Advanced Science and Technology Letters
Vol.129 (Mechanical Engineering 2016), pp.19-24
http://dx.doi.org/10.14257/astl.2016.129.05

Bug Severity Prediction by Classifying Normal Bugs with Text and Meta-Field Information

Kwanghue Jin[1], Amarmend Dashbalbar[1], Geunseok Yang[1], Jung-Won Lee[2] and Byungjeong Lee[1]

[1]Dept. of Computer Science, University of Seoul
Siribdae-ro 163, Dongdaemun-Gu, Seoul 20504, KOREA
[2]Dept. of Electrical and Computer Engineering, Ajou University
16-1 Woncheon, Suwon, Gyonggi-do, 443-749, KOREA
dct193@uos.ac.kr, bjlee@uos.ac.kr

Abstract. New software systems have been continuously developed and maintained to be useful and satisfy changing needs. Bug fixing is a fundamental activity in such constant software development and maintenance. If it is possible to predict bug severity accurately, it will be a huge contribution toward effective software development. Thus, activity in support of bug severity prediction is important for software development. If bug severity is predicted incorrectly, it brings about increased workloads for developers and increased expense. Therefore, in this study we present a method to improve prediction of bug severity. In this method we added text and meta-fields of the bug reports for our classifier model, and included 'normal' severity bug reports which represent a large amount of total reported bugs. The use of added fields and normal bugs were not taken into consideration in other studies. The bugs used in the experiments we performed cover large parts of the bug reports available in open source projects such as Mozilla and Eclipse. The experiments showed that the prediction accuracy was improved by the method we present.

Keywords: Severity prediction, normal bugs, meta-field based classification, bug fixing, software development and maintenance.

1 Introduction

New software systems have been developed continuously, and are used in a variety of fields. Thus, software bug fixing is an important phase of both open source and commercial software [1], so bugs should be fixed appropriately to their severity levels because they have different severity levels. Therefore, accurate prediction of bug

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2014M3C4A7030504), and by Seoul Creative Human Development Program funded by Seoul Metropolitan Government (No.CAC15106).

severity is required for successful software development and maintenance that developers are able know which bug need to be fixed urgently [2]. Previous studies used only text information of the bug report for their prediction method. Thus, in order to improve bug fixing, we introduce a reliable method of bug severity prediction that added the Description and Reporter fields because bug reporters record text attributes (Summary and Description) of the report. We also use parts of the datasets which other study did not use.
Below is the contribution of this study.

- Past studies did not use a significant amount of bug data that is classified as pertaining to a normal level of severity in the prediction methods. In this study, we used most of the bug reports except reports with the severity status of Enhancement.
- Past studies did not show high resultant reliability due to using only text information in a bug report for bug severity prediction on a limited basis. In this study, the Reporter attribute, which records text information, was added as classification.
- In order to evaluate the proposed method, comparisons with other studies that only used text information were made and they showed that the proposed method is more efficient and more accurate.

The rest of our paper is structured as follows. Section 2 introduces some previous related studies on bug severity prediction. Section 3 introduces the severity prediction method and shows a severity prediction example. Section 4 shows experiment result and finally Section 5 concludes the paper.

2 Related Work

In previous research on bug severity prediction, Lamkanfi et al. [3, 4] described text mining algorithms to find out which is the most effective algorithm for predicting bug severity, by its text information. Their study compared four text mining algorithms including Naive Bayes (NB), MNB, SVM and K-nearest [4]. The MNB classifier generally outperformed other classification.The SEVERIS, an automated method which predicts the severity of a reported bug, was introduced in the study by Menzies et al. [5]. SEVERIS also combined those well-known text mining and machine learning algorithms.. However, the dataset was not sufficient since 1 to 617 bug reports were used per severity level and 5 levels were defined in the paper. The experiment result for 79000 terms in 775 bug reports ranged between 65%-98% in terms of the F measure.

3 Bug Severity Prediction

Information on bugs that are found during program runtime is an essential factor for the successful development and maintenance of a project. Therefore, predicting bug severity levels is fundamental for project planning. But if bug severity levels are predicted inaccurately, that increases project expense and developers' workloads. In this paper we used text attributes such as Summary and Description, and additional attributes (Product, Component, Reporter, and Severity) of the bug report for our classifier model which was trained by a training data set that other studies did not use in order to make a reliable and highly accuracy classifier.

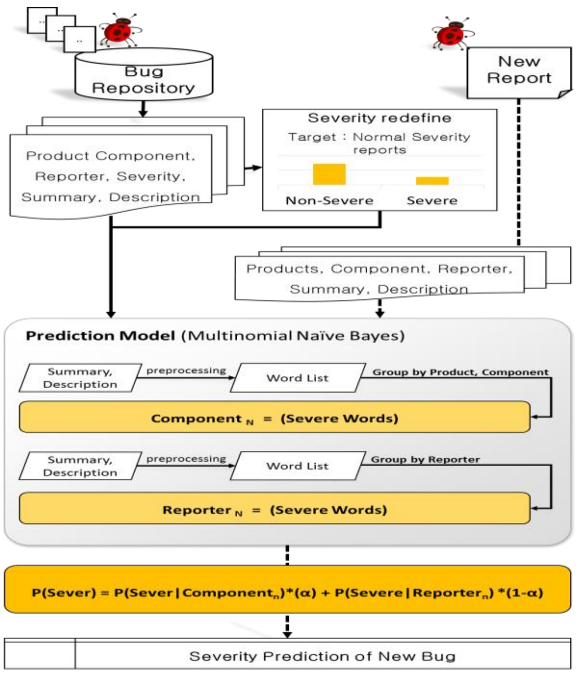
Bug Repository
New Report
Product Component, Reporter, Severity, Summary, Description
Severity redefine
Target : Normal Severity reports
Non-Severe
Severe
Products, Component, Reporter, Summary, Description
Prediction Model (Multinomial Naïve Bayes)
Summary, Description
preprocessing
Word List
Group by Product, Component
Component N = (Severe Words)
Summary, Description
preprocessing
Word List
Group by Reporter
Reporter N = (Severe Words)
P(Sever) = P(Sever|Componentn)*(α) + P(Severe|Reportern) *(1-α)
Severity Prediction of New Bug

Fig. 1. Flow of proposed method

Fig. 1 shows the process flow of the proposed method and how it predicts newly reported bugs by the severity levels that we redefined. The process flow of the proposed method is defined as follows: Firstly, we collect the reported bugs and extract features such as product, component, reporter, severity, summary and description. Then, it proceeds to preprocessing tasks [6]. After the preprocess task on the extracted text is finished, the activity that re-classifies normal bugs runs. We reclassify the normal bugs in order to involve terms that are included in both normal and non-normal bug reports. The final stage for defining a bug severity prediction method is grouping all terms included in redefined bug reports by Reporter and Component fields, respectively into two classes, non-sever and severe. Major, critical and blocker levels in bug reports are severe, where trivial and minor levels are non-severe. By the prediction method defined above, when the reporter reports new bug, it goes through pre-processing based on predefined information in the component and reporter fields before the model classifies it to a suitable level through the text similarity technique.

Table 1. Example for Naïve Bayes usage

Bug Report		Words	Severity
Existing Bugs	No.1	serious, extremely, serious	Severe
	No.2	serious, serious, very	Severe
	No.3	serious, peculiarly	Severe
	No.4	weak, light, serious	Non Severe
New Bug	No.5	serious, serious, serious, weak, light	Severity Prediction

Prior probability of the example shown in Table 1 is divided by severe and non-severe types and is computed below.

Prior probability:

P (Sever) = 3/4   P (Non-Severe) = 1/4

Next, each conditional probability of words in a new bug report is shown below.

Conditional probability:

P (serious │Sever)= (5+1)/(8+6) = 3/7
P (weak │Sever) = (0+1)/(8+6) = 1/14
P (light │Sever) = (0+1)/(8+6) = 1/14

P (serious │Non Sever) = (1+1)/(3+6) = 2/9
P (weak │Non Sever) = (1+1)/(3+6) = 2/9
P (light │Non Sever) = (1+1)/(3+6) = 2/9

Finally, the result for 5 words' probability is 0.0003% possibility for severe and 0.0001 for non-severe.

Severity level selection:

P (Sever │5words) = 3/4*(3/7)3*1/14*1/14 = 0.0003
P (Non-Sever │5words) = 1/4*(2/9)3*2/9*2/9 = 0.0001

Thus, newly reported 5th bug is predicted as severe bug.

4. Experiments

Bug reports includes a status field which shows the current status of bug, a resolution field which presents information related to the bug, an assignee field that shows who is responsible for fixing the bug and a Reporter field which presents the reporter of bug. Those fields contain single-attribute values and other fields such as CC list, keywords and flags are multiple-attribute values. Bug report fields such as Product, Component, Reporter, Severity, Description and Summary were used in the experiment. These six fields of the bug report were selected for experiment in order to improve prediction accuracy for bug severity. The criteria for severity prediction may vary depending upon the value of Product and Component fields and also on who reported the bug. Therefore, Severity, Description and Summary fields are also essential for determining the severity level of bugs. Proposed method works as follows: First the method groups bug reports by their Product field and again groups them by Component and Reporter fields. After grouping them, the method extracts information from Description, Summary and Severity fields. Next, Multinomial Naïve Bayes(MNB) is applied by using the extracted terms.

The comparison of the proposed model with those of Lamkanfi [3] method is shown in Fig. 2. In the future, optimization of an execution engine with JavaScript characteristics applied which will operate more efficiently than the web based interpreter. For purposes of comparing these methods, we used the dataset used in Lamkanfi. The comparison result shows that the proposed method outperformed Lamkanfi method and the F measure of the Eclipse and Mozilla datasets has been improved.

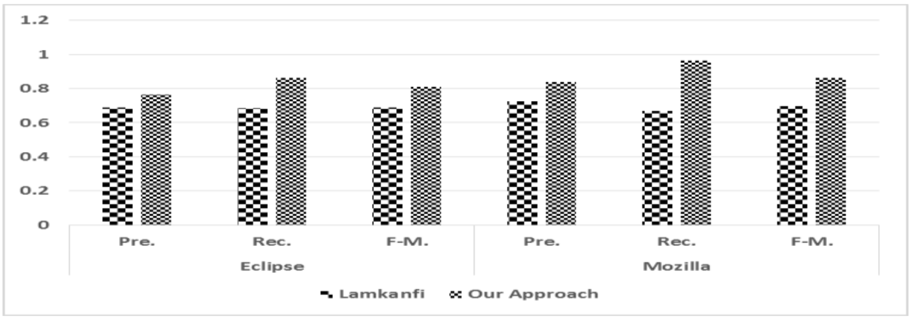

Fig. 2. Comparison result based on F-measure value

5 Conclusion

In this paper text data and meta-field data in bug reports which have not been used in other studies were added in order to improve the prediction accuracy of the classification model. The training dataset also included bugs, not used in other studies, which are set to normal status by the reporter when the report is initially submitted. The experimental results showed that the prediction accuracy of our proposed model has been improved. However, other fields aside from Component and Reporter were analyzed but have not shown any effect on improvement. Analyzing the factors affecting the bug severity prediction through experiment is expected to further improve performance. Finally, for bugs submitted by reporters who are doing so for the first time, the accuracy may fall for the reason that only the component field is used for prediction because no past report history exists. Text information used for prediction is also objective. Thus, further research that presents objective criteria to be effective for prediction of bug severity is needed.

References

1. Chaturvedi, K.,Singh V.: Determining bug severity using machine learning techniques. In Proc. of 2012 CSI Sixth International Conference on Software Engineering (CONSEG), IEEE, 2012
2. Giger, E., Pinzger, M., Gall, H.: Predicting the Fix Time of bugs. In Proc. of International Workshop on Recommendation System for Software Engineering (RSSE), pp. 52-56, 2010
3. Lamkanfi, A., Demeyer, S, Giger, E., Goethals, B.: Predicting the Severity of a Reported Bug: In Proc. of IEEE Working Conference on Mining Software Repositories, pp. 1-10, 2010
4. Lamkanfi, A., Demeyer, S., Soetens, Q. D., Verdonck T.: Comparing Mining Algorithms for Predicting the Severity of a Reported Bug. In Proc. of 15th European Conference on Software Maintenance and Reengineering (CSMR 2011), pp. 249-258, 2011
5. Menzies, T., Marcus, A.: Automated Severity Assessment of Software Defect Reports. In Proc. of 24th IEEE International Conference Software Maintenance, pp. 346-355, 2008
6. Feldman, R., James, S.: The text mining handbook: advanced approaches in analyzing unstructured data. Cambridge University Press, 2007.