A Threats to Validity

Our survey was conducted according to well-established literature review guidelines for systematic literature reviews (SLRs) in software engineering (SE) [3]. Naturally, as with many other SLRs, our survey has room for improvement. We discuss a few points below.

A.1 Search Strategy Threats

Threats to the search strategy can occur if our search methods do not cover enough relevant studies. To mitigate this, we employed a systematic search strategy [1, 2] across multiple databases. We carefully refined the search string using the QGS [5] to ensure a wide and accurate coverage of relevant studies. Additionally, we performed a comprehensive snowballing search to identify any significant studies that may have been missed.

A.2 Study Selection Threats

Threats to study selection arise when the inclusion and exclusion criteria are not properly designed, potentially leading to biased results. To address this, we defined clear and precise inclusion and exclusion criteria before the selection process. Multiple researchers independently screened the studies, and any discrepancies were resolved through discussion or consulting a third party, ensuring a balanced and unbiased selection of studies. Furthermore, with the merging trend of LMs, most primary studies are from arXiv and are not peer-reviewed, posing a validity threat. To handle this, we conducted rigorous study quality assessments using specialized quality assessment criteria (QAC) to filter out low-quality studies [2, 4].

A.3 Data Collection Threats

Threats to data collection can affect the validity of our findings if the data extraction process is flawed or subjective. To mitigate this, we developed a standardized table to record all data collected in this process, and any modifications to the table were independently reviewed by at least two researchers to verify the accuracy of the extracted data [1]. Any inconsistencies were discussed and resolved collaboratively to maintain data integrity.

By addressing these potential threats to validity, we aim to enhance the robustness and reliability of our survey findings.

References

[1] Jingzhi Gong and Tao Chen. Deep configuration performance learning: A systematic survey and taxonomy. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 2024.

[2] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology*, 2023.

[3] Barbara Kitchenham and Stuart M. Charters. Guidelines for performing systematic literature reviews in software engineering, 2007.

[4] Simin Wang, Liguo Huang, Amiao Gao, Jidong Ge, Tengfei Zhang, Haitao Feng, Ishna Satyarth, Ming Li, He Zhang, and Vincent Ng. Machine/deep learning for software engineering: A systematic literature review. *IEEE Trans. Software Eng.*, 49(3):1188–1231, 2023.

[5] He Zhang, Muhammad Ali Babar, and Paolo Tell. Identifying relevant studies in software engineering. *Information and Software Technology*, 53:625–637, 2011.