Artisan: Automated Artifact Evaluation with Agents

ANONYMOUS AUTHOR(S)

Artifact evaluation has been shown to be a valuablue way to ensure the reproducibility of software artifacts. However, in its current form, it remains notoriously labor-intensive. To mitigate this, recent work has introduced benchmarks that evaluate LLM agents on automatically reproducing research results. We identify several limitations of existing benchmarks: (1) They largely frame the task as output prediction. (2) Their usefulness is limited: LLM agents successfully predicting the outcome of research does not raise the confidence that the paper is reproducible. (3) They cover a narrow domain of computer science and programming languages. To address these issues, we present SEAE-Bench, an automated artifact evaluation benchmark for software engineering research. SEAE-Bench has the following characteristics: (1) It measures an agent's ability to generate code that reproduces a paper's particular claim; not only outputs but also the script used to produce them are graded. (2) The submitted script serves as a reproduction script for the particular claim, and when the result of the script does not match with the results of the paper, it acts as a credible evidence of the transcription or rounding errors on the side of the paper. (3) It spans diverse software engineering techniques and programming languages. Informed by existing LLM agents' struggles on SEAE-Bench, we develop Artisan, an agentic approach to automated artifact evaluation. Artisan includes specialized tools for domain-specific challenges of artifact evaluation. Our experiments show that, while most LLM agents struggle on SEAE-Bench, Artisan is effective, producing TODO reproduction scripts and outperforming the baseline by TODO. In addition, during the development of SEAE-Bench, we uncovered TODO inconsistencies between papers and their artifacts, demonstrating the strength of our approach to benchmark construction.

CCS Concepts: • **Software and its engineering → Software maintenance tools**.

Additional Key Words and Phrases: Artifact Evaluation, Benchmarks, LLM agents

ACM Reference Format:
Anonymous Author(s). 2025. Artisan: Automated Artifact Evaluation with Agents. 1, 1 (September 2025), 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Since the pilot of the artifact evaluation at ESEC/FSE 2011, artifact evaluation has become a standard practice at a software engineering and programming language communities [13]. One of the main goals of artifact evaluation is reproducibility, which is defined as obtainment of the similar experimental outcome with the same experimental setup by the different team [4]. Reproducibility is one of the stated goals of artifact evaluation in the Calls for Artifacts and generally regarded as one of the most important goal of the artifact evaluation by the participants. [9]. In addition, there has been studies reporting that artifact evaluation leads to more available, maintained, and documented artifacts [16].

Despite these benefits, artifact evaluation practiced in its current form is notoriously labor-intensive. There has been report that an individual artifact evaluation could take from eight to ten hours, which is more than the usual time for the paper review [8]. There has been mixed evidence whether papers that underwent artifact evaluation has more visibility, with some studies suggesting higher visibility [10] and some studies finding no correlation [16]. Considering the significant

ACM XXXX-XXXX/2025/9-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Table 1. Comparison of prior benchmarks and our benchmark. ML stands for Machine Learning. NLP stands for Natural Language Processing. CS stands for Computer Science. SS stands for Social Science. Med stands for Medicine. SE stands for Software Engineering.

	SUPER [5]	**CORE-Bench** [14]	**Repro-Bench** [11]	**SEAE-Bench (Ours)**
Domain	ML, NLP	CS, SS, Med	SS	SE
#Paper	45	90[1]	112	35
#Task	45[2]	270	112	162
#Programming Languages	1	2	6	10
#Inconsistensies Found	0	0	0	10
Time Limit	30 minutes	45 minutes	2 hours	8 hours
Judging	Accuracy of outputs	Accuracy of outputs	Accuracy of reproducibility scores	#Script that reproduces the claim

[1] 37 for CS [2] In terms of Expert set. SUPER additionally provide 152 masked sets and 604 Auto sets.

manual effort, it is unclear whether the benefit for individual researchers as to do artifact evalutaion outweight the cost as both the authors and the users of the artifact.

To mitigate this, there has been a recent effort to benchmark the performance of LLM agents on reproducing research results. Table 1 summarizes these recent efforts. SUPER [5] is a benchmark that evaluates the capability of LLM agents in reproducing results from ML and NLP papers. CORE-Bench [14], while having similar overall goal, differs by targeting broader domain (Social Science, Medicine) and including vision-language tasks. Repro-Bench [11] focuses on reproduction of social science research, and makes the problem setup more realistic.

However, we find several limitations with the existing benchmarks: (1) Existing benchmarks largely formulate the problem as an output prediction problem, which undermines its credibility. SUPER and CORE-Bench only measures the accuracy of the reproduction outcome by the agents; how they obtained the result is not part of the evaluation. This disagrees with the goals of the artifact evaluation as practiced in the software engineering community. Furthermore, it is also potentially more prone to data leakages. Repro-Bench measures the accuracy of the reproducibility scores from one to four compared to the ground truth. While artifact evaluation badge (similar to the reproducibility score used by Repro-Bench) is one of the important side-effect of the artifact evaluation process which signals the quality of the artifact, it is not the main goal of the artifact evaluation. (2) Due to such problem formulation, they have limited usefulness. Contrary to human artifact evaluation which adds confidence that the result of papers are reproducible, exercise of benchmarks do not confer similar confidence. Existing benchmarks thus can be used to measure LLM agents' performance on the formulated task, but they can not augment or replace the current artifact evaluation. (3) They are limited in terms of domain of computer science (ML, NLP) and programming languages (Python, R, Stata).

To mitigate this problem, we present SEAE-Bench, an automated artifact evaluation benchmark in the domain of software engineering. SEAE-Bench has the following characteristics: (1) SEAE-Bench measures LLM agents' ability to generate a code that reproduces the particular claim of the paper; not only the output but the code behind is graded. It also distinguishes fast-path and short-path that LLM agents can take to reproduce the results, which is a crucial distinction not yet highlighted by the previous work. (2) SEAE-Bench thus has two applications: A script submitted by LLM agents can be used as a reproduction scripit that reproduces the particular claim of the paper. In the case that the outcome of the submitted script differs from that of the paper, it acts as a credible evidence that paper actually is the one with a transcription or rounding error. (3) SEAE-Bench covers diverse software engineering techniques and programming languages. It covers diverse software engineering problems (empirical study, fuzzing, static analysis) and diverse real-world Programming Languages (C, C++, Go, JS, Java, Scala, Rust, Ocaml, etc.).

Informed by the existing agents' struggle with the SEAE-Bench, we also develop Artisan, an agentic approach for the automated artifact evalutaion. Artisan is equipped with many specialized tools to deal with domain-specific problems of artifact evaluation. TODO: Many more technical novelty for the side of Artisan?

Our evaluation shows that while most LLM agents struggle with SEAE-Bench, Artisan is the most effective LLM agent by producing TODO reproduction scripts, outperforming the baseline by TODO. In addition, during our development of SEAE-Bench, we have uncovered TODO inconsistencies between the paper and the artifact, which demonstrates the strength of our approach in benchmark construction.

In summary, this paper contributes the following:

- SEAE-Bench, a novel benchmark that evaluates the capabilities of AI agents to reproduce Software Engineering papers using script submission.
- Artisan, an LLM-based agent which incorporates several tools to aid the reproduction of software engineering research artifacts.
- Empirical evaluation which shows that (1) SEAE-Bench is challenging for the current state of the art LLM agents and (2) Artisan effectively addresses the common pitfalls faced by LLM agents.
- We make SEAE-Bench and Artisan publicly available: https://github.com/doehyunbaek/artisan

2 Background

2.1 Artifact Evaluation

3 Benchmark

In this section, we discuss methodologies we use to construct SEAE-Bench. We first discuss important requirements we aim our benchmark to have (Section 3.1), how we select papers in the benchmark (Section 3.2), how we select individual tasks that constitute the benchmark finally (Section 3.3), how individual task of SEAE-Bench looks like(. Section 3.4), and how individual submission is graded (Section 3.5).

3.1 Requirements

There are two core requirements we have for our benchmark. 1. The tasks in the benchmark should reflect a real-world problem realistic enough so that good results in the benchmark generalize to the real-world use cases. This is motivated by limitations of the previous benchmarks as we view previous benchmarks [5, 11, 14] not to fully reflect the real-world artifact evaluations practiced in the software engineering communities. 2. Manual validation is desired. While SWE-Bench [12] has been widely influential as a benchmark of evaluating capabilities of LLMs on real-world software engineering issues, efforts like SWE-Bench Verified [7] and [15] has highlighted the important of manual validation not to overestimate or underestimate the capabilities of LLM agents. Although both are required for good benchmarks, there are some factors where there is a conflict. For instance, task size of benchmarks should be extensive enough to cover diverse use cases but small enough to

Table 2. Paper selection criteria

Criterion	Count
Flagship SE conference papers in 2024 with Available and Reusable badges	114
Packaged using Docker	76
No non-public API use	68
No GPU use	46
Less than eight hours per task	35
Manual reproduction successful	30

enable manual validation. In addition, task size too large leads to more time and monetary cost for reproduction, which might be too resource-intensive for resource-constrained academic groups [6]

3.2 Paper Selection

Guided by these requirements, we select a set of 30 papers which we select through the process outlined in Table 2. (1) We gather 114 papers published in four flagship software engineering conferences (ICSE, FSE, ASE, ISSTA) in 2024 with ACM artifacts available and reusable badges [4]. We also manually search the web to find the link to the artifacts accompanying the paper. This is straightforward in the case of FSE and ISSTA papers which contain a link to the related artifact in ACM Digital Library; in other conferences where this is not the case, we use a popular search engine (Google) with the title of the paper. (2) We select 77 papers whose artifacts that are packaged using Docker. This is motivated by the fact that Docker and similar virtualization technologies have become standard requirements in recent artifact evaluation processes [1–3]. To implement this, we do text search with docker (case-insensitive) on markdown and pdf files. (3) We select 67 papers that do not make use of non-public APIs. This is to ensure reproducible outcome. To implement this, we do text search with openai or anthropic as a keyword on Python and Jupyter Notebook files. All eight excluded papers make use of OpenAI APIs. (4) We select 46 papers which do not make use of GPU or any specialized hardware. This is to make benchmark accessible to a wide variety of hardware setups. To implement this, we do text search for common GPU keyords (cuda, cudnn, tensorflow, nvidia-smi, –gpus) files with extensions excluding common non-code extensions (.txt, .csv, .json, .log). (5) We select 35 papers which do not take eight hours per each task. To do the selection, we manually search each paper texts with keyword "hour" to see if there is any explicit mention of timeout setting. If there is such mention of such setting over eight hours, we filter them out. (6) We manually try to reproduce the major claims of the paper with maximum of eight hours. If we fail to set up the environment or reproduce the major claims, we exclude the papers.

3.3 Task Selection

3.4 Example Task

3.5 Grading

4 Approach

5 Evaluation

6 Discussion

6.1 Threats To Validity

6.2 Limitations

7 Related work

References

[1] ASE 2024 2024. *Artifact Evaluation Track — ASE 2024 (Call for Artifacts)*. ASE 2024. https://conf.researchr.org/track/ase-2024/ase-2024-artifact-evaluation-track#Call-for-Artifacts

[2] ICSE 2024 2024. *Artifact Evaluation — ICSE 2024*. ICSE 2024. https://conf.researchr.org/track/icse-2024/icse-2024-artifact-evaluation

[3] ESEC/FSE 2024 2024. *Artifacts — FSE 2024 (Call for Artifacts)*. ESEC/FSE 2024. https://2024.esec-fse.org/track/fse-2024-artifacts

[4] Association for Computing Machinery. 2020. *Artifact Review and Badging — Current*. https://www.acm.org/publications/policies/artifact-review-and-badging-current Version 1.1.

[5] Ben Bogin, Kejuan Yang, Shashank Gupta, Kyle Richardson, Erin Bransom, Peter Clark, Ashish Sabharwal, and Tushar Khot. 2024. SUPER: Evaluating Agents on Setting Up and Executing Tasks from Research Repositories. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, 12622–12645. doi:10.18653/V1/2024.EMNLP-MAIN.702

[6] Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Aleksander Madry, and Lilian Weng. 2025. MLE-bench: Evaluating Machine Learning Agents on Machine Learning Engineering. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net. https://openreview.net/forum?id=6s5uXNWGIh

[7] Neil Chowdhury, James Aung, Chan Jun Shern, Oliver Jaffe, Dane Sherburn, Giulio Starace, Evan Mays, Rachel Dias, Marwan Aljubeh, Mia Glaese, Carlos E. Jimenez, John Yang, Leyton Ho, Tejal Patwardhan, Kevin Liu, and Aleksander Madry. 2024. Introducing SWE-bench Verified. https://openai.com/index/introducing-swe-bench-verified/

[8] Ben Hermann. 2022. What Has Artifact Evaluation Ever Done for Us? *IEEE Secur. Priv.* 20, 5 (2022), 96–99. doi:10.1109/MSEC.2022.3184234

[9] Ben Hermann, Stefan Winter, and Janet Siegmund. 2020. Community expectations for research artifacts and evaluation processes. In *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*, Prem Devanbu, Myra B. Cohen, and Thomas Zimmermann (Eds.). ACM, 469–480. doi:10.1145/3368089.3409767

[10] Robert Heumüller, Sebastian Nielebock, Jacob Krüger, and Frank Ortmeier. 2020. Publish or perish, but do not forget your software artifacts. *Empir. Softw. Eng.* 25, 6 (2020), 4585–4616. doi:10.1007/S10664-020-09851-6

[11] Chuxuan Hu, Liyun Zhang, Yeji Lim, Aum Wadhwani, Austin Peters, and Daniel Kang. 2025. REPRO-Bench: Can Agentic AI Systems Assess the Reproducibility of Social Science Research?. In *Findings of the Association for Computational Linguistics, ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (Eds.). Association for Computational Linguistics, 23616–23626. https://aclanthology.org/2025.findings-acl.1210/

[12] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. 2024. SWE-bench: Can Language Models Resolve Real-world Github Issues?. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=VTF8yNQM66

[13] Shriram Krishnamurthi and Jan Vitek. 2015. The real software crisis: repeatability as a core value. *Commun. ACM* 58, 3 (2015), 34–36. doi:10.1145/2658987

[14] Zachary S. Siegel, Sayash Kapoor, Nitya Nadgir, Benedikt Stroebl, and Arvind Narayanan. 2024. CORE-Bench: Fostering the Credibility of Published Research Through a Computational Reproducibility Agent Benchmark. *Trans. Mach. Learn. Res.* 2024 (2024). https://openreview.net/forum?id=BsMMc4MEGS

[15] You Wang, Michael Pradel, and Zhongxin Liu. 2025. Are "Solved Issues" in SWE-bench Really Solved Correctly? An Empirical Study. *CoRR* abs/2503.15223 (2025). doi:10.48550/ARXIV.2503.15223 arXiv:2503.15223

[16] Stefan Winter, Christopher Steven Timperley, Ben Hermann, Jürgen Cito, Jonathan Bell, Michael Hilton, and Dirk Beyer. 2024. A Retrospective Study of one Decade of Artifact Evaluations. In *Software Engineering 2024, Fachtagung des GI-Fachbereichs Softwaretechnik, Linz, Austria, February 26 - March 1, 2024 (LNI, Vol. P-343)*, Rick Rabiser, Manuel Wimmer, Iris Groher, Andreas Wortmann, and Bianca Wiesmayr (Eds.). Gesellschaft für Informatik e.V., 95–96. doi:10.18420/SW2024_28