An Improved Approach for Bug Report Summarization Using Data Processing

Priya Karambe[1], Vidya Bharde[2]

[1] PG Scholar, [2]Professor, Computer Engineering,

MGMCET, Kamothe, Navi Mumbai (India)

Abstract *- Software developers access bug reports in a project's bug repository to help with a number of different tasks, including understanding how previous changes have been made and understanding multiple aspects of particular defects. A developer's interaction with existing bug reports often requires perusing a substantial amount of text. In this will investigate whether it is possible to summarize bug reports automatically so that developers can perform their tasks by consulting shorter summaries instead of entire bug reports. We investigated whether existing conversation-based automated summarizers are applicable to bug reports and found that the quality of generated summaries is similar to summaries produced for e-mail threads and other conversations [1]. So we have proposed Trained Summarizer for improved Summaries (TSIS). We trained a summarizer on a bug report corpus. This summarizer produces summaries that are statistically better than summaries produced by existing conversation-based generators. To determine if automatically produced bug report summaries can help a developer with their work. We found that summaries helped the participants to save time.*

Keywords: Original Bug reports, TSIS, *summarize bug reports, Automated summarizers, Trained summarizer.*

I. INTRODUCTION

While working software project, the developer may consult the bug repository to understand reported defects in more details, or to understand how changes were made on the project in the past. A software project's bug repository provides a rich source of information. Sometimes a developer can determine relevance based on a quick read of the title of the bug report [1], other times a developer must read the report, which can be lengthy, involving discussions amongst multiple team members and other stakeholders. So, summary requires that represents information in the report to help other developers who later access the report in future. Summarization is the creation of a shortened version of a text. Bug report summaries could help developers perform duplicate detection tasks [2] in less time with no indication of accuracy degradation. As far, existing conversation-based, e-mail threads applicable to bug reports. But Trained [6] a summarizer on a bug report corpus produces summaries better than existing [9] conversation-based generators reduce the time a developer spends getting to the right artefacts to perform their work is to provide a summary of each artefact.

II. SYSTEM MODEL

Given the evolving nature of bug repositories and the limited time available to developers, this optimal path is unlikely to occur. As a result, we investigate the automatic production of summaries to enable generation of up-to-date summaries on-demand. Bug reports vary in length. Some are short, consisting of only a few words. Others are lengthy and include conversations between many developers and users. Bug reports are automatically summarized so that developers can perform their tasks by consulting shorter summaries instead of entire bug reports. Fig. 2.1 shows Models for training and detection of TSIS (Trained Summarizer for Improved summary). So, summaries that produced can be statistically better than summaries produced by existing conversation-based generators become needed. We investigated trained summarizer; using frequency analysis optimizes performance by converting large reports into short integer vectors for faster analysis as compared to string comparisons. An automatically produced bug report summary helps a developer with their work without degrading accuracy [8].

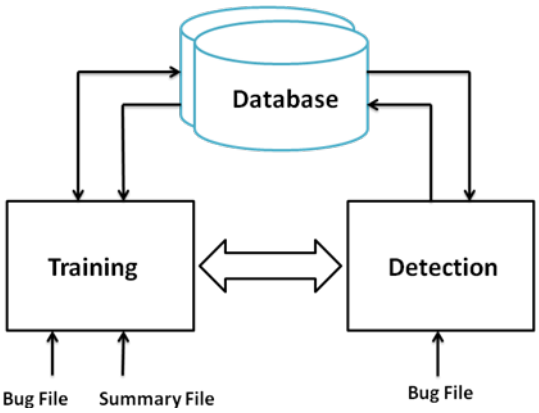

Fig. 2.1 System Model for improved Summarizer

Then an abstractive summary can be generated by identifying patterns that abstract over multiple sentences. Software developers must perform similar activities, such as understanding what bugs have been filed against a particular component of a system. However, developers must perform these activities without the benefit of summaries, leading them to either expend substantial effort

to perform the activity thoroughly or resulting in missed information. In this paper, we have investigated summarizer that automatically generates summaries of bug reports, to provide developers with the benefits others experience daily in other domains. We also found that an extractive summary generator trained on bug reports produces the best results. Generated bug report summaries could help developers perform tasks in less time with no indication of accuracy degradation, confirming that bug report summaries help software developers in performing software tasks.

III. REVIEW OF WORK

Sarah Rastkar, Gail C. Murphy, Gabriel Murray, "Automatic Summarization of Bug Reports", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL.40, NO. 4, APRIL 2014

Paper focuses on extractive techniques. Bug reports varies depending upon the system being used to store the reports, much of the information in a bug report resembles a conversation. Beyond the fixed fields with pre-defined values, such as the status field that records whether the bug is open or closed or some other state, a bug report usually involves free-form text, including a title or summary, a description and a series of time-stamped that capture a conversation between developers (and sometimes users) related to the bug [1].

- Used a summarization technique developed for generic conversations.
- bug reports summaries generated by this technique help developers future efforts may focus more
- on domain-specific features of bug reports to improve generated summaries

R. K. Taware , "Complete Bug Report Summarization using Task-Based Evaluation: A Survey", International Journal of Engineering Research and General Science Volume 2, Issue 6, October-November, 2014

Paper discussed Summarizations methods like, Key word extraction and abstraction, Document summarization, and summarizing email threads. The summarization system computes the frequency of the key words in the text, which sentences they are existing in, and where these sentences are in the text. Performance of extraction based summarization is better than abstractive approach. But, corpus for different language's stop words is not available, also more accuracy needed [2].

Paul W. Mc Burney and Collin McMillan, "Automatic Documentation Generation via Source Code Summarization of Method Context," ICPC '14, June 2–3, 2014

A documentation generator is a programming tool that creates documentation for software by analysing the statements and comments in the software's source code. While many of these tools are manual, in that they require specially formatted metadata written by programmers, new research has made inroads towards automatic generation of documentation. These approaches work by stitching together keywords from the source code into readable natural language sentences [7]. They can describe the behaviour of a Java method, but not why the method exists or what role it plays in the software. This paper, propose a technique that includes this context by analysing how the Java methods are invoked. Documentation generator is a programming tool that creates documentation for software by analysing the statements and comments in the software's source code. Many of these tools are manual, in that they require specially formatted metadata written by programmers [3].

R. Lotufo, Z.Malik, and K.Czarnecki, "Modelling the 'Hurried' Bug Report Reading Process to Summarize Bug Reports," Proc. IEEE 28thInt'l Conf. Software Maintenance (ICSM'12), pp. 430-439, 2012

As with most extractive summarization approaches, paper proposed techniques to rank sentences by relevance and select the most relevant sentences to compose the summary. For summarization approach, paper estimate the relevance of a sentence based on the probability of a reader focusing his attention on that sentence, if the reader were only allowed to focus his attention on a limited number of sentences while skimming through the bug report and still wanted to maximize his knowledge about the bug [4].

Y. Surendranadha Reddy , Dr. A.P. Siva Kumar, "An Efficient Approach for Web document summarization by Sentence Ranking ," International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 7, July 2012

This paper presents a summarization system that produces a summary for a given web document based on sentence importance measures. Most of the existing approaches for single document summarization provide the summary by using only the information present in the given document. And some approaches use the neighbour documents to get better document summary for the given document. Paper describes approach for single document summarization which uses the two sentence importance measures: Frequency of the terms in the sentence and similarity to the other sentences. The sentences are ranked according to their respective scores and the sentences with top ranks are

selected for summary. The summary is evaluated by using 'recall' evaluation measure [5].

IV. PROPOSED METHODOLOGY

System proposed a technique to automatically generate a natural language summary to understand the main goal. In a separate work, we developed Summarizer for automatic generation of natural language summaries. To the best of our knowledge, the work presented in this paper, is the attempt to generate meaningful summaries of bug reports and to evaluate the usefulness of the generated summaries in the context of a software task. As another example, as part of the annotation process, we also gathered information about the intent of sentences, such As whether a sentence indicated a 'problem,' 'suggestion,' 'fix,' 'agreement,' or 'disagreement', this information can be used to train classifiers to map sentences of a bug report to appropriate labels. Fig. 4.1 describes methodology of TSIS.

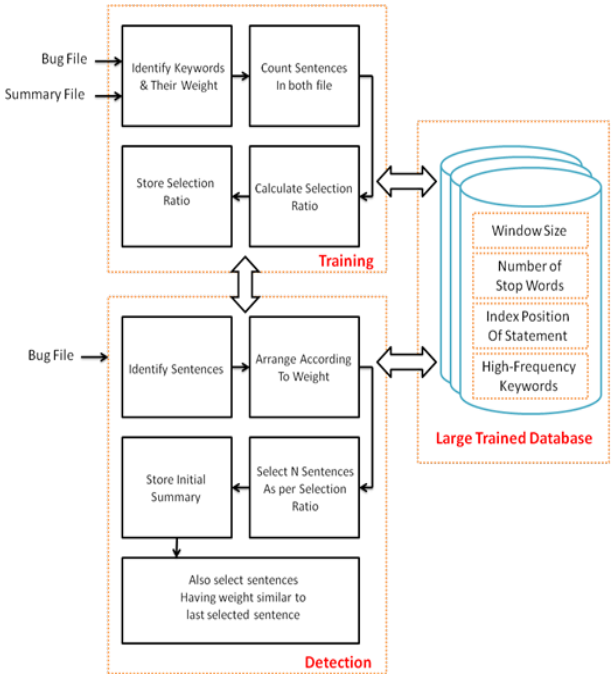

Fig. 4.1 TSIS System.

Overall process of TSIS consists of mainly two phases:

A. Training
B. Detection

A. Training:

In this section database will train to work with bug report to summarize them. Here original bug report and sample summary file as input will provided to summarizer. Then number of sentences in original bug report (N1) as well as that of summary file (N2) will be calculated. Selection Ratio (R) of these two parameters calculated as following:

$$R= N2/N1$$

Where,

R= Selection Ratio

N1= number of sentences in original bug report

N2= number of sentences in Summary File

Selection ratio will be calculate and stored for future reference in Detection process. Fig. 4.2 gathers indexing information of sentences as per their weight as a part of training.

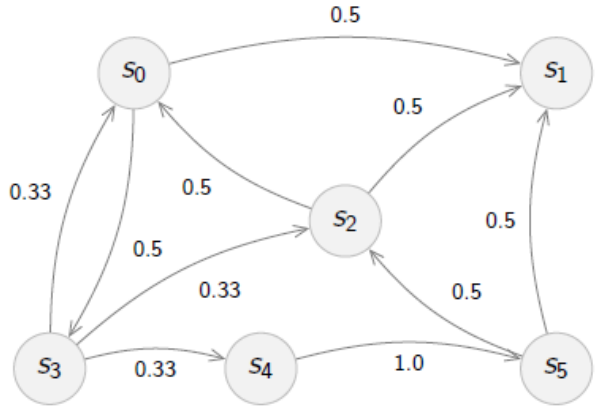

Fig. 4.2 Indexing sentences as per their weight

B. Detection:

Here bug file which user needs to summarize will provide as input. Number of sentences of that file (N) will be calculated and stored in database. Sentences with its index value will be store in database. Weight of each sentence as per keywords will be calculated. Sentences will be arranged in descending order of their weight and then count for sentences to be select (X) will be calculated as per following:

$$X= R*N$$

Where,

X=number of sentences to be select

R=selection ratio (calculated from training process)

N=number of sentences of file need to be summarize

Top X sentences based on weight of keywords similar to trained keywords will be selected. Using index values stored in database these sentences will be arranged and will be stored in database as generated summary.

V. EXPERIMENTAL RESULTS

We investigated whether existing conversation-based automated summarizers are applicable to bug reports and

found that the generated summaries are similar to summaries produced for e-mail threads and other conversations. TSIS (Trained Summarizer for Improved Summaries) using frequency analysis optimizes performance by converting large reports into short integer vectors for faster analysis as compared to string comparisons using regular expressions. Fig. 5 shows Comparison bases on accuracy for originals and summaries.

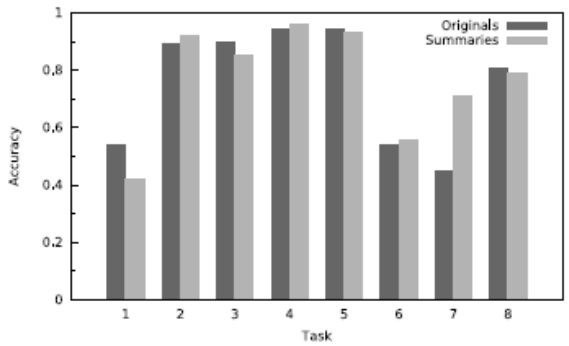

Fig.5.1 Comparison of aspects based on accuracy.

This summarizer produces improved summaries that are statistically better than summaries produced by existing conversation-based generators. An automatically produced bug report summary helps a developer with their work within proved accuracy.

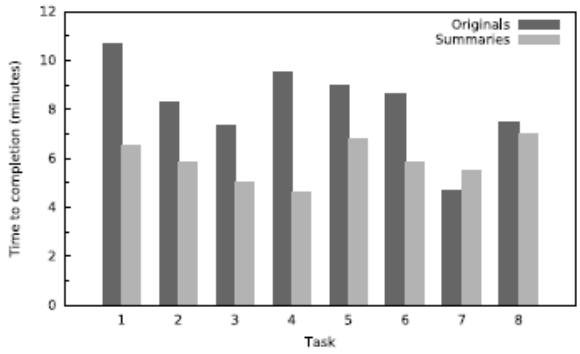

Fig. 5.2 Comparison of aspects based on Time complexity.

We found that summaries helped participants save time. Fig 5.2 shows time Complexity for originals and Summaries.

VI. CONCLUSION

Software bug reports are important project artefacts that evolve throughout the life of a software project. A developer's interaction with existing bug reports often requires perusing a substantial amount of text. Summarize bug reports automatically so that developers can perform their tasks by consulting shorter summaries instead of entire bug reports. This summarizer produces summaries that are statistically better than summaries produced by existing conversation-based generators .Summaries helped the study participants save time, that there was no evidence that accuracy degraded. Summaries were helpful in the context of duplicate detection tasks. In this section, discuss possible ways to improve the summaries produced and to evaluate their usefulness. Summarization technique developed for generic conversations to automatically summarize bug reports. Bug reports summaries generated by this technique help developers by improving accuracy of Summaries and reducing the time complexity of the process.

VII. FUTURE SCOPES

Here the system using frequency analysis optimizes performance by converting large reports into short integer vectors so in future analysis become faster as compared to string comparisons using regular expressions. As it uses Weight based analysis in terms of location of keywords improves accuracy of detection leading to low false-rate as compared to uniform text summarization.

REFERENCES

[1] Sarah Rastkar, Gail C. Murphy, Gabriel Murray, "Automatic Summarization of Bug Reports", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL.40, NO. 4, APRIL 2014.

[2] R. K.Taware , "Complete Bug Report Summarization using Task-Based Evaluation: A Survey ", International Journal of Engineering Research and General Science Volume 2, Issue 6, October-November, 2014 .

[3] Paul W. Mc Burney and Collin McMillan, "Automatic Documentation Generation via Source Code Summarization of Method Context," ICPC '14, June 2–3, 2014.

[4] R. Lotufo, Z.Malik, and K.Czarnecki, "Modelling the 'Hurried' Bug Report Reading Process to Summarize Bug Reports," Proc. IEEE 28thInt'l Conf. Software Maintenance (ICSM'12), pp. 430-439, 2012.

[5] Y. Surendranadha Reddy , Dr. A.P. Siva Kumar, "An Efficient Approach for Web document summarization by Sentence Ranking ," International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 7, July 2012.

[6] Jyoti Gautam1, Ela Kumar, "An Integrated and Improved Approach to Terms Weighting in Text Classification ," IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 1, No 1, January 2013.

[7] L. Moreno, J. Aponte, G. Sridhara, A Marcus, L. Pollock, and K. Vijay-Shanker, "Automatic Generation of Natural Language Summaries for Java Classes," Proc. IEEE 21st Int'l Conf. Program Comprehension (ICPC '13), 2013.

[8] Chengnian Sun, David Lo, Xiaoyin Wang, Jing Jiang, Siau-Cheng Khoo, "A Discriminative Model Approach for Accurate Duplicate Bug Report Retrieval, ICSE'10, May 2–8, 2010.

[9] Shamima Yeasmin Chanchal K. Roy Kevin A. Schneider, "Interactive Visualization of Bug Reports using Topic Evolution and Extractive Summaries.

[10] Rafael Lotufo, Zeeshan Malik, Krzysztof Czarnecki, "Modelling The 'Hurried' Bug Report Reading Process To Summarize Bug Reports", 978-1-4673-2312-3/12/$31.00 c 2012 IEEE.