Open Access Journal of Information Systems (OAJIS)

Available at http://is.its.ac.id/pubs/oajis/index.php

The Third Information Systems International Conference

Reuse Bug Solution Using bugFix Recommender System

Chin Yoon May, and Sa'adah Hassan

Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, 43400 UPM, Serdang, Selangor,Malaysia

Abstract

In improving the software maintenance activity, it is essential to automatically detect duplicate or similar bug reports stored in the repository, and then reusing its solution if possible. Thus, a bugFix Recommender System is proposed to assist user to get a recommendation of a solution from duplicate or similar bug reports that available in the repository. The proposed system acts like a help desk system. The system is developed based on Case-Based Reasoning approach and natural language processing techniques. This paper discusses in details the framework of the proposed system. The system has potential to provide benefits to triagers as well as end-users.

Keywords: case-based reasoning; recommender systems; natural langauge processing;

1. Introduction

Software bugs are inevitable and typically occurred due to carelessness, technical issues, poor defined specifications or misunderstanding of the problem. Bug reporting can be treated as part of software testing and maintenance activities. Triaging bug reports require developers to spent majority of their effort in this activity. Normally, the triaging scenario is start with end-user submits a report expressing the bug he/she is facing with. The triager receives and views the report and then looking for a list of potential duplicate reports stored in the repository. The triager then further investigates each of them and then provides solution on how to fix the bug to the end-user. The bug report submitted by the end-user can be a new one, or it can be similar or same with that already exist in repository. Thus, if the bug report is duplicated (i.e., have been solved), it is a waste of time and resources for finding solution for the bug that has been already fixed. Besides, some of the bug problems can be solved by the end-user themselves. Thus, supporting end-user by providing a solution promptly will also contribute towards better maintenance activity. Or at least, provides a recovery step as a temporary solution while waiting for the bug to be fixed by the developer, is also useful.

There are many research work on auto filtering duplicate bug reports as a solution to reduce manual effort of developers, increased productivity and accelerate the defect management process. However, those work mostly focus on performance in retrieving or detecting duplicate bug reports. Thus, in addition to that, our work assists users (i.e., both the triager and end-user) in providing recommendation of a solution for the bug problem. A bugFix recommender system is proposed to assist user to find

similar bug reports from repository and then reuse its solutions. It is an integration of Case Base Reasoning (CBR) approach and natural language processing (NLP) techniques. In which, CBR is a common approach used in recommender systems that resolves problem which rely heavily on the retrieval concept and similarity metric. CBR depend on historical solved problems in a repository as their main problem solving source. While, NLP techniques were applied for the text pre-processing and for retrieving similar information. The system solve user's bug problem (i.e., bug report) by retrieving and then re-using solution of the most similar bug reports stored in the repository. If necessary, triager will revise the recommended solution and store as a new bug solution into repository for future reference. Potentially, it helps to enhances software maintenance activity in solving bug problems.

This paper is divided into five sections. The following section describes the related area and efforts that motivate this work. Section 3 elaborates the proposed system and Section 4 discussed the results. Finally, we conclude our work in Section 5.

2. Background and Motivation

This section presents issues and related areas that motivate this work.

2.1. Dealing with bug reports

Bettenburg, N., *et al*. [1] stated that many software development projects offer platform to report and track bugs, and to store these bug reports in a bug tracker system. Large software projects with huge of end-users normally have huge amount of bug reports received. As reported by Anvik, J. *et al*. [2], more than 420,000 bug reports obtained in Mozilla project and up to 225, 000 bug reports received in Eclipse project. Triaging work consumes large amount of time and resources when taken a consideration on the amount of bugs reports received daily. Moreover, some of the bug reports received was reported as duplicates. If unable to detect whether the given bug report is either a duplicate bug report or not, then treating it as unique bug report and finding solution for that duplicate bug report are waste of time. Anvik, J. *et al*. [2] also suggested that an open bug repository is a necessary element to be included in most of the open source software system. This is to provide a platform to the software users to easily report and track bug reports and thus, improves the software quality.

There are studies (e.g., [3], [4], [5], [6], [7]) addressing issues regarding duplicate bug reports. However, most of the efforts work on improving the performance of retrieving duplicate bug reports. Those efforts are useful to assists bug triagers for better maintenance activity. In other aspect, aiding end-user to get solution in fixing bug is also essential. Moreover, some of the bug problem can be solved by the end-user themselves, providing that the solution is given. Or at least, provide solution as a recovery step while waiting for the bug to be fixed is also useful.

2.2. Case-Based Reasoning (CBR) in Recommender Systems

There are lots of applications in a wide range of domains created based on the CBR concept [8], [9]. CBR is an effective approach used in customer service area or helpdesk that dealing with users complains and requests (e.g. [10], [11], [12], [13], [14]). CBR is an iterated and integrated problem solving methodology that resolves a new problem by retrieving the most similar and solved case stored in the case base or repository and then re-using and adapting that retrieved case to solve the new problem [15]. In general, there are four principal steps implemented in CBR known as "4REs" which are retrieve, reuse, revise and retain, as shown in Fig.1.

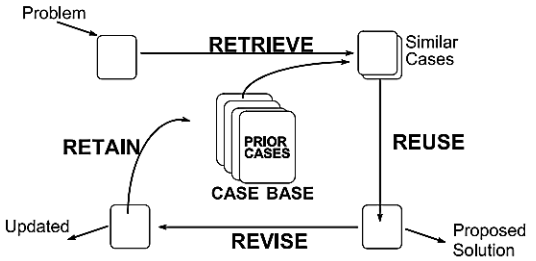

Fig. 1 CBR process cycle (adopted from [16])

In order to illustrate the cycle, given in the basic scenario where user wants to search for a solution; the user submits description of a problem (new problem). The system look through the case base for similar problems based on the past cases that suite with the user's description. The case base stores prior cases, in which each case consists of problem description and its solution. Similarity metric was applied in order to retrieve similar cases. The solution of the case with highest similarity will be reused and recommended to the user as a solution for the submitted problem. But, most of the time, revise process is necessary. This happened when user dissatisfied with the recommended solution. Thus, the solution needs to be revised until the user is satisfied. Once the user satisfies with the revised solution, the problem and revised solution are retained as a new case in the case base for future reference.

2.3. NLP Techniques in Duplicate Detection

The use of NLP to detect duplicate bug report is becoming a popular topic in software engineering research area (e.g. [3], [17], [18], [19]). There have been few efforts that have focused on adopting a more semantically-rich model, including use of natural language artifacts to support identification and detection of duplicates bug reports. Runeson, P. *et al*. [3] used textual similarity to detect bug reports which are known duplicate. NLP have been applied in their prototype in order to analyze percentage of duplicate bug reports at Sony Ericson Mobile Communications. While, Jalbert, N. *et al*. [17] presents a system that able to automatically detect duplicate bug reports to save developer time. This system combines the surface features, textual similarity metrics, and graphs clustering algorithm in order to identify duplicate bug reports. Sureka, A. *et al*. [18] has proposed an approach to help support detection of duplicate bug reports between a given user's problem (query) and existing bug reports in defect management system. They applied character N-grams-based model as low-level features for computing text similarity (based on the title and description of the user bug report). While, work by Minh, P. N. [19] has used basic NLP technique, N-grams feature and cluster shrinkage for supporting duplicate bug reports detection. The method is proposed to better improve the performance of the approaches proposed by previous researchers (i.e., [3], [17] and [18]). Table 1 summarizes and also shows the techniques that mostly used by the researches for detecting duplicate bug reports.

Table 1. Summary of the techniques used

Techniques \ Author	[3]	[17]	[18]	[19]
Pre-processing (tokenization/stemming/ stop words removal)	✓(including synonyms & spellchecking)	✓	x	✓(tokenization only)
N-grams	x	x	✓	✓
Cosine Similarity	✓	✓	x	✓
TF-IDF method	x	✓	✓	x
Clustering	x	✓	x	✓(clustering shrinkage technique)

3. Proposed System

The framework design for bugFix recommender system is based on the 4R cycle in CBR, which are retrieving, reusing, revising, and retaining. The NLP techniques were applied at the retrieval stage. The bugFix prototype was implemented by using NetBeans IDE 8.0.2, including Apache web server, MySQL database and phpMyAdmin. Fig.2 below shows the framework design for bugFix.

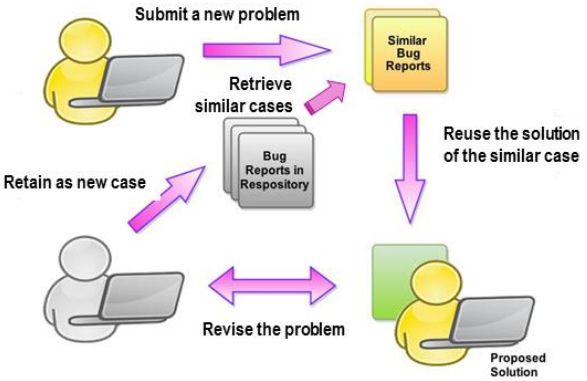

Fig. 2. The bugFix framework based on CBR process

There are five main steps involved in the process. The end-user and triager are the two roles that involved in the process. Each of the steps is discussed as below:

3.1 Problem (User bug report)

Initially, end-user submits the detail of bug problem in a new bug report form (including, title and description of the problem).

3.2 Retrieve

Based on the specification of the bug report, the system will retrieved duplicate bug reports in repository with its similarity scores. Pre-processing and cosine similarity approach were used to retrieve bug reports and compute the similarity score. There are important steps involved in pre-processing which are tokenization, stop words removal, and keywords extraction. After the pre-processing, the character level N-grams method will be applied. N-gram is applied in both, the report

title and the description to get K-bag of words. It is then treated as a vector to be used in cosine similarity calculation. Various character-level N-grams like 2-grams, 3-grams and 4-grams were experimented in both the title and description, it showed that 2-grams is the best N-grams in helping to detect duplicate bug reports. A stop word list with more than 600 words was applied in order to remove all the common words appeared in the text. This stop words list is a combination of several common stop words used by others in extracting keywords. While, cosine similarity measurement is used to compute the similarity score between the user's bug report with bug reports stored in repository. The higher the similarity score means that the corresponding reports share more weighted words, and hence, the more similar of the two reports.

$$\text{Similarity} = \cos(\theta) = \frac{\mathrm{A} \cdot \mathrm{B}}{\|\mathrm{A}\|\|B\|} = \frac{\sum_{i=1}^{n} A\mathrm{i} \times B\mathrm{i}}{\sqrt{\sum_{i=1}^{n}(A\mathrm{i})^2} \times \sqrt{\sum_{i=1}^{n}(B\mathrm{i})^2}} \quad (1)$$

Above is the cosine similarity function, where Ai represents the character-level bigrams get from bug report, and Bi represents the character-level bigrams get from existing bug reports in repository. NlpTools [20], a library for NLP written in PHP script, is a third party source code used for the calculation on similarity score (i.e., cosine similarity function).

3.3 Reuse

The solution of the most similar report is reused and recommended to solve the user's bug report. The process will end here if the user is satisfied with the proposed solution, or else, the report needs to be revised.

3.4 Revise

User satisfaction level is the key factor in this stage. If the user dissatisfied with the recommended solution (i.e., unsolved the problem), the triager needs to further investigate and revise the bug report and its solution. This step will repeat until the user satisfied with the proposed solution.

3.5 Retain

If the proposed solution solves the user's bug problem, then, the bug report with its solution will be retained in repository as a new bug solution.

4. Discussion

The bugFix recommender system provides a platform for software user to seek for advice or recommendation of a solution from the system in order to resolve the bug problem. It involved an easy and simple process, in which user just needs to enter bug's problem description and then gets recommendation of its solution. The bugFix is also a platform to assist triager in handling bug report and managing bug reports in the repository. It helps to manage bug reports easily and faster in providing solution to the user. Other significant benefits are direct recommendation of a solution to user without involving further triaging stage, and as a self-learning mechanism in which knowledge of solving bug problem will evolve as new case is retain.

However, there are improvements and future enhancements that could be considered in the future work. For example, to adopt report classification approach (e.g., k-nearest-neighbor approach or pattern-based classification) in order to get rich feature set that allow for better detection of duplicate bug reports.

If the detection or retrieval performance is improved, then the end-user will get better recommendation of a solution.

5. Conclusion

In this article, we proposed a bugFix recommender system with the aims to improve the maintenance activity as well as in providing support to the end-users. It is developed based on CBR approach and implementation of NLP techniques. It acts as a platform for software user to report and get recommendation of a solution from the system. It also assists triaging process by reducing workload of triagers in browsing through similar bug reports and in proposing solution or recommendation to the user.

Acknowledgements

This research received support from the UPM Research Grant.

References

[1] Bettenburg, N., Premraj, R., Zimmermann, T., & Kim, S. Duplicate bug reports considered harmful really? Software Maintenance. ICSM 2008. IEEE International Conference, 2008, p.337-345.

[2] Anvik, J., Hiew, L., & G. C. Murphy. Coping with an Open Bug Respository. OOPSLA workshop on Eclipse technology eXchange, 2005, p.35-39.

[3] Runeson, P., Alexandersson, M., and Nyholm, O. Detection of duplicate defect reports using natural language processing. 29th International Conference on Software Engineering (ICSE), 2007, p. 499–510.

[4] C. Sun, D. Lo, S.-C. Khoo, and J. Jiang, Towards more accurate retrieval of duplicate bug reports, in *ASE*, 2011.

[5] C. Sun, D. Lo, X. Wang, J. Jiang, and S.-C. Khoo, A discriminative model approach for accurate duplicate bug report retrieval, in ICSE, 2010.

[6] Wang, X., Zhang, L., Xie, T., Anvik, J., & Sun, J. An approach to detecting duplicate bug reports using natural language and execution information. 30th international conference on Software engineering (ICSE'08), 2008, p.461–470.

[7] Chengnian, S., David, L., Siau-Cheng, K., Jin, J.. Towards more accurate retrieval of duplicate bug reports. Automated Software Engineering (ASE), 26th IEEE/ACM International Conference, 2011.

[8] Watson, I.,. *Techniques for enterprise systems: applying case-based reasoning*. USA: Morgan Kaufmann;1997

[9] Aha, D.W. The Omnipresence of case-based reasoning in science and application. *Knowledge-based systems*, 1998;11, p.261-273.

[10] Anglano C, Montani S. Achieving self-healing in autonomic software systems: a case-based reasoning approach. *In: Proceedings of the international conference on self organization and adaptation of multiagent and grid systems*, IOS Press, Amsterdam, 2005, p. 267–281

[11] Hassan, S., McSherry, D. and Bustard, D.W.,. Autonomic self-healing and recovery informed by environment knowledge, *Artificial intelligence review*, 2006; 26 (1-2), p.89-101.

[12] McSherry, D., Hassan, S., and Bustard, D. Conversational Case-Based Reasoning in Self-healing and Recovery. In *Proceedings of the 9th European Conference on Advances in Case-Based Reasoning* (Trier, Germany, September 01 - 04, 2008). K. Althoff, R. Bergmann, M. Minor, and A. Hanft, Eds. *Lecture Notes in Artificial Intelligence*, vol. 5239. Springer-Verlag, Berlin, Heidelberg, 340-354. DOI= http://dx.doi.org/10.1007/978-3-540-85502-6_23

[13] Tsatsoulis, C., and Kashyap, R.L., Case-Based Reasoning and Learning in Manufacturing with the TOLTEC Planner, *IEEE Expert*, Vol. 23(4), August 1993, p. 1010-1025.

[14] Acorn, T., and Walden, S., SMART: support management automated reasoning technology for Compaq customer service, Proceedings of the Fourth Innovative Applications of Artificial Intelligence Conference, AAAI Press, 1992.

[15] Lorenzi, F. and Ricci, F. Case-based recommender systems: a unifying view. International conference on Intelligent Techniques for Web Personalization, ITWP'03, 2003, p. 89-113.

[16] Mantaras, R.L.D., Mcsherry, D., Bridge, D., Leake, D., Smyth, B., Craw, S., Faltings, B., Maher, M.l., Cox, M.T., Forbus, K., Keane, M., Aamodt, A. and Watson, I. Retrieval, reuse, revision and retention in case-based reasoning. *The Knowledge Engineering Review*, 2006; Vol. 20:3, p.215–240. United Kingdom: Cambridge University Press.

[17] Jalbert, N., & Weimer, W. Automated duplicate detection for bug tracking system. International Conference on Dependable Systems & Network: Alaska, IEEE, 2008, p.52-61.

[18] Sureka, A., & Jalote, P. Detecting duplicate bug report using character n-gram-based features. Asia Pacific Software Engineering Conference, IEEE, 2010, p.366-373.

[19] Minh, P. N. An approach to detecting duplicate bug reports using n-gram features and cluster shrinkage technique. *International Journal of Scientific and Research Publications*, 2014; Vol 4(5).

[20] Trilla, A., 2013, accessed on Jun 2015 from http://nlptools.atrilla.net/web/index.php**,** http://atrilla.net/index.php?article=pub