Diversity in Software Engineering Research Review - By Ahmed El Shatshat

Problem to be Solved

One of the primary goals of Software Engineering Research is to achieve generality. To do so, it is essential to collect a diverse and representative set of projects for analysis, as focusing on a narrow group of projects within a relevant population can lead to skewed data. However, in practice, determining whether a selected sample has high coverage across relevant subgroups is challenging. This highlights the need for a systematic way to measure sample diversity, ensuring that research findings are broadly applicable across the software space.

New Idea

The authors introduce a measure called sample coverage, defined as the percentage of projects in a population that are similar to a given sample. This measure not only provides an accurate way to assess the quality of a sample but also helps highlight projects that could be added to improve its representativeness.

To calculate similarity metrics, the authors introduce a vocabulary consisting of universe, space, and configuration:

- Universe represents a large set of projects, also known as the population. Possible universes include all open-source projects, all closed-source projects, or all web applications.
- Space refers to the dimensions a project covers, such as total lines of code, number of developers, and main programming language. The research focus determines which dimensions are relevant.
- Configuration defines how similarity is determined between projects. A configuration consists of a list of similarity functions, and for a project to be considered similar within a universe, it must be similar in all selected dimensions.

The authors emphasize that coverage scores do not determine whether research is important; instead, they provide a way to reason about research findings. Achieving generality in software engineering research is difficult, but understanding the context in which a study is conducted helps researchers gain deeper insights into its results, even when they differ.

Positives

Thorough Evaluation of an Integral Part of Research: A bad sampling can completely devalue the worth of a paper, especially if it is not reported on. The authors provides both good arguments for proper sampling, as well as a thorough methodology for assessing your project's coverage

Well Formatted: The authors do a great job of clearly illustrating how their sampling method works, provide a clear example, and follow it with an insightful discussion of the results**.** Graphs are easy to understand and provide further insight into results

Universally Applicable: Almost all, if not all, research projects have to at least reference other similar projects. As a result, the research presented provides value to all research; an extremely generalizable result

Negatives

Rambles a Bit Near the End: I feel as though the paper could have been cut off much earlier, especially given that the Related Work section feels a little backloaded.

Feels a Touch Too Conversational At Times: At times the tone leans a little too conversational for a research paper **(**e.g. "consider a researcher who wants to investigate a hypothesis about say distributed development…")

Future Work

It would be interesting to integrate a tool with research archival platforms like ACM Digital Library to perform these checks.

- Would help with finding similar papers across dimensions
- Currently it's still quite a pain to find relevant research papers on these platform

I wonder if this methodology could extend to other research domains?

- Would need communication with domain experts to see if the Universe/Space/Configuration model would be applicable
- Would also need to understand what keywords would be relevant if such a model is applicable

Rating

I give the paper a 5/5. This paper provides both great insight, and a thorough methodology for the problem it seeks to address.

Discussion Points

1. Does a bad sampling ever dissuade you from referencing a research paper?
    a. Would that decision be swayed by the paper reporting its sampling as a threat to validity?
2. Do you believe that better sampling coverage and reporting sampling scores will lead to more robust results in the SE domain?
3. What other considerations do you think could be included in determining coverage?

Summary of Discussion

We began the discussion with the first discussion point above. One student gave this experiences with dealing with poor sampling in research papers, saying that oftentimes it is unavoidable to have a poor sample; sometimes there just isn't enough data. However, he still agreed that in such cases, reporting the issues with a sample (whatever they may be) lends more credibility to the authors than not.

Another student gave their thoughts on the score coverage metric itself, stating that most projects evaluated with the sample coverage metric had low scores. They believed that such a result implies that the sample coverage metric itself isn't very useful. However, I argued—in accordance with the authors' thoughts as well—that such a metric is merely to provide more insight by which both author and reader can reason upon their results. There is also the use of using the sample coverage metric to find new projects to add to your sample that you may not have considered.

On the topic of other considerations that could be included in determining metrics, one student noted that there isn't any way to prioritize any dimensions as more important to the sample coverage. We determined that some sort of weighting system could be helpful in such cases.

We also explored that, while the paper itself covers software projects as a primary use case, the model presented could also be applied to finding research papers broadly. Formally submitted papers have keywords as part of the general format; the only issue is that different papers may use different keywords to describe the same concept. Some form of fuzzing or grouping algorithm could help with this issue.