

# 1ShotBench: Demonstrating One-Shot Prompt Reproducibility for Autonomous Coding Agents

Neng Li\*  
University of Waterloo  
Waterloo, Canada  
n262li@uwaterloo.ca

Yijun Ge\*  
University of Waterloo  
Waterloo, Canada  
l2ge@uwaterloo.ca

Xueguang Ma  
University of Waterloo  
Waterloo, Canada  
x93ma@uwaterloo.ca

Jimmy Lin  
University of Waterloo  
Waterloo, Canada  
jimmylin@uwaterloo.ca

## Abstract

The growing use of LLM agents raises a reproducibility problem: given the same task specification, different agents may follow different workflows, consume different resources, and produce varying artifacts and outcomes. We introduce the concept of *One-Shot Prompt Reproducibility* (1SPR): the consistency of one-shot execution across different agents given a fixed specification, without iterative human guidance. To operationalize this, we present *1ShotBench*, a toolkit that repeatedly executes the same product requirement document (PRD) across isolated agent workspaces, capturing full traces, artifacts, and metadata, along with an LLM judge to evaluate results. We demonstrate a case study using software engineering tasks built around the Anserini information retrieval toolkit, though our system itself supports arbitrary tasks. *1ShotBench* allows users to compare runs side by side, inspect how execution and implementation paths diverge despite a shared specification, and analyze how workflow differences relate to cost, latency, and correctness. Our central contribution is a reproducibility-oriented lens for examining agentic software development.

## CCS Concepts

• Information systems → Evaluation of retrieval results.

## Keywords

Software Engineering, Benchmarking, Information Retrieval

## ACM Reference Format:

Neng Li, Yijun Ge, Xueguang Ma, and Jimmy Lin. 2026. 1ShotBench: Demonstrating One-Shot Prompt Reproducibility for Autonomous Coding Agents. In *Proceedings of the 35th ACM International Conference on Information and Knowledge Management (CIKM '26)*, November 07–11, 2026, Rome, Italy. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/3799682.3840271>

## 1 Introduction

Autonomous coding agents powered by large language models (LLMs) are increasingly used to build software from natural language specifications such as issue reports and product requirement documents (PRDs). In practice, two one-shot runs of the same specification can both succeed while differing substantially in tool usage,

implementation strategy, runtime, token consumption, and produced artifacts. Standard end-state metrics such as task completion, passing tests, or benchmark accuracy are necessary but do not expose this variation.

This paper argues that such variation should be treated as an important evaluation target. We call this perspective *One-Shot Prompt Reproducibility* (1SPR): given a fixed specification and no iterative human steering, how consistently do agents reproduce workflows, resource profiles, artifacts, and outcomes? The one-shot setting matters for two reasons: it isolates coding-agent behavior from operator intervention, since repeated re-prompting makes it hard to attribute an outcome to the agent versus the operator; and it reflects the long-term goal of delegating well-scoped tasks to agents with minimal supervision.

To support this line of inquiry, we present *1ShotBench*,<sup>1</sup> a toolkit for repeated, controlled execution of software tasks by coding agents. *1ShotBench* runs the same PRD across isolated workspaces using different underlying LLMs, capturing the resulting agent traces, resource usage, and artifacts for side-by-side inspection and judge-based evaluation. Agent workspaces can also be provisioned with reusable skills [1] that encode task-specific operational knowledge; our case study uses this mechanism with the popular information retrieval (IR) research toolkit Anserini [9, 13].

This paper makes three primary contributions. First, it formulates 1SPR as a reproducibility problem specific to autonomous coding agents. Second, it presents *1ShotBench* as an execution and inspection toolkit for repeated PRD-driven software tasks, providing an open, extensible playground for studying 1SPR. Third, it demonstrates the 1SPR approach with *1ShotBench* on Anserini-based IR development tasks.

## 2 Related Work

*1ShotBench* is not intended as a replacement for established agent benchmarks. Instead, it demonstrates a gap left by systems whose primary objective is final-task evaluation:

*General agent benchmarks.* AgentBench evaluates reasoning and decision making across eight interactive environments, including operating systems, databases, knowledge graphs, and games [10].  $\tau$ -bench emulates dynamic conversations between a user and a tool-using agent in policy-constrained domains, and its  $pass^k$  metric measures outcome reliability across repeated trials [14]; 1SPR (Section 3) instead decomposes consistency into four dimensions, so that behavior beyond simple task success remains visible. These benchmarks also pose fixed, curated interactive tasks; *1ShotBench*

\*Both authors contributed equally to this work.



This work is licensed under a Creative Commons Attribution 4.0 International License. *CIKM '26, Rome, Italy*

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2539-5/2026/11

<https://doi.org/10.1145/3799682.3840271>

<sup>1</sup><https://github.com/castorini/1shotbench>

supports running user-provided PRDs, letting researchers and practitioners choose an open-ended task and vary how much implementation detail the specification provides.

*Domain-specific agent benchmarks.* SWE-bench gives a model a real GitHub repository and issue description, then evaluates the resulting patch with repository tests [5]. PaperBench asks agents to reproduce machine-learning research from papers and grades the resulting artifacts with detailed rubrics [12]. These benchmarks substantially broaden the complexity of agent evaluation, but like AgentBench and  $\tau$ -bench they ask whether an agent can solve a given task, scored against benchmark-specific objectives. 1SPR asks a different question about the same kinds of tasks: if a fixed specification is implemented more than once by different agents, how consistent are their results?

### 3 One-Shot Prompt Reproducibility

We define 1SPR as the consistency of behavior and outcomes when different agents independently execute a fixed specification without iterative human guidance. In the terminology the ACM uses to distinguish reproducibility from replicability and repeatability [2, 7, 8], reproducibility is characterized as *different team, same experimental setup*: an independent team obtains the same result using the original author’s own artifacts. 1SPR fits this pattern if the PRD itself, rather than a shared codebase or pipeline, is treated as the reused artifact: different agents (the “independent teams”) each execute the same fixed PRD and are checked for similar outcomes.

This differs from the traditional sense of reproducibility as repeating one identical experiment over time: 1SPR instead holds the specification fixed and varies the agent. This is motivated by a practical concern as coding agents become common: different users’ agents, potentially built on different models or harnesses, should independently reach compatible outcomes for the same task. Our reported experiments hold the specification fixed and vary the model while holding the harness fixed; systematically studying same-model, same-harness run-to-run variation is a narrower question about LLM determinism rather than 1SPR itself, so we do not focus on it. Varying the harness as well is future work. This consistency can be examined along four dimensions:

- **Workflow consistency:** whether agents follow similar sequences of inspection, editing, tool calls, and debugging.
- **Resource consistency:** whether runtime, token usage, and cost remain within a predictable range.
- **Artifact consistency:** whether the generated code and its design converge or diverge across agents.
- **Outcome consistency:** whether different agents reach comparable functional results.

The idea of 1SPR also aligns with recent calls for reproducible and explainable evaluation of coding agents used for software engineering, especially around exposing thought-action-result style traces and interaction data [6].

## 4 1ShotBench System

1ShotBench operationalizes 1SPR by providing a toolkit to run different models on the same task and evaluate results. For workflow

consistency, we have structured event traces and readable logs to record exactly how a model executes the task, as well as a web interface to run and observe what different models are doing side by side in real time. For resource consistency, we document wall-clock duration, token usage details, and costs, which are provided in a summary report generated at the end of each run. For artifact consistency, all generated code from experiments is available for inspection and further analysis. For outcome consistency, an agentic judge evaluates each generated product against the original PRD via a custom skill, detailed further in Section 4.2.

### 4.1 Implementation Details

Our system currently supports Pi, an open-source agent harness covering a range of models, skills, tool integration, and structured event-stream execution [4]. We add an orchestration layer with isolated agent workspaces for repeatable experiments, where each agent can only access its own workspace and a skills directory with user-specified skills. Users supply a custom PRD or prompt and select the models to run the task with. Since the initial PRD may be LLM-generated, there is an optional stage for each model to rewrite the PRD for itself before implementing to reduce potential model bias due to prompt style; this stage was disabled for the results reported in Table 1, so all agents implemented the identical, unmodified PRD. The toolkit then invokes the Pi CLI command from each agent workspace and preserves the metadata needed to audit the run, such as the submitted prompt and command configuration.

Each agent runs in a separate workspace wrapped by a platform sandbox (sandbox-exec on macOS, Bubblewrap on Linux) that permits ordinary process activity but denies reads and writes to sibling agent workspaces, preventing an agent from inspecting or modifying another agent’s implementation.

Workspace configurations may declare reusable agent skills. Pre-flight verifies that required skills are available before a run, and the same skills are visible from each workspace. Our Anserini tasks provide skills for installing and verifying the Anserini fatjar, using its command-line interface, and following reproduction workflows. Skills make task-specific operational knowledge available without embedding that knowledge into 1ShotBench itself.

The execution interface supports cross-model comparisons: users can select multiple agent workspaces for a single invocation. As described in Section 3, our reported experiments vary the model across agent workspaces while holding the harness fixed; the same CLI can also be invoked repeatedly with an identical model selection, but we do not study that run-to-run variation systematically. Each completed run retains a structured summary, the generated workspace, resource usage, and judge outputs when available.

### 4.2 Agentic Judge

1ShotBench includes a judge system to evaluate applications produced by agent runs, with a Pi-harness-based judge that performs setup, collects browser evidence, and returns evidence-grounded verdicts. Evaluations begin by using the PRD to generate task-specific functional checks: concrete behaviors the generated application should exhibit, such as exposing a required index list or returning a retrieval metric. Separately, before testing, the same

**Table 1: Available run-level results for the three Anserini case studies. Correctness is the evidence-grounded evaluation score described in Section 4.2.**

| Task          | Provider    | Model                  | Duration (s)        | Input tokens | Output tokens | Cost   | Correctness |
|---------------|-------------|------------------------|---------------------|--------------|---------------|--------|-------------|
| frontend      | OpenAI      | GPT-5.5                | 383.7               | 705,589      | 13,914        | \$1.06 | 100%        |
|               | Anthropic   | Claude Opus 4.7        | 367.4               | 737,970      | 16,494        | \$0.99 | 100%        |
|               | Google      | Gemini 3.1 Pro Preview | 444.0               | 922,760      | 10,287        | \$0.68 | 100%        |
|               | DeepSeek    | DeepSeek V4 Pro        | 250.8               | 519,645      | 10,832        | \$0.02 | 100%        |
|               | Z.ai        | GLM-5.1                | 500.7               | 1,157,424    | 10,255        | \$0.38 | 100%        |
|               | Moonshot AI | Kimi K2.6              | 460.3               | 1,232,255    | 13,604        | \$0.33 | 100%        |
|               | MiniMax     | MiniMax-M3             | 933.1               | 2,401,901    | 23,793        | \$0.37 | 100%        |
|               | Xiaomi      | MiMo V2.5 Pro          | 404.9               | 377,623      | 5,810         | \$0.11 | 100%        |
| evaluator     | OpenAI      | GPT-5.5                | 411.4               | 883,731      | 17,083        | \$1.17 | 100%        |
|               | Anthropic   | Claude Opus 4.7        | 523.5               | 1,717,292    | 36,688        | \$2.16 | 100%        |
|               | Google      | Gemini 3.1 Pro Preview | 440.4               | 2,831,291    | 16,012        | \$1.10 | 100%        |
|               | DeepSeek    | DeepSeek V4 Pro        | 856.3               | 627,235      | 14,410        | \$0.03 | 100%        |
|               | Z.ai        | GLM-5.1                | 1005.5              | 3,273,449    | 27,712        | \$1.06 | 100%        |
|               | Moonshot AI | Kimi K2.6              | 488.9               | 1,267,864    | 16,621        | \$0.35 | 100%        |
|               | MiniMax     | MiniMax-M3             | 1800.1 <sup>†</sup> | 6,276,850    | 44,184        | \$0.92 | 100%        |
|               | Xiaomi      | MiMo V2.5 Pro          | 627.5               | 3,196,794    | 18,718        | \$0.79 | 100%        |
| nfcopus-repro | OpenAI      | GPT-5.5                | 748.9               | 1,648,167    | 20,975        | \$2.00 | 100%        |
|               | Anthropic   | Claude Opus 4.7        | 1228.6              | 2,458,774    | 40,953        | \$2.78 | 100%        |
|               | Google      | Gemini 3.1 Pro Preview | 312.6               | 1,644,082    | 16,979        | \$0.84 | 100%        |
|               | DeepSeek    | DeepSeek V4 Pro        | 1382.3              | 3,331,344    | 43,719        | \$0.07 | 100%        |
|               | Z.ai        | GLM-5.1                | 743.8               | 3,286,118    | 23,700        | \$1.10 | 100%        |
|               | Moonshot AI | Kimi K2.6              | 1222.8              | 2,927,119    | 28,215        | \$0.69 | 100%        |
|               | MiniMax     | MiniMax-M3             | 1800.1 <sup>†</sup> | 1,642,573    | 29,024        | \$0.34 | 80%         |
|               | Xiaomi      | MiMo V2.5 Pro          | 1800.2 <sup>†</sup> | 1,803,530    | 24,474        | \$0.49 | 60%         |

<sup>†</sup> Run reached the 1800-second timeout. Despite the timeout, agents could still produce partially correct or fully correct implementations.

judge model runs setup commands to get the application running based on the PRD, README files, and referenced skills.

For each functional check, the judge agent plans and executes browser actions live through a Playwright-based tool, guided by the check’s description and the app’s observed interface rather than a fixed script, taking a bounded follow-up pass if the initial evidence proves insufficient. It captures visible text, accessibility-tree snapshots, screenshots, interactive elements, and console and network errors as evidence. Finally, the judge assigns *pass*, *fail*, or *uncertain* using only this evidence. The reported correctness percentage is the fraction of functional checks judged *pass*; uncertain judgments count as not passed. Evidence packets, judgments, screenshots, setup results, and summary reports remain available for inspection, letting a human reviewer audit the verdict.

### 4.3 Workflow

A user starts from a single PRD, such as one for an Anserini-based application, and selects the agent models to implement it with. 1ShotBench executes that PRD across isolated agent workspaces, and an overview page reports each run’s status, resource usage, and links to its workspace and artifacts as runs complete. From there, a user can select an individual run, open its workspace to inspect the generated code and traces, or invoke the judge to evaluate and compare outcomes across implementations. This workflow is intended for researchers and practitioners working on agent evaluation, as well as those building and maintaining agent-assisted development pipelines where reproducibility, observability, and cost predictability are operational concerns.

## 5 Case Study

We study the four dimensions of 1SPR through a case study of three IR applications built around Anserini [13], with varying complexity. The simplest is the *frontend* task, which requests a search interface over the MS MARCO passage collection [11]. The harder *evaluator* task requests a local web interface for browsing prebuilt Lucene indexes and running reproducible retrieval and TREC evaluations. The hardest *nfcopus-repro* task requests a Dockerized, Render-deployable NFCorpus diagnostics workbench [3] with live Anserini-backed search, BM25 evaluation, observed-vs-expected metrics, command traces, and artifact paths; screenshots of this task’s generated artifacts are shown in Figure 1.

Table 1 reports the resources used by each model for each task, as well as the outcome evaluated by our judge.

Our work focuses on reproducibility rather than ranking: a single stochastic run per model and task cannot rigorously evaluate model strength. These values instead show why artifact and resource measurements are informative alongside outcome evaluation. Complete data and generated artifacts from our experiments are available in our GitHub repository.

Viewed through the four dimensions of 1SPR, the Anserini results show high outcome consistency across agents but weaker resource consistency. On the *frontend* and *evaluator* tasks, all eight agents achieved 100% correctness, yet cost varied by more than 50× and tokens by up to 10×. The hardest *nfcopus-repro* task reduced outcome consistency: six agents achieved 100% correctness, while two timeout-limited runs received 80% and 60%, connecting resource consistency to outcome consistency, since approaching

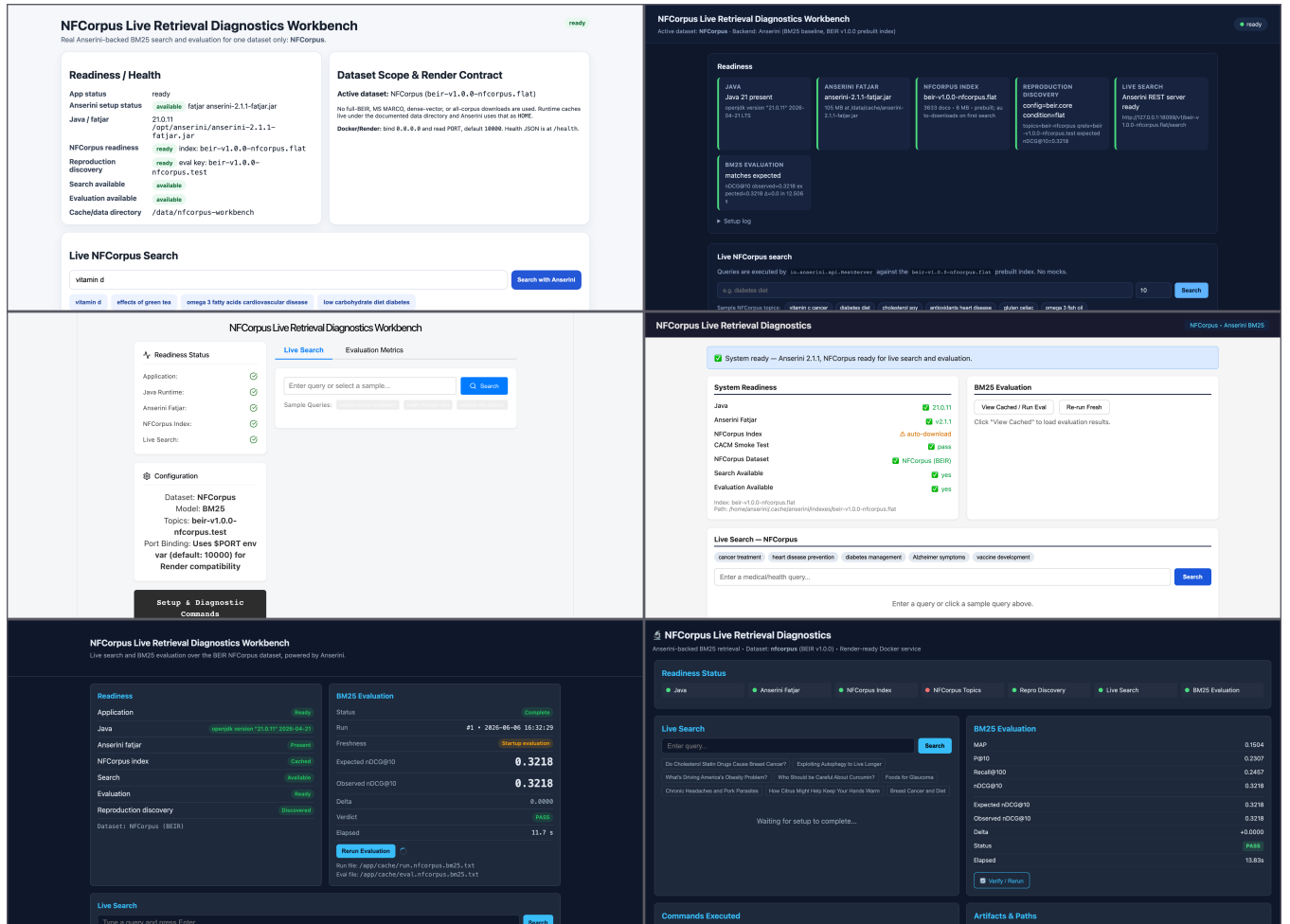


Figure 1: Six model-backed agents implementing the same *nfcopus-repro* PRD produce different web interfaces and operational states. Top to bottom: GPT-5.5 and Claude Opus 4.7; Gemini 3.1 Pro Preview and DeepSeek V4 Pro; Kimi K2.6 and GLM-5.1.

the execution budget can prevent an otherwise plausible implementation from fully satisfying the PRD. More broadly, the table illustrates why 1SPR treats workflow, resource, artifact, and outcome behavior as separate but related signals: final correctness can be reproducible even when cost and latency are not, while harder tasks can turn resource divergence into observable differences in final functionality.

## 6 Conclusion

This paper introduces *One-Shot Prompt Reproducibility* (1SPR): given a fixed specification and no iterative human steering, how consistently do coding agents reproduce workflows, resource profiles, artifacts, and outcomes? We operationalize 1SPR with *1ShotBench*, a toolkit that executes the same PRD across isolated agent workspaces, captures structured traces and resource usage, and evaluates the resulting artifacts with an agentic judge.

A natural extension is applying 1ShotBench to non-IR PRDs, such as web-application or data-pipeline specifications, to test how the four 1SPR dimensions generalize beyond the Anserini-based

tasks studied here. Outcome-consistency judgments currently rely solely on an LLM-based judge; we plan a human-validation study reporting agreement statistics to surface blind spots. Every agent in our experiments runs through the same Pi harness; we plan a controlled ablation to separate harness effects from model effects.

Our case study shows why this framing matters: even where every agent reached full correctness, cost and workflow diverged sharply, and on the hardest task, resource pressure itself caused some agents to fall short. 1SPR turns these connections into properties that can be examined systematically, and 1ShotBench lets researchers and practitioners observe them directly instead of inferring them from a single pass/fail number. Treating workflow, resource, artifact, and outcome consistency as separate, inspectable signals, rather than one score, is the step this paper takes toward evaluation that tracks agent behavior, not merely success.

## Acknowledgments

This research was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

## Generative AI Usage Disclosure

Generative AI tools assisted multiple parts of this work. During writing, an AI assistant helped draft, edit, and revise portions of this manuscript; all content was reviewed and verified by the authors. During development, AI coding assistants helped implement and debug parts of the 1ShotBench codebase, again under author review. Generative AI is also intrinsic to our experiments: the coding agents under study are themselves LLM-based, and our judge system uses an LLM to generate functional checks, propose setup commands, and evaluate outcomes, as described in Section 4.2.

## References

- [1] Anthropic. 2025. Equipping Agents for the Real World with Agent Skills. <https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills> Accessed: 2026-08-17.
- [2] Association for Computing Machinery. 2023. Artifact Review and Badging Version 1.1. <https://www.acm.org/publications/policies/artifact-review-and-badging-current> Accessed: 2026-08-23.
- [3] Vera Boteva, Demian Gholipour Ghalandari, Artem Sokolov, and Stefan Riezler. 2016. A Full-Text Learning to Rank Dataset for Medical Information Retrieval. In *Advances in Information Retrieval (Lecture Notes in Computer Science, Vol. 9626)*. Springer, 716–722. doi:10.1007/978-3-319-30671-1\_58
- [4] Earendil Inc. and Contributors. 2026. Pi Documentation. <https://pi.dev/docs/latest> Accessed: 2026-06-06.
- [5] 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 *Proceedings of the Twelfth International Conference on Learning Representations*. doi:10.48550/arXiv.2310.06770
- [6] Jingyue Li and André Storhaug. 2026. Reproducible, Explainable, and Effective Evaluations of Agentic AI for Software Engineering. *arXiv preprint arXiv:2604.01437* (2026). doi:10.48550/arXiv.2604.01437
- [7] Jimmy Lin. 2022. Building a Culture of Reproducibility in Academic Research. *arXiv preprint arXiv:2212.13534* (2022). doi:10.48550/arXiv.2212.13534
- [8] Jimmy Lin. 2026. Reproducibility in the Era of AI Agents. *ACM SIGIR Forum* 60, 1 (2026).
- [9] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2356–2362. doi:10.1145/3404835.3463238
- [10] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. 2024. AgentBench: Evaluating LLMs as Agents. In *Proceedings of the Twelfth International Conference on Learning Representations*. doi:10.48550/arXiv.2308.03688
- [11] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated Machine Reading Comprehension Dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches (CEUR Workshop Proceedings, Vol. 1773)*. doi:10.48550/arXiv.1611.09268
- [12] Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. 2025. PaperBench: Evaluating AI’s Ability to Replicate AI Research. *arXiv preprint arXiv:2504.01848* (2025). doi:10.48550/arXiv.2504.01848
- [13] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1253–1256. doi:10.1145/3077136.3080721
- [14] Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R. Narasimhan. 2025.  $\tau$ -bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. In *Proceedings of the Thirteenth International Conference on Learning Representations*. doi:10.48550/arXiv.2406.12045