Linking Stack Overflow to Issue Tracker for Issue Resolution

Tao Wang[†], Gang Yin[†], Huaimin Wang[†], Cheng Yang[†], Peng Zou[†,‡]
[†]Key Lab. of Parallel and Distributed Computing
Colledge of Computer, National University of Defence Technology, ChangSha, Hunan, China
[‡]Academy of Equipment Command and Technology, Beijing, 100000, China
{taowang2005, jack.nudt,hmwang,zpeng}@nudt.edu.cn, delpiero710@126.com,

ABSTRACT

Issue resolution is a central task for software development. The efficiency of issue fixing largely relies on the issue report quality. Stack Overflow, which hosts rich and real-time posts about programming-specific problems, is a valuable external source of knowledge for issue resolution. In this paper, we present CrossLink, an analysis framework that automatically introduce related posts in Stack Overflow to issues in Android Issue Tracker. This helps developers to leverage the abundant and professional knowledge from massive programmers in Stack Overflow for issue resolution. CrossLink explores the semantic similarities as well as the temporal associations between the two types of repositories to recommend Stack Overflow posts to Android issues. The internal links in Stack Overflow are also employed to improve the linking accuracy. The experiments prove the effectiveness of CrossLink with precision of 62.51% for top-10 recommendations, which is significantly higher than the state-of-art method.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management

General Terms

Documentation

Keywords

Issue Resolution, Android Issue Tracker, Stack Overflow

1. INTRODUCTION

Issue resolution is a central task in software development. It is estimated to take as much as 50% of the efforts among all development tasks[3]. To resolve an issue, developers often resort to bug report for collecting crucial information such as steps to reproduce, stack traces and testing cases. The time required for locating and fixing an issue largely relies on the quality of the issue reports submitted. However, the bug reports in issue tracking systems often provide inadequate or even incorrect information [2], especially that a large proportion of the bugs are reported by end-users who may have little programming experience.

Promoting the quality of the bug report is critical for improving the issue fixing efficiency. Many researches have been done, such as leveraging the external information to enrich the bug descriptions [4, 2]. Besides, there are works on revising the current bug tracking systems in mechanisms. Lotufo et. al studied how to improve the traditional bug tracking system by introducing the game mechanism which are successfully adopt in Stack Overflow into bug tracking system[6].

Stack Overflow(SO), which was released at 2008, has emerged as the dominant Q&A site in software programming fields. Focusing on programming-specific issues, it has attracted more than 2,000,000 professional developers who contributed more than 15,600,000 posts on programming questions until Sept. 2013. The voting and gaming mechanisms adopted by SO ensure the quality of the questions and answers in it. Among the posts in Stack Overflow a large proportion of them are related to software issues. The rich and high-quality contents in Stack Overflow make it a valuable external source of knowledge for issue resolution. Bacchelli et. al [1] designed a plugin to integrate Stack Overflow with Eclipse to solve developers' questions when programming. Different from this work, we mainly focus on employing Stack Overflow on issue resolution.

In this paper, we present *CrossLink*, an analysis framework that recommend related posts in SO to Android issues to provide rich and professional information, which helps issue resolution. *CrossLink* firstly analyses the internal links in SO and make posts into clusters. Then it explores the semantic correlations and temporal associations between issues and post clusters. Highly related clusters will be linked to the correlated issues. We conduct extensive experiments on Android Issue Tracker and SO. The results proves the effectiveness of our approach. We describe the design of *CrossLink* and discuss the experiment results in this paper.

2. MOTIVATION

As one of the most visible Q&A sites for programmers and software engineers, Stack Overflow has captured significant mindshare among software developers. Many projects such as jQuery even take it as an official support channel. The Stack Overflow can benefit much for issue resolution from

INTERNETWARE'14, November 17, 2014, Hong Kong, China
ACM 978-1-4503-3303-0/14/11
http://dx.doi.org/10.1145/2677832.2677839

several perspectives.

Firstly, the extensive participants in SO provide rich information for issue fixing. Taking Android project as an example. In Stack Overflow, there are more than 370,000 developers gathered in SO and discuss about Android-related programming problems. What they discuss covers about 87% of Android platform API classes [7]. These users provide information and solutions around an issue from different perspectives, and thus result in more precise issue reports.

Secondly, the high-quality contents in SO provide more precise information for issue fixing. SO designs a voting and reputation mechanism which motivates users to provide high-quality questions and answers. It also leverages the power of crowds to filter out poor contents. In addition, most of the participants in SO are those who have certain programming experience. Thus, the information accumulated in SO should be professional and therefore useful for issue fixing.

Thirdly, the quick response in SO can motivate bug reporters to participate in and provide more additional information. In Android Issue Tracker, the median time of first responses for reported issue is about 31 days. Such a long delay of response will dampen the reporters' willing to contribute. While in SO, it achieves almost real-time responses of about 13 minutes in median for first answers for Android-related questions. Introducing the quick responses in SO to issues will motivate reporters to provide more information and accelerate issues resolving.

Overall, linking related SO posts to issues will introduce the external crowds in SO to Issue Tracker for issue fixing, and thus improve the efficiency of issue resolution.

3. APPROACH

CrossLink aims to build links between Stack Overflow posts and issues so as to leverage the external knowledge for issue resolution. The overview of our approach and detailed presented in this sections.

3.1 Overview of CrossLink

CrossLink employs the semantic similarity and temporal associations between issue reports and SO posts to reveal the potential links between them. The internal links in SO posts are also explored to promote the linking accuracy. CrossLink mainly consists of three stages, including *Data Extraction*, *Similarity Model Training* and *Links Recommendation*. The framework of CrossLink is presented in Figure 1

In the *Data Extraction* stage, three types of information are extracted from the issues and SO posts. The first is the text contents including the titles and tags of issues/posts, issue/post contents and corresponding answers. For a given issue in Android Issue Tracker or a question post in SO, the issue/post and the corresponding answers are organized into issue thread or post thread. We treat the text contents of an issue/post thread as a whole to represent the issue/post. The second is the internal links in SO posts which are added by questioners or answerers to refer to other related posts. The third is the temporal attributes, including the reporting and responding time of issue and post threads.

In the *Similarity Model Training* stage, the internal links are employed to cluster posts in SO. The extracted thread texts are used to build the Semantic Similarity Model, and the temporal attributes are employed to build the Temporal-Locality Model. At building the semantic and temporal models, the post clusters instead of each single post thread are used. Then these models are integrated as the Synthesized Similarity Model.

In the *Links Recommendation* stage, the thread texts and temporal attributes of a query issue are used as the input of the Synthesized Similarity Model. Based on the synthesized similarity between the issue and all the post clusters, a list of post clusters will be returned as potential associations. The details are described in the following sections.

3.2 Internal Links and Semantic Similarity

To mine the potential associations between issues and posts, the semantic similarity is the basic measurement. In the previous work [4], Correa et al. calculated the semantic similarities between the issue and every single post thread based on the their titles and tags. Differently, we firstly cluster the closely related posts which focus on similar issues. Such post clusters can cover different aspects of the issue and give more comprehensive information about it. Then for a given issue, we consider the similarity between it and the post cluster instead of single post thread.

The internal links in SO posts and the semantic similarities between them are leveraged to cluster the posts. Stack Overflow contains quite a large number of internal links which are added by participants to refer to other related posts. These links connect posts around similar questions and form inter-connected knowledge graph [5]. However, among these links a proportion of them may transfer to other topics which are not so coherent to the core question. To find the closely related posts, we focus on two metrics to divide large connected graph into small isolated clusters. The first is semantic similarity and the second is the diameter of the connected graphs. For two connected posts, we can determine that the questions two posts discuss not coherent if the semantic similarity is under a certain threshold. For such links we can delete them. The semantic similarity is only used to measure the coherency between two directly connected posts. For a given path in the graph, even the semantic similarities between the pairs of directly connected posts are all above the similarity threshold, the focus of two posts with large distance can differ a lot. This is similar to the spread of influence in social network. We use the diameter as a metric to depart the large connected graph into ones with diameter smaller than a given threshold.

We firstly construct a weighted undirected graph, in which the nodes are the posts and the edges are the existing links. For a post, if no links in and out, it will be an isolated note in the graph. The weight of an edge is the semantic similarity between two connected posts. We delete these edges with weights less than the predefined threshold. This departs the large connected graph into smaller closely coupled ones. Then the remain large graphs will be departed by deleting the links with smallest weight until the diameters of all connected graphs are less than the given threshold. By several iterations of experiments and manual analysis, we set the similarity threshold as 0.3 and the diameter threshold as 4 which can cluster these posts with best internal cohesion.

We view each isolated graph as a post cluster. For a given issue and a post cluster, we firstly calculate the semantic similarity between the issue and each post in the cluster. Then the largest value of similarity is selected to represent the semantic similarity between the issue and the post cluster. This is a simple yet effective strategy. To calculate the

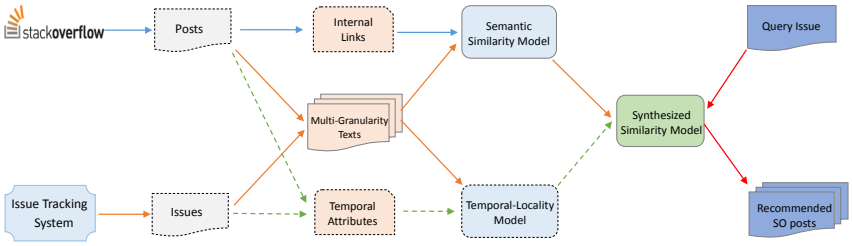

Figure 1: The framework of CrossLink

semantic similarity between a pair of issue and post documents, many techniques can be employed. We adopt Cosine Similarity in *CrossLink* for its simpleness and effectiveness. Vector Space Model and TFIDF are used to represent the issues and posts, which transforms documents into weighted vectors.

3.3 Temporal Locality

Android has attracted hundreds of thousands of developers to develop Apps for it and even more end-users (There are 58,112 participants in Android Issue Tracker and 171,152 ones in Android-related posts in SO). If there are defects in the platform which result in problems, they will be perceived quickly by the large number of users. Issue reports and programming questions will arise in Android Issue Tracker and Stack Overflow. As rooted in the same defect in Android system, the time the issue report arise in Issue Tracker and that the questions posted in SO should be temporally related, which we call temporal-locality.

In addition to the reporting-time locality, there is another kind of temporal locality over responding time. There are many shared participants who are active in both communities. For a given participant, his activity (like questioning, answering and commenting) over the same topic in both communities should be temporally close. Based on this intuition, if a post and an issue are similar in semantic and meanwhile the responding time are close, we can assume that the responses may come from the same developer, and thus raise the probability that the post and issue are correlated. This is another type of temporal locality we can utilize.

To take advantage of such temporal locality, we calculate the temporal similarity between issue and post cluster according to Equation 1.

$$Sim_{temporal}(i,j) = \frac{1}{\min(time_{intervals}(i,j)) + 1} \quad (1)$$

In Equation 1, $time_{intervals}(i,j)$ is a set of time intervals between the issue and posts cluster i and j, including the reporting and responding time intervals with unit of "day". For a given pair of issue and post thread, we adopt the minimum interval as the final result. To be consistent with the value of semantic similarity, we uniform the temporal similarity as its reciprocal.

3.4 Synthesized Similarity

To identify related issue and post threads, we combine the semantic and temporal factors together and propose the synthesized similarity model. For those issues and post clusters which have similar level of semantic similarity, if the reporting time or responding time is close, the probability that they are correlated will be higher than that with large time interval. We calculate the synthesized similarity with Equation 2.

$$Sim(i,j) = \alpha \times Sim_{content}(i,j) + \beta \times Sim_{temporal}(i,j) \quad (2)$$

In Equation 2, $Sim_{content}(i,j)$ and $Sim_{temporal}(i,j)$ stand for the similarity based on $issue_i$ and $cluster_j$(or $cluster_i$ and $issue_j$) contents and reporting/responding time respectively. α and β are the weight factors whose sum should be equal to one. In this paper we set both α and β as 0.5 for which get best performance.

When do recommendation, we take the strongly-connected posts as a whole, and the post here is a cluster of connected posts instead of a single one. After the synthesized similarities are calculated, the potential linked post clusters are recommended according to the synthesized similarities.

4. EVALUATION AND IMPLEMENTATION

In this section, we first introduce the experiment dataset and the evaluation metrics, and then we present the experiment results in detail.

4.1 Experiment Datasets

We conduct experiments on issues in Android Issue Tracker and Android-related posts in SO. We collect Android issues data from Nov. 2007 to Sept. 2013 which contains 30,572 issue threads. For Stack Overflow, we use the publicly available data which contains 387,422 post threads from Jul. 2008 to Sept. 2013(`https://archive.org/details/stackexchange`).

In these two datasets,there are 653 issue threads containing links to posts and 2753 post threads containing links to issues (totally 2315 unique issues are linked with posts). These links are added by the developers manually. We use these issues as the query issues and the contained links as the ground-truth to test the tool *CrossLink*. The details of the dataset is presented in Table 1.

Table 1: Statistics of Experiment Datasets

Community	#Total threads	#Threads with links	Time period
Android Issue Tracker	30,572	653	2007-11-12 ~ 2013-09-02
Stack Overflow (Android)	387,422	2,753	2008-07-31 ~ 2013-09-06

4.2 Evaluation Metrics

We measure the performance of CrossLink with precision and Mean Reciprocal Rank(MRR) of *Top-K*, which represent how many links are correctly predicted and the average rank of correct links in Top-K recommendations. Precision

can be calculated as Equation 3.

$$P@K = \frac{1}{|Q|}\sum_{i=1}^{|Q|} result_i \quad (3)$$

where Q means the set of query samples. For $result_i$, if the correct answer appear in the top-k results, then its value is 1, elsewhere 0. and MRR is calculated as Equation 4.

$$MRR@K = \frac{1}{|Q|}\sum_{i=1}^{|Q|} \frac{1}{Rank_{K_i}} \quad (4)$$

where Q means the same as in Equation 4, and $Rank_{K_i}$ is the rank of the correct answer in the top-k recommendations.

4.3 Experiments Results

We compare the performances of different approaches which based on the *S*emantic *S*imilarity(SS), combination of *S*emantic similarity and *T*emporal *S*imilarity(STS), and the combination of *S*emantic, *T*emporal and internal *L*inks(STL) respectively for recommendation. Table 2 presents the detailed results for recommending top 10 candidates.

Table 2: Recommendation performance for different approaches

Approach	Precision	MRR
SS	0.4942	0.2926
STS	0.5620	0.3371
STL	0.6251	0.3704

Overall, STL approach performs best with average precision of 62.51% and MRR of 0.3704. Compared with the previous work [4] which only using the titles and tags, we can see that our approach is significantly higher than the results presented in the paper which is only 33.16%.

To be more precise, we record the precision for recommending different number of candidates, ranging from 5 to 100. Figure 2 presents the detailed results.

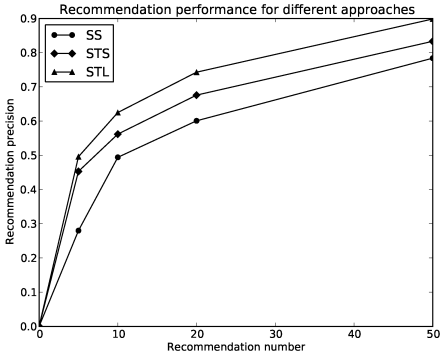

Figure 2: Recommendation precision for different approaches

As can be seen from the figure, as the recommending number grows, the hit precision will increase as well. When recommending top 5, the precision for STL is 49.59%. While the precision reach 89.85% for recommending to 50. This means that, for about 90% of the testing issues, the truly correlated posts are corrected returned in the top 50 candidates. Comparing the three approaches, we can see that STL always performs better that the other two approaches no matter how many candidates are recommended.

One thing need to note is that, we only view these existing links in issues as the correct answers, and the others even they are duplicate to the standard answers will be viewed as incorrect. For some issues, although the standard answers are not contained in the top-k recommendations, the duplicate or related ones of the correct answer may be returned in the top-K recommendations which are actually correct. Although recommending clusters instead of single post will alleviate this problem, the actual precision should be higher than the results presented here.

The CrossLink do recommendation relying on the wealth data in Stack Overflow. Currently we use the offline SO data instead of directly connecting to Stack Overflow. The whole SO dataset is about 2GB, which makes it not feasible to provide a standalone tool to download. In the future, we may further optimize the tool and connect it with Stack Overflow directly which will be able keep pace with the up-to-date data in SO.

5. CONCLUSION

In this paper, we explore the potential benefits for linking Stack Overflow posts to Android issue, and propose a novel approach *CrossLink* to automatic this process for issue resolution. It employs the semantic similarities, temporal localities and internal links to reveal the associations between issues and posts. The experiments show that it can achieve a precision of about 62.51% which outperforms the previous work for Top 10 recommendations. Although it is only validated on Android Issue Tracker and Stack Overflow in this paper, CrossLink can be easily revised to be applied to other open source projects.

6. ACKNOWLEDGMENT

This research is supported by the National High Technology Research and Development Program of China under Grant No. 2012AA011201, and the NSFC under grant No. 61432020 and 61472430.

7. REFERENCES

[1] A. Bacchelli, L. Ponzanelli, and M. Lanza. Harnessing stack overflow for the ide. In *Recommendation Systems for Software Engineering (RSSE), 2012 Third International Workshop on*, pages 26–30. IEEE, 2012.

[2] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, and T. Zimmermann. What makes a good bug report? In *FSE*, 2008.

[3] B. Boehm and V. R. Basili. Software defect reduction top 10 list. *Foundations of empirical software engineering: the legacy of Victor R. Basili*, 426, 2005.

[4] D. Correa and A. Sureka. Integrating issue tracking systems with community-based question and answering websites. In *ASWEC*, 2013.

[5] C. Gómez, B. Cleary, and L. Singer. A study of innovation diffusion through link sharing on stack overflow. In *MSR*, 2013.

[6] R. Lotufo, L. Passos, and K. Czarnecki. Towards improving bug tracking systems with game mechanisms. In *MSR*, 2012.

[7] C. Parnin, C. Treude, L. Grammel, and M.-A. Storey. Crowd documentation: Exploring the coverage and the dynamics of api discussions on stack overflow. *Georgia Institute of Technology, Tech. Rep*, 2012.