mining Software Repositories, 2009, p. 10.
- [14] T. Menzies and A. Marcus, "Automated severity assessmentof software defect reports," in IEEE Internationa IConference on Software Maintenance, 28 2008-Oct. 4 2008, pp. 346–355.