CS846-001 Week 8 Reviews

Jun Qing Lim
University of Waterloo
jq3lim@uwaterloo.ca

1 Diversity in Software Engineering Research

1.1 Problem Being Solved

The paper presents a measure of determining the coverage of a paper through an algorithm of computing the coverage scores. It addresses the issue of not being able to determine how broadly a paper covers (generality), where a narrowedly covered paper will not have findings that are representative and useful for other domains. Through this proposed measure, it hopes to enhance researchers' ability of reasoning and validating the findings of a study, with an objective of making its studies not just relevant to its own domain but other domains of the entire population too.

1.2 New Idea Proposed

The authors first combined ideas from diversity and representativeness concepts and introduced a sample coverage (percentage of projects similar to a given sample) - the higher this score, the higher the coverage and generality. The authors then introduced an example of how to compute this sample coverage through an evaluation of an empirical case study of a software engineering research. The following findings were then highlighted and acknowledged:

- coverages scores do not reflect how important a research is, but rather it enhances the ability of reasoning the research - low score does not imply low value of study
- a low coverage score may cause a researcher not to report the results as it felt it was undervalued, but a low coverage paper should not be discouraged to publish as it may still have value (especially in a space that is highly covered)
- a paper should not also report the coverage scores, but also report how they are covered (i.e., universe, space, and configuration) of report scores

The authors hoped that this proposed technique will assist researchers in making informed decisions on the selection of project to study.

1.3 Positive Points

Firstly, the authors highlighted an important point in software engineering research (diversity & generality) in achieving research outcomes that are useful to multiple domains, this helps create awareness among researchers and practitioners when publishing papers. Secondly, the authors introduced a new idea of measuring coverageness of a paper, this helps to assist researchers in pinpointing and determining the relevance of a paper to the community. Thirdly, although the paper was written close to a decade ago, but the ideas illustrated are still relevant to the current industry practitioners as it highlights the importance of producing research that is of wide range of contexts and usefulness.

1.4 Negative Points

Firstly, although the paper provided a sample of empirical case study, but it lacks practical guidelines on how to implement these approaches in practice. Secondly, the paper does not illustrate how accurate the coverage scores are (i.e., no discussion on the proposed technique of computing coverage scores), as such it may be challenging to convince readers on its reliability.

1.5 Future Work

One extension would be conduct further research on how diversity of software engineering research affects the community (will low coverage findings lead to better or worse outcomes of extended research?). The other extension would be to conduct a case study on the efficacy, relevance of this proposed technique; this allows us to have a deeper understanding on the usefulness of the coverage scores.

1.6 Rating

3 out of 5. The paper lacks detailed discussion and reports on the relevance of the proposed technique.

1.7 Discussion Points

- How do researchers typically select the coverage of study?
- How broadly is a typical study covered?

2 Fair and Balanced? Bias in Bug-Fix Datasets

2.1 Problem Being Solved

The paper presents a study on software projects to explore evidences of systematic bias in bug-fix datasets to evaluate software quality. It highlights that datasets may be skewed towards bugs that are easier to fix with higher chance of occurrences, which results in inaccuracies in prediction models. It aims to address the issue of inability to build effective bug-prediction system due to poor quality of software and datasets adopted. The authors hoped that this paper could raise awareness of these problems and to encourage researchers and practitioners in developing balanced (and less bias) datasets in improving the accuracy of software evaluation.

2.2 New Idea Proposed

The authors did not propose any new idea, but provided an analysis of bias is bug-fix datasets. The authors examined several possibilities of bugs in the category of: bug type, properties of bug fixer and properties of fixing process. The findings of the bug-feature bias were then as follow:

- bug-feature bias is only important when it affects the hypothesis of one's research
- the bias towards certain bug types could lead to an overestimation of effectiveness of certain bug-fixing techniques
- bug-fix datasets have significant impact on software quality that leads to inaccurate assessments of software quality
- biases could be reduced if researchers adopt higher transparency such as reporting the sample used, this could help improving the fairness of a software evaluation.

2.3 Positive Points

Firstly, the paper provided a deep analysis on bug-fix datasets with convincing explanations on its arguments such as frequency of bugs, severity, and fix time, this provides readers an overview on the nature of biases. Secondly, the paper provided practical suggestions on addressing bias such as random sampling, increasing number of bugs, etc.; these help researchers in providing an idea to develop balanced bug-fix datasets.

2.4 Negative Points

Firstly, the paper is slightly outdated as it was published in 2009 - while the findings are still relevant, there have been significant development in software quality in the past decade, thus the ideas proposed may not be fully digestable in the current context. Secondly, the paper lacks in-depth discussions (or did not acknowledge) on how other factors could also affect software quality (such as software complexity, methodologies adopted, etc.)

2.5 Future Work

One extension could be exploring how biases in bug-fix affect prediction models, doing so allow us to understand the relationship between datasets and prediction model in building a more robust and accurate models. Besides that, we could also study the effects of biases on software quality to build better awareness for researchers to identify strategies to mitigate biases.

2.6 Rating

2 out of 5. The paper is a general study and was written with many statistical terms/explanations. This makes the paper hard to be understood by researchers with general background. Paper will be better off if layman-terms were used instead.

2.7 Discussion Points

- How can we measure the impact of bias on software quality evaluation?
- What is the long-term consequences of using biased bug-fix datasets in our research?
- Are there any types of software that are more prone to bias in bug-fix datasets?

3 It's Not a Bug, It's a Feature: How Misclassification Impacts Bug Prediction

3.1 Problem Being Solved

The paper presents a study on how misclassification of bugs impact prediction of bugs. It explores how misclassification leads to an underestimation of number of bugs in a software and how this negatively impacts bug prediction models. The authors conducted an empirical survey on multiple open-source projects, aiming to show that misclassification is a common problem and proposes a few solutions to improve bug classification and prediction.

3.2 New Idea Proposed

The authors conducted the study on five open-source projects and classifies the categories of issues of the project as: BUG, RFE (feature enhancement), IMPR (improving functionality), DOC (documentation), REFAC (refactoring), OTHER. Two authors were then selected to manually classify the issues; the results were then compared and the following findings were obtained:

- close to 40% of all issues are wrongly classified, as such misclassification is a common issue
- misclassification leads to significant underestimation on the number of bugs in the system, leading inaccurate findings
- current bug prediction models were trained to predict changes instead of bugs; as such prediction models are trained to have biases

The authors then hoped that these findings could build better awareness in the need of improving bug classification and prediction models for better accuracy and reliability of software.

3.3 Positive Points

Firstly, the paper presents an empirical study using a set of open-source projects to investigate the impact of misclassification on bug prediction, where the evidences help increase its reliability and relevance. Secondly, the paper was well-written in consistent structure with detailed explanations (i.e., started of with an overview in its introduction with a brief idea), giving readers a good flow when reading it. Thirdly, the paper did not just explore the aspect of impact of misclassification but it also illustrated the sources of misclassification, giving a concrete understanding in the aspect of classification for readers without a background knowledge in the field of prediction modeling.

3.4 Negative Points

Firstly, the paper lacks implementation details where it proposes ideas to address misclassification and improving bug prediction system but does not provide practical guidelines on how to achieve these. Secondly, the paper only conducted the study on Java-based projects, as such it may not be relevant to projects of other languages and the findings may not be generalizable.

3.5 Future Work

We could conduct a study in developing advanced bug prediction models to address the issue of misclassification and prediction inaccuracies with the use of newer machine learning algorithms. The other extension is that we could investigate how misclassification affect tasks in other software domains, giving a better overview on the impacts and consequences of bugs misclassification.

3.6 Rating

5 out of 5. The paper is written in a very concise manner and it has high relevance to the current state of industry (as misclassification is an adverse isuse). As such, this paper has given readers a good understanding and awareness in building prediction models.

3.7 Discussion Points

- How important is classification of bugs to be accurate?
- What are the factors that could contribute to misclassification?
- Are there any practical limitations of the paper written?