AN IDENTIFICATION METHOD OF RELATED GROUP THREADS FOR A RECENT BUG THREAD BY PEAK CHARACTERISTICS OF SIMILARITIES

Yuuki Imanara, Kota Itakura, Masaki Samejima and Masanori Akiyoshi
Graduate School of Information Science and Technology, Osaka University, 2-1, Yamadaoka, Suita, Osaka, Japan

Keywords: Bug tracking system, Related group thread, SVM, Peak characteristics of similarities.

Abstract: This paper addresses the problem to identify the related group threads that has dependent relationships with recent bug threads. Because most of recent bug threads have no dependent relationships with group threads, basic approach based on similarity regards them as having dependent relationships wrongly. In this paper, we propose an identification method of related group threads by peak characteristics of similarities. The proposed method removes recent bug threads that have no dependent relationships by Support Vector Machine based on vectors representing peak characteristics of similarities between the recent bug thread and group threads. The application result shows that the precision rate is improved by 49% and the recall rate is kept 76% on average using the proposed method.

1 INTRODUCTION

In open source software development, communities for developers are organized, and they discuss who fixes the bug and how to fix it. In order to support their discussion and manage bug information, bug tracking systems (Serrano and Ciordia, 2005; Matsushita et al., 2005) are introduced.

The bug tracking system generally consists of bug threads posted by developers. Each bug thread has a title, the progress to fix, developers' comments, and the dependent relationship. The dependent relationship indicates the relationship that one bug can not be fixed unless the other bug is fixed (Souza et al., 2007). The bug threads that have dependent relationships each other are organized as "group thread". Every time a recent bug is reported, developers find the group thread which the recent bug thread has dependent relationships with bug threads in, that is called "related group thread" to improve their discussion (Black, 2002; Chen et al., 2010). Because there are dozens of group thread, it is difficult for developers to find the related group thread (Zimmermann, 2009; Gall et al., 2003). The purpose of this research is to identify the related group thread for the recent bug thread automatically.

Since threads that has dependent relationships each other have common symptom of the bugs, comments on the threads are often similar (Nagwani and Singh, 2009). So, the similar group thread to the recent bug thread can be regarded as the related group thread. With this concept, the basic approach is to derive similarities between the recent bug thread and each group thread by Cosine Similarity (Sullivan, 2001), and to decide the related group thread as the group thread that has the highest similarity more than a threshold. The threshold is derived from similarities among existing bug threads. However, some of the recent bug threads are similar to the thread group, but do not have dependent relationship with the existing bug threads because the recent bug does not have enough comments and the similarity is not correctly derived. This causes misidentification of the related group thread. So, it is necessary to extract characteristics of the misidentified related group thread and remove the recent bug thread before the identification (Imanara et al., 2011).

We propose an identification method of related group thread by peak characteristics of similarities. In case that the recent bug thread has dependent relationships with the related group thread, the similarity with the related group thread is very high but the similarity with the other group thread is low. We call the characteristics of similarities "peak characteristics". So, the peak characteristics of similarities can be on these similarities with the related group thread. Two kinds

Imanara Y., Itakura K., Samejima M. and Akiyoshi M..
AN IDENTIFICATION METHOD OF RELATED GROUP THREADS FOR A RECENT BUG THREAD BY PEAK CHARACTERISTICS OF SIMILARITIES.
DOI: 10.5220/0003506401790184
In *Proceedings of the 6th International Conference on Software and Database Technologies* (ICSOFT-2011), pages 179-184
ISBN: 978-989-8425-77-5
Copyright © 2011 SCITEPRESS (Science and Technology Publications, Lda.)

of feature vectors, higher similarities and differential of similarities, generated from the peak characteristics. And Support Vector Machine (SVM) based on the feature vectors classify the recent bug thread not to have dependent relationships.

2 IDENTIFICATION OF RELATED GROUP THREADS FOR A RECENT BUG THREAD

2.1 Problem on Identification of Related Group Threads

Fig. 1 shows the outline of identification of related group threads for a recent bug thread. Bug threads in the bug tracking system have dependent relationships each other. The dependent relationship between bug threads indicates that one bug thread can not be fixed until the other is fixed. In Fig. 1, the dependent relationship between bug threads α, β is shown as an arrow: $\alpha \rightarrow \beta$ means that α blocks β or β depends on α, which is the case that β can not be fixed until α is not fixed. The bug threads that have dependent relationships each other, called as "dependent bug threads" organize a group of bug threads. We define the group of bug threads as "group thread".

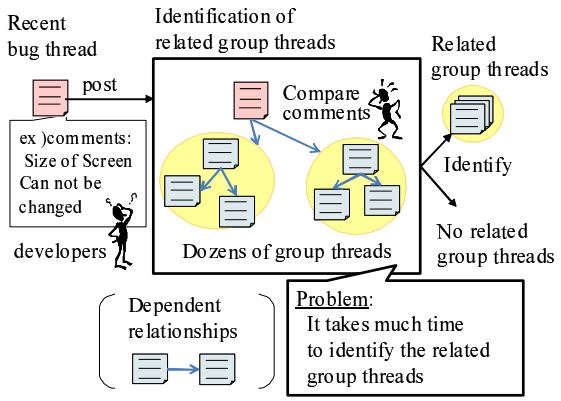

Figure 1: The outline of identification of related group threads.

When the developer receives the bug report, the developer posts the recent bug thread to the bug tracking system. Developers compare comments on the recent bug thread to comments on the existing bug threads in group threads. Finding the group thread which the recent bug thread has a dependent relationship with a bug thread in, developers address fixing the bug with referring the group thread called "related group thread". However, there are dozens of group threads in the bug tracking systems and some of existing bug threads have many comments. Developers read comments of bug threads in the group threads and identify whether the related group thread for the recent bug thread exists or not. However, because the task is a time-consuming for developers, some dependent relationships are not still identified in the bug tracking systems. The goal of this research is to identify the related group thread if the recent bug has a dependent relationship with the related group thread.

2.2 Similarity based Approach

The bugs that have the dependent relationships tend to have the common characteristics: using the same module, causing similar troubles (e.g. software crash), being under the same environment (e.g. Operating System), and so on. So, our approach is based on similarities between the recent bug thread and the group threads. The similarity based approach is shown in Fig. 2.

Because the dependent bug threads include some common words, the group thread is characterized by the common words. We call these common words "topic words". The topic words consist of the common words CW_k between the dependent bug threads on the kth dependent relationship in a group thread:

$$Topic\ Words = \bigcup_k CW_k$$

The common words CW_k are union of words W_α in the bug thread α and words W_β in the bug thread β where the kth dependent relationship exists (W_α is a group of words that appear in comments except for stop words):

$$CW_k = W_\alpha \cap W_\beta$$

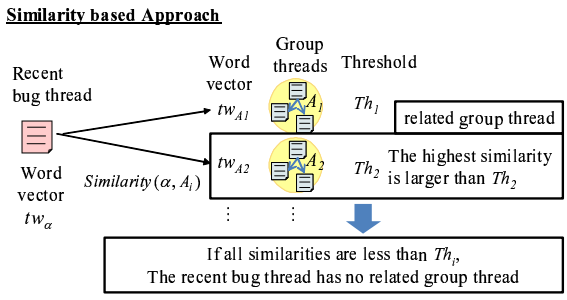

Figure 2: Similarity based approach.

Then the similarity between the recent bug thread and the group thread on the topic words is decided by Cosine Similarity (Sullivan, 2001). The value of Cosine Similarity is derived from the frequencies $tw_{\alpha,i}$, $tw_{A,i}$ of each topic word i in the recent bug thread α

and the group thread A:

$$\begin{aligned} TW_\alpha &= \{tw_{\alpha,1}, tw_{\alpha,2}, \cdots tw_{\alpha,m}\} \\ TW_A &= \{tw_{A,1}, tw_{A,2}, \cdots tw_{A,m}\} \\ Similarity(\alpha, A) &= \frac{TW_\alpha \cdot TW_A}{\|TW_\alpha\| \|TW_A\|} \end{aligned}$$

where the group thread A is a union of the bug threads in the group thread.

It is considered that $Similarity(\alpha, A)$ is high if the recent bug thread α has a dependent relationship with the group thread A. So, the basic approach is to decide that the recent bug thread has no related group thread if the similarity is small. This is judged by a threshold for similarity on each group thread. After that, the group thread that has higher similarity than any other group threads can be regarded as the related group thread.

However, the recent bug threads that have no related group threads tend to have relatively high similarities with group threads. And, most of the recent bug threads have no related group threads, which are from 80% to 90% of the recent bug threads in a bug tracking system "Bugzilla@Mozilla"[1] . So, these recent bug threads are not identified correctly by just similarity based approach.

3 IDENTIFICATION METHOD BY PEAK CHARACTERISTICS OF SIMILARITIES

3.1 Peak Characteristics of Similarities

As described in Section 2.2, the recent bug threads that have no related group threads tend to have relatively high similarities with group threads. On the other hand, the recent bug threads that have related group thread tend to have very high similarity with the related group thread but small similarities with the other group threads. The Fig. 3 shows similarities in the case of the recent bug threads that have related group threads and otherwise. On the Fig. 3, the horizontal axis indicates the group threads and the vertical axis indicates the similarities with them, this results are generated from bug threads in Bugzilla@Mozilla.

As shown in Fig. 3, we can see a few group threads that have much higher similarities than any other group threads and define these characteristics as "peak characteristics". Because peak characteristics are not appeared in the recent bugs that have no related thread group. Therefore, we propose an identification method by peak characteristics of similarities. The outline of the proposed method is shown in Fig. 4.

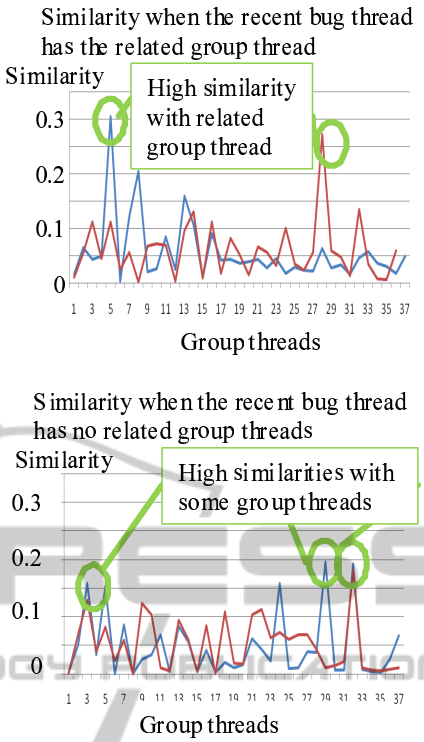

Figure 3: Peak characteristics of similarities.

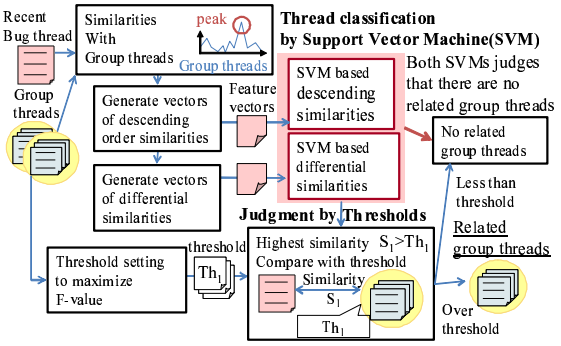

Figure 4: Outline of the proposed method.

When a developer inputs the recent bug thread to the proposed method, the proposed method derived similarities with group threads and applies Support Vector Machine (SVM) based on vectors of these similarities to classify the recent bug thread that have no related group threads. In order to derive the similarity between the recent bug thread and the group thread, the frequency of each topic word in the recent bug thread and in the bug threads in the group thread. The similarity is derived from the frequencies by the Cosine similarity as described in Section 2.2. How to

[1] https://bugzilla.mozilla.org/

generate vectors for SVM is described in Section 3.2.

Additionally, as well as similarity based approach, the group thread that has higher similarity than any other group threads and its own threshold can be regarded as the related group thread. Because there are many bug threads in the group threads, an appropriate threshold for each group thread can be decide by the bug threads. The automatic setting method of the thresholds is described in Section 3.3

3.2 Identification Method using Support Vector Machine (SVM)

The recent bug thread that has no related group threads has relatively high similarities with group threads. On the other hand, the recent bug thread that has the related group thread has very high similarities with the related group thread. We think that this difference on the peak characteristics of similarities is useful to classify the recent bug thread that has no related group threads by SVM.

In order to extract the peak characteristics, the similarities with group threads are sorted in descending order as shown Fig. 5. There are differences of similarities in high order. So, we design the two kinds of vectors for SVM: one is a vector V_1 of similarities in the top k and the other is a vector V_2 of differences between a similarity in the top l and in the top $(l+1)$:

$$\begin{aligned} V_1 &= Sort_1(Similarity(\alpha,A_1),\cdots Similarity(\alpha,A_n)) \\ V_2 &= Sort_1(Similarity(\alpha,A_1),\cdots Similarity(\alpha,A_n)) \\ &\quad -Sort_2(Similarity(\alpha,A_1),\cdots Similarity(\alpha,A_n)) \end{aligned}$$

where $Sort_r(\cdot)$ is a function to change similarities in the top $(r-1)$ to 0, and to arrange similarities in descending order.

Using both vectors $V=\{V_1,V_2\}$, SVMs in the proposed method judge that the recent bug thread has no related group thread p by the following function:

$$y_p = sign(w_p^T V - h_p)$$

where y_p indicates a result of the judgment: $y_p = -1$ means that the recent bug thread does not have dependent relationships and $y_p = 1$ means that the recent bug thread has ones in the group thread p. $sign(u)$ indicates the identification function on SVM: $sign(u) = -1$ on $u \leq 0$ and $sign(u) = 1$ on $u > 0$. w_p is a vector of weight parameters and h_p is a vector of thresholds in SVM, which are decided by training with the existing bug threads in thread groups. Because it can be decided whether existing bug threads t in the thread group p is in a thread group or not, the vector of word frequencies $a_{t,p}$ and whether the bug thread t has dependent relationships or does not

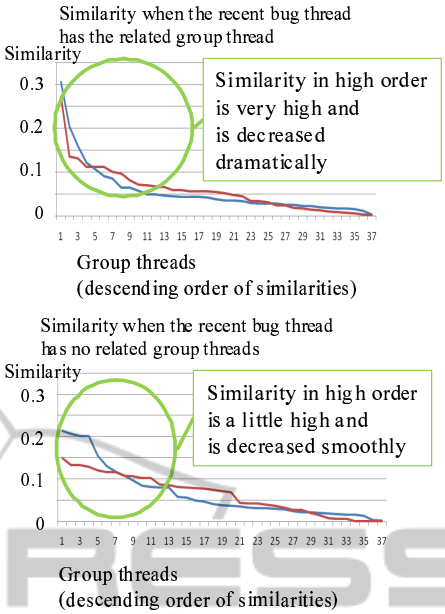

Figure 5: Similarities in descending order.

$b_{t,p} = \{-1,1\}$ can be used for the training. The training process is formulated as the following optimization problem:

$$\begin{aligned} &minimize \quad ||w_p|| \\ &subject\ to \quad b_{t,p}(a_{t,p}w_p - h_p) - 1 \geq 0 \ \ (t = 1,2\cdots) \end{aligned}$$

In order to prevent from removing the recent bug thread that has related group thread, only if SVMs based on both vectors $V=\{V_1,V_2\}$ judge that the recent bug thread has no related group thread, the recent bug thread is regarded to have no related group thread. When either of SVMs judges that the recent bug thread has related group thread, the related group thread is judged by thresholds in the next step.

3.3 Automatic Setting of Thresholds

There are many bug threads in the bug tracking system. So, the proposed method searches the thresholds to maximize $F-measure$ in inputting the bug threads by changing thresholds slightly. The $F-measure$ is decided by the following formula:

$$\begin{aligned} F-measure &= \frac{2\cdot precision\cdot recall}{precision+recall} \\ precision &= \frac{N_c}{N_c+N_w} \\ recall &= \frac{N_c}{N_c+N_u} \end{aligned}$$

where N_c is the number of correctly identified threads, N_w is the number of wrongly identified thre-

ads and N_u is the number of not-identified threads.

The detailed process of automatic setting is described in the following:

1. For initializing thresholds Th_p for the group thread p, values of all thresholds Th_p are set to 0.
2. All combinations of values of thresholds are generated with increasing values of thresholds Th_p by ΔTh_p for all p.
3. F-measure are decided with the combinations of thresholds and the threshold that makes F-measure maximize is used for the proposed method.

4 EVALUATION EXPERIMENT

4.1 Target of Experiment

We extract the bug threads from "bugzilla@mozilla" about two kinds of open source software: "firefox" and "thunderbird". The number of bug threads on firefox is 6272, the number of group threads is 37 and 137 bug threads belong to them. The number of bug threads on thunderbird is 674, the number of group threads is 62 and 185 bug threads belong to them. We compare four methods: similarity based approach, similarity based approach and SVM with similarity vector V_1, similarity based approach and SVM with similarity vector V_2 and the proposed method. In this experiment, 50 bug threads belonging to group threads and 50 bug threads isolated from group threads are randomly extracted and they are used for training data of SVM and automatic threshold setting.

4.2 Experimental Result

Fig. 6 shows the result of *recall* and *precision* described in Section 3.3. According to Fig. 6, the proposed method can improve *precision* dramatically. The method using either of SVMs decreases the recall rate. But the proposed method uses both SVMs and identify the bug thread that has no related group thread only when both SVM judges that the recent bug thread has no related group thread.

Fig. 7 shows the number of the recent bug threads that are wrongly identified as having the related group threads regardless of having no related group threads. By using SVMs based on peak characteristics of similarities, we can decrease the wrong identifications by 90% compared to similarity based approach.

5 CONCLUSIONS

We proposed an identification method of related group threads by peak characteristics of similarities. The proposed method removes recent bug threads that have no dependent relationships by Support Vector Machine based on vectors representing peak characteristics of similarities between the recent bug thread and group threads. The application result showed that the precision rate was improved by 49% and the recall rate was kept 76% on average using the proposed method.

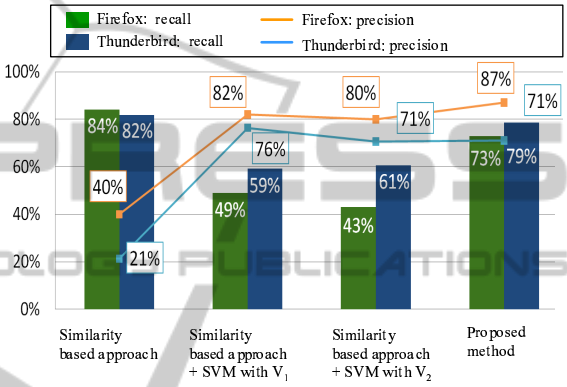

Figure 6: Recall and precision by each method.

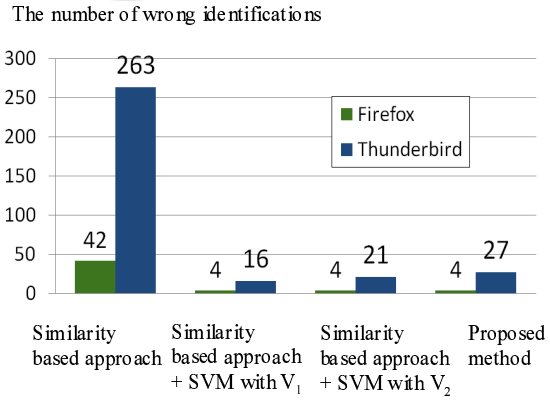

Figure 7: The number of wrong identifications by each method.

REFERENCES

Black, R. (2002). *Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software Testing*. John Wiley & Sons.

Chen, I., Li, C., and Yang, C. (2010). Mining co-location relationships among bug reports to localize fault-prone modules. *IEICE Transactions on Information and Systems*, 93(5):1154–1161.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

Gall, H., Jazayeri, M., and Krajewski, J. (2003). CVS release history data for detecting logical couplings. In *Proc. of International Workshop on Principles of Software Evolution (IWPSE2003)*, pages 12–23.

Imanara, Y., Itakura, K., Samejima, M., and Akiyoshi, M. (2011). A detection system of dependent relationships among bug report threads. In *Proc. of the 4th Japan-China Joint Symposium on Information Systems (JCIS2011)*, pages 65–68.

Matsushita, M., Sasaki, K., and Inoue, K. (2005). Coxr: open source development history search system. *Proc. of Supporting Knowledge Collaboration in Software Development*, pages 821–826.

Nagwani, N. K. and Singh, P. (2009). Weight similarity measurement model based, object oriented approach for bug databases mining to detect similar and duplicate bugs. In *Proc. of the International Conference on Advances in Computing, Communication and Control (ICAC3 '09)*, pages 202–207.

Serrano, N. and Ciordia, I. (2005). Bugzilla, itracker, and other bug trackers. *IEEE Software*, 22(2):11–13.

Souza, C. R. D., Quirk, S., Trainer, E., and Redmiles, D. F. (2007). Supporting collaborative software development through the visualization of socio-technical dependencies. In *Proc. of the 2007 international ACM conference on Supporting group work (GROUP '07*, pages 147–156.

Sullivan, D. (2001). *Document Warehousing and Text Mining: Techniques for Improving Business Operations, Marketing, and Sales*. John Wiley & Sons.

Zimmermann, T. (2009). Changes and bugs mining and predicting development activities. In *Proc. of IEEE International Conference on Software Maintenance (ICSM2009)*, pages 443–446.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS