Indian Journal of Science and Technology, Vol 9(21), DOI: 10.17485/ijst/2016/v9i21/89198, June 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

An Empirical Analysis on Reducing Open Source Software Development Tasks using Stack Overflow

Tirath Prasad Sahu[1]*, Naresh Kumar Nagwani[2] and Shrish Verma[3]

[1]Department of Information Technology, National Institute of Technology Raipur, Raipur - 492010, Chhattisgarh, India; tirsahu.it@nitrr.ac.in
[2]Department of Computer Science and Engineering, National Institute of Technology Raipur, Raipur - 492010, Chhattisgarh, India; nknagwani.cs@nitrr.ac.in
[3]Department of Electronics and Telecommunication, National Institute of Technology Raipur, Raipur - 492010, Chhattisgarh, India; shrishverma@nitrr.ac.in

Abstract

Objectives: The cross repository analysis between Open Source Software (OSS) and Community Question Answering (CQA) site is presented in order to speed the development process of OSS. **Methods/Analysis:** The OSS development is becoming popular nowadays due to fact that the source codes, the developer specifications and bug lists are made available online to the public. Anyone can contribute to the development of software by referring these files. Similarly, Stack Overflow is an interactive CQA site that caters programming related questions with their answers online and turned into repositories of software engineering knowledge. In order to track the correlation of such sites with software development tasks, we employ the two repositories to find the semantic similarity between bugs and Question and Answer (Q&A) posts posted on OSS projects and Stack Overflow respectively. The semantic similarity is analyzed by integrating the contents of the repositories based on text mining approach. The relationship between a bug and Q&A post is established through the semantic similarity and metadata features. **Findings:** The statistics of our analysis is presented for five OSS projects in terms of number of bugs and average bug fix time. The statistical result shows that the bug fix time can be reduced by posting the bugs into Stack Overflow. **Application/Improvement:** The presented approach can be utilized to find the similar Q&A posts for reported OSS bug and helps developers of OSS projects to resolve the bugs quickly by leveraging programming skills of users' in the form of Q&A posts.

Keywords: Open Source Software, Community Question Answering, Stack Overflow, Cross Repository Analysis, Bug Tracking System, Bug Fixing

1. Introduction

OSS is gaining popularity by keeping their development files online so that anyone can contribute in the development tasks. Bug fixing is an important task so as to improve the quality for competitive software products, to maintain higher customer satisfaction, to keep effort, schedule and cost of development on track[1]. Therefore, the development teams are always desired to fix the bugs reported by the users' quickly. The online contribution by other users' or community can be one solution to improve the bug fixing time[2-5].

The online community such as CQA sites have become an important source of knowledge over the years in the form of Q&A[6]. The CQA sites can be classified in two classes: 1. General Q&A sites such as Quora and Yahoo! Answers, 2. Domain specific Q&A sites such as Stack Overflow and Ask Ubuntu, which have catered to programming related questions and their answers and turned into repositories of software engineering knowledge. Stack Overflow is an interactive CQA site for software development knowledge by hosting communities of millions of users (developers)[7,8]. The voting and gamification ensures the quality of the contents in Stack Overflow[9].

*Author for correspondence

Stack Overflow is consisting of more than 29 million posts by more than 4 million professional developers since its inception in 2008 until February 2016. The large proportion of the posts is related to software bugs of the projects. Therefore, it can be seen as the important source of software knowledge that can help to fix the reported bugs in bug tracking system. The developer can post query to Stack Overflow related to the bug in the form of question and will get the solution to resolve that bug in the form of answer. The huge and unstructured contents of the repositories are posing several challenges for researchers to establish the relationship between them for analysis[2-5,7,8].

CQA sites can be utilized effectively for making the software development tasks efficient. The most closely related work is CrossLink: a linkage of Stack Overflow to issue tracker[2], which explores the semantic similarity and temporal association between them. They have concentrated to improve the linking accuracy using text mining approach for issue resolution. Similarly[3], also links the issue tracking systems of android and Chromium by looking the links to Stack Overflow, Wikipedia and other external websites in the bug description. They use the metric "mean time to repair" (which we refer as the average fixing time in this study) to judge whether the presence of these links in the bug description leads to faster repairing. The linking of Stack Overflow to GitHub (the largest coding repository) is presented by the authors[10] to match questions in Stack Overflow with codes which are forked in GitHub. The aim is to find the contribution of Stack Overflow into the development of GitHub projects by classifying whether the Q&A posts are related to commit event of a project or not[11], presents a study of Stack Overflow, the reasons of its success and dynamics, which gives a proper description of working of Stack Overflow[12]. The literature gives us the basic knowledge of how Stack Overflow and bug tracking systems can be linked accurately based on the contents. In addition to linking, we present the impact of Stack Overflow into the development of OSS projects irrespective of presence of links in bug description using average fix time of posted and non-posted bugs into Stack Overflow.

In this paper, we present cross repository analysis by integrating the contents based on text mining approach in order to fix the bugs as early as possible. First, the dataset is extracted from repositories by matching projects name with tags of the posts and temporal features. Then, text mining approach is applied to find the semantic similarity[13] between bugs and posts. The high similarity signifies the posts related to the bugs. The empirical analysis has been carried out by comparing the average bug fix time of posted and non-posted bugs in Stack Overflow.

The rest of the paper is organized as follows. Section 2 describes the proposed framework with dataset description. The empirical analysis is presented with statistics of results in Section 3. We conclude our work with their future scope in Section 4. Section 5 consist acknowledgement. Finally, the references to the presented study are listed in Section 6.

2. Proposed Framework

In this section, a framework is proposed to analyze the contribution of Stack Overflow into the development of OSS projects based on bug fix time. For this, text mining approach is applied on real data extracted from bug tracking system of OSS projects and CQA site Stack Overflow.

2.1 Dataset Description

The empirical analysis is based on two datasets extracted from two online repositories i.e. Bug Tracking System of OSS projects and Stack Overflow CQA site.

2.1.1 Bug Tracking System

A bug tracking system is a software application to make bug repository that keeps track of reported software bugs found in the development of software projects. We collect the fixed bugs of five OSS projects namely Open Office, Mozilla, Apache, Ruby and Python, reported in the year 2014 for our experimental analysis. The dataset extracted from the bug tracking system of the OSS projects con-

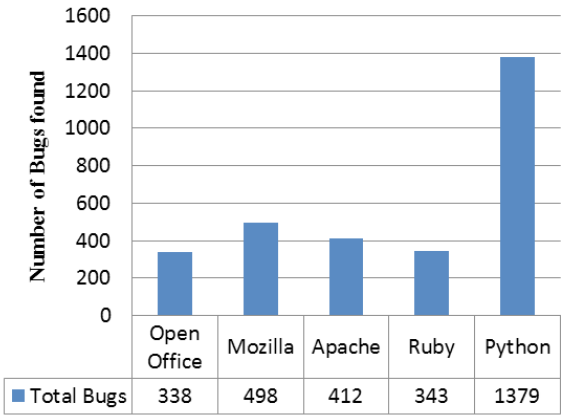

Figure 1. Bugs from OSS projects.

tains data consisting of the bug id, summary, description, status, opening and closing date etc.[14] Figure 1 shows the statistics of extracted bugs from OSS projects.

2.1.2 Stack Exchange Data Explorer

The Stack Overflow dataset is publicly available through data dump and stack exchange data explorer. Stack exchange data explorer provides the way to extract the data by firing query online. We use data explorer to extract accepted Q&A posts of year 2014 based on post tags related to OSS project name of bug repository. The attributes of Stack Overflow posts are: postid, userid, title, body, tags, accepted answerid and creation date etc. Figure 2 shows the statistics of extracted Q&A posts from Stack Overflow.

2.2 Text Mining Approach

Text mining approach is used to uncover the potential information and association in text corpus. Textual features are identified that represents the intrinsic quality metrics related to tangible features of the text documents. Textual features can be automatically generated using a sequence of procedures and are usually done using computer programs that includes text pre-processing, textual similarity, clustering etc.

2.2.1 Text Pre-Processing

Pre-processing of the document is the preparation of the dataset before applying any operation on it. Text pre-processing is the process of identifying and extracting interesting and non-trivial information from unstructured text documents such as CQA posts. Our approach of pre-processing of the Q&A posts includes Tokenization, Stopping, Stemming and code snippet filtering.

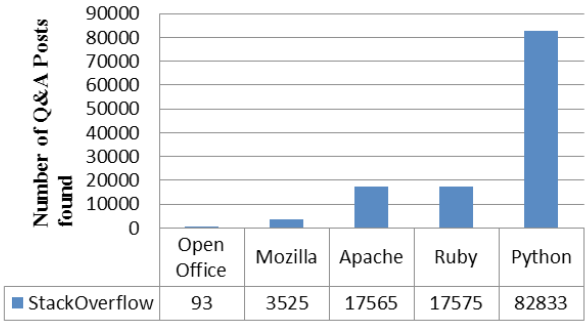

Figure 2. Q&A posts from Stack Overflow.

- **Tokenization:** It is a process of breaking a stream of text into words, phrases, symbols, or other meaningful elements called tokens. The aim of the tokenization is the exploration of the words in a sentence.
- **Stopping:** It is a process of filtering out most common words called stop words from text documents. The removal of stop words is done through the list of stop words. The example of stop words are the prepositions (e.g. above, across, before), determiners (e.g. a, an, the) etc.
- **Stemming:** It is a process of removing the commoner morphological and inflexional endings from the words in English. Its main use is as a part of term normalization process that is usually done in information retrieval systems. It stems the words to their root words. For Example, abate, abates, abated, abatement, abatements are all stemmed to the root word 'abate'.
- **Code Snippet Filtering:** It is a process of filtering the source code present in the Q&A posts. The example is represented in Table 1.

2.2.2 Term-Document Matrix Generation

A collection of text documents is represented by a Term-by-Document Matrix (TDM) consisting of m rows and n columns, where m is the number of terms used to index the n documents. Each element a_{ij} of the matrix describes the frequency of term i that occurs in j^{th} document and is suitable measures to identify the importance of term i with respect to the j^{th} document and the entire document collection. There are different weighting scheme to find the importance of terms with respect to documents such as term frequency, Term Frequency-Inverse Document Frequency (TF-IDF), Log-IDF etc. We use TF-IDF weighting scheme to represent the pre-processed text documents[15].

2.2.3 Term Frequency-Inverse Document Frequency (TF-IDF)

The importance of terms in collection of text documents can be identified by referring Term-by-Document Matrix (TDM). There are various weighting scheme to identify the importance of terms in connection with documents. TF-IDF is one such weighting scheme that employs two types of weight: local and global. Local weight of term i

Table 1. Code snippet filtering

Post with Code Snippet	Post without Code Snippet

I am trying to remove all mentions of anyone from a string, I was wondering if there was a faster way to do this?
<pre><code>text = “hey @foo say hi to @bar”
textsplit = text.split()
n = -1
ts2 = textsplit
for x in textsplit:
  n += 1
  if x[0]== “@”:
  del ts2[n]
text = ‘ ‘.join(ts2)
</code></pre>
 Thanks in advance. (This is sort of like Removing elements from a list containing specific characters but this one is a little different.)
	I am trying to remove all mentions of anyone from a string, I was wondering if there was a faster way to do this?
	 Thanks in advance. (This is sort of like Removing elements from a list containing specific characters but this one is a little different.)

w.r.t. document j is the frequency of that term i in document j.

$$\text{Weight}_{\text{local}}(i,j) = \text{tf}(i,j) \quad (1)$$

The global weight of term i is defined as follows:

$$\text{Weight}_{\text{global}}(i) = \text{idf}(i) = \log_b\left(\frac{N_{\text{docs}}}{N_i}\right) = \frac{\log_{10}\left(\frac{N_{\text{docs}}}{N_i}\right)}{\log_{10}(b)} \quad (2)$$

Now, the weight of term i w.r.t. to document j in TDM is calculated as:

$$\text{Weight}(i,j) = \text{Weight}_{\text{local}}(i,j) * \text{Weight}_{\text{global}}(i) \quad (3)$$

$$\text{weight}(i,j) = \text{tf}(i,j) * \text{idf}(i) = \text{tf}(i,j) * \log_b\left(\frac{N_{\text{docs}}}{N_i}\right) = \text{tf}(i,j) * \frac{\log_{10}\left(\frac{N_{\text{docs}}}{N_i}\right)}{\log_{10}(b)} \quad (4)$$

where N_{docs} is the number of documents in TDM and N_i is the number of documents containing the term i.

2.2.4 Cosine Similarity

We use the cosine similarity between bug as query (B) and each individual Q&A post (Q) as document. Both query and documents are first converted into TDM with TF-IDF weighting scheme. The first column of TDM represents bug as query vector and remaining columns represents related Q&A posts as document vectors.

$$\text{Sim}_{\cos}(B,Q) = \frac{\sum_{i=1}^{n} B_i * Q_i}{\sqrt{\sum_{i=1}^{n} B_i^2} * \sqrt{\sum_{i=1}^{n} Q_i^2}} \quad (5)$$

where n is the number of terms in TDM.

2.3 Methodology

Once the data is imported into the local database after applying the matching criteria based on temporal features, the text pre-processing is applied to remove non-useful information contained on them. Thereafter TDM is generated for each matched bug consisting Q&A posts as documents to decide further whether the bug is posted or not based on cosine similarity. After all bugs of an OSS project are classified into two class i.e. posted and non-posted bugs w.r.t. Q&A posts, the average bug fix time is calculated for analysing the contribution of Stack Overflow. The detailed steps are described in Table 2 and Figure 3.

3. Empirical Analysis and Results

The empirical analysis is done by integrating bug tracking system and Stack Overflow, linking bugs to related Q&A posts (if possible) based on text mining approach and comparing the average bug fix time of posted and non-posted bugs. The temporal features and cosine similarity is used to decide whether a bug is posted into Stack Overflow or not. The illustrative example is shown in the Table 3.

The above example shows that the bug (bugid = 9776) of “Ruby” OSS projects is posted into Stack Overflow as question (postid = 23282342). The temporal features of bug-Q&A post are compatible as creation date and time of question (25-04-2014) is greater than the creation date and time of bug (25-04-2014) and the closing time (26-04-

Table 2. Algorithm: Average fix time of posted and non-posted bugs

Input: Bugs and Q&A posts of a project
Output: Average Fix Time of Posted and Non-Posted Bugs

Procedure: for each bug of a project do:
for each Q&A post related to project do:
if (Metadata features matched for Bug-Q&A pair) then
Apply pre-processing on bug once and create TDM & treat it as query;
Apply pre-processing on every Q&A and insert them into TDM for all matched Q&A posts related to the bug;
Calculate Sim_{cos} between query and individual Q&A posts of TDM;
if (threshold ≤ any Sim_{cos}) then
Record bug as posted bug with Fix Time;
else
Record bug as Non-posted bug with Fix Time;
end if-else
else
Record bug as Non-posted bug with Fix Time;
end if-else
end for loop
end for loop

Return: Average Fix Time of Posted and Non-Posted Bugs

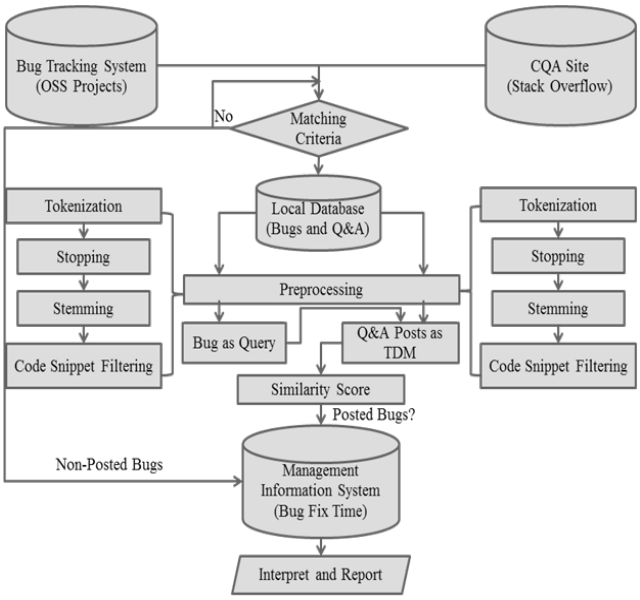

Figure 3. Research methodology.

2014) of that question with accepted answer (25-04-2014) is less than the closing time of that bug with fixed status. The bug summary and Q&A post title also reveals that they are similar. The cosine similarity is used to find the similarity with some threshold value. The bug which satisfies the matching criteria and cross the threshold value in similarity measure is classified as posted bugs. The number of posted and non-posted found is represented in Figure 4. For the selected data, next, we find the average fix time i.e. the average time required for the bugs to be resolved since the time of its creation for both posted and non-posted bugs.

Table 3. Illustrative example

Project Name = Ruby	**Tag Names = "Ruby", "Ruby Trunk"**
Bugid = 9776	Postid = 23282342
Creation Date = 25-04-2014	Creation Date = 25-04-2014
Closing Date = 26-04-2014	Accepted Answer Date = 25-04-2014
Bug Summary = "Ruby double-splat operator unexpectedly modifies hash"	Q&A Title = "Double-splat operator destructively modifies hash - is this a Ruby bug?"

The data obtained from the results of the operation on the five chosen OSS projects is presented in the Table 4. Clearly, it is observed that the average fix time of non-posted bugs is greater than the average fix time of posted bugs. The statistical analysis of the results applied to Open Office, Mozilla, Apache, Ruby and Python is presented in the Figure 5.

Table 4. Average fix time (in days) of posted vs. non-posted bugs

	Open Office	Mozilla	Apache	Ruby	Python
Posted Bugs	60	71	239	25	183
Non-Posted Bugs	114	87	268	35	207

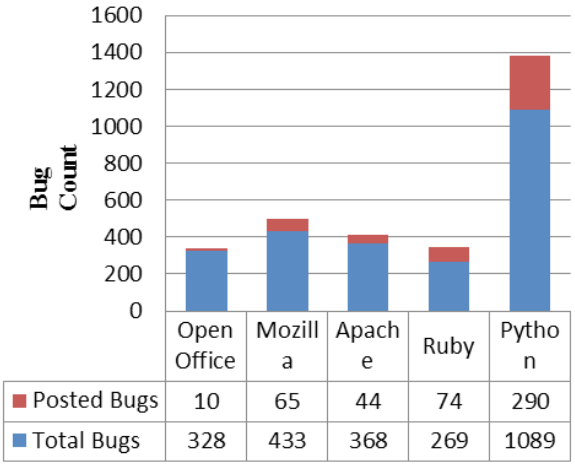

Figure 4. Posted Vs. non-posted Bugs.

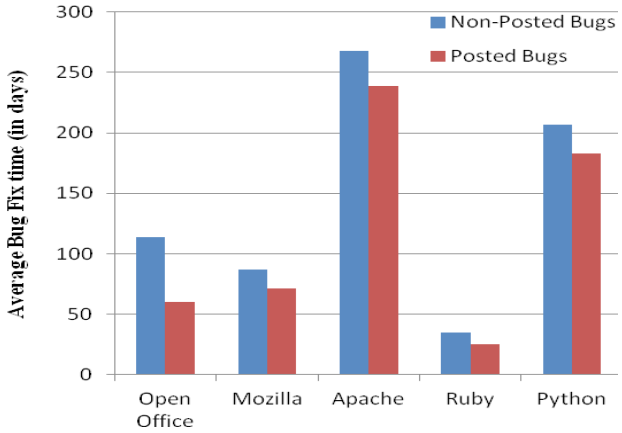

Figure 5. Average Bug Fix Time of Posted Vs Non-Posted Bugs

4. Conclusion and Future Scope

The OSS project development is becoming popular because anyone can make online contribution. There are two fundamental questions: First, how to contribute? Second, what is the impact of contribution? In this paper, the empirical analysis is presented to give the answer of these two questions using cross repository analysis between Bug Tracking System (OSS Projects) and Stack Overflow (Programming related CQA site). We have shown that the users' of Stack Overflow are contributing well by giving the solution (answer) of query (bugs in the form of question) posted by users' of OSS projects. The impact is analysed by comparing average fix time of posted and non-posted bugs into Stack Overflow. The average bug fix time of five OSS projects for the year 2014 are analysed and found that it will take less time to fix the bugs if it is posted in Stack Overflow. This leads us to believe that there is positive influence of Stack Overflow into the development of OSS projects. This means that there is a trend, where the developers seem to be looking into CQA sites like Stack Overflow to solve difficult issues by the experts in these communities who are ready to answer these queries.

The online repositories are playing an important role by hosting the data online resulting collaborative network of worldwide users. In future, the multidimensional cross repository analysis can be studied to improve the linking accuracy that increases the authorization of the online resources. But the main problem is integration of the repositories because of heterogeneous structure. The topic modelling in textual contents can also be applied that can suggests us to identify the topical authority of online users'.

5. Acknowledgment

We would like to thank KRK Mithra for gathering the datasets and National Institute of Technology Raipur to provide the necessary environment to carry out the presented work.

6. References

1. Akila V, Zayaraz G, Govindasamy V. Bug triage in open source systems: A review. International Journal of Collaborative Enterprise. 2014; 4(4):299-319.
2. Wang T, Yin G, Wang H, Yang C, Zou P. Linking Stack Overflow to issue tracker for issue resolution. Proceedings of the 6th Asia-Pacific Symposium on Internetware on Internetware; 2014. p. 1-14.
3. Correa D, Sureka A. Integrating issue tracking systems with community-based question and answering websites. Australian Software Engineering Conference (ASWEC); 2013. p. 88-96.
4. Treude C, Barzilay O, Storey MA. How do programmers ask and answer questions on the web? Nier track. 33rd International Conference on Software Engineering (ICSE); 2011. p. 804-7.
5. Storey MA, Treude C, van Deursen A, Cheng LT. The impact of social media on software engineering practices

and tools. Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research; 2010. p. 359-64.
6. Mamykina L, Manoim B, Mittal M, Hripcsak G, Hartmann B. Design lessons from the fastest Q&A site in the west. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems; 2011. p. 2857-66.
7. Barua A, Thomas SW, Hassan AE. What are developers talking about? An analysis of topics and trends in Stack Overflow. Empirical Software Engineering. 2014; 19(3):619-54.
8. Wang S, Lo D, Jiang L. An empirical study on developer interactions in Stack Overflow. Proceedings of the 28th Annual ACM Symposium on Applied Computing; 2013. p. 1019-24.
9. Deterding S. Gamification, designing for motivation. Interactions. 2012; 19(4):14-7.
10. Vasilescu B, Filkov V, Serebrenik A. Stack Overflow and GitHub: Associations between software development and crowd sourced knowledge. International Conference on Social Computing (SocialCom); 2013. P. 188-95.
11. Anderson A, Huttenlocher D, Kleinberg J, Leskovec J. Discovering value from community activity on focused question answering sites: A case study of Stack Overflow. Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2012. P. 850-8.
12. Parnin C, Treude C, Grammel L, Storey MA. Crowd documentation: Exploring the coverage and the dynamics of API discussions on Stack Overflow. Georgia Institute of Technology, Technical Report. 2012.
13. Jayalakshmi S, Sheshasaayee A. Question classification: A review of state-of-the-art algorithms and approaches. Indian Journal of Science and Technology. 2015; 8(29):1-40.
14. Nagwani NK, Verma S. On studying the effect of sample size in evaluation of bug classifiers. Indian Journal of Science and Technology. 2013; 6(1):3849-55.
15. Salton G, Buckley C. Term-weighting approaches in automatic text retrieval. Information Processing and Management. 1988; 24(5):513-23.