Paper Review CS846, Feb. 27
Parse that data! Practical tips for preparing your raw data for analysis **and** *Characterizing and predicting which bugs get fixed: An empirical study of Microsoft Windows*

Wenhan Zhu (Cosmos)
w65zhu@uwaterloo.ca
University of Waterloo
Waterloo, Canada

ABSTRACT

This week's chapter from the book being reviewed here is **Parse that data! Practical tips for preparing your raw data for analysis** by *Philip Guo* and **Characterizing and predicting which bugs get fixed: An empirical study of Microsoft Windows** by *Philip J. Guo, Thomas Zimmermann, Nachiappan Nagappan and Brendan Murphy*

Guo's chapter talks about his experiences when dealing with parsing raw data and how to prepare it for further analysis.

Guo et al. is about an empirical study in *Microsoft* which focused on which bugs gets fixed for *Windows Vista, Windows 7*. Their results showed that bug reports by reputed personals are more likely to be fixed and also for bug reports submitted by the same group of people. They also implemented a predictor which have around 65% of accuracy when predicting which bugs get fixed.

KEYWORDS

paper review

1 SUMMARY

Guo's chapter is very compact and full of useful information. He mentioned that due to the nature of data we are using there exists many outliers and often we can not guarantee the consistency or correctness of data. So we need to clean up the data for further analysis. He suggests that when writing the parser, we should use assertions at every place when we make assumptions of the data. So that we will no when ever the assumption if not met by the raw data. Printing out data that violates assertions will make debugging easier and get to know the data better. When dealing with data that is categories, we should count each occurrences so that we don't missed out outliers that likely to be noises in the data. Our parser should support restarting at relative points, since when doing parsing it's likely in the middle of a huge dataset the parser will break and restarting from the very beginning is a waste of time in the majority of times since we already parsed them. However, we should always rerun the parsing when all things are fixed to make sure we don't make a regression mistake. When prototyping the parser, we should work on a small subset of the data so that the prototyping phase is really fast. Store standard output data to log files so that we can revisit them when needed. We can also store raw data with cleaned data so we can verify our parsing at any time when space permits. And when everything is finished write a verifying program to make sure the cleaned data meets our expectations.

Guo et al.'s paper is an empirical study on which bugs get fixed on *Window Vista, Windows 7* for *Microsoft*. The authors extracted information from bug reports for these systems and processed the data to fit their analysis. They found out some very interesting results that people with previous more successful bug fixed are more likely to have their bug fixed in the future. The authors assumes that it is because the higher quality of bug reports they produce compared to other people. Another results from this is that the more people interested in a specific bug it's more likely to be fixed. They also found out that reassignments and reopening does not always affect the likelihood of whether a bug gets fixed. They found out that bug reports from another country or team are less likely to be fixed due to fewer communications. Using this data, the authors built a logistic regression model to predict the likeliness of whether a bug gets fixed. The model have around 65% success rate for both the systems.

2 THOUGHTS

This combo of chapter and paper is an advice for how to parse data and an actual usage of such techniques on predicting bug report resolutions for *Windows* systems.

The chapter gave very good suggestions for how to parse data in practice. I think the suggestion made are very useful for parsing data for research. I will definitely implement the suggestions in future research so that I can increase productivity and improve the precision of my parser.

I like the paper's idea on predicting which bugs are more likely to be fixed inside *Microsoft*, however, it seems to me that some information are not very useful for people outside of *Microsoft*. For example, the results for the likeliness of whether a bug will be fixed when the submitter is from another country or group is not applicable for open source programs and it's also likely affected by how software development happens inside *Microsoft*. For example, if the team sits in the same office, they might communicate during coffee break about the bug and one of the previous papers we have read suggested that the life of a bug does not necessary only lies in the bug reports since not all communications are documented. I wonder how this work can be replicated on open source projects and if the results are the same in these circumstances.

3 RATINGS

I would rate *Guo*'s chapter 5/5. To me the content is extremely useful and it's written in a compact way that I can follow each and everyone of them. He also gave explanation on each suggestion he made and says about the downside for some of them and let the reader make the judgement.

I would rate *Guo et al.*'s work a 2/5. It's idea is nice, but I don't like many of the things are *Microsoft* only and the method does not seem novel to me (Maybe due to it's a 2010 paper).