CS846-001 Week 5 Reviews

Jun Qing Lim
University of Waterloo
jq3lim@uwaterloo.ca

1 World of Code: An Infrastructure for Mining the Universe of Open Source VCS Data

1.1 Problem Being Solved

The paper presents an infrastructure of a collection of version control data from open-source projects called World of Code (WoC), aiming to provide as a tool to conduct research and studies on software development projects. Due to the natural sheer size of data involved in open-source projects, it is challenging to conduct a large-scale studies of it without proper techniques. This infrastructure aims to address the issue of lacking a comprehensive and robus ecosystem for gathering and analysing version control system (VCS) data It hopes to provide abilities to efficiently extract and analyze up-to-date data at a large scale with minimal resources used.

1.2 New Idea Proposed

The authors first propose the process of implementing WoC that contains four stages of:

1. Project discovery: discovers open-source projects via several approaches such as Search API, Search Engine, Keyword Search and retrieves local copies of them
2. Data retrival: extracts data objects of ecah repository and store them in a single database
3. Correction: perform correction over updated repositories
4. Reorganization: reorganizes the data to perform specific data analytics

This infrastructure is then evaluated on its applicability over multiple research tasks and concluded the following findings:

1. WoC is able to be scaled and is an efficient infrastructure to store and process VCS data
2. WoC can support diverse research tasks for most researchers
3. WoC has demonstrated its effectiveness and usefulness through several actual case-studies
4. WoC will be made publicly available in supporting large-scale studies in software development

1.3 Positive Points

Firstly, the paper addressed a crucial issue in software engineering of having terabytes of data over the internet, which makes it difficult to organize and conduct research and subsequently proposed a scalable and efficient solution, which act as a great resource for both developers and researchers. Secondly, the paper has an in-depth explanation of how the WoC infrastructure is built, which convinces the readers over its legitimacy and credibility. Thirdly, the authors did not just propose a new code infrastructure but also test and verify the effectiveness of the solution by conducting multiple research work, this helps in evaluating the actual capability of the proposed infrastructure.

1.4 Negative Points

Firstly, this code aggregation infrastructure was built without consultation from the experts on how useful this WoC would be towards the research community, thus it may not be relevant to the state of current industry. Secondly, the paper mentioned the evaluation of WoC infrastructure, but did not specify the metrics used in evaluating the comprehensiveness, which may create reliability issues. Thirdly, the paper does not mention how beneficial this WoC would be in comparison to other ideas of VCS data processing, and its similarities and differences.

1.5 Future Work

One extension is to conduct research on the relevance and usefulness of this infrastructure such as if this WoC could be integrated with any of existing software development tools or systems, providing greater accessibility and comprehensiveness of data analytics. The other extension is a survey on the relevance of this infrastructure for both academia and industry to better understand the effectiveness.

1.6 Rating

4 out of 5. The paper is of high-quality and it uses multiple sources of open-source projects, and written in well-detailed and well-organized manner.

1.7 Discussion Points

- What are some examples of existing work (of VCS data aggregation and processing)?
- What are some of the existing approaches in which researchers use to conduct study and analytics on VCS data?
- How did open-source VCS data evolved until today?

2 How Often Do Single-Statement Bugs Occur? The ManySStuBs4J Dataset

2.1 Problem Being Solved

The paper presents a dataset of single statement bug-fix changes of Java projects, known as ManySStuBs4J (where SStuBs refers to "simple stupid bugs"), it aims to address the issue of trying to determine the frequency of simple bugs that occur in a software development project and filling up the gap with this dataset. The purpose of the study is to provide an understanding of single-statement bugs in achiving an acceptable performance of a program. The authors proposed and hope that the dataset would serve as a valuable resource for both program repair and studies in empirical software engineering.

2.2 New Idea Proposed

The authors collected single statement bugs changes mined from popular open-source Java projects with codes that match a set of bug templates in studying the role of single-statement bugs in software development. The authors analyzed the the dataset and concluded the following findings:

1. on average there is about two single statement bugs occur for every project commit
2. some single-statement bugs have more than one bug pattern
3. the three most common bug pattern are: *Change Identifier User*, *Wrong Function Name*, *Change Numeric Literal*
4. static bug detectors cannot detect all single-statement bugs, developers would still need to go through the code to locate the exact bug

2.3 Positive Points

Firstly, the paper proposed a comprehensive new dataset that originates from existing open-source projects, serving as an important novel point for researchers to expand study on single-statement bugs. Secondly, the paper provided detailed information on the methodology adopted in crafting the ManySStuBs4J dataset, with clear explanation in each step of the process, providing a strong reliability of the dataset. Thirdly, the paper's study is relevant to the current state of software development whereby developers often have issues of single-line bugs and not knowing how often it occurs; through this paper, developers and researchers could better understand of the bugs and its impact in software development field.

2.4 Negative Points

Firstly, the dataset used is limited to only Java projects, which may not be easily generalize to projects of other languages. Secondly, the paper lacks clear and deep explanations on frequency of simple bugs (i.e., lacks comparison), which may not had necessarily addressed its original objective.

2.5 Future Work

The authors pointed out that single-statement bugs occur frequently (with mostly due to human coding error), one extension that we could make is to conduct an analysis and survey to figure out the reasons behind single-statement bugs, why the occur, so that they can be better prevented in the future. The other extension is extending the study to cover other programming languages, which provides a more comprehensive study on single-statement bugs.

2.6 Rating

3 out of 5. A paper that provides moderate insights and it is difficult to see what can be extended from this study because the paper lacks convincing explantions.

2.7 Discussion Points

- How does this compare with multi-statement bugs? Does multi-statement bugs occur less frequent (due to its larger code snippets, hence being put into greater attention and have less bugs)?
- Do we have any tools that assist developers in detecting single-statement bugs (linters, etc.)? How impactful and accurate are they?

3 Mea culpa: How developers fix their own simple bugs differently from other developers

3.1 Problem Being Solved

The paper presents a study of the difference in characteristics of bug fixes made by original authors and non-original authors using a collection of bug-fixing commits SStuBs dataset. It compares the bug fixes in terms of size and scope of commit, and the time taken to fix, while seeking to find how these differences affect the frequency of bug fix, time taken, and the overall quality of commit bug fixes. The paper aims to provide an understanding of developers on bug-fixing, while hoping to improve the process of fixing bugs (and its practices) through this study.

3.2 New Idea Proposed

The authors utilized the SStuBs dataset (that contains simple-statement bug fixes) with some mathematical formulas and discovered the following:

1. original authors fix more than half of all bug fixes
2. original authors fix bugs with a much lesser time than non-original authors
3. original authors make larger commits (higher lines of code) compared to non-original authors

3.3 Positive Points

Firstly, the paper's methodology is written in a well-ordered and well-explained manner, giving the readers an understanding on how the dataset is used and evaluated. Secondly, the paper is relevant to the current state of the industry and it provides a valuable contribution to developers; as the demand for software rises, by knowing the bug fixes differences between authors and non-authors, it allows organizations or developers to better improve their software development process, helping to increase efficiency and productivity. Thirdly, the paper conducted the empirical study using SStuBs dataset, providing a strong credibility and reliability over its research findings.

3.4 Negative Points

Firstly, the paper took in consideration of the "results" from the dataset and does not take in consideration of the others factors, such as experience of developers, complexity of bugs, codebase size, which may affect the authors' research outcomes. Secondly, the paper is a relatively small study that used only approximately 10K instances of single-statement bugs, such small sample size could create biases in the results of the study (eg:, Java code in generally are longer than Python, which could affect how developers chose to commit changes).

3.5 Future Work

The authors could expand this study by taking in consideration of other factors that affect the results such as *human factors*, *technical factors*, etc., which could help create a more comprehensive review on bug fixing differences. The other extension which could be done is using a larger dataset that comprise of many different programming languages and multiple domains of bugs, so that the results could be much more reliable.

3.6 Rating

3 out of 5. Although the paper provides valuable ideas to bug fixing studies with well-written findings, the paper is comparatively short and lack many specific details.

3.7 Discussion Points

- Could programming languages affect bug fixing practices across developers?
- What are the tools available in the industry that could assist developers in fixing bugs to raise efficiency and productivity?
- How does experience of developers (number of experiences) affect the overall process of bug fixing?

4 SOTorrent: Reconstructing and Analyzing the Evolution of Stack Overflow Posts

4.1 Problem Being Solved

The paper presents a dataset called SOTorrent that contains data of Stack Overflow posts to investigate its evolution history such as post content changes, which posts or code snippets get edited and relationship between post edits and comments. It addresses issue of lacking of a comprehensive Stack Overflow dataset at a fine-grained level in conducting a thorough study on the evolution of SO posts and its role in the software development environment. This paper provides insights on SO posts and allows developers and researchers to able to conduct study on various aspects of Stack Overflow including its posts and behaviour, to better understand the software development field.

4.2 New Idea Proposed

The authors aggregated data from the official SO data dump that contains access to the version history at a whole level post / individual text / code blocks and used this open dataset to conduct various study on SO posts evolution. The findings are as follow:

1. Many of the SO posts edit have been edited at least once with mostly modification on a single line of text / code
2. Changes on code blocks are usually accompanied buy changes in texts
3. Post edits are mostly made on the same day of post creation and made by post authors themselves
4. Posts with higher frequency of edits have more comments made

The authors hope to utilize the findings of this study in helping us to better understand the future of Stack Overflow in software development and how it evolves over time.

4.3 Positive Points

Firstly, the paper first addressed its objective and decided to come out with a more comprehensive dataset instead of using the default Stack Overflow data dump (that does not contain detailed information), this allows the study to be done in a thorough manner, giving more accurate representation of its findings. Secondly, the authors created the SOTorrent and made it open source, which is a beneficial and useful resource for both developers and researchers in conduct similar study in the field. Thirdly, the paper provided statistical details on how the study was conducted in evaluating the SO posts, which helps to convince the readers on its conclusion / findings.

4.4 Negative Points

Firstly, the data used might be limited in studying software development practices, because it only uses Stack Overflow as dataset, and the findings may not be generalized to other question-and-answer website for developers (such as Quora, Code Project, GitHub). Secondly, the authors built the SOTorrent without evaluating the accuracy of the dataset, which could cause the findings to be inaccurate.

4.5 Future Work

As illustrated under the negative points, we could extend the work by using dataset across different common websites (such as Quora, GitHub), giving a much broader and comprehensive findings of their roles in software development field. We could also enhance the reliability of the SOTorrent by conducting evaluation on its accuracy, allow us to prevent publication of unreliable or inaccurate information.

4.6 Rating

4 out of 5. The findings are useful for us to better understand the trends of developers in the internet world in solving a technical problem.

4.7 Discussion Points

- Are there any other question-and-answer platforms which may give different findings with a different set of user behaviours?
- How did the evolution of internet access affected the idea of "online communities" in the software development world?
- How can we expand the usefulness of the proposed SOTorrent dataset? What are the ways which we could enhance it further?