Rillor

METHODS NOTE

Evaluation sets for agents: building questions with known answers

An agent training dataset and an agent evaluation set share one record: task, starting state, tools and a checkable known answer. How to write, verify and keep them apart.



Rillor

Contents

01 Summary

02 Background

03 Method

04 Findings

05 Implications

06 Limits of this analysis

07 Sources

Summary

Published agent benchmarks, the audits of them and evaluation guides show how to write tasks whose answers a program can check, and how to keep test tasks out of an agent training dataset.

- SWE-bench, tau-bench, OSWorld and GAIA store a known answer and a check with every task: tests that must pass in SWE-bench, a database end state in tau-bench, a final file or window state in OSWorld and a short exact answer in GAIA [1, 4, 8, 10]. Training sets such as SWE-Gym use the same task format [14].
- In SWE-bench and SWE-bench Verified, most candidate tasks failed screening. SWE-bench kept 2,294 tasks from about 90,000 pull requests [1], and annotators filtered out 68.3% of 1,699 screened SWE-bench samples to build the 500-task SWE-bench Verified [2].
- Each answer needs an independent check. GAIA validated a question only when two new annotators reached the original answer, and 68% passed as written [10]. A February 2026 audit still found material issues in 59.4% of 138 SWE-bench Verified problems that a frontier model failed to solve consistently [3].
- A set needs tasks where the right action is to stop, scored so that doing nothing fails. On tau-bench's airline tasks, an agent that returned empty responses scored 38% [11].
- Program checks of end states and tests are the safer default. An audit of ten agentic benchmarks found that WebArena's string matching and its unvalidated model judge overestimate agent performance by 1.4% to 5.2% [11], and NIST's January 2026 draft practices for automated benchmark evaluations list comparing a model judge's grading with human grading as an emerging practice [31].
- Evaluation tasks stay apart from training data through separate sources, withheld answers, canary strings (a marker in every test file that training pipelines can filter on), n-gram decontamination (removing training items that share long word sequences with test items) and tasks dated after a model's cutoff [10, 14, 17, 21, 23]. Paraphrased copies of test items evade n-gram checks [22].
- Small sets give wide intervals. A 90% pass rate on 20 tasks spans 69.9% to 97.2% at 95% confidence (Wilson interval, derived), and tasks that share a template call for clustered standard errors [27, 28].
- Answers and environments drift. Only 44.4% of ToolBench's API calls succeeded when StableToolBench's authors retested them in 2024 [20], and by July 2025 OSWorld's maintainers had fixed more than 300 reported issues, including changed websites and tasks whose dates had passed [9].

Background

An agent acts over many steps with tools, so its result is checked against an answer fixed before the run: a test suite, a database state, a file, a short string or the decision to stop. The answer key goes by several names. GAIA calls the answer the ground truth [10], and the human-written fix behind a SWE-bench task is known as the gold patch [3].

Teams often start from public benchmarks, and audits of those benchmarks keep finding flaws. A review of 445 language model benchmarks found that 12.3% relied only on convenience sampling and another 27.0% used it in part, 42.6% reused items from existing benchmarks, and 16.0% used uncertainty estimates or statistical tests to compare results (2025) [12]. Kapoor and colleagues found that most of 17 agent benchmarks lacked an appropriate held-out set, 7 of them with none [13]. The Agentic Benchmark Checklist (ABC) found outcome-validity flaws in seven of ten popular agentic benchmarks, task-validity issues in seven and reporting limits in all ten [11].

Public sets also leak into training. In February 2026 the developer that released SWE-bench Verified wrote that the set no longer measures frontier coding capabilities, and that "benchmarks sourced from publicly available material carry contamination risk, where training-data exposure can silently inflate scores" [3]. Every frontier model it tested could reproduce the gold patch or verbatim problem details for some tasks [3].

The same pipelines now produce training and test tasks. SWE-rebench (November 2025) offers over 21,000 interactive tasks "suitable for reinforcement learning" and a stream of fresh tasks for a contamination-free benchmark [16]. Training and evaluation tasks therefore share one record, and the work is to write tasks whose answers are right and checkable, then keep the evaluation split away from training.

This note is for teams writing an evaluation set for their own agent, or an agent training dataset with checkable answers. Audits of the public benchmarks themselves are covered in a companion report on what agent benchmarks measure.

Method

What was examined. The task records, construction steps, answer verification, checks, decontamination rules and upkeep of the main agent benchmarks: SWE-bench and SWE-bench Verified, tau-bench and tau2-bench, WebArena and WebArena-Verified, OSWorld and OSWorld-Verified, and GAIA. Also agent training sets built in the same shape (SWE-Gym, SWE-smith, SWE-rebench, AgentTuning, ToolBench), precedents from question answering (SQuAD 2.0, XSTest, FreshQA), decontamination methods, and the statistics of small evaluation sets.

Sources. Thirty-five sources, each opened on 8 November 2026 ([35] again on 12 November 2026) and read for the passages cited: 26 research papers read in full from arXiv, 10 of them published at a named conference, 2 posts by a model developer read in Internet Archive captures, a benchmark repository, a research lab's blog post, NIST's initial public draft on automated benchmark evaluations, 2 pages of an open evaluation framework's documentation, an engineering guide from a model developer and a cloud provider's documentation.

How. Each element of a task record and each procedure is described from the source that defines it. Figures are given with the version and date they come from, because benchmark audits change counts between versions. Figures marked "derived" are arithmetic on a source's own numbers or formula. The intervals use the 95% Wilson score interval that Bowyer and colleagues recommend for small sets [28]. No model is ranked.

Scope of the claims. The note describes published methods and results. It sets no required size, pass rate or tool, since no source reviewed sets one.

Findings

1. What a task with a known answer contains

In the benchmarks below, a task has five parts: what the agent is told, the state it starts in, the tools it may use, the end state or answer that counts as success, and the check that compares them.

Part	SWE-bench [1]	tau-bench [4]	OSWorld [8]	GAIA [10]
Instruction	A GitHub issue	A hidden instruction for a simulated user	A task in a desktop or web app	A question
Starting state	The repository before the fix	Users, orders or reservations in a database	A setup configuration	An attached file for some questions, otherwise none: only the answer is scored [10]
Known answer	Tests that fail before the fix and pass after	Ground-truth database writes	The final state of files and windows	A string, number or list
Check	Apply the patch, run the tests	Compare the final database with the goal	An execution-based script	Quasi exact match

SWE-bench. Each task has at least one fail-to-pass test, 40% have at least two, and a median of 51 additional tests check that prior functionality still works [1]. A patch passes when it applies and all of these tests pass [1].

tau-bench. A task pairs an instruction for the user simulation, hidden from agents, with "an annotation of the ground truth database write actions" and optionally the outputs the agent owes the user [4]. The instruction is written to guarantee "only one possible outcome under the domain policy" [4]. The reward multiplies an action score by an output score, each 0 or 1 [4]. The authors note that a reward of 1 "might be a necessary but not sufficient condition", since an agent might issue a return without explicit user confirmation [4].

WebArena. Its 812 tasks come from 241 templates, about 3.4 per template (derived) [6]. Answers are scored by exact match, by inclusion or by a language model judging semantic equivalence. Site outcomes are scored by programs that read the site's state, often its underlying database, through a database query, an API call or a JavaScript element selection [6].

OSWorld. Each of 369 tasks has "a detailed initial state setup configuration and a custom execution-based evaluation script" [8]. Across the set, 134 evaluation functions pair a getter, which extracts the modified file or window text from the final state, with an evaluator [8].

GAIA. Each answer is a few words, a number or a comma-separated list. "There is only one correct answer", so scoring is quasi exact match after normalization tied to the answer's type [10].

Frameworks. Inspect stores a sample as an input, an optional target, an id, metadata, a sandbox type, files copied into the sandbox and a setup script, read from CSV, JSON or JSON Lines [32]. A cloud provider's documentation adds a reference trajectory, "The expected tool use for the agent to satisfy the query" [35].

2. Where the tasks come from

Real failures first. An engineering guide from a model developer (January 2026) states that "20-50 simple tasks drawn from real failures is a great start", drawn from the manual checks run during development and, in production, from the bug tracker and support queue [34]. Converting user-reported failures into test cases "ensures your suite reflects actual usage" [34].

Real artifacts, filtered hard. SWE-bench collected about 90,000 pull requests and kept 2,294 tasks: merged pull requests that resolve an issue and change test files, with at least one fail-to-pass test and no installation or runtime errors [1]. OSWorld's infeasible tasks come from features "raised by real users" that are deprecated or do not exist [8].

Written by people who know the domain. GAIA's questions rest on sources of truth that were "often specified in the question to avoid ambiguity", such as trusted web pages unlikely to disappear [10]. tau-bench's authors write that manual annotation "is difficult and requires a deep understanding of both the domain and agent capabilities" [4].

The effort is measurable. GAIA took about two hours of annotator time per question, validation and repairs included [10]. OSWorld took 9 students about 1,800 hours over 3 months, with about 1 hour per task for the starting state, about 2 hours for its evaluation script and 400 hours for double-checking [8].

Generated, with proof. tau2-bench composes tasks from atomic scenarios defined by initialization, solution and assertion functions, a method its authors say "ensures provable correctness of tasks" [5]. Of 2,285 composed tasks, it kept 114 balanced across intents and numbers of subtasks [5]. SWE-smith generates tasks that break existing tests and built 50,000 from 128 repositories, with about 18 hours of human review for the environments [15].

Generated tasks carry their own risk. Bean and colleagues recommend vetting test items for contamination "when an LLM is used to generate task examples" [12].

3. One correct answer, verified

Design for one outcome. tau-bench's authors call it "the key challenge" to make each instruction lead to a unique database outcome [4]. They ran each retail task more than 40 times, polished instructions from the trajectories until "no ambiguities exist", and checked every task with zero or low success [4].

Independent answers. GAIA asked two new annotators to answer each question on their own, and validated a question only when all three agreed [10]. 68% of the questions were "good as is", and the rest were corrected or removed [10]. WebArena annotated each reference answer twice, with a third annotator settling disagreements [6]. In OSWorld two other authors cross-checked each task for feasibility, ambiguity and alignment with its source, and two authors who had not written a task attempted it [8].

Screening an existing set. For SWE-bench Verified (August 2024), engineers first hand-labeled 50 samples for annotator onboarding tests [2]. Then 93 developers labeled 1,699 samples, each three times [2]. 38.3% were flagged for underspecified problem statements, 61.1% for tests that may reject valid solutions, and 68.3% were filtered out [2]. SWE-bench Verified's 500 tasks were then drawn from the rest [2]. In 2026 the same developer audited 138 of those 500, the problems a frontier model failed to solve consistently over 64 runs, with at least six engineers per case [3]. 59.4% still had material issues, and the developer wrote that "Catching these problems took multiple extensive human labeling campaigns" [3].

Checks the ABC lists. Annotated ground truth is verified for correctness, each task is verified to be solvable, and the benchmark includes an oracle solver that solves every task automatically [11]. Pilot runs flag outliers: "if agents consistently fail on easy tasks, this may indicate that tasks are impossible", while success only on hard ones may indicate shortcuts [11]. The engineering guide calls a 0% pass rate across many trials "most often a signal of a broken task", describes a good task as one where two domain experts would independently reach the same verdict, and suggests a reference solution for each task that passes all graders [34]. NIST's draft asks reviewers whether a person given the same instructions would be confused, with several reviewers for consistency [31].

Answer keys carry errors too. Northcutt and colleagues estimate at least 3.3% label errors on average across ten test sets, at least 6% in the ImageNet validation set [29].

4. Tasks where the right action is to stop

Unanswerable questions. SQuAD 2.0 added 53,775 unanswerable questions whose paragraph holds a plausible answer [24]. A system scoring 86% F1 on the earlier version scored 66% F1 on SQuAD 2.0 [24]. WebArena cites that work for its unachievable tasks, labelled "N/A", where the agent is expected to give an equivalent response [6].

Infeasible tasks. OSWorld includes 30 infeasible tasks and scores 1 "if the agent accurately predicts failure for an infeasible task" [8]. tau-bench includes intentionally unattainable tasks, such as changing a non-refundable ticket, which agents are meant to recognize and reject [11].

The shortcut. On tau-bench's airline tasks, "a trivial agent that returns empty responses is considered successful on intentionally impossible tasks", and scores 38% [11]. OSWorld scores a stop case as passed only when the agent predicts failure [8], while ABC found that WebArena's model judge accepted an empty reply on its "N/A" tasks, so a trivial agent passed them [11].

Both sides. XSTest pairs 250 safe prompts that well-calibrated models should not refuse with 200 unsafe contrasts, because "even clearly safe prompts are refused if they use similar language to unsafe prompts" [25]. FreshQA adds a false-premise class whose questions must be rebutted [26]. The engineering guide advises testing "both the cases where a behavior should occur and where it shouldn't", warning that "One-sided evals create one-sided optimization" [34].

5. The check itself

String rules. ABC groups answer checks into whole-string matching, substring matching and model judges, and found that WebArena's string matching and its unvalidated model judge overestimate agent performance by 1.4% to 5.2% [11]. In the 445 benchmarks Bean and colleagues reviewed, exact matching was used at least partially by 81.3% and exclusively by 40.7% [12]. Inspect's built-in scorers run from substring inclusion to exact match after normalization, token-overlap F1 and model grading against guidance in the target [33].

State checks and tests. tau-bench's authors write that one possible outcome lets "subjective and noisy human judgments" give way to "simple and objective database state comparisons" [4]. OSWorld chose execution checks because evaluation that assumes a single solution for each task "wrongfully penalizes alternative correct solutions" [8]. WebArena-Verified removed model judging and substring matching "in favor of type-aware normalization and structural comparison" [7].

Path checks. A cloud provider's metrics compare the agent's tool calls with a reference trajectory: exact match in the same order, in-order match that allows extra calls, any-order match, precision and recall [35]. The engineering guide notes that "it's often better to grade what the agent produced, not the path it took" [34].

Model graders. Zheng and colleagues found strong model judges reached over 80% agreement with human preferences, the level of agreement between humans, and examined position, verbosity and self-enhancement biases [30]. ToolBench's grader agreed with human annotators 87.1% on pass rate and 80.3% on win rate [19]. NIST's draft lists comparing a model judge's grading with human grading, and computing interrater agreement, as an emerging practice [31]. The engineering guide recommends "deterministic graders where possible, LLM graders where necessary" [34].

The environment. ABC asks that leftover state be cleared between runs, that the agent be isolated from ground truth information and that the setup contain no flaw an agent could use to pass without doing the task [11]. NIST warns that agents can cheat by finding solution files left in the testing environment, or by "grader gaming", such as "removing the failing tests" [31]. The engineering guide starts each trial from a clean environment, since shared state "can cause correlated failures" [34].

6. Keeping the evaluation split out of training

What to hold out. Kapoor and colleagues write that "the more general the intended generality of the agent, the more the held-out set should differ from the training set", from new samples of the same distribution up to whole new domains [13]. The held-out set "would need to be kept secret" [13].

Withheld answers and marks. GAIA released a developer set of 166 annotated questions and kept the answers to 300 for its leaderboard [10]. BIG-bench put a canary string in every task file to help researchers filter its data out of training corpora collected from the web [23]. NIST's draft lists canary strings as an emerging practice [31]. The 2026 SWE-bench Verified post asks for care "in how datasets are posted (i.e. password protected)" and for "strict adherence to canary strings" [3].

Separate sources. SWE-Gym's 11 training repositories "are separate from those used in SWE-Bench to avoid contamination" [14]. SWE-smith left all 12 SWE-bench test repositories out [15].

Decontaminate the training data. Tülu 3 used 8-gram matching on prompts: a test item overlaps a training item when more than 50% of its tokens have 8-gram matches with it, and a training set whose items overlap more than 2% of an evaluation's items is contaminated [21]. It removed every training set contaminated with its unseen evaluations [21]. AgentTuning matched tokenized 10-grams with up to 4 tokens of mismatch and "found no evidence of data leakage" [18].

Where n-grams fail. Yang and colleagues found that paraphrase and translation "can easily bypass these decontamination measures", that 8% to 18% of the HumanEval benchmark overlaps two pretraining sets, and that a 13-billion-parameter model trained on rephrased items can "achieve drastically high performance" [22].

Date the tasks. LiveCodeBench tags each problem with a release date and scores newer models only on problems released after their cutoff [17]. SWE-rebench marks an evaluation as possibly contaminated when the issue predates the model's release [16]. NIST's draft notes the same use of data generated after the system [31].

7. How many tasks

No standard size. NIST's draft states that the number of items "affects whether results (statistically) support evaluation objectives" and that power analysis can support the choice [31]. The engineering guide's 20 to 50 tasks is a starting point [34].

Intervals on small sets. Bowyer and colleagues find that methods based on the central limit theorem "fail to provide adequate uncertainty estimates" for small benchmarks, usually with error bars that are too small, and recommend Wilson score or Bayesian intervals [28].

Tasks passed	Pass rate	95% Wilson interval (derived)
18 of 20	90%	69.9% to 97.2%

Tasks passed	Pass rate	95% Wilson interval (derived)
45 of 50	90%	78.6% to 95.7%
90 of 100	90%	82.6% to 94.5%
450 of 500	90%	87.1% to 92.3%

Source: computed with the interval recommended in [28].

Clusters. Miller recommends clustered standard errors when questions come in related groups, which "can be over 3X larger than naive standard errors", reporting the cluster count beside the question count, and comparing two models on question-level paired differences [27]. WebArena's 812 tasks from 241 templates are such clusters [6, 27].

Repeated trials. tau-bench trades quantity for quality, arguing that "a small set of high-quality tasks" run for multiple trials can reveal rich insights [4]. Its pass^k is "the chance that all k i.i.d. task trials are successful, averaged across tasks", and its agents scored below 25% on pass^8 in retail [4].

8. Versions and drift

Date the evidence. GAIA calls a question ambiguous when it names no version of a web page whose content differs across versions, and advises giving the page's date [10]. FreshQA sorts 600 questions by how fast their answers change, from never-changing to fast-changing, and its authors committed in November 2023 to updating the answers on a regular schedule [26].

Tools decay. StableToolBench (2024) found the status of 55.6% of ToolBench's online tools inconsistent, and only 44.4% of API calls successful [20]. Its fix is a virtual API server with a cache and API simulators [20]. ABC asks that the setup not change over time [11], and Kapoor and colleagues note that WebArena does not model drift [13].

Fixes cost time. OSWorld-Verified (July 2025) took about two months of a team of about ten to work through more than 300 pieces of feedback, mostly by changing evaluators, to keep changes to the tasks small [9]. The problems included changed web content, vague instructions, overly strict evaluators and tasks with future dates or reservations [9]. Changes are listed in a public fix log [9]. WebArena-Verified manually reviewed and corrected every task, reference answer and evaluator, and adds a 258-task hard subset [7]. tau2-bench reviewed the older domains for underspecification, overspecification and non-unique solutions and refined their instructions [5].

Retire what saturates. The engineering guide notes that "An eval at 100% tracks regressions but provides no signal for improvement", and calls a suite "a living artifact that needs ongoing attention and clear ownership" [34]. The top SWE-bench Verified score rose from 74.9% to 80.9% in the six months before February 2026 [3].

Bean and colleagues suggest procedurally generated tasks to keep benchmarks current [12].

9. From evaluation set to agent training dataset

Filter trajectories on the known answer. AgentTuning's AgentInstruct holds 1,866 verified interaction trajectories from six agent tasks [18]. For all tasks but one, trajectories were kept only at a final reward of 1, and training on unfiltered trajectories did significantly worse on held-in and held-out tasks [18]. Where a reference solution existed, a trajectory was kept when its answer matched the reference's [18].

Tests as rewards. SWE-Gym's 2,438 tasks each bundle a codebase, an executable environment, unit tests and a task in plain language [14]. Its test suites label sampled trajectories as successful or not [14]. Fine-tuning a 32-billion-parameter model on only 491 trajectories sampled using SWE-Gym raised resolution rates by 12.3 points on SWE-bench Lite and 13.6 points on SWE-bench Verified [14]. Tülu 3's reinforcement learning with verifiable rewards "only gets a reward if the model's completions are verified to be correct", with development and unseen evaluations kept apart [21].

Scale with checks. SWE-smith's 50,000 tasks each break existing tests [15]. SWE-rebench keeps tasks whose failing tests pass after the solution patch while the passing tests keep passing [16].

Without a known answer. ToolBench was generated automatically from 16,464 APIs in 49 categories, with instructions and solution paths written by a language model [19]. Its grader is a model, measured against human annotators [19]. Without a fixed answer, the check becomes a measured instrument, and its agreement rate is reported with the data.

Implications

For teams building an evaluation set.

1. Start from logged failures and real requests, then add written tasks with a named source behind each answer [1, 10, 34].
2. Store each task as instruction, starting state, tools, known answer and check, in a format an evaluation framework reads [32].
3. Prefer tests and end-state checks, and record any normalization rule with the answer [4, 7, 8].
4. Have a second person answer each task independently, and keep a reference solution that passes the check [10, 34].
5. Run agents on the set before trusting it, and review tasks that every agent fails or that only strong agents pass [11, 34].
6. Include stop cases scored on a specific response, and look-alikes where the agent should proceed [6, 8, 11, 25].
7. Measure any model grader against human labels and report its agreement [30, 31].
8. Report intervals, and clustered errors when tasks share a template or repository [27, 28].
9. Date the source of every answer, log each fix and retire saturated tasks [9, 10, 34].

For teams building an agent training dataset. Draw training tasks from sources disjoint from the evaluation set, as SWE-Gym and SWE-smith did [14, 15]. Decontaminate prompts against every evaluation with n-gram matching, and add a paraphrase check, since n-grams miss rephrased items [21, 22]. Keep only trajectories whose outcome passed the check [18]. Put a canary string in evaluation files, hold their answers back, and add tasks dated after the training data [3, 10, 17, 23].

For readers of a reported score. Ask how the answers were verified, how stop cases are scored, whether the tasks predate the model's training data, and how many tasks and clusters stand behind each figure [3, 11, 12, 27].

Rillor's custom datasets service scopes and builds training and evaluation data that does not exist yet, and keeps it current.

Limits of this analysis

- The benchmarks reviewed cover code, web, desktop, customer service and question answering. Agents in other settings may need checks these sources do not describe.
- Counts and scores change between benchmark versions. Each figure is given with its version and date, and later fixes may change them.
- No source sets a required size or pass rate for an evaluation set. The intervals marked "derived" treat tasks as independent, and clustered tasks widen them [27].
- No source reviewed measures how often agent training trajectories leak into agent evaluation sets in general. The evidence comes from checks of specific sets [3, 16, 17, 18].
- Human effort figures are in hours [8, 10, 15]. The only dollar figures are for generated tasks: SWE-smith cost $1,360 to build, and generating an issue cost 2.54 cents on average [15]. No source gives a cost per human-written task.
- No source shows that canary strings are honored by any particular training pipeline [3, 23, 31].
- Two sources are posts by a model developer about a benchmark subset it built [2, 3], one is another model developer's engineering guide [34], and three are framework or provider documentation that can change after the viewing date [32, 33, 35].

Sources

1. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? Carlos E. Jimenez, John Yang, Alexander Wettig and others, ICLR 2024, arXiv 2310.06770, version 3, 11 November 2024, viewed 8 November 2026. https://arxiv.org/abs/2310.06770

2. Introducing SWE-bench Verified. OpenAI, 13 August 2024, updated 24 February 2025, Internet Archive capture of 1 January 2026, viewed 8 November 2026. https://web.archive.org/web/20260101233457/https://openai.com/index/introducing-swe-bench-verified/

3. Why SWE-bench Verified no longer measures frontier coding capabilities. OpenAI, 23 February 2026, Internet Archive capture of 15 October 2026, viewed 8 November 2026. https://web.archive.org/web/20261015182204/https://openai.com/index/why-we-no-longer-evaluate-swe-bench-verified/

4. τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. Shunyu Yao, Noah Shinn, Pedram Razavi and Karthik Narasimhan, arXiv 2406.12045, 17 June 2024, viewed 8 November 2026. https://arxiv.org/abs/2406.12045

5. τ^2-Bench: Evaluating Conversational Agents in a Dual-Control Environment. Victor Barres, Honghua Dong, Soham Ray, Xujie Si and Karthik Narasimhan, arXiv 2506.07982, 9 June 2025, viewed 8 November 2026. https://arxiv.org/abs/2506.07982

6. WebArena: A Realistic Web Environment for Building Autonomous Agents. Shuyan Zhou, Frank F. Xu, Hao Zhu and others, ICLR 2024, arXiv 2307.13854, version 4, 16 April 2024, viewed 8 November 2026. https://arxiv.org/abs/2307.13854

7. WebArena-Verified, repository README. ServiceNow on GitHub, viewed 8 November 2026. https://github.com/ServiceNow/webarena-verified

8. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. Tianbao Xie, Danyang Zhang, Jixuan Chen and others, arXiv 2404.07972, version 2, 30 May 2024, viewed 8 November 2026. https://arxiv.org/abs/2404.07972

9. Introducing OSWorld-Verified. XLANG Lab, 28 July 2025, viewed 8 November 2026. https://xlang.ai/blog/osworld-verified

10. GAIA: a benchmark for General AI Assistants. Grégoire Mialon, Clémentine Fourrier, Craig Swift and others, arXiv 2311.12983, 21 November 2023, viewed 8 November 2026. https://arxiv.org/abs/2311.12983

11. Establishing Best Practices for Building Rigorous Agentic Benchmarks. Yuxuan Zhu, Tengjun Jin, Yada Pruksachatkun and others, arXiv 2507.02825, version 5, 7 August 2025, viewed 8 November 2026. https://arxiv.org/abs/2507.02825

12. Measuring what Matters: Construct Validity in Large Language Model Benchmarks. Andrew M. Bean, Ryan Othniel Kearns, Angelika Romanou and others, NeurIPS 2025 Datasets and Benchmarks Track, arXiv 2511.04703, 3 November 2025, viewed 8 November 2026. https://arxiv.org/abs/2511.04703

13. AI Agents That Matter. Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel and others, arXiv 2407.01502, 1 July 2024, viewed 8 November 2026. https://arxiv.org/abs/2407.01502

14. Training Software Engineering Agents and Verifiers with SWE-Gym. Jiayi Pan, Xingyao Wang, Graham Neubig and others, ICML 2025, arXiv 2412.21139, version 2, 6 June 2025, viewed 8 November 2026. https://arxiv.org/abs/2412.21139

15. SWE-smith: Scaling Data for Software Engineering Agents. John Yang, Kilian Lieret, Carlos E. Jimenez and others, arXiv 2504.21798, version 2, 21 May 2025, viewed 8 November 2026. https://arxiv.org/abs/2504.21798

16. SWE-rebench: An Automated Pipeline for Task Collection and Decontaminated Evaluation of Software Engineering Agents. Ibragim Badertdinov, Alexander Golubev, Maksim Nekrashevich and others, NeurIPS 2025, arXiv 2505.20411, version 2, 4 November 2025, viewed 8 November 2026. https://arxiv.org/abs/2505.20411

17. LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code. Naman Jain, King Han, Alex Gu and others, arXiv 2403.07974, version 2, 6 June 2024, viewed 8 November 2026. https://arxiv.org/abs/2403.07974

18. AgentTuning: Enabling Generalized Agent Abilities for LLMs. Aohan Zeng, Mingdao Liu, Rui Lu and others, arXiv 2310.12823, version 2, 22 October 2023, viewed 8 November 2026. https://arxiv.org/abs/2310.12823

19. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. Yujia Qin, Shihao Liang, Yining Ye and others, arXiv 2307.16789, version 2, 3 October 2023, viewed 8 November 2026. https://arxiv.org/abs/2307.16789

20. StableToolBench: Towards Stable Large-Scale Benchmarking on Tool Learning of Large Language Models. Zhicheng Guo, Sijie Cheng, Hao Wang and others, arXiv 2403.07714, version 5, 5 March 2025, viewed 8 November 2026. https://arxiv.org/abs/2403.07714

21. Tülu 3: Pushing Frontiers in Open Language Model Post-Training. Nathan Lambert, Jacob Morrison, Valentina Pyatkin and others, arXiv 2411.15124, version 5, 14 April 2025, viewed 8 November 2026. https://arxiv.org/abs/2411.15124

22. Rethinking Benchmark and Contamination for Language Models with Rephrased Samples. Shuo Yang, Wei-Lin Chiang, Lianmin Zheng and others, arXiv 2311.04850, version 2, 11 November 2023, viewed 8 November 2026. https://arxiv.org/abs/2311.04850

23. Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao and others, arXiv 2206.04615, version 3, 12 June 2023, viewed 8 November 2026. https://arxiv.org/abs/2206.04615

24. Know What You Don't Know: Unanswerable Questions for SQuAD. Pranav Rajpurkar, Robin Jia and Percy Liang, ACL 2018, arXiv 1806.03822, 11 June 2018, viewed 8 November 2026. https://arxiv.org/abs/1806.03822

25. XSTest: A Test Suite for Identifying Exaggerated Safety Behaviours in Large Language Models. Paul Röttger, Hannah Rose Kirk, Bertie Vidgen and others, NAACL 2024, arXiv 2308.01263, version 3, 1 April 2024, viewed 8 November 2026. https://arxiv.org/abs/2308.01263

26. FreshLLMs: Refreshing Large Language Models with Search Engine Augmentation. Tu Vu, Mohit Iyyer, Xuezhi Wang and others, arXiv 2310.03214, version 2, 22 November 2023, viewed 8 November 2026. https://arxiv.org/abs/2310.03214

27. Adding Error Bars to Evals: A Statistical Approach to Language Model Evaluations. Evan Miller, arXiv 2411.00640, 1 November 2024, viewed 8 November 2026. https://arxiv.org/abs/2411.00640

28. Position: Don't Use the CLT in LLM Evals With Fewer Than a Few Hundred Datapoints. Sam Bowyer, Laurence Aitchison and Desi R. Ivanova, ICML 2025, arXiv 2503.01747, version 3, 28 May 2025, viewed 8 November 2026. https://arxiv.org/abs/2503.01747

29. Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks. Curtis G. Northcutt, Anish Athalye and Jonas Mueller, NeurIPS 2021 Datasets and Benchmarks Track, arXiv 2103.14749, version 4, 7 November 2021, viewed 8 November 2026. https://arxiv.org/abs/2103.14749

30. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. Lianmin Zheng, Wei-Lin Chiang, Ying Sheng and others, NeurIPS 2023 Datasets and Benchmarks Track, arXiv 2306.05685, version 4, 24 December 2023, viewed 8 November 2026. https://arxiv.org/abs/2306.05685

31. Practices for Automated Benchmark Evaluations of Language Models, NIST AI 800-2, Initial Public Draft. National Institute of Standards and Technology, Center for AI Standards and Innovation, January 2026, viewed 8 November 2026. https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.800-2.ipd.pdf

32. Datasets, Inspect documentation. UK AI Security Institute, viewed 8 November 2026. https://inspect.aisi.org.uk/datasets.html

33. Scorers, Inspect documentation. UK AI Security Institute, viewed 8 November 2026. https://inspect.aisi.org.uk/scorers.html

34. Demystifying evals for AI agents. Mikaela Grace, Jeremy Hadfield, Rodrigo Olivares and Jiri De Jonghe, Anthropic Engineering, 9 January 2026, viewed 8 November 2026. https://www.anthropic.com/engineering/demystifying-evals-for-ai-agents

35. Evaluate Gen AI agents, Gemini Enterprise Agent Platform documentation. Google Cloud, last updated 10 November 2026, viewed 12 November 2026. https://docs.cloud.google.com/gemini-enterprise-agent-platform/models/evaluation-agents

About this report

Published by Rillor on 13 November 2026. Superintelligence (SI) tools assist research and drafting, and every figure cites its source. Rillor builds and secures SI systems, supplies the data they learn from and delivers the infrastructure they run on. We apply SI to financial and compute markets.

Cite as: Rillor. Evaluation sets for agents: building questions with known answers. 13 November 2026. https://rillor.com/insights/evaluation-sets-for-agents

Work with Rillor

Send the question the data should answer, the sources you know of, the coverage you need and how often it should update. Contact us at rillor.com/contact?type=custom-data or contact@rillor.com. More on data services: rillor.com/data