

How to run an AI benchmark yourself for under \$20

Keep an AI benchmark run under \$20 by fixing the question, sampling tasks, estimating tokens, using deterministic graders, and saving a reproducible run record.

Updated October 15, 2026

You can run a useful AI benchmark for under \$20 if you ask one narrow question, price the run before sending requests, and refuse to turn a small sample into a grand leaderboard.

The budget is enough for a smoke test, a regression check, or a focused comparison. It is not enough to establish every capability of a model. Treat the limit as a design constraint: spend tokens where they can change a decision.

Pick one decision

Choose a benchmark that matches a concrete choice. You might compare two models for tool-error recovery, check whether a prompt revision weakens honesty, or confirm that a new model still passes a coding gate. Write the claim you hope to support and the failure that would disprove it.

NIST's automated-evaluation guidance recommends defining objectives before selecting a benchmark. For a small run, this is also cost control. A precise objective stops you from paying for unrelated categories merely because they are included in a popular suite.

Read the licence and access instructions. Download only from the benchmark's primary source. Pin a released version or commit, and record the retrieval date. If the tasks are gated, follow the access terms rather than copying a convenient mirror.

Price tokens before the run

For a simple API evaluation, estimate:

$\text{tasks} \times \text{average input tokens} \times \text{input price per million} / 1,000,000$

plus

$\text{tasks} \times \text{average output tokens} \times \text{output price per million} / 1,000,000.$

Then add retries, judge calls, and tool-model calls. Current provider prices change, so read the official pricing page on the day you run. Note whether cached inputs, batch processing, or long-context rates apply; do not assume one advertised number covers every token.

Suppose 200 tasks average 2,000 input tokens and 500 output tokens. At \$2 per million input tokens and \$10 per million output tokens, the base model cost is \$1.80: \$0.80 for input and \$1.00 for output. A second model at the same rates doubles that to \$3.60. A 20 percent retry allowance brings it to \$4.32. That leaves room for a small judge sample while staying far below \$20.

Use the [eval run cost estimator](#) with the prices you actually see. It reports a base estimate and a retry-adjusted high estimate, not a quote.

Sample deliberately

If the full suite is too expensive, use a seeded sample. Stratify it when categories or difficulty bands matter. Twenty random easy tasks can miss the exact failure you care about; fifty tasks spread across relevant slices may teach you more.

Freeze the task IDs before running any model. Do not resample until you get a preferred outcome. Save the sampling seed and selection script. If you later run the full suite, distinguish that result from the pilot.

Small samples have wide uncertainty. Report the numerator and denominator—“42 of 50”—not only “84%.” Bootstrap intervals or a binomial interval are better than false decimals. Avoid ranking two systems when their difference is one task and run-to-run variation is similar.

Prefer deterministic grading

Exact matching, schema validation, unit tests, and simulator state checks are cheap and repeatable. Use them before paying another model to judge every output. Normalize only what the benchmark specification allows; aggressive cleanup can hide real formatting failures.

For open-ended answers, write the rubric first. Judge a subset during the pilot, and inspect disagreements by hand. If an LLM judge is necessary, pin its model and prompt and include its tokens in the budget. Randomize answer order for pairwise judging so position does not silently favor a system.

A benchmark like [Benchmark Tool Errors](#) can use simulator state to check retries and idempotency without a judge call for every case. [Benchmark Test Weakening](#) can run tests and inspect whether the agent changed protected test files. Deterministic signals make a small budget go further.

Put hard limits in the runner

Set maximum output tokens, request timeouts, a retry cap, and a total-spend stop. Log a failed request once; do not let an unbounded loop eat the budget. Start with five tasks and inspect raw requests, responses, and grades before releasing the full batch.

Concurrency saves wall time but not money. It can also amplify a bad prompt quickly. Ramp from one worker only after the smoke batch looks correct.

Keep secrets in environment variables. Never write an API key into the repository, task file, or run artifact. Strip sensitive production data before it reaches a third-party model.

Save a run another person can inspect

A useful result bundle contains:

- benchmark name, version, task IDs, and source URL;
- model provider, exact model ID, date, and parameters;
- system and user prompt templates;

- grader version and rubric, if any;
- per-task response, score, and reason;
- token counts, cost, latency, failures, and retries;
- aggregate score with an uncertainty interval.

HELM demonstrates the value of exposing scenario, prompt, metric, and reproducibility details. Your \$20 run can follow the same principle without copying its scale.

Spend the last dollars on verification

Do not automatically use the full budget. First inspect every failure in the pilot. A malformed request, wrong stop sequence, or broken answer parser invalidates the comparison. Fix the pipeline and start a new named run; do not quietly overwrite results.

If the result will influence a purchase, reserve budget for repetition. Two independent seeds on 100 tasks can reveal instability better than one 200-task run presented as exact. If a few high-risk cases matter most, run them more than once and publish that choice.

Finish by stating what the run does not show. A focused safety probe does not establish general safety. A code test does not establish browser skill. A cheap, honest result with full provenance is more useful than an impressive score whose configuration and cost cannot be reconstructed.

Frequently asked questions

Can a small sample give a trustworthy leaderboard?

Usually not. It can catch large failures and compare configurations, but report uncertainty and avoid fine-grained rank claims.

What costs are easy to miss?

Retries, judge-model calls, long tool traces, cached-input rules, and failed requests can all change the bill.

Sources

[OpenAI API pricing](https://platform.openai.com/pricing) — <https://platform.openai.com/pricing>

[NIST Practices for Automated Benchmark Evaluations of Language Models](https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.800-2.ipd.pdf) — <https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.800-2.ipd.pdf>

[Stanford HELM](https://crfm.stanford.edu/helm/index.html) — <https://crfm.stanford.edu/helm/index.html>