INFORMATION
Volume 19, Number 2, pp.651-659

ISSN 1343-4500

Utilizing Feature based Classification and Textual Information of Bug reports for Severity Prediction

Kwanghue Jin*, Eun Chul Lee*, Amarmend Dashbalbar*, Jungwon Lee ** and Byungjeong Lee**

** Department of Computer Science, University of Seoul, Seoul, Korea*
E-mail: {dct193, fdc517, bjlee}@uos.ac.kr

*** Department of Electrical and Computer Engineering, Ajou University, Seoul, Korea*
jungwony@ajou.ac.kr

Abstract

Predicting bug severity is important task in software development and maintenance. If bug severity is predicted accurately, it would be a significant assistance for software developers to allocate resource and fix bugs. Thus, this paper presents a way to predict bug severity. First, this study trains bugs in bug repository which stores previously reported bug reports.

This study applies feature based classification using meta-fields in bug reports in training where Multinomial Naive Bayes(MNB) is used. Next, when new bug is reported, this study predicts its severity by using textual information (Summary, Description) of new bug and existing bugs. We evaluate the performance of our method using two large-scale open-source projects, including Eclipse, and Mozilla. The experimental results reveal that our approach outperforms other severity prediction method.

Key Words: Bug reports, Severity prediction, Text similarity, Open source projects, Software maintenance

1. Introduction

Many software systems are increasingly developed and used in various fields. Software complexity is growing day by day and big software systems which consist of various components are being developed. Bug severity prediction is desired in error fixing task and allocating resources effectively such as manpower and budget. Normally, other studies on bug severity prediction mainly used textual information of bug report for its prediction method [1, 2, 3]. They did not show good result. Therefore, in this paper we used not only textual information but also additional attributes to predict bug severity to obtain better result than other studies.

Contributions of our study are:

- This study introduces a new effective method based on textual information and meta fields (attributes) of bug reports.

- This paper presents comparisons with other method and shows that our method produces better prediction result.

The rest of the paper is structured as follows. Section 2 provides brief information on other related works. Section 3 introduces severity prediction method and Section 4 describes results of our experiment. Finally, we conclude our study in Section 5.

2. Related work

There are several works for improving severity prediction of reported bugs. Most of them are based on text mining methods using textual information in bug reports to predict severity of bug using terms included in the bug reports.

Menzies[2] introduced automated method named SEVERIS which automatically predicts severity of reported bugs. SEVERIS is based on text mining and machine learning algorithms such as Tokenization, Stop word removal, Stemming, etc. This tool was applied to data sets of five anonymous PITS projects run by NASA's Independent Verification and Validation Facility. This paper distinguished severity level one to five and training data set sizes are 1 to 617 bug reports per severity. Result of experiment that used 79000 terms in 775 bug reports ranged between 65%-98% in terms of F-measure. The authors concluded their work that it has shown that usage of text mining and machine learning methods which make it possible to automatically generate levels of bug severity from the free text entered into PITS.

Lamkanfi[3] studied text mining algorithms to compare them and explore which algorithm is the most effective for predicting severity of bug based on its textual information. Four text mining algorithms were compared in this study which Naive Bayes(NB), MNB, K-nearest and SVM. In order to examine the efficiency of these algorithms Eclipse and GNOME bug databases were used. Area Under Curve results have shown MNB classifier outperformed other classification methods except GEF/Draw 2D case. The study suggested that around 250 bug reports of each severity are suitable size for training set.

Tian[4] proposed an approach for automatically predicting bug severity using BM25-based document similarity function. Total of more than 65000 bugs were collected from three big open source project bug databases Eclipse, Open Office and Mozilla. The study used textual attributes of bug report such as summary, description and non-textual ones such as product, component. Those attributes were used for comparing features of bug reports and similarity measure is then used in a nearest-neighbor method to assign severity label to bug report. The result showed that we can improve the F-measure of the state-of-the-art approach especially on bug severity that hard to predict.

Yang[5] suggested that using attributes other than textual information of bug reports can be more efficient way to predict severity of bug. The attributes that were investigated in this study are stack traces, report length, attachments, steps to reproduce and it also combined the textual information in bug report summary with these attributes. MNB classification was used for all configurations and firstly it examined the existence of the attributes in the bug report and then quantity of them. Eclipse bug database was used for the experiment. The result showed that the stack traces have the most effective contribution than other indicators and explained why most of severe bug reports contained stack traces.

Roy[6] studied improving prediction accuracy based bi-grams and text mining methods. The study used Eclipse and Mozilla bug report database for its study and adopted NB classifier for the classification. Also the study identified the best terms which have the highest probability included in the severe bug report such as "crash" and "deadlock". However, the study did not classify non-severe bugs accurately because most of non-severe bugs include less characterizing terms in their textual information of bug report.

3. Predicting bug severity

3.1 Overview

This paper introduces a new prediction method and presents comparison with other studies. Figure 1 shows an overview of the severity prediction method that we propose. In this study we train the bug reports which is stored in bug repository by using MNB. Training data is classified into severe and non-severe bugs, where severe bugs have severity level of "major", "critical", "blocker" and non-severe bugs have "trivial", "minor" levels [7].

Training set is trained twice. First, it is trained by the reporter field and then component field with summary and description information respectively. Thus, bug reports in training set we used are classified into severe and non-severe. Table 1 shows severity levels of training set.

Table 1. Severity levels of training set

SEVERITY	NON SEVERE	SEVERE
Severity Levels	trivial, minor	major, critical, blocker

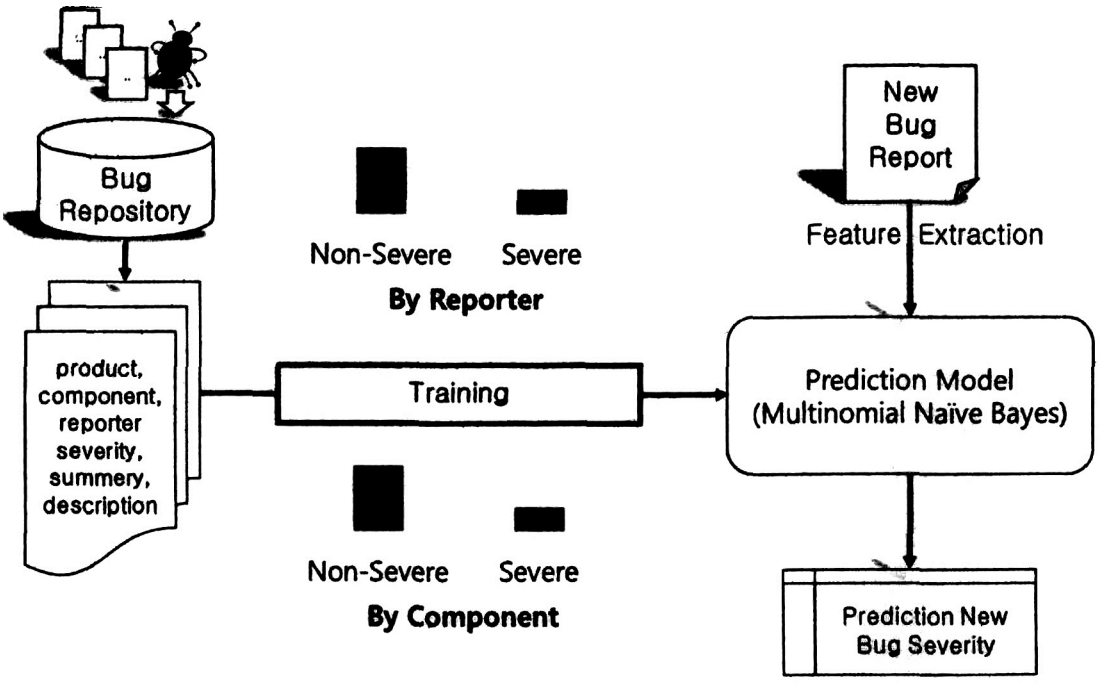

Fig. 1. Overview of severity prediction

3.2 Classifier

NB classifier is based on Bayes rule. Bayes rule expresses the relation between prior and post probability with the naïve assumption of independence between every pair of features. Bayes classifier performs far better than we expected in spite of its simple design and assumption. MNB is one of Bayes variant algorithms which mainly used in text classification. NB only considers only the existence of terms in document but MNB considers the number of occurrence of terms. According to Lamkanfi[3], MNB showed better prediction accuracy than NB.

4. Experiment

4.1 Experiment Data

Our data set includes total 9852 bug reports that reported between January, 2008 and December, 2013 which collected from 6 products of Eclipse and Mozilla open source projects.

Bug reports with severity level of "major", "critical" or "blocker" are considered as severe bug and "minor", "trivial" ones are non-severe. The bug reports of which severity attribute is "enhanced" and "normal" were excluded from experiment because they were considered as unreliable bug just like other studies did [1]. Data set is shown in Table 2.

Table 2. Number of sever and non-severe bug reports

Project	Product	# of Bug Reports		
		Total	Severe	Non-Severe
Eclipse	JDT	1412	732	680
	PDE	728	507	221
	Platform	3729	2634	1095
Mozilla	Bugzilla	1315	545	770
	SeaMonkey	1466	994	472
	Calendar	1202	832	370

Experiment data were used for training our classifier and evaluating classifier performance. In order to check whether our training set is big enough to perform stable result or not we divide subset into 10 parts and use 9 parts for train our classifier and use the remaining 1 part to measure classifier performance.

4.2 Evaluation

In this work we used confusion matrix in order to analyze bug trend more accurately. Table 3 shows confusion matrix of binary classification. For example, n11 is number of severe which were classified correctly and n12 is number of severe sample which were classified wrongly. Confusion Matrix shows result of our classier that how it classifies severity level wrongly.

Table 3. Data set

Result / Classify	ω_1	ω_2
ω_1	n_{11} (TP)	n_{12} (FN)
ω_2	n_{21} (FP)	n_{22} (TN)

The result of experiment can be divided into four cases. First case is True Positive (TP). TP represents severe samples which classified correctly, second case True Negative(TN) represents non-severe samples which classified correctly, then False Negative(FN) is severe samples which recognized as non-severe bug and classified wrongly, lastly non-severe samples which recognized as severe bug is represented by False Positive(FP). In order to evaluate the performance of our severity prediction method, we used most widely used evaluation measures: precision, recall and F-measure. The formula used is as follows.

- Precision = $\frac{TP}{TP + FP}$
- Precision indicates experiment result accuracy.
- Recall = $\frac{TP}{TP + FN}$
- Recall indicates the coverage of the experiment result.
- F-Measure = $\frac{2 * Precision * Recall}{precision + Recll}$
- F-Measure takes the precision and the recall into consideration. It is a combined precision and recall.

4.3 Experiment Result

Table 4 shows precision, recall and F-measure by severe and non-severe bug results separately. Data set were gathered from 3 products (Platform, PDE, and JDT) of Eclipse and 3 products (Calendar, SeaMonkey and Bugzilla) of Mozilla.

Table 4. Experiment result (Precision, Recall, F-Measure)

	Severe			Non-Severe		
Eclipse	Prec.	Rec.	F-M.	Prec.	Rec.	F-M.
Platform	0.85	0.68	0.76	0.49	0.72	0.57
PDE	0.85	0.58	0.69	0.46	0.78	0.57
JDT	0.84	0.64	0.73	0.69	0.86	0.76
Mozilla	Prec.	Rec.	F1	Prec.	Rec.	F1
Calendar	0.86	0.91	0.89	0.74	0.65	0.69
SeaMonkey	0.83	0.83	0.83	0.66	0.67	0.66
Bugzilla	0.72	0.67	0.69	0.78	0.82	0.79
Eclipse	Prec.	Rec.	F-M.	Prec.	Rec.	F-M.

4.4 Comparison analysis

Lamkanfi[1] used two attributes (Component, Summary) of bug report and NB algorithm for training. Whereas, in this study we used four attributes (Component, Reporter, Summary, and Description) and MNB algorithm for training. Our method showed better result than A.Lamkanfi[1]'s study because our study used more fields and MNB that makes us possible to use effective vector representation than NB which was used in A.Lamkanfi[1] (Table 5).

Table 5. Method comparison

Method	Classifier	Attributes
Lamkanfi [1]	NB	(2) Component, Summary
Our Approach	MNB	(4) Component, Reporter, Summary, Description

Figure 2 shows comparison result between our method and Lamkanfi[1] for Eclipse and Mozilla open source projects.

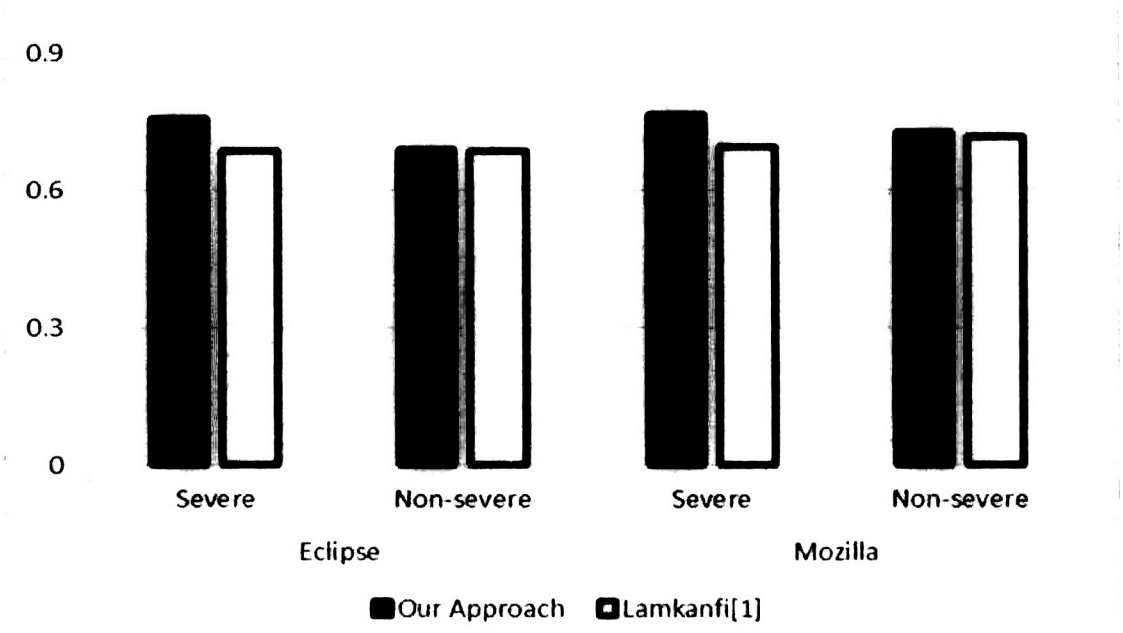

Fig. 2. Quantitative comparison

The comparison shows that our method predicts more accurate than A. Lamkanfi[1]. From this comparison we can expect that applying MNB with more attributes is an effective way to perform more accurate prediction.

5. Conclusion

In this study we introduced a new bug severity prediction method using previously reported bug information. First, we trained bug reports by using information in meta fields of them in bug repository and used the set to predict the severity level of new bug when it is reported. The experimental results revealed that our approach outperformed other severity prediction method.

In non-severe bug reports, terms are diverse and frequency of terms is very low. For that reason prediction performance of our study is similar to that of the previous study in predicting non-severe bugs. Therefore, more investigation is required to improve prediction of non-severe bug reports.

If bug reporter is a beginner, prediction accuracy may be not good because there is no trained terms for the reporter. Also bug reporter may often use the very similar format for different types of bug. It may result in low prediction accuracy. Thus we have to investigate improving this low accuracy. Finally, we need to analyze other attributes that can affect the prediction accuracy in order to obtain more reliable result

6. Acknowledgments

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 Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(No. 2012-0007149).

References

[1] A. Lamkanfi, S. Demeyer, E. Giger, and B. Goethals, "Predicting the Severity of a Reported Bug," In Proc. of IEEE Working Conference on Mining Software Repositories, (2010), 1-10.

[2] T. Menzies and A. Marcus, "Automated Severity Assessment of Software Defect Reports," In Proc. of 24th IEEE International Conference Software Maintenance, (2008), 346-355.

[3] A. Lamkanfi, S. Demeyer, Q. D. Soetens, and T. Verdonck, "Comparing Mining Algorithms for Predicting the Severity of a Reported Bug," In Proc. of 15th European Conference on Software Maintenance and Reengineering (CSMR 2011), (2011), 249-258.

[4] Y. Tian, D. Lo, and C. Sun, "Information Retrieval Based Nearest Neighbor Classification for Fine-Grained Bug Severity Prediction," In Proc. of 19th Working Conference on Reverse Engineering (WCRE '12), (2012), 215-224.

[5] C.-Z. Yang, K.-Y. Chen, W.-C. Kao and C.-C. Yang, "Improving Severity Prediction on Software Bug Reports using Quality Indicators," In Proc. of 5th IEEE International Conference on Software Engineering and Service Science (ICSESS), (2014), 216 – 219.

[6] N.K.S. Roy and B. Rossi, "Towards an Improvement of Bug Severity Classification," In

Proc. of 40th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), (2014), 269 – 276.

[7] I. Herraiz, D. M. German, J. M. Gonzalez-Barahona, and G. Robles. "Towards a Simplification of the Bug Report Form in Eclipse," In Proc. of International Working Conference on Mining Software Repositories, (2008), 145–148.

* Corresponding author: Byungjeong Lee, Ph.D.
Department of Computer Science and Engineering,
University of Seoul,
163 Seoulsiripdaero (90 Jeonnong-dong),
Dongdaemun-gu, Seoul 130-743 KOREA
E-mail: bjlee@uos.ac.kr