EVOCLAWBENCH: CAN AGENTS LEARN REUSABLE SKILLS FROM THEIR OWN RUNS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing agent benchmarks primarily test task completion, tool use, or skill utility, but do not isolate whether a runtime can convert evidence from its own runs into reusable skills that improve fresh executions after authoring overhead. We introduce **EvoClawBench**, a benchmark for this closed-loop skill-learning question on repeated, fixture-backed tasks. EvoClawBench compares direct execution without skills, PRESKILL authoring before execution, and POSTSKILL summarization from first-run evidence followed by a fresh second execution of the same task. The suite contains 100 tasks and 502 sub-problems across coding, data, office, security, operations, and domain-document workflows, with support for multiple agent runtimes. We evaluate five models under OpenClaw and nanobot on the full 100-task suite, report percentage-point changes against a skill-free rerun inside each run, and log execution status for timeouts, empty transcripts, and fallback replies (Section D). Self-authored skills change mean scores by at most about 3.7 points on the four stable model rows and by about 5–6 points on `Qwen3.6-Plus`, comparable to the -3.3 to $+3.1$-point gap between two skill-free executions of the same model; `DeepSeek-V4-Pro` is the exception, with nanobot skill-mode scores near 19% while skill-free reruns stay near 82%. Reuse is not cheaper: excluding the first run, the summary and second execution still use about $2.6\times$ the tokens of a direct run. These results indicate that learning reusable skills from an agent's own runs is selective and cost-sensitive, rather than an automatic benefit of adding skill authoring to an agent loop.

1 INTRODUCTION

Large language model agents are increasingly used to turn foundation models into interactive workers. Recent evaluations place agents in repositories, terminals, browsers, graphical interfaces, tool APIs, and executable environments (Fourney et al., 2024; Wang et al., 2025; Xu et al., 2025; Agashe et al., 2025; Cursor Research et al., 2026). As a result, the benchmark ecosystem has expanded rapidly. SWE-Bench and SWE-agent-style evaluations emphasize repository-level software engineering (Jimenez et al., 2024; Yang et al., 2024); Terminal-Bench, AppWorld, WebArena, and OSWorld measure command-line, app-world, browser, and desktop interaction (Zhou et al., 2024; Trivedi et al., 2024; Xie et al., 2024; Merrill et al., 2026); and SkillsBench evaluates skills as first-class artifacts across diverse tasks (Li et al., 2026). These benchmarks have made agent capability more measurable, but they leave a closed-loop learning question under-specified. They primarily test task completion, tool use, environment interaction, or the utility of curated and self-generated skills. They do not isolate the process in which the same agent runtime creates a skill, summarizes evidence from its own completed run, reuses that skill in a fresh execution, and pays the full token, cost, and time overhead. We therefore ask: *Can LLM agents learn reusable skills from their own runs?*

To answer this question, we introduce EvoClawBench, a benchmark designed around this question. EvoClawBench evaluates an agent on repeated, structured tasks that contain multiple related sub-problems. This design creates an opportunity for the agent to recognize shared patterns and, in the own-run condition, convert first-run evidence into reusable skills. The benchmark focuses on reusable skills because they are becoming a practical way to specialize LLM agents for recurring tool-use tasks. SkillsBench reports 84,192 skills collected within 136 days (Li et al., 2026), suggesting a shift from prompt engineering by individual users toward a marketplace of reusable procedural memory. Alongside stronger foundation models and richer harnesses, a complementary line of work treats

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

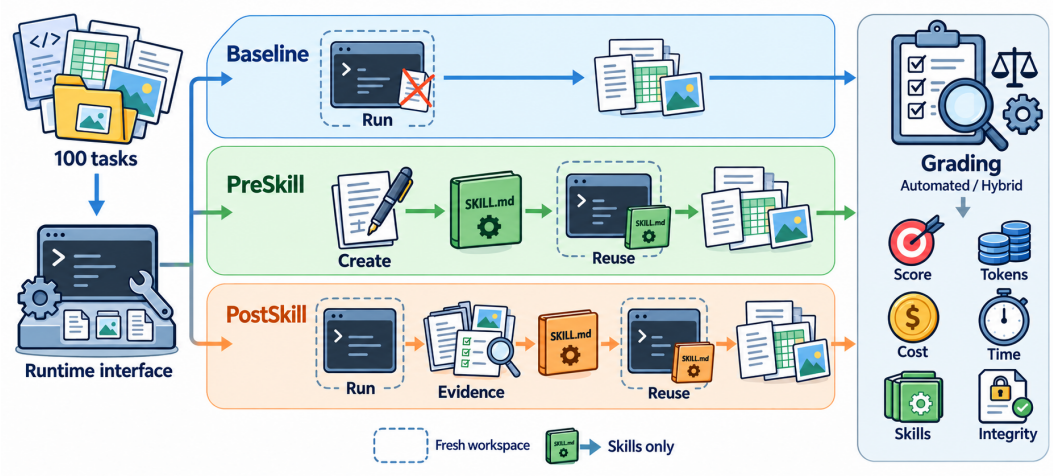


Figure 1: Overview for EvoClawBench. The task-suite panel summarizes the 100 non-sanity tasks in the official suite; Figure 2 breaks down their families and fixture formats. The pipeline routes tasks through a runtime-agnostic agent interface, compares BASELINE, PRESKILL, and POSTSKILL in fresh workspaces, and reports grader-backed quality, resource, skill, and mutation-integrity metrics.

procedural artifacts as external skills or learned procedures that can be derived, refined, reused, or moved into compact task-family state from agent experience (Yang et al., 2026b; Ni et al., 2026; Wang et al., 2026; Yang et al., 2026a; Xie et al., 2026). In this paper, we use *skills* for lightweight, inspectable procedural documents with optional scripts, templates, and references, because such artifacts can be installed, edited, and injected into an agent's context without retraining the model.

For each task, we compare three strategies. In BASELINE, the agent solves the task directly and is forbidden from creating or editing skills. In PRESKILL, the same model and runtime first create one or more task-specific skills, then a fresh execution workspace solves the task using only those generated skills. In POSTSKILL, the agent first solves the task without skills, receives a compact first-run context containing the prompt, grading summary, output previews, and transcript summary, then writes reusable skills that are copied into a fresh second execution. This protocol isolates two distinct forms of self-authored skill construction. PRESKILL is a pre-execution control that measures whether an agent can infer a useful reusable procedure before solving a task. POSTSKILL tests the title question in its within-task form: because the second execution reruns the same task and its first-run context includes grading feedback, it measures whether an agent can turn evidence from its own attempt into a procedure that improves a fresh attempt, not transfer to held-out sub-problems. Both are compared against BASELINE under execution-only and end-to-end metrics, so a skill workflow must both improve task score and justify its token, cost, and time overhead.

Our paper makes the following contributions:

- **Benchmark.** EvoClawBench is a benchmark testing whether LLM agents can learn reusable skills from self-authored skill workflows and their own first-run evidence. It covers 100 tasks and 502 sub-problems with fixtures, automated or hybrid graders, and runtime adapters.
- **Three-mode evaluation protocol.** We formalize BASELINE, PRESKILL, and POSTSKILL conditions, separating execution-only performance from end-to-end workflow cost and detecting skill mutation violations during reuse via before/after skill-file hashes.
- **Empirical findings.** We report the three modes with percentage-point deltas, a skill-free rerun inside each run, and execution-status logging. Full-suite results show small, mixed skill effects for most models, a persistent token overhead, and a model-specific `DeepSeek-V4-Pro` collapse under skill modes. Subset checks cover repeated runs, skill-content controls, and hybrid-judge robustness, each reported with its task subset and number of runs.
- **Open artifact.** We release the benchmark tasks, fixtures, graders, scripts, and reported result exports at `https://anonymous.4open.science/r/EvoClawBench-9380/`.

Table 1: Comparison of EvoClawBench with existing benchmarks. "Skill Cond." marks benchmarks that include agent skills, "Det. Verifier" marks deterministic (non-LLM) verification, and "Closed Loop" marks whether the same runtime creates and reuses skills in fresh runs.

Benchmark	Size	Skill Cond.	Tool Use	Det. Verifier	Closed Loop
SWE-Bench Verified	500	None	✓	✓	✗
Terminal-Bench 2.0	89	None	✓	✓	✗
WebArena	812	None	✓	✗	✗
AppWorld	750	None	✓	✓	✗
OSWorld	369	None	✓	✗	✗
SkillsBench	84	✓	✓	✓	✗
EvoClawBench	**100**	✓	✓	✓	✓

2 Related Work

Agent and software engineering benchmarks. Repository-level benchmarks test whether agents can modify realistic codebases to pass executable checks. SWE-Bench evaluates real GitHub issue resolution (Jimenez et al., 2024), while SWE-agent studies agent-computer interfaces for software engineering (Yang et al., 2024). Terminal-Bench focuses on realistic command-line tasks in isolated environments (Merrill et al., 2026). MLE-bench extends this line to end-to-end machine-learning engineering on Kaggle competitions (Chan et al., 2025). AgentBench and GAIA evaluate broader agentic reasoning, tool use, and multi-step problem solving across heterogeneous environments or questions (Liu et al., 2024; Mialon et al., 2024). These benchmarks measure task completion but do not isolate whether an agent can write reusable skills at runtime and benefit from them.

Web, GUI, and tool-use benchmarks. Several benchmarks focus on agents that act through browsers, operating systems, applications, or APIs. WebArena builds realistic web environments for long-horizon browser tasks (Zhou et al., 2024), and Mind2Web provides open-ended web tasks collected from real websites (Deng et al., 2023). WorkArena++ targets compositional knowledge-work workflows in ServiceNow (Boisvert et al., 2024), while OSWorld evaluates multimodal agents on real desktop-computer tasks (Xie et al., 2024). ToolLLM and ToolBench evaluate API-use capability at large scale (Qin et al., 2023). AppWorld and τ-Bench broaden the scope to controllable app worlds and tool-agent-user interaction settings (Trivedi et al., 2024; Yao et al., 2024). These environments stress planning and tool execution, but their evaluation conditions do not center on agent-authored artifacts that are reused in fresh runs.

Skill and procedural memory evaluation. Skills can be viewed as external procedural memory for agents, related to work on agent skills, reflection, experience accumulation, procedural memory, and open-ended skill acquisition (Zhang et al., 2025; Yao et al., 2023; Shinn et al., 2023; Zhao et al., 2024; Fang et al., 2026; Wang et al., 2023; Xu & Yan, 2026; Wu & Zhang, 2026). SkillsBench is closest to our setting because it evaluates skills as first-class artifacts, including curated and self-generated skill conditions (Li et al., 2026). Unlike SkillsBench, which evaluates the utility of curated Skills and a pre-solution self-generated Skills condition, our benchmark separates pre-execution skill authoring from post-run skill learning. In particular, PreSkill controls for skills authored before task execution, while PostSkill directly tests whether an agent can convert evidence from its own prior run into a reusable skill for later execution. Taken together, these lines of work motivate task and tool-use evaluation, whereas EvoClawBench makes the skill lifecycle itself the evaluated object: authoring, own-run distillation, reuse, mutation checks, and end-to-end cost. Table 1 summarizes the benchmark position at a high level. EvoClawBench tests whether agent-authored and own-run-derived skills improve task success after the full cost of creating and reusing them.

3 EvoClawBench Construction

EvoClawBench is constructed to test whether an agent can recognize repeated structure across sub-problems and encode that structure as a reusable skill. The benchmark implementation is organized around task definitions, workspace fixtures, runtime adapters, graders, and metric aggregation.

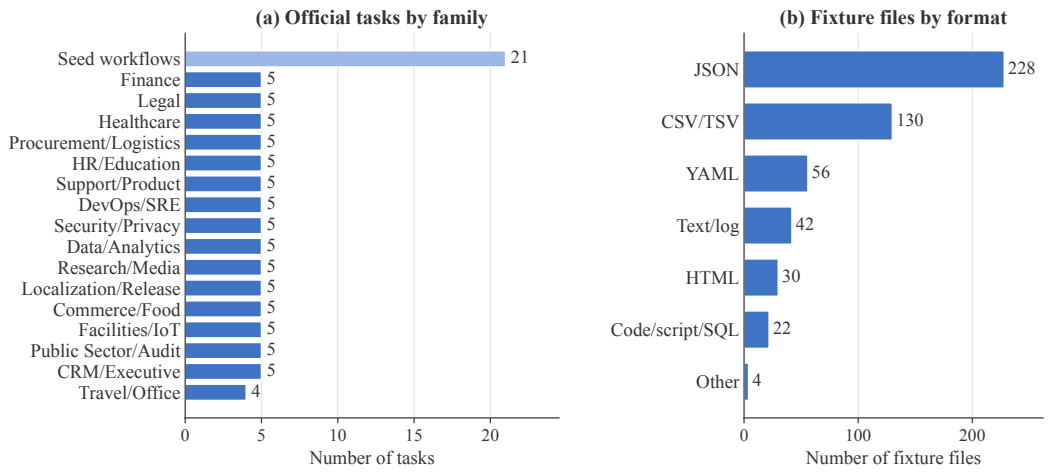


Figure 2: Official-suite task and fixture distributions in EvoClawBench. Panel (a) counts the 100 non-sanity tasks by seed or generated family. Panel (b) counts repository-local workspace fixture files by grouped file format. These metadata counts describe benchmark coverage, not model performance.

3.1 Task Suite

Tasks are specified as markdown files with YAML frontmatter, a natural-language prompt, expected behavior, sub-problem descriptions, grading criteria, and automated or hybrid grading logic. Each task typically contains five to ten structurally related sub-problems, creating repeated situations in which an agent can benefit from recording a general procedure rather than solving each instance independently. The repository also includes a sanity task used by the loader, but we exclude it from official suite counts. After this exclusion, the suite contains 100 tasks and 502 sub-problems. It covers recurring workflows such as data transformation, log analysis, API scaffolding, test generation, configuration migration, security review, document extraction, database operations, Excel analytics, web extraction, document generation, data pipelines, email and invoice processing, shell automation, CI generation, dependency audit, environment configuration, and metrics anomaly detection. The full suite extends these surfaces into harder repository-local families, including finance, legal, healthcare, procurement, HR, support, DevOps/SRE, privacy, localization, commerce, facilities, public-sector audit, CRM, and travel/office coordination. Many hard-mode fixtures are synthetic repository-local cases with competing evidence packets, stale revisions, and decoy records. This design reduces privacy risk while preserving repeated, artifact-producing workflows. Figure 2 summarizes official-suite coverage from task metadata rather than model results: the suite combines seed workflows, generated hard-mode families, and repository-local fixtures in multiple file formats.

3.2 Workspace and Fixtures

Isolation is per task phase: all sub-problems of a task are solved in one agent session inside one workspace, and every execution, skill-authoring, and summary phase starts from a fresh workspace. The benchmark copies input assets from its fixture directory into that workspace, and the agent must write grader-visible outputs to the expected paths. This design avoids relying on conversation-only answers and instead requires concrete artifacts, including JSON files, scripts, reports, Dockerfiles, CI files, and structured extraction outputs. Task definitions instruct agents not to modify input fixtures, while graders inspect output artifacts rather than trusting the agent's final natural-language answer.

The benchmark supports both local subprocess execution and Docker-based execution. The reported experiments use local subprocess execution with 32 parallel workers and one run per task and mode, without the Docker sandbox or its resource limits.

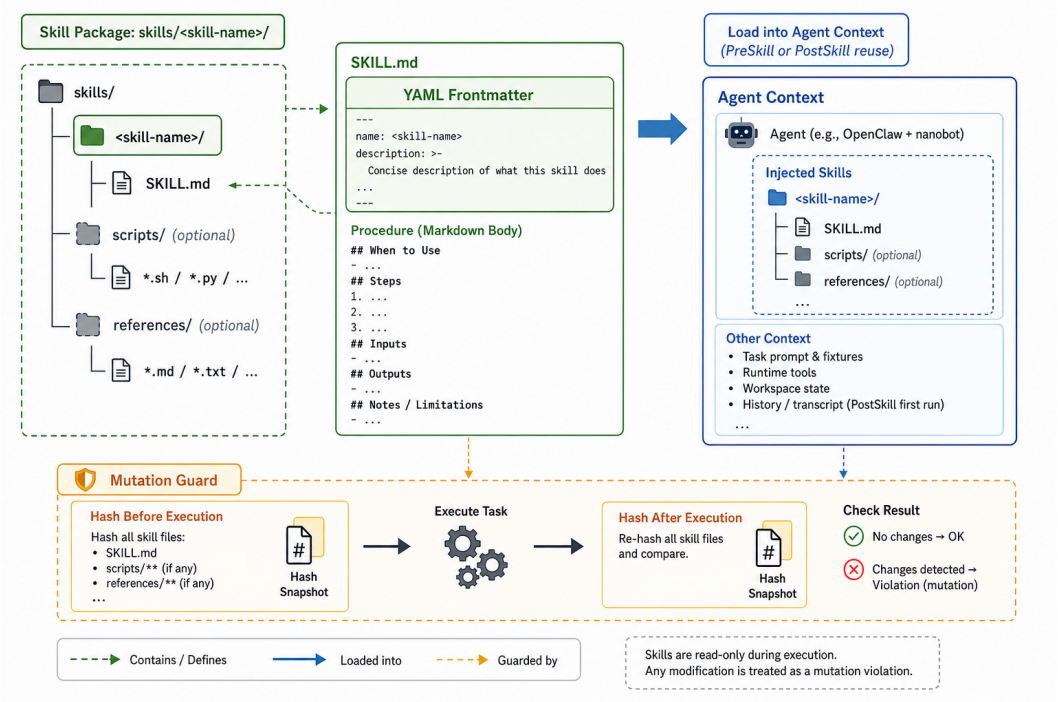


Figure 3: Structure of a reusable skill artifact in EvoClawBench. Generated skills use a `SKILL.md` document with optional supporting scripts or references, are injected into the agent context during reuse, and are protected by before/after mutation checks during execution.

3.3 Skill Authoring and Reuse

EvoClawBench represents reusable procedures with the standard `SKILL.md` format adopted by the supported runtimes (Figure 3). During skill-authoring phases, the workspace is seeded with a `skill-creator` bundle that instructs the agent to create a well-formed skill. The benchmark then scans generated skills under `skills/` in the authoring workspace and copies them into fresh execution workspaces for reuse. A generated skill must include `SKILL.md`, optional scripts and references are counted when present, and the seeded `skill-creator` bundle is excluded from created-skill counts. To prevent a reuse run from becoming another authoring run, EvoClawBench hashes all non-seeded skill files before and after execution. If a reuse execution adds, deletes, or edits a skill, the benchmark records a `skill_mutation_violation`. Thus, PreSkill and PostSkill execution may consult generated skills but may not revise them while solving the task.

3.4 Runtimes

EvoClawBench currently supports OpenClaw and nanobot. Each runtime is invoked as a separate command-line subprocess with the task workspace as its working directory and the assembled prompt as its input; Section A lists the exact invocation syntax for reproducibility. The runtime abstraction keeps task prompts, workspaces, grading, and metrics shared across claws, allowing the benchmark to compare the same model under different agent scaffolds. Additional runtimes can be added by implementing the same execution interface and usage extraction logic.

4 Evaluation Protocol

4.1 Three Execution Strategies

For a task t, model m, and runtime r, EvoClawBench evaluates the three strategies shown in Figure 4.

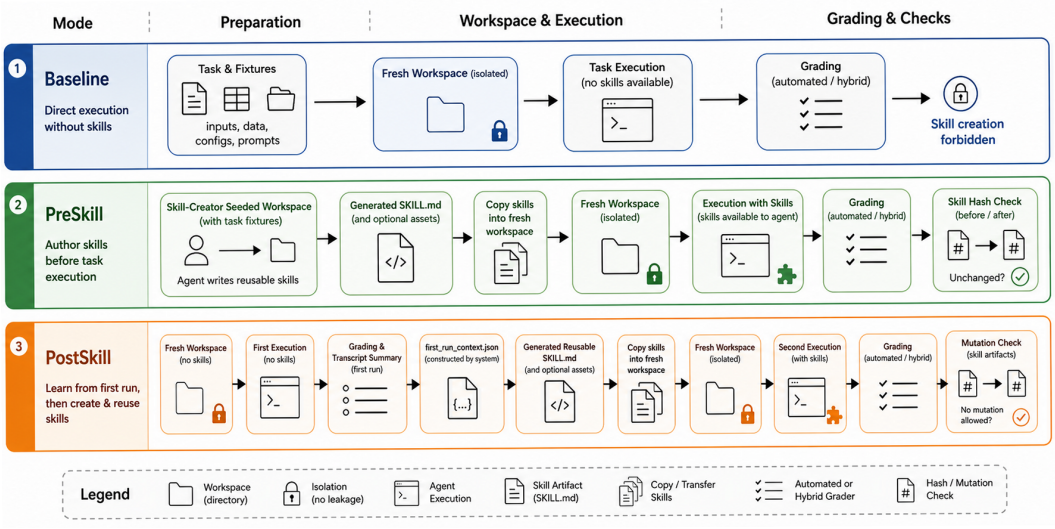


Figure 4: Three-mode evaluation protocol. BASELINE executes the task without skills; PRESKILL creates task-specific skills and reuses them in a fresh workspace; POSTSKILL distills skills from first-run evidence for a second run. Both reuse phases are checked for skill mutations.

BASELINE. The agent receives the task prompt and solves it directly. The prompt prefix forbids creating, editing, or deleting skills. This condition measures the agent's direct task-solving ability.

PRESKILL. The agent first enters a skill-authoring phase. It reads the skill-creator bundle and writes one or more task-specific skills under `skills/<skill-name>/SKILL.md`. The final task outputs are not graded during this phase. The benchmark then creates a fresh workspace, copies the generated skills into it, and runs the task execution phase with skill mutation disabled.

POSTSKILL. The agent first solves the task in a no-skill execution phase. The benchmark writes a compact file containing the task prompt, grading details, output summaries, and transcript summaries. The same runtime and model then summarize reusable skills from that first-run evidence. A fresh second workspace executes the same task using the summarized skills.

4.2 METRICS

Let S_b, S_p, and S_q denote mean task scores in percent for BASELINE, PRESKILL, and POSTSKILL. We report the percentage-point changes $\Delta_p = S_p - S_b$ and $\Delta_q = S_q - S_b$ rather than score ratios, which are unstable when S_b is close to zero. The POSTSKILL first execution uses the BASELINE prompt without skills, so its score $S_q^{(1)}$ is a skill-free rerun of the same tasks, and $S_q^{(1)} - S_b$ estimates run-to-run variation at the scale of the full suite.

EvoClawBench reports three resource scopes. **Execution-only** compares only the final execution phases: direct baseline execution, preskill reuse execution, and postskill second execution. **Reuse-only** adds the skill summary to the POSTSKILL second execution and treats the first execution as a sunk, baseline-equivalent run. **End-to-end** includes all workflow cost: skill generation for PRESKILL, first execution and skill summary for POSTSKILL, and the single direct execution for BASELINE.

For each scope, the benchmark reports token, cost, and time usage. Given baseline usage U_b and candidate usage U_x, efficiency gain is computed as:

$$E_x = \frac{U_b}{U_x}. \tag{1}$$

Values above 1.0 indicate that the skill workflow uses fewer resources than the baseline. The benchmark also reports created-skill counts, heuristic skill quality, and skill mutation violations. Execution-only metrics test whether a skill improves the final attempt, whereas end-to-end metrics

Table 2: Execution-only results over the 100 official tasks. Δ is the percentage-point change from BASELINE; Rerun is the skill-free POSTSKILL first execution minus BASELINE. Skills lists PRESKILL / POSTSKILL created-skill counts. Bold marks the highest mode score per row.

Runtime	Model	Tasks	BASELINE	PRESKILL	Δ_p	POSTSKILL	Δ_q	Rerun	Skills
nanobot	GPT-5.4	100	96.09	**96.70**	+0.60	96.13	+0.03	−0.42	100 / 100
	Qwen3.6-Plus	100	84.87	79.04	−5.83	78.61	−6.26	+1.35	96 / 89
	DeepSeek-V4-Pro	100	**82.47**	18.56	−63.91	19.31	−63.16	−0.54	88 / 84
	MiniMax-M2.7	100	90.88	92.83	+1.95	**94.44**	+3.56	+3.00	98 / 99
	GPT-5.4 mini	100	88.59	87.29	−1.30	**89.67**	+1.08	−3.28	99 / 81
OpenClaw	GPT-5.4	100	96.09	**96.25**	+0.16	95.52	−0.58	−0.39	99 / 98
	Qwen3.6-Plus	100	**86.20**	81.09	−5.11	84.05	−2.15	+1.11	97 / 92
	DeepSeek-V4-Pro	100	**85.05**	77.81	−7.24	79.42	−5.63	+0.00	96 / 91
	MiniMax-M2.7	100	91.94	93.87	+1.93	**94.58**	+2.64	+2.04	97 / 96
	GPT-5.4 mini	100	89.18	88.39	−0.79	**90.03**	+0.85	−3.20	99 / 83

test whether the full workflow justifies its authoring and summary overhead. A skill may thus preserve task score yet remain unattractive if it increases token use, cost, or wall time.

5 RESULTS

5.1 EXPERIMENTAL SETUP

We evaluate five models under OpenClaw 2026.3.22 and nanobot 0.1.4.post5 with local subprocess execution; Section A gives the commands. Scores are execution-only means over the 100 official tasks. The 12 hybrid tasks add an LLM-judge component to deterministic checks, while the other 88 tasks, including all 79 generated tasks, are graded by deterministic checks alone. Hybrid grading uses `MiniMax-M2.7` as the judge; the main-table result files predate judge-model serialization, so this setting is taken from the run commands, and Section 5.3 bounds judge sensitivity separately.

Execution logging. Each final execution is logged for runtime timeouts, fully empty assistant transcripts, and nanobot's fallback reply ("I've completed processing but have no response to give"), which the harness records as a successful run (Section D). No final execution has a fully empty transcript, each runtime-model-mode cell has at most 3 timeouts and 5 fallback replies among its 100 executions, and fallback executions score at most 1.3 points on average in any mode. Table 2 reports execution-only means over the 100 official tasks, with Δ denoting the percentage-point change from BASELINE, Rerun denoting the skill-free POSTSKILL first execution minus BASELINE, and Skills listing PRESKILL / POSTSKILL created-skill counts. Section E decomposes each row into its 21 seed tasks and 79 generated hard-mode tasks with bootstrap intervals.

5.2 EVALUATION FINDINGS

Finding 1: Strong models align across runtimes on the full suite. On the 100 official tasks, `GPT-5.4`, `MiniMax-M2.7`, and `GPT-5.4 mini` differ by at most 1.1 points between OpenClaw and nanobot on BASELINE and by at most 1.1 points on POSTSKILL; `Qwen3.6-Plus` differs by 1.3–2.1 points. On the 21 seed tasks alone, OpenClaw BASELINE meets or exceeds nanobot on the same tasks (Section E). Runtime therefore remains part of the experimental condition—it determines prompt wrapping, tool invocation, timeouts, and usage extraction—but the full-suite scores do not support a large inherent gap between these two scaffolds for the stronger models. The exception is `DeepSeek-V4-Pro` PRESKILL, where OpenClaw remains near 78% but nanobot falls to 19% (Table 2).

Finding 2: Skill effects are small relative to rerun variation. Excluding `DeepSeek-V4-Pro`, PRESKILL changes the mean score by -6.26 to $+1.93$ points and POSTSKILL by -6.26 to $+3.56$ points across the nine remaining rows in Table 2. The skill-free rerun inside the same runs differs from BASELINE by -3.28 to $+3.00$ points, and paired task-bootstrap 95% intervals for the non-`DeepSeek-V4-Pro` deltas include zero (Section E). For example, the largest gain, $+3.56$ for `MiniMax-M2.7` under POSTSKILL, is only 0.56 points above that run's own skill-free first execution. `Qwen3.6-Plus` loses about 5–6 points under both skill modes rather than improving over BASELINE.

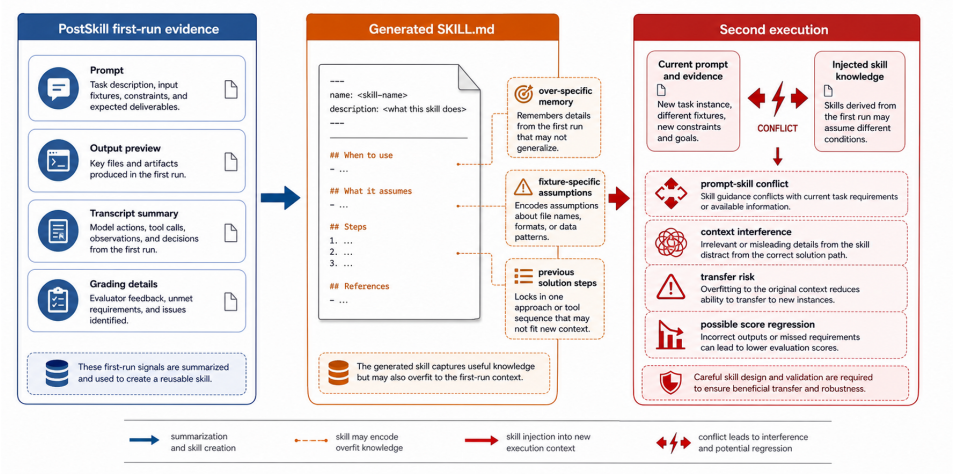


Figure 5: Schematic failure mode for experience-derived skills. In POSTSKILL, first-run evidence can help produce reusable procedural memory, but it can also encode over-specific assumptions, such as file names or formats, that interfere with the second execution context.

OpenClaw `DeepSeek-V4-Pro` loses 7.2 points under PRESKILL and 5.6 under POSTSKILL on the full suite, whereas nanobot on the same model collapses to about 19% under both skill modes while the skill-free rerun stays near 82%. These intervals reflect task sampling only; the repeated subset runs in Section 5.3 add run-to-run variation. Neither skill timing reliably improves over direct execution, and neither consistently outperforms the other across models or runtimes.

Finding 3: Reuse does not pay for itself in tokens. Token usage is available for OpenClaw only, because the nanobot adapter did not extract provider usage in these runs. On the 21 OpenClaw seed tasks where usage was logged (Section F), PRESKILL end-to-end token efficiency E is 0.21–0.41 across the five models. For models whose POSTSKILL phases completed on those seed tasks, end-to-end efficiency is 0.27–0.29; treating the first execution as a sunk baseline-equivalent run, reuse-only efficiency is still 0.38–0.39, i.e., about 2.6× BASELINE tokens. Even the skill-guided executions alone reach only 0.64–0.93 for POSTSKILL and 0.51–1.02 for PRESKILL, so repeated reuse cannot amortize authoring cost in tokens. The modest score changes in Table 2 therefore do not come with cheaper reuse. Output tokens are 2–4% of BASELINE tokens, so the overhead comes from reading context and skills.

Finding 4: `DeepSeek-V4-Pro` is the main skill-mode failure case. For `GPT-5.4`, `MiniMax-M2.7`, and `GPT-5.4 mini`, skill-mode scores stay within a few points of BASELINE and of the skill-free rerun. `DeepSeek-V4-Pro` is different: a matched 20-task nanobot subset still scores 70.36, 15.83, and 16.67 in the three modes with no fallback replies and no skill mutations (Section G), and the full-suite nanobot row in Table 2 stays near 19% under both skill modes while the skill-free rerun remains near 82%. OpenClaw on the same model stays near 78–79% under skill modes, so the collapse is severe under nanobot but partial under OpenClaw. The empty-scaffold control in Section 5.3 is a different condition, a normally executing pipeline with an intentionally empty skill, so it does not by itself explain the nanobot `DeepSeek-V4-Pro` skill-mode collapse.

Where executions complete, a delta still does not isolate skill content. Figure 5 illustrates one mechanism for POSTSKILL: first-run evidence can make a skill too specific to one execution context, so that reuse preserves an incidental assumption or skips checks that the second execution still requires, such as checks on file names or output formats.

5.3 SUBSET REPEATS, COST, ABLATION, AND JUDGE CHECKS

Focused subset experiments use OpenClaw with `GPT-5.4 mini` on the 12 hybrid tasks, all of which are seed tasks; the ablation uses four of them, and details are in Section C.

Table 3: Repeated runs and end-to-end cost on the 12-task hybrid subset (OpenClaw, `GPT-5.4 mini`). Scores are percentages. Break-even is the reuse count at which amortized wall time falls below BASELINE; neither skill mode breaks even on tokens or USD.

	Score				End-to-end cost (run 1)				
Mode	**Run 1**	**Run 2**	**Run 3**	**Mean ± s.d.**	**Tokens**	**USD**	**Time h**	**Requests**	**Break-even**
BASELINE	80.96	77.05	70.09	76.03 ± 5.51	1.94M	1.262	0.305	84	–
PRESKILL	79.77	69.61	80.75	76.71 ± 6.17	7.07M	3.745	0.689	270	14
POSTSKILL	79.21	68.54	76.11	74.62 ± 5.49	7.71M	4.128	0.833	306	9

Repeated runs. We executed the subset three times. Runs 2 and 3 changed the model route, endpoint, judge, timeout, and transcript handling, so they are operational repeats rather than controlled seeds, and 5 of their 72 cells had registration errors that are kept as zeros. Across runs, the mean and standard deviation are 76.03 ± 5.51 for BASELINE, 76.71 ± 6.17 for PRESKILL, and 74.62 ± 5.49 for POSTSKILL (Table 3). This 5–6-point spread matches the largest non-`DeepSeek-V4-Pro` deltas in Table 2, those of `Qwen3.6-Plus`, and exceeds all others.

Cost and amortization. In the first run, both skill workflows lower the score slightly (-1.19 and -1.75 points) while using 3.6–4.0$\times$ the BASELINE tokens (Table 3), i.e., -0.23 and -0.30 points per million extra tokens. Because a reuse execution alone is not cheaper in tokens or provider cost than a direct BASELINE run, neither workflow breaks even on these resources at any reuse count; wall-clock time breaks even only after 14 PRESKILL or 9 POSTSKILL reuses of the same skill under the local wall times observed in run 1 (Table 3, last column).

Skill-content ablation. On four tasks (`task_02`, `task_07`, `task_15`, and `task_21`), normal PRESKILL reuse changes the score by $+4.05$ points, an empty-skill scaffold by $+19.84$, and an irrelevant skill by $+1.52$; normal POSTSKILL reuse gives $+1.47$, and discarding the summarized skill gives -4.23 (Table 6). Because the empty scaffold moves the score more than any real skill, the deltas in Table 2 measure the combined effect of skill content, the reuse prompt, and the workspace reset, not skill content alone. The ablation is a single run on four tasks, so we use it for this attribution argument rather than as a causal estimate of skill content.

Judge robustness. Only the 12 hybrid tasks use an LLM judge, which is `MiniMax-M2.7` in both the main table and the subset run. Regrading the 36 hybrid task-mode outputs while holding the automated sub-scores fixed gives a Pearson correlation of 0.997 and a mean absolute score delta of 0.014 with `Qwen3.6-Plus` as the alternate judge, and 0.951 and 0.061 with `GPT-5.4 mini`, whose largest disagreement is -0.40 on `task_11` under BASELINE. Judge choice thus changes individual hybrid scores but not the subset trend, and it does not affect the 88 deterministically graded tasks that carry most of Table 2, including all 79 generated hard-mode tasks.

6 CONCLUSION

EvoClawBench evaluates the full lifecycle of procedural memory: skill creation, own-run skill distillation, reuse, and library maintenance. The current results suggest that agents are not yet reliable at this lifecycle: for most models, neither PRESKILL nor POSTSKILL moves scores beyond rerun variation, reuse does not recover its token cost, and Figure 5 shows how own-run summaries can overfit a second execution context. The skill-content ablation further shows that an empty skill scaffold can shift scores more than a real skill, so mode-level deltas conflate skill content with the reuse prompt and workspace reset. `DeepSeek-V4-Pro` remains the clearest counterexample, with large skill-mode drops on nanobot despite a stable skill-free rerun. Future skill systems should use selective creation policies and validation before reuse. Benchmark reports should audit execution status, include a skill-free rerun and a skill-content control, and report end-to-end resources before attributing score changes to skills or ranking runtimes.

AI USE STATEMENT

We used generative AI tools only to polish the writing of the manuscript. The models evaluated in Section 5 are objects of study rather than writing aids. All AI-polished text was reviewed by the authors. We take full responsibility for the final content of this work.

ETHICS STATEMENT

EvoClawBench is intended as a research benchmark for evaluating agent skill authoring, own-run skill distillation, and reuse, not as a deployment recipe for autonomous skill libraries. The benchmark uses repository-local task fixtures and synthetic examples whenever sensitive domains are represented; it does not involve human subjects, crowdworkers, or newly collected personal data. Several tasks simulate workflows involving privacy, security, legal, healthcare, finance, or public-sector records, but these are benchmark scenarios for controlled evaluation rather than operational recommendations. The main risk is dual use: better skill creation from an agent's own runs could make agents more effective at legitimate repeated work, but could also preserve unsafe procedures, overfit to private context, or automate harmful operational tasks if deployed without review. The benchmark therefore records skill mutation violations and separates execution-only quality from end-to-end cost, but it does not solve policy questions about which generated skills should be installed, shared, or trusted. Systems using self-authored skills should include human review, privacy checks, provenance tracking, and mechanisms for disabling or deleting unsafe skills.

REPRODUCIBILITY STATEMENT

The benchmark artifact used for this manuscript is the repository-local `evoclawbench/` package. It contains task definitions under `tasks/`, input fixtures under `assets/`, grading and metric code under `scripts/`, and reported OpenClaw and nanobot result exports under `results/`. `pyproject.toml` declares the package license as MIT. For anonymous review, the artifact is available at `https://anonymous.4open.science/r/EvoClawBench-9380/`. The reported task fixtures are benchmark cases rather than newly collected human-subject data, and the manuscript does not rely on private external datasets. Sections A and B give the exact mode prompts and the POSTSKILL first-run context, Sections H to J list every official task with its outputs and grading criteria, Sections D to G give execution-status counts, seed/generated splits with bootstrap intervals, phase-level usage, and matched reruns supporting Table 2, and Sections P and Q describe the result JSON schema and validity checks.

REFERENCES

Saaket Agashe, Kyle Wong, Vincent Tu, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent s2: A compositional generalist-specialist framework for computer use agents, 2025. URL https://arxiv.org/abs/2504.00906.

Léo Boisvert, Megh Thakkar, Maxime Gasse, Massimo Caccia, Thibault L De Chezelles, Quentin Cappart, Nicolas Chapados, Alexandre Lacoste, and Alexandre Drouin. Workarena++: Towards compositional planning and reasoning-based common knowledge work tasks. *Advances in Neural Information Processing Systems*, 37:5996–6051, 2024.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. Mlebench: Evaluating machine learning agents on machine learning engineering. 2025. URL https://arxiv.org/abs/2410.07095.

Cursor Research, Aaron Chan, Ahmed Shalaby, Alexander Wettig, Aman Sanger, Andrew Zhai, Anurag Ajay, Ashvin Nair, Charlie Snell, Chen Lu, et al. Composer 2 technical report. *arXiv preprint arXiv:2603.24477*, 2026.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory, 2026. URL https://arxiv.org/abs/2508.06433.

Adam Fourney, Gagan Bansal, Hussein Mozannar, Cheng Tan, Eduardo Salinas, Erkang, Zhu, Friederike Niedtner, Grace Proebsting, Griffin Bassman, Jack Gerrits, Jacob Alber, Peter Chang, Ricky Loynd, Robert West, Victor Dibia, Ahmed Awadallah, Ece Kamar, Rafah Hosn, and Saleema Amershi. Magentic-one: A generalist multi-agent system for solving complex tasks, 2024. URL https://arxiv.org/abs/2411.04468.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/edac78c3e300629acfe6cbe9ca88fb84-Paper-Conference.pdf.

Xiangyi Li, Wenbo Chen, Yimin Liu, Shenghan Zheng, Xiaokun Chen, Yifeng He, Yubo Li, Bingran You, Haotian Shen, Jiankai Sun, Shuyi Wang, Binxu Li, Qunhong Zeng, Di Wang, Xuandong Zhao, Yuanli Wang, Roey Ben Chaim, Zonglin Di, Yipeng Gao, Junwei He, Yizhuo He, Liqiang Jing, Luyang Kong, Xin Lan, Jiachen Li, Songlin Li, Yijiang Li, Yueqian Lin, Xinyi Liu, Xuanqing Liu, Haoran Lyu, Ze Ma, Bowei Wang, Runhui Wang, Tianyu Wang, Wengao Ye, Yue Zhang, Hanwen Xing, Yiqi Xue, Steven Dillmann, and Han chung Lee. Skillsbench: Benchmarking how well agent skills work across diverse tasks, 2026. URL https://arxiv.org/abs/2602.12670.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. 2024:52989–53046, 2024.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, E. Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Jenia Jitsev, Di Lu, Orfeas Menis Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Jesse Hu, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces, 2026. URL https://arxiv.org/abs/2601.11868.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. In *International Conference on Learning Representations*, volume 2024, pp. 9025–9049, 2024.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. Trace2skill: Distill trajectory-local lessons into transferable agent skills, 2026. URL https://arxiv.org/abs/2603.25158.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. 2023. URL https://arxiv.org/abs/2307.16789.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. 2023. URL https://arxiv.org/abs/2303.11366.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. Appworld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, 2024.

Chenxi Wang, Zhuoyun Yu, Xin Xie, Wuguannan Yao, Runnan Fang, Shuofei Qiao, Kexin Cao, Guozhou Zheng, Xiang Qi, Peng Zhang, and Shumin Deng. Skillx: Automatically constructing skill knowledge bases for agents, 2026. URL https://arxiv.org/abs/2604.04804.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. 2023. URL https://arxiv.org/abs/2305.16291.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Daniel Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An open platform for AI software developers as generalist agents. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 65882–65919, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/a4b6ad6b48850c0c331d1259fc66a69c-Paper-Conference.pdf.

Yaxiong Wu and Yongyue Zhang. Agent skills from the perspective of procedural memory: A survey. *Authorea Preprints*, 2026.

Haoyang Xie, Xinyuan Wang, Yancheng Wang, Puda Zhao, and Feng Ju. From history to state: Constant-context skill learning for llm agents, 2026. URL https://arxiv.org/abs/2605.05413.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. 2024. URL https://arxiv.org/abs/2404.07972.

Frank (Fangzheng) Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. TheAgentCompany: Benchmarking LLM agents on consequential real world tasks, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/0d744742f6fac4d1134c019b7cef3c8a-Paper-Datasets_and_Benchmarks_Track.pdf.

Renjun Xu and Yang Yan. Agent skills for large language models: Architecture, acquisition, security, and the path forward. 2026. URL https://arxiv.org/abs/2602.12430.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. *Advances in Neural Information Processing Systems*, 37:50528–50652, 2024.

Min Yang, Jinghua Piao, Xu Xia, Xiaochong Lan, Jiaju Chen, Yongshun Gong, and Yong Li. Skillmaster: Toward autonomous skill mastery in llm agents, 2026a. URL https://arxiv.org/abs/2605.08693.

Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, Bo Zhang, and Liang He. Autoskill: Experience-driven lifelong learning via skill self-evolution, 2026b. URL https://arxiv.org/abs/2603.01145.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. 2023. URL https://arxiv.org/abs/2210.03629.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-Bench: A benchmark for tool-agent-user interaction in real-world domains. 2024.

Barry Zhang, Keith Lazuka, and Mahesh Murag. Equipping agents for the real world with agent skills, 2025.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. 2024. doi: 10.1609/aaai.v38i17.29936. URL https://doi.org/10.1609/aaai.v38i17.29936.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. 2024:15585–15606, 2024.

A Mode Prompt Prefixes

EvoClawBench controls each execution condition by prepending a mode-specific prefix to the task prompt loaded from the task markdown file. The prefix text below reproduces the benchmark runtime helper wording; trailing blank lines are omitted and long lines are wrapped by LaTeX for typesetting. The assembled prompt is always the selected prefix followed by the task prompt; task prompts themselves remain stored in `evoclawbench/tasks/task_*.md`.

Each runtime receives the assembled prompt in one command-line call, with the task workspace as working directory. OpenClaw runs `openclaw agent --agent <id> --session-id <id> --message '<prompt>'`; nanobot runs `nanobot agent --workspace <dir> --config <path> --message '<prompt>'`. Both calls run as local subprocesses with a per-task timeout. The reported runs use OpenClaw 2026.3.22 and nanobot 0.1.4.post5 and are launched with `uv run scripts/benchmark.py --mode all --workers 32 --environment local --judge openai/MiniMax/MiniMax-M2.7`, adding `--runtime openclaw` or `--runtime nanobot`. Each model-runtime pair is one invocation that runs all three modes over all loaded tasks and writes one result file.

Baseline execution prefix — `BASELINE MODE:`

```
Complete the task directly. You must NOT create, edit, or delete any skills or
SKILL.md files. Solve each sub-problem independently from scratch, and focus on
producing the required grader-visible outputs.
```

Preskill authoring prefix — `PRESKILL AUTHOR MODE:`

```
Before solving the task, create a task-specific reusable skill. Read
‘skills/skill-creator/SKILL.md‘ and write one or more new skills under
‘skills/<skill-name>/SKILL.md‘ that would help an agent solve this task later. Do not
produce the final task outputs in ‘outputs/‘; this phase is only for skill authoring.
Do not replace or delete ‘skills/skill-creator‘.
```

Postskill summary prefix — `POSTSKILL SUMMARY MODE:`

```
Summarize a reusable task-specific skill from a completed first run. Read
‘.evoclawbench/first_run_context.json‘ for the first run prompt, grading details,
workspace output summary, and transcript summary. Create one or more skills under
‘skills/<skill-name>/SKILL.md‘ so a later agent can solve the same task more
accurately and efficiently. Do not redo the task and do not write final task outputs
in ‘outputs/‘. Do not replace or delete ‘skills/skill-creator‘.
```

Skill reuse execution prefix — `SKILL REUSE EXECUTION MODE:`

```
Complete the task using the existing skills in ‘skills/‘ when helpful. You must NOT
create, edit, or delete any skills or SKILL.md files during this execution phase. Focus
on producing the required grader-visible outputs and validate them before finishing.
```

Table 4: Prompt assembly by benchmark phase.

Mode	Phase	Prompt assembly	Graded behavior
baseline	baseline execution	Baseline prefix + task prompt	No skill creation or mutation; grade final outputs.
preskill_author	preskill authoring	Preskill author prefix + task prompt	Seed skill-creator; write skills; final task outputs are not graded.
preskill_execute	preskill reuse execution	Skill reuse prefix + task prompt	Copy generated skills into a fresh workspace; forbid skill mutation.
postskill_first	postskill first execution	Baseline prefix + task prompt	Solve without skills; grade outputs and write first-run context.
postskill_summary	postskill summary	Postskill summary prefix + task prompt	Read first-run context; write reusable skills; do not redo final outputs.
postskill_second	postskill reuse execution	Skill reuse prefix + task prompt	Copy summarized skills into a fresh workspace; forbid skill mutation.

B PostSkill First-Run Context

In PostSkill, the first execution is graded before any summary skill is written. The benchmark then writes `.evoclawbench/first_run_context.json` into the first workspace and copies that file into the summary workspace. This prevents the summary phase from relying on hidden state while still giving it compact evidence from the completed attempt.

Table 5: First-run context fields serialized for the summary phase.

Field	Purpose
task_id	Stable task identifier copied from the task markdown frontmatter.
task_name	Human-readable task name.
prompt	The original task prompt, without the mode prefix.
expected_behavior	The task-level expected behavior section.
grading_criteria	Parsed checklist items from the grading criteria section.
grade	Serialized grade result from the first no-skill execution.
outputs	Small summaries of files produced under the first-run workspace outputs directory.
transcript_summary	A compact role/text transcript summary capped by the benchmark helper.

The summary phase receives the same task prompt after the PostSkill summary prefix, but is told not to redo the task or write final outputs under `outputs/`. Its skills are copied into a fresh second execution workspace and mutation-checked before and after execution.

C Subset Audit Details

Table 3 in Section 5.3 reports the repeated runs, end-to-end cost, and break-even counts, and the table below gives the skill-content ablation. The cost, break-even, and judge-robustness numbers in Section 5.3 all come from the first of the three subset runs.

Table 6: Skill-content ablation on a 4-task representative subset. Scores and deltas are percentages; resources (tokens, USD, and requests) are end-to-end for each variant.

Variant	Score	Delta	Tokens	USD	Requests
BASELINE	54.24	0.00	0.80M	0.236	35
PRESKILL normal	58.28	4.05	2.80M	0.658	102
Empty scaffold	74.08	19.84	0.54M	0.196	24
Irrelevant skill	55.76	1.52	0.92M	0.241	38
POSTSKILL normal	55.70	1.47	2.79M	0.810	108
Summary, no skill	50.01	-4.23	2.59M	0.743	102

D EXECUTION AUDIT

For every final execution phase, the audit counts runtime timeouts, executions whose assistant messages are all empty, and nanobot executions whose transcript contains the fallback reply "I've completed processing but have no response to give". The harness records the last two as successful runs, although fallback executions average at most 1.3 points per mode. Counts are over the 100 official tasks and are summarized in Table 7. A failed judge call scores the LLM half of a hybrid task as 0; judge failures occur only in `DeepSeek-V4-Pro` runs on the 12 hybrid tasks. The `MiniMax-M2.7` nanobot POSTSKILL export contains 99 task objects because `task_85_smart_home_support_diagnosis` is missing; its mean and deltas in Table 2 use the 99 available tasks.

Table 7: Final-phase execution audit over the 100 official tasks. Each cell lists timeouts / fully empty assistant transcripts / fallback replies among the 100 executions.

Runtime	Model	BASELINE	PRESKILL	POSTSKILL
OpenClaw	`GPT-5.4`	1 / 0 / 0	0 / 0 / 0	2 / 0 / 0
OpenClaw	`Qwen3.6-Plus`	0 / 0 / 0	1 / 0 / 0	1 / 0 / 0
OpenClaw	`DeepSeek-V4-Pro`	0 / 0 / 0	2 / 0 / 0	1 / 0 / 0
OpenClaw	`MiniMax-M2.7`	1 / 0 / 0	1 / 0 / 0	1 / 0 / 0
OpenClaw	`GPT-5.4 mini`	1 / 0 / 0	0 / 0 / 0	2 / 0 / 0
nanobot	`GPT-5.4`	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0
nanobot	`Qwen3.6-Plus`	0 / 0 / 4	0 / 0 / 4	0 / 0 / 5
nanobot	`DeepSeek-V4-Pro`	0 / 0 / 2	0 / 0 / 5	0 / 0 / 3
nanobot	`MiniMax-M2.7`	3 / 0 / 0	0 / 0 / 1	0 / 0 / 0
nanobot	`GPT-5.4 mini`	0 / 0 / 0	0 / 0 / 0	0 / 0 / 0

E SEED AND GENERATED SPLITS WITH BOOTSTRAP INTERVALS

Table 8 separates the 21 seed tasks from the 79 generated hard-mode tasks for each runtime and mode. For every row, the 100-task mean in Table 2 equals $0.21 \times$ seed mean $+ 0.79 \times$ generated mean for BASELINE, PRESKILL, and POSTSKILL. Table 9 reports paired task-bootstrap 95% intervals (10,000 resamples of tasks) for the deltas in Table 2. The intervals capture task-sampling uncertainty only, not run-to-run variation, which Table 3 estimates separately.

Table 8: Execution-only mean scores (percent) by task split.

		Seed (21 tasks)			Generated (79 tasks)		
Runtime	**Model**	**B**	**P**	**Q**	**B**	**P**	**Q**
OpenClaw	GPT-5.4	84.85	82.16	84.85	99.08	100.00	98.35
OpenClaw	Qwen3.6-Plus	87.37	78.83	87.22	85.90	81.70	83.20
OpenClaw	DeepSeek-V4-Pro	91.36	67.68	91.36	83.37	80.51	76.24
OpenClaw	MiniMax-M2.7	86.04	81.74	87.11	93.51	97.09	96.57
OpenClaw	GPT-5.4 mini	87.16	88.65	88.37	89.72	88.32	90.47
nanobot	GPT-5.4	84.85	84.28	87.75	99.08	100.00	98.35
nanobot	Qwen3.6-Plus	81.02	69.05	61.31	85.90	81.70	83.20
nanobot	DeepSeek-V4-Pro	84.74	13.54	20.29	81.87	19.90	19.05
nanobot	MiniMax-M2.7	80.96	76.80	86.53	93.51	97.09	96.54
nanobot	GPT-5.4 mini	84.38	83.39	86.62	89.72	88.32	90.47

Table 9: Percentage-point deltas from BASELINE with paired task-bootstrap 95% intervals. Rerun is the skill-free POSTSKILL first execution minus BASELINE. Intervals for DeepSeek-V4-Pro exclude zero; other rows include zero.

Runtime	**Model**	**Tasks**	Δ_p **[95% CI]**	Δ_q **[95% CI]**	**Rerun [95% CI]**
OpenClaw	GPT-5.4	100	$+0.16\ [-0.31, +1.99]$	$-0.58\ [-2.15, +2.18]$	$-0.39\ [-1.97, +1.27]$
OpenClaw	Qwen3.6-Plus	100	$-5.11\ [-11.42, +1.26]$	$-2.15\ [-8.33, +3.96]$	$+1.11\ [-3.90, +6.12]$
OpenClaw	DeepSeek-V4-Pro	100	$-7.24\ [-14.26, -0.76]$	$-5.63\ [-12.86, +1.24]$	$+0.00\ [-3.33, +3.33]$
OpenClaw	MiniMax-M2.7	100	$+1.93\ [-4.05, +7.97]$	$+2.64\ [-1.93, +9.62]$	$+2.04\ [-2.85, +9.13]$
OpenClaw	GPT-5.4 mini	100	$-0.79\ [-5.55, +3.05]$	$+0.85\ [-2.80, +5.00]$	$-3.20\ [-8.21, +1.33]$
nanobot	GPT-5.4	100	$+0.60\ [-0.31, +1.99]$	$+0.03\ [-2.15, +2.18]$	$-0.42\ [-1.97, +1.27]$
nanobot	Qwen3.6-Plus	100	$-5.83\ [-12.00, +1.50]$	$-6.26\ [-13.56, +1.22]$	$+1.35\ [-8.06, +10.66]$
nanobot	DeepSeek-V4-Pro	100	$-63.91\ [-71.94, -55.24]$	$-63.16\ [-71.29, -54.38]$	$-0.54\ [-3.50, +2.50]$
nanobot	MiniMax-M2.7	100	$+1.95\ [-4.05, +7.97]$	$+3.56\ [-1.93, +9.62]$	$+3.00\ [-2.85, +9.13]$
nanobot	GPT-5.4 mini	100	$-1.30\ [-5.55, +3.05]$	$+1.08\ [-2.80, +5.00]$	$-3.28\ [-8.21, +1.33]$

F PHASE-LEVEL TOKEN USAGE

Table 10 breaks OpenClaw token usage on the 21 seed tasks into workflow phases, with input, output, and cache-read tokens reported separately because providers price them differently. Provider usage was not captured for some POSTSKILL phases in the stored OpenClaw logs and is marked n/a below. The nanobot adapter did not extract provider usage in the reported runs, so nanobot rows are omitted. Efficiency ratios in Finding 3 divide BASELINE total tokens by the sum of the relevant phases: PRESKILL execution for PRESKILL execution-only, authoring plus execution for PRESKILL end-to-end, second execution for POSTSKILL second-only, summary plus second execution for reuse-only, and all three phases for POSTSKILL end-to-end.

Table 10: OpenClaw phase-level usage on the 21 seed tasks. Tokens are in millions, and time is wall-clock hours summed over the 21 tasks for each phase.

Model	**Phase**	**Input**	**Output**	**Cache read**	**Total**	**Time h**
GPT-5.4	BASELINE	2.44	0.103	2.26	4.80	1.27
	PRESKILL authoring	3.90	0.111	3.11	7.12	1.36
	PRESKILL execution	3.57	0.105	2.53	6.21	1.90
	POSTSKILL first execution	n/a	n/a	n/a	n/a	0.88
	POSTSKILL summary	n/a	n/a	n/a	n/a	1.68
	POSTSKILL second execution	n/a	n/a	n/a	n/a	1.70
Qwen3.6-Plus	BASELINE	4.90	0.190	0.00	5.09	1.68
	PRESKILL authoring	8.55	0.244	0.00	8.79	1.97
	PRESKILL execution	4.76	0.205	0.00	4.97	2.09
	POSTSKILL first execution	4.24	0.171	0.00	4.41	1.63
	POSTSKILL summary	7.28	0.133	0.00	7.41	1.92
	POSTSKILL second execution	5.29	0.193	0.00	5.49	2.22
DeepSeek-V4-Pro	BASELINE	1.27	0.224	3.69	5.18	1.76

Model	Phase	Input	Output	Cache read	Total	Time h
	PRESKILL authoring	1.51	0.193	5.82	7.52	1.69
	PRESKILL execution	1.25	0.241	3.69	5.18	2.27
	POSTSKILL first execution	n/a	n/a	n/a	n/a	0.89
	POSTSKILL summary	n/a	n/a	n/a	n/a	1.71
	POSTSKILL second execution	n/a	n/a	n/a	n/a	1.72
MiniMax-M2.7	BASELINE	0.58	0.139	3.20	3.92	1.41
	PRESKILL authoring	1.03	0.148	9.52	10.70	1.49
	PRESKILL execution	0.69	0.180	6.80	7.67	2.10
	POSTSKILL first execution	0.60	0.122	3.33	4.05	1.24
	POSTSKILL summary	0.76	0.067	3.47	4.30	1.71
	POSTSKILL second execution	0.69	0.165	5.29	6.14	2.40
GPT-5.4 mini	BASELINE	2.04	0.106	1.33	3.48	0.99
	PRESKILL authoring	3.98	0.170	2.44	6.59	1.23
	PRESKILL execution	2.33	0.105	1.91	4.35	1.71
	POSTSKILL first execution	2.02	0.104	1.12	3.25	1.00
	POSTSKILL summary	3.00	0.076	1.63	4.70	1.74
	POSTSKILL second execution	2.39	0.109	1.81	4.30	1.80

G MATCHED DEEPSEEK-V4-PRO RERUN

To test whether the `DeepSeek-V4-Pro` collapse is tied to the fallback reply, we reran nanobot `DeepSeek-V4-Pro` in `mode=all` with 32 workers on 20 tasks: five seed tasks (`task_02`, `07`, `14`, `17`, `21`) and 15 generated tasks spanning 14 of the 16 generated families. The rerun used a different model route and a newer harness build, so it is a diagnostic rather than a replacement row. No execution returned the fallback reply, and no reuse execution mutated a skill file; nevertheless both skill modes stayed far below BASELINE. The diagnostic subset is reported separately from the full-suite rows in Table 2.

Table 11: Matched 20-task nanobot `DeepSeek-V4-Pro` rerun.

	BASELINE	PRESKILL	POSTSKILL
Score (%)	70.36	15.83	16.67
Completed / timed out	17 / 3	16 / 4	18 / 2
Fallback replies	0	0	0
Created skills	–	10	1

H OFFICIAL TASK INVENTORY

The loader currently finds 101 task files. The official benchmark size reported in the paper excludes `task_00_sanity`, leaving 100 tasks and 502 sub-problems. The table uses ASCII-safe names because the review build uses `pdflatex` without CJK font configuration.

Table 12: Official task inventory, excluding `task_00_sanity`.

Task	Name	Family/category	Grade	Sub.	Files	Ext.
task_01	Batch Data Transform	Seed / data transformation	automated	6	6	csv, json, tsv, xml, yaml
task_02	Log Analysis	Seed / data analysis	hybrid	5	5	log
task_03	API Integration Scaffold	Seed / code generation	automated	5	5	json
task_04	Test Generation	Seed / testing	automated	6	6	py
task_05	Config Migration v1 to v2	Seed / data transformation	automated	5	5	json
task_06	Security Code Review	Seed / code analysis	hybrid	5	5	go, js, py
task_07	Document Data Extraction	Seed / data extraction	hybrid	5	5	txt
task_08	Database Schema Operations	Seed / database	automated	5	5	sql

Task	Name	Family/category	Grade	Sub.	Files	Ext.
task_09	Excel Analytics Report Generation	Seed / data transformation	automated	5	5	csv
task_10	Git Repository History Analysis	Seed / data analysis	automated	5	5	sh
task_11	Web Page Structured Data Extraction	Seed / data extraction	hybrid	5	5	html
task_12	Word Document Generation	Seed / code generation	automated	5	5	json
task_13	Multi-Dataset Statistical Analysis Pipeline	Seed / data analysis	hybrid	5	15	csv
task_14	Enterprise Email Thread Analysis	Seed / office automation	hybrid	5	5	json
task_15	Shell Automation Script Generation	Seed / automation	hybrid	5	5	yaml
task_16	Dockerfile and CI Pipeline Generation	Seed / devops	hybrid	5	5	json
task_17	Enterprise Invoice Expense Processing	Seed / office daily	hybrid	5	5	csv, json, tsv, txt, xml
task_18	Dependency Security Audit	Seed / security	hybrid	5	5	json, mod, none, txt, xml
task_19	Meeting Notes Structured Extraction	Seed / office daily	hybrid	5	5	txt
task_20	Multi-Environment Config Generation	Seed / devops automation	automated	5	5	json
task_21	System Metrics Anomaly Detection	Seed / data analysis	hybrid	5	5	csv
task_22	Finance Ledger Reconciliation	Finance	automated	5	5	csv
task_23	Subscription Revenue Audit	Finance	automated	5	5	csv
task_24	Bank KYC Packet Review	Finance	automated	5	5	csv
task_25	Loan Document Checklist	Finance	automated	5	5	csv
task_26	Tax Form Consistency	Finance	automated	5	5	csv
task_27	Contract Clause Extraction	Legal	automated	5	5	txt
task_28	Discovery Document Tagging	Legal	automated	5	5	txt
task_29	Policy Exception Mapping	Legal	automated	5	5	txt
task_30	Nda Obligation Review	Legal	automated	5	5	txt
task_31	Vendor Contract Risk	Legal	automated	5	5	txt
task_32	Appointment Referral Triage	Healthcare	automated	5	5	json
task_33	Pharmacy Inventory Reorder	Healthcare	automated	5	5	json
task_34	Clinical Trial Eligibility	Healthcare	automated	5	5	json
task_35	Insurance Prior Auth Review	Healthcare	automated	5	5	json
task_36	Lab Result Followup Queue	Healthcare	automated	5	5	json
task_37	Procurement Bid Scoring	Procurement/Logistics	automated	5	5	yaml
task_38	Purchase Order Three Way Match	Procurement/Logistics	automated	5	5	yaml
task_39	Shipping Exception Resolution	Procurement/Logistics	automated	5	5	yaml
task_40	Warehouse Picklist Validation	Procurement/Logistics	automated	5	5	yaml
task_41	Supplier Risk Packet	Procurement/Logistics	automated	5	5	yaml
task_42	Employee Onboarding Checklist	HR/Education	automated	5	5	json
task_43	Interview Feedback Calibration	HR/Education	automated	5	5	json
task_44	Admissions Packet Screening	HR/Education	automated	5	5	json
task_45	Assignment Rubric Grading	HR/Education	automated	5	5	json
task_46	Training Completion Audit	HR/Education	automated	5	5	json
task_47	Support Ticket Escalation	Support/Product	automated	5	5	json
task_48	Kb Gap Analysis	Support/Product	automated	5	5	json
task_49	App Review Theme Mining	Support/Product	automated	5	5	json
task_50	Bug Report Deduplication	Support/Product	automated	5	5	json
task_51	Call Center Quality Review	Support/Product	automated	5	5	json
task_52	Kubernetes Policy Review	DevOps/SRE	automated	5	5	yaml
task_53	Terraform Plan Drift	DevOps/SRE	automated	5	5	yaml
task_54	SLO Burn Rate Analysis	DevOps/SRE	automated	5	5	yaml
task_55	CI Pipeline Hardening	DevOps/SRE	automated	5	5	yaml
task_56	Backup Restore Drill	DevOps/SRE	automated	5	5	yaml

Task	Name	Family/category	Grade	Sub.	Files	Ext.
task_57	Security Alert Correlation	Security/Privacy	automated	5	5	json
task_58	Phishing Report Triage	Security/Privacy	automated	5	5	json
task_59	Vulnerability Exception Review	Security/Privacy	automated	5	5	json
task_60	PII Redaction Release	Security/Privacy	automated	5	5	json
task_61	DSR Request Routing	Security/Privacy	automated	5	5	json
task_62	SQL Schema Migration Review	Data/Analytics	automated	5	5	csv
task_63	Messy Csv Normalization	Data/Analytics	automated	5	5	csv
task_64	Dashboard Metric Reconciliation	Data/Analytics	automated	5	5	csv
task_65	Data Quality Rule Authoring	Data/Analytics	automated	5	5	csv
task_66	Experiment Results Analysis	Data/Analytics	automated	5	5	csv
task_67	Research Claim Evidence	Research/Media	automated	5	5	html
task_68	Newsroom Fact Check	Research/Media	automated	5	5	html
task_69	Literature Table Extraction	Research/Media	automated	5	5	html
task_70	Public Web Directory Extraction	Research/Media	automated	5	5	html
task_71	Citation Deduplication	Research/Media	automated	5	5	html
task_72	Localization Placeholder Qa	Localization/Release	automated	5	5	json
task_73	Glossary Compliance Review	Localization/Release	automated	5	5	json
task_74	Release Notes Curation	Localization/Release	automated	5	5	json
task_75	Changelog Impact Matrix	Localization/Release	automated	5	5	json
task_76	Feature Flag Cleanup Plan	Localization/Release	automated	5	5	json
task_77	Ecommerce Catalog Normalization	Commerce/Food	automated	5	5	csv
task_78	Retail Returns Root Cause	Commerce/Food	automated	5	5	csv
task_79	Restaurant Inspection Summary	Commerce/Food	automated	5	5	csv
task_80	Food Delivery Refund Review	Commerce/Food	automated	5	5	csv
task_81	Marketplace Listing Policy	Commerce/Food	automated	5	5	csv
task_82	Facilities Maintenance Prioritization	Facilities/IoT	automated	5	5	json
task_83	Fleet Maintenance Schedule	Facilities/IoT	automated	5	5	json
task_84	Iot Sensor Anomaly Review	Facilities/IoT	automated	5	5	json
task_85	Smart Home Support Diagnosis	Facilities/IoT	automated	5	5	json
task_86	Robotics Workcell Event Report	Facilities/IoT	automated	5	5	json
task_87	Civic Service Request Routing	Public Sector/Audit	automated	5	5	json
task_88	Public Records Redaction	Public Sector/Audit	automated	5	5	json
task_89	Grant Application Completeness	Public Sector/Audit	automated	5	5	json
task_90	Audit Evidence Collection	Public Sector/Audit	automated	5	5	json
task_91	Risk Register Rollup	Public Sector/Audit	automated	5	5	json
task_92	CRM Pipeline Hygiene	CRM/Executive	automated	5	5	csv
task_93	Sales Forecast Variance	CRM/Executive	automated	5	5	csv
task_94	Marketing Campaign Qa	CRM/Executive	automated	5	5	csv
task_95	Board Packet Preparation	CRM/Executive	automated	5	5	csv
task_96	Executive Action Item Tracking	CRM/Executive	automated	5	5	csv
task_97	Calendar Conflict Resolution	Travel/Office	automated	5	5	json
task_98	Travel Itinerary Exception	Travel/Office	automated	5	5	json
task_99	Meeting Notes Action Items	Travel/Office	automated	5	5	json
task_100	Inbox Rules Classification	Travel/Office	automated	5	5	json

I Per-Task Prompt and Output Surfaces

The following table expands the inventory with the prompt focus and grader-visible output surface for every official task. It is generated from the task markdown files and is intended to make the 100-task composition auditable without requiring reviewers to open each fixture directory.

Table 13: Per-task prompt and output surfaces.

Task	Name	Prompt focus	Grader-visible output surface	Checks
task_01	Batch Data Transform	You have 6 data files in artifact that each contain user records in different formats.	outputs/users_01.json, outputs/users_02.json, outputs/users_03.json ...	automated; 6 criteria
task_02	Log Analysis	You have 5 log files in artifact from different microservices.	outputs/api_gateway_report.json, outputs/auth_service_report.json, outputs/notification_service_report.json ...	hybrid; 6 criteria

Task	Name	Prompt focus	Grader-visible output surface	Checks
task_03	API Integration Scaffold	You have 5 API specification files in artifact.	outputs/analytics_client.py, outputs/inventory_client.py, outputs/notifications_client.py ...	automated; 6 criteria
task_04	Test Generation	You have 6 Python utility modules in artifact.	outputs/test_data_utils.py, outputs/test_date_utils.py, outputs/test_file_utils.py ...	automated; 6 criteria
task_05	Config Migration v1 to v2	You have 5 application configuration files in artifact that use a legacy flat key format (v1).	outputs/gateway_v2.json, outputs/mailer_v2.json, outputs/scheduler_v2.json ...	automated; 7 criteria
task_06	Security Code Review	You have 5 source code files in artifact that contain intentional security vulnerabilities.	outputs/admin_panel_review.json, outputs/api_handler_review.json, outputs/file_manager_review.json ...	hybrid; 7 criteria
task_07	Document Data Extraction	You have 5 text documents in artifact representing different document types.	outputs/contract_001.json, outputs/expense_report_001.json, outputs/invoice_XX.json ...	hybrid; 8 criteria
task_08	Database Schema Operations	You have 5 SQL schema files in artifact representing existing PostgreSQL tables.	outputs/migrate_orders.sql, outputs/migrate_products.sql, outputs/migrate_reviews.sql ...	automated; 7 criteria
task_09	Excel Analytics Report Generation	You are given 5 regional sales CSV files in artifact.	outputs/report_central.xlsx, outputs/report_east.xlsx, outputs/report_north.xlsx ...	automated; 7 criteria
task_10	Git Repository History Analysis	You are given 5 bash setup scripts in artifact.	task-specific grader-visible files under outputs/	automated; 6 criteria
task_11	Web Page Structured Data Extraction	You are given 5 HTML files in artifact that simulate web pages.	task-specific grader-visible files under outputs/	hybrid; 7 criteria
task_12	Word Document Generation	You are given 5 JSON data files in artifact.	task-specific grader-visible files under outputs/	automated; 6 criteria
task_13	Multi-Dataset Statistical Analysis Pipeline	You are given 5 groups of related CSV files in artifact.	task-specific grader-visible files under outputs/	hybrid; 6 criteria
task_14	Enterprise Email Thread Analysis	Complete enterprise email thread analysis and write grader-visible outputs.	outputs/thread_XX_report.json	hybrid; 7 criteria
task_15	Shell Automation Script Generation	You have 5 automation task specification files in artifact.	outputs/task_XX_db_dump.sh, outputs/task_XX_dir_monitor.sh, outputs/task_XX_file_backup.sh ...	hybrid; 9 criteria
task_16	Dockerfile and CI Pipeline Generation	You have 5 application specification files in artifact.	outputs/app_XX/Dockerfile	hybrid; 8 criteria
task_17	Enterprise Invoice Expense Processing	Complete enterprise invoice expense processing and write grader-visible outputs.	outputs/invoice_XX_parsed.json	hybrid; 8 criteria
task_18	Dependency Security Audit	You have 5 dependency manifest files in artifact from different application stacks.	outputs/gemfile_audit.json, outputs/go_audit.json, outputs/package_audit.json ...	hybrid; 8 criteria
task_19	Meeting Notes Structured Extraction	Complete meeting notes structured extraction and write grader-visible outputs.	outputs/meeting_XX_minutes.json	hybrid; 8 criteria
task_20	Multi-Environment Config Generation	You have 5 application specification files in artifact.	outputs/data-exporter/config.dev.yml, outputs/ml-inference/config.dev.yml, outputs/notification-service/config.dev.yml ...	automated; 8 criteria
task_21	System Metrics Anomaly Detection	You have 5 system metrics CSV files in artifact.	outputs/db_query_perf_report.json, outputs/disk_io_report.json, outputs/network_traffic_report.json ...	hybrid; 8 criteria
task_22	Finance Ledger Reconciliation	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_23	Subscription Revenue Audit	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_24	Bank KYC Packet Review	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_25	Loan Document Checklist	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_26	Tax Form Consistency	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_27	Contract Clause Extraction	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_28	Discovery Document Tagging	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_29	Policy Exception Mapping	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_30	Nda Obligation Review	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_31	Vendor Contract Risk	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_32	Appointment Referral Triage	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_33	Pharmacy Inventory Reorder	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_34	Clinical Trial Eligibility	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_35	Insurance Prior Auth Review	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria

Task	Name	Prompt focus	Grader-visible output surface	Checks
task_36	Lab Result Followup Queue	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_37	Procurement Bid Scoring	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_38	Purchase Order Three Way Match	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_39	Shipping Exception Resolution	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_40	Warehouse Picklist Validation	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_41	Supplier Risk Packet	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_42	Employee Onboarding Checklist	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_43	Interview Feedback Calibration	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_44	Admissions Packet Screening	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_45	Assignment Rubric Grading	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_46	Training Completion Audit	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_47	Support Ticket Escalation	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_48	Kb Gap Analysis	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_49	App Review Theme Mining	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_50	Bug Report Deduplication	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_51	Call Center Quality Review	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_52	Kubernetes Policy Review	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_53	Terraform Plan Drift	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_54	SLO Burn Rate Analysis	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_55	CI Pipeline Hardening	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_56	Backup Restore Drill	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_57	Security Alert Correlation	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_58	Phishing Report Triage	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_59	Vulnerability Exception Review	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_60	PII Redaction Release	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_61	DSR Request Routing	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_62	SQL Schema Migration Review	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_63	Messy Csv Normalization	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_64	Dashboard Metric Reconciliation	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_65	Data Quality Rule Authoring	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_66	Experiment Results Analysis	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_67	Research Claim Evidence	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_68	Newsroom Fact Check	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_69	Literature Table Extraction	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_70	Public Web Directory Extraction	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_71	Citation Deduplication	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_72	Localization Placeholder Qa	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_73	Glossary Compliance Review	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_74	Release Notes Curation	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria

Task	Name	Prompt focus	Grader-visible output surface	Checks
task_75	Changelog Impact Matrix	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_76	Feature Flag Cleanup Plan	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_77	Ecommerce Catalog Normalization	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_78	Retail Returns Root Cause	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_79	Restaurant Inspection Summary	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_80	Food Delivery Refund Review	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_81	Marketplace Listing Policy	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_82	Facilities Maintenance Prioritization	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_83	Fleet Maintenance Schedule	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_84	Iot Sensor Anomaly Review	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_85	Smart Home Support Diagnosis	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_86	Robotics Workcell Event Report	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_87	Civic Service Request Routing	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_88	Public Records Redaction	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_89	Grant Application Completeness	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_90	Audit Evidence Collection	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_91	Risk Register Rollup	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_92	CRM Pipeline Hygiene	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_93	Sales Forecast Variance	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_94	Marketing Campaign Qa	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_95	Board Packet Preparation	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_96	Executive Action Item Tracking	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_97	Calendar Conflict Resolution	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_98	Travel Itinerary Exception	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_99	Meeting Notes Action Items	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria
task_100	Inbox Rules Classification	Select the authoritative evidence packet, discard decoys, derive strict JSON case reports.	outputs/case_XX_report.json	automated; 8 criteria

J PER-TASK GRADING SURFACES

This table records how each official task is checked. It separates the grading type from the concrete surface that the grader inspects, which is useful because hybrid tasks still require file artifacts and automated tasks may include rich schema or content checks.

Table 14: Per-task grading surfaces.

Task	Name	Grade	Check surface	Representative criteria
task_01	Batch Data Transform	automated	automated code	Each of the 6 output files exists in artifact; Each output file contains valid JSON; Each record conforms to the target schema (all 6 fields present)
task_02	Log Analysis	hybrid	automated code, LLM rubric, weighted blend	Each of the 5 output report files exists; Each report contains valid JSON; Each report has all required top-level fields

Task	Name	Grade	Check surface	Representative criteria
task_03	API Integration Scaffold	automated	automated code	Each of the 5 output .py files exists; Each file contains valid Python (parseable by ast.parse); Each file contains credential-handling code
task_04	Test Generation	automated	automated code	Each of the 6 test files exists in artifact; Each file contains valid Python syntax; Each file imports pytest
task_05	Config Migration v1 to v2	automated	automated code	All 5 output files exist in outputs/; All output files are valid JSON; All output files use nested structure (no flat dot-notation keys)
task_06	Security Code Review	hybrid	automated code, LLM rubric, weighted blend	All 5 output files exist in outputs/; All output files are valid JSON; Each report has a "vulnerabilities" array with at least 3 entries
task_07	Document Data Extraction	hybrid	automated code, LLM rubric, weighted blend	All 5 output files exist in outputs/; All output files are valid JSON; Invoice has line_items array and grand_total
task_08	Database Schema Operations	automated	automated code	All 5 output files exist in outputs/; All files contain valid SQL statements; Each migration contains the required ALTER TABLE statements
task_09	Excel Analytics Report Generation	automated	automated code	Output .xlsx files exist for all 5 regions; Each workbook has exactly 3 sheets named correctly; Raw Data sheet contains the expected number of rows (20 data rows + header)
task_10	Git Repository History Analysis	automated	automated code	Output JSON files exist for all 5 repositories; artifact matches expected value exactly; All expected contributors appear in the contributors list
task_11	Web Page Structured Data Extraction	hybrid	automated code, LLM rubric, weighted blend	All 5 output JSON files exist; Each file contains a list with at least 5 items; Product catalog: prices are numeric, product IDs match source
task_12	Word Document Generation	automated	automated code	All 5 .docx files exist in outputs/; Files are valid Word documents (openable by python-docx); Each document contains the correct title as a Heading 1
task_13	Multi-Dataset Statistical Analysis Pipeline	hybrid	automated code, LLM rubric, weighted blend	All 5 analysis JSON files exist; artifact / artifact / artifact / artifact match expected counts; Numeric statistics are present and of correct type
task_14	Enterprise Email Thread Analysis	hybrid	automated code, LLM rubric, weighted blend	All 5 output files exist; each report is valid JSON; each report contains all required top-level fields
task_15	Shell Automation Script Generation	hybrid	automated code, LLM rubric, weighted blend	All 10 output files exist (5 scripts + 5 READMEs); Each artifact file starts with artifact or artifact; Each artifact file contains artifact or artifact
task_16	Dockerfile and CI Pipeline Generation	hybrid	automated code, LLM rubric, weighted blend	All 10 output files exist (5 Dockerfiles + 5 workflow YAMLs); Each Dockerfile starts with artifact; Multi-stage Dockerfiles (apps 01–04) contain at least 2 artifact instructions
task_17	Enterprise Invoice Expense Processing	hybrid	automated code, LLM rubric, weighted blend	All 6 output files exist (5 parsed files plus 1 summary); each parsed file is valid JSON; parsed files contain all required top-level fields
task_18	Dependency Security Audit	hybrid	automated code, LLM rubric, weighted blend	All 5 output files exist; Each report contains valid JSON; Each report has all required top-level fields

Task	Name	Grade	Check surface	Representative criteria
task_19	Meeting Notes Structured Extraction	hybrid	automated code, LLM rubric, weighted blend	All 5 output files exist; each file is valid JSON; each file contains all required top-level fields
task_20	Multi-Environment Config Generation	automated	automated code	All 15 output YAML files exist; Each YAML file is parseable (valid YAML); artifact has artifact
task_21	System Metrics Anomaly Detection	hybrid	automated code, LLM rubric, weighted blend	All 5 output report files exist; Each report contains valid JSON; Each report has all required top-level fields
task_22	Finance Ledger Reconciliation	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_23	Subscription Revenue Audit	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_24	Bank KYC Packet Review	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_25	Loan Document Checklist	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_26	Tax Form Consistency	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_27	Contract Clause Extraction	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_28	Discovery Document Tagging	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_29	Policy Exception Mapping	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_30	Nda Obligation Review	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_31	Vendor Contract Risk	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.

Task	Name	Grade	Check surface	Representative criteria
task_32	Appointment Referral Triage	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_33	Pharmacy Inventory Reorder	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_34	Clinical Trial Eligibility	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_35	Insurance Prior Auth Review	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_36	Lab Result Followup Queue	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_37	Procurement Bid Scoring	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_38	Purchase Order Three Way Match	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_39	Shipping Exception Resolution	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_40	Warehouse Picklist Validation	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_41	Supplier Risk Packet	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_42	Employee Onboarding Checklist	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_43	Interview Feedback Calibration	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.

Task	Name	Grade	Check surface	Representative criteria
task_44	Admissions Packet Screening	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_45	Assignment Rubric Grading	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_46	Training Completion Audit	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_47	Support Ticket Escalation	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_48	Kb Gap Analysis	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_49	App Review Theme Mining	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_50	Bug Report Deduplication	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_51	Call Center Quality Review	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_52	Kubernetes Policy Review	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_53	Terraform Plan Drift	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_54	SLO Burn Rate Analysis	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_55	CI Pipeline Hardening	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.

Task	Name	Grade	Check surface	Representative criteria
task_56	Backup Restore Drill	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_57	Security Alert Correlation	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_58	Phishing Report Triage	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_59	Vulnerability Exception Review	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_60	PII Redaction Release	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_61	DSR Request Routing	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_62	SQL Schema Migration Review	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_63	Messy Csv Normalization	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_64	Dashboard Metric Reconciliation	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_65	Data Quality Rule Authoring	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_66	Experiment Results Analysis	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_67	Research Claim Evidence	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.

Task	Name	Grade	Check surface	Representative criteria
task_68	Newsroom Fact Check	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_69	Literature Table Extraction	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_70	Public Web Directory Extraction	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_71	Citation Deduplication	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_72	Localization Placeholder Qa	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_73	Glossary Compliance Review	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_74	Release Notes Curation	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_75	Changelog Impact Matrix	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_76	Feature Flag Cleanup Plan	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_77	Ecommerce Catalog Normalization	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_78	Retail Returns Root Cause	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_79	Restaurant Inspection Summary	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.

Task	Name	Grade	Check surface	Representative criteria
task_80	Food Delivery Refund Review	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_81	Marketplace Listing Policy	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_82	Facilities Maintenance Prioritization	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_83	Fleet Maintenance Schedule	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_84	Iot Sensor Anomaly Review	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_85	Smart Home Support Diagnosis	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_86	Robotics Workcell Event Report	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_87	Civic Service Request Routing	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_88	Public Records Redaction	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_89	Grant Application Completeness	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_90	Audit Evidence Collection	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_91	Risk Register Rollup	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.

Task	Name	Grade	Check surface	Representative criteria
task_92	CRM Pipeline Hygiene	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_93	Sales Forecast Variance	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_94	Marketing Campaign Qa	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_95	Board Packet Preparation	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_96	Executive Action Item Tracking	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_97	Calendar Conflict Resolution	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_98	Travel Itinerary Exception	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_99	Meeting Notes Action Items	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.
task_100	Inbox Rules Classification	automated	automated code	All five artifact files exist.; Each report is valid JSON and contains every required field.; The report ignores draft, superseded, invalid-checksum, and decoy packet records.

K Task Families and Fixture Mechanics

The suite combines heterogeneous seed tasks with generated hard-mode families. The generated families share a common evidence-packet protocol, but each family changes the business action, output schema, and grader-visible decision surface. This is intended to make label diversity insufficient: an agent must handle repeated mechanics while mapping them to distinct artifacts.

Table 15: Task-family summary.

Family	Tasks	Sub-problems	Fixture and output mechanics
Original seed tasks	21	107	Early heterogeneous tasks covering code, logs, documents, office files, shell automation, CI, dependencies, and metrics; these use task-specific fixtures and graders rather than the shared hard-mode packet protocol. Tasks: 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21.
CRM/Executive	5	25	Pipeline hygiene, forecast variance, campaign QA, board packets, and executive action tracking with repeated business-report outputs. Tasks: 92, 93, 94, 95, 96.
Commerce/Food	5	25	Delivery refunds, inspections, marketplace policies, maintenance-like retail cases, and supportable operational decisions over generated local fixtures. Tasks: 77, 78, 79, 80, 81.
Data/Analytics	5	25	Normalization, reconciliation, rule-authoring, experiment, and extraction tasks that stress repeatable data derivation under decoys and stale revisions. Tasks: 62, 63, 64, 65, 66.
DevOps/SRE	5	25	Policy, drift, burn-rate, pipeline-hardening, and restore-drill tasks with infrastructure-oriented evidence channels and output artifacts. Tasks: 52, 53, 54, 55, 56.
Facilities/IoT	5	25	Facilities, fleet, sensor, smart-home, and workcell event tasks that derive maintenance or anomaly reports from selected evidence packets. Tasks: 82, 83, 84, 85, 86.
Finance	5	25	Ledger, revenue, loan, tax, and KYC-style reconciliation with selected evidence packets, stale revisions, numeric deltas, list edits, and boolean compliance gates. Tasks: 22, 23, 24, 25, 26.
HR/Education	5	25	Onboarding, interview, admissions, grading, and completion-audit cases requiring structured checklist or rubric outputs from decoy-rich evidence. Tasks: 42, 43, 44, 45, 46.
Healthcare	5	25	Clinical-administration tasks such as triage, pharmacy reorder, trial eligibility, prior authorization, and follow-up queues with strict output schemas. Tasks: 32, 33, 34, 35, 36.
Legal	5	25	Clause, discovery, policy-exception, NDA, and contract-risk review tasks that require choosing authoritative records and writing structured legal-operation outputs. Tasks: 27, 28, 29, 30, 31.
Localization/ Release	5	25	Placeholder QA, release notes, flag cleanup, catalog normalization, and returns analysis tasks with structured release or commerce-adjacent artifacts. Tasks: 72, 73, 74, 75, 76.
Procurement/ Logistics	5	25	Supplier, bid, purchase-order, warehouse, and shipping cases built around conflicting packets, exception lists, and derived routing or scoring reports. Tasks: 37, 38, 39, 40, 41.
Public Sector/Audit	5	25	Service routing, records redaction, grant completeness, evidence collection, and risk register tasks using synthetic civic/audit fixtures. Tasks: 87, 88, 89, 90, 91.
Research/Media	5	25	Claim checking, table extraction, directory extraction, citation deduplication, and glossary compliance with evidence selection and citation-like outputs. Tasks: 67, 68, 69, 70, 71.
Security/Privacy	5	25	Alert, phishing, exception, release-redaction, and request-routing tasks that test privacy/security triage without using real sensitive data. Tasks: 57, 58, 59, 60, 61.
Support/Product	5	25	Ticket, knowledge-base, app-review, bug-deduplication, and call-quality tasks that convert repeated records into grader-visible JSON reports. Tasks: 47, 48, 49, 50, 51.

Family	Tasks	Sub-problems	Fixture and output mechanics
Travel/Office	4	20	Calendar, itinerary, meeting-action, and inbox-rule cases that convert office coordination evidence into structured decisions. Tasks: 97, 98, 99, 100.

L GENERATED FAMILY DESIGN MATRIX

Table 16 expands the family summary with the repeated structure that is intended to make skill creation plausible, and the anti-shortcut mechanism that prevents a shallow schema-only answer from satisfying the grader. The matrix is written at the family level rather than the task level because generated tasks within a family share the same evidence-selection contract while changing the concrete business operation and required output fields.

Table 16: Design matrix for generated hard-mode task families.

Family	Repeated structure	Anti-shortcut mechanism	Grader-visible artifact
CRM/Executive	Repeated business-status packets must be reconciled into executive-ready decisions.	Decoy opportunities, stale campaign revisions, and conflicting forecast evidence make visible totals unreliable.	Structured pipeline, forecast, campaign, board, or action-item JSON reports.
Commerce/Food	Repeated commerce cases map operational evidence to refund, inspection, marketplace, or returns decisions.	Draft and superseded packets expose plausible but wrong SKUs, refund IDs, or policy codes.	Per-case operational JSON reports under `outputs/`.
Data/Analytics	Repeated analytics records require applying numeric, list, and text derivation channels consistently.	Schema-only outputs fail because selected packets must be verified before computing rows, rules, or experiment winners.	Strict JSON reports with derived counts, deltas, failures, and winners.
DevOps/SRE	Repeated infrastructure cases ask for drift, burn-rate, policy, pipeline, or restore conclusions.	Stale revisions and decoy resource identifiers make direct copying from fixtures unsafe.	JSON reports that encode affected resources, exception IDs, and remediation signals.
Facilities/IoT	Repeated sensor, fleet, smart-home, and workcell packets require deriving maintenance or anomaly actions.	Conflicting packet states and boolean gates separate approved evidence from noisy telemetry.	Prioritization, diagnosis, schedule, or event-summary JSON reports.
Finance	Repeated finance cases share ledger-style aggregation, exception lists, tax-like numeric fields, and compliance gates.	Invalid checksums, superseded packets, and decoy ledgers prevent copying visible totals.	Reconciliation or packet-review JSON reports.
HR/Education	Repeated people-process cases map evidence to checklists, rubrics, screening, or calibration decisions.	Decoy records include plausible missing items or scores that must be ignored unless final and selected.	Checklist, rubric, admissions, interview, or completion-audit JSON reports.
Healthcare	Repeated clinical-administration cases derive triage, reorder, eligibility, authorization, or follow-up decisions.	Synthetic healthcare-like evidence avoids real patient data while still requiring selected-packet derivation.	Strict healthcare-administration JSON reports.
Legal	Repeated legal-operation cases extract obligations, clauses, exceptions, or contract risks.	Draft clauses and stale revisions test whether the agent follows packet provenance instead of surface wording.	Clause, discovery, NDA, policy, or risk JSON outputs.

Family	Repeated structure	Anti-shortcut mechanism	Grader-visible artifact
Localization/ Release	Repeated release-management cases reconcile placeholders, glossary issues, flags, changelog impacts, or catalog data.	Multiple revisions and alias/remove list actions require ordered list derivation rather than keyword matching.	Release, localization, flag, changelog, or catalog-normalization JSON reports.
Procurement/ Logistics	Repeated sourcing and logistics cases derive vendors, purchase-order exceptions, warehouse issues, or shipment actions.	Competing packet weights and revisions expose believable but wrong vendor and exception values.	Procurement, warehouse, and shipping JSON reports.
Public Sector/Audit	Repeated public-service and audit packets require routing, redaction, evidence, grant, or risk decisions.	Synthetic public-sector records contain missing-evidence decoys and privacy-like fields without real PII.	Civic-routing, redaction, grant, evidence, and risk-register JSON reports.
Research/Media	Repeated research and media cases derive evidence tables, citations, claims, facts, or glossary compliance.	Text candidates and list edits force deterministic selection rather than free-form summarization.	Claim, fact-check, extraction, citation, or compliance JSON reports.
Security/Privacy	Repeated security and privacy cases derive incidents, redactions, phishing, exceptions, or DSR routing.	Decoy indicators and boolean gates separate final selected security evidence from raw noisy records.	Alert, privacy, phishing, exception, or release-redaction JSON reports.
Support/Product	Repeated support and product cases derive escalations, KB gaps, app-review themes, bug clusters, or call quality.	Alias and remove actions test whether the agent can update lists rather than aggregate all visible labels.	Support/product JSON reports with routed or clustered decisions.
Travel/Office	Repeated office-coordination cases derive conflicts, itinerary exceptions, meeting actions, or inbox classifications.	Stale calendar or inbox-rule revisions make the latest visible item insufficient without packet selection.	Calendar, travel, meeting, or inbox-rule JSON reports.

M HARD-MODE EVIDENCE PROTOCOL

Generated hard-mode tasks are written so that visible values in a fixture are not necessarily authoritative. Each case contains a packet manifest plus evidence records. The agent must select the approved packet with no superseding packet and a valid checksum equal to the first 16 hexadecimal characters of the hash of the benchmark salt, task id, case id, packet id, and nonce. If more than one packet remains, it chooses the highest revision, then the highest source weight, then the lowest packet identifier. Only final records from the selected packet may drive the output.

```
sha256(evoclawbench-difficulty-hardening-20260524-v4|
<task_id>|<case_id>|<packet_id>|<nonce>)
```

Table 17: Evidence-channel derivation rules used by generated hard-mode tasks.

Output type	Record field	Derivation rule
numeric	numeric_delta	Apply signed amount_minor / scale; subtract when operator is subtract; round non-count values to two decimals.
list	list_action	Apply include, remove, and alias rows in revision order; emit sorted unique strings.
dict	dict_delta	Sum deltas by bucket and omit zero-valued buckets.
boolean	boolean_gate	Emit true only when every selected gate has observed equal to expected.
text	text_candidate	Choose the candidate maximizing score - penalty; break ties lexicographically.

The protocol appears across JSON, YAML, CSV, text, and HTML fixtures. JSON and YAML expose manifest and record arrays directly; CSV fixtures use section-tagged rows; text fixtures use JSON lines; HTML fixtures embed JSON in script blocks. The grader checks the derived files under `outputs/` and does not accept a final natural-language explanation as a substitute for those artifacts.

N Representative Output Schemas

For generated hard-mode tasks, the prompt names the exact JSON fields required in each case report. The table lists one schema per generated family; seed tasks use task-specific formats such as scripts, workbooks, SQL migrations, Dockerfiles, CI files, or extraction JSON.

Table 18: Representative output schemas for generated hard-mode families.

Family	Representative task	Required output fields
CRM/Executive	task_92: CRM Pipeline Hygiene	stale_opportunities, forecast_delta, campaign_errors, action_items, executive_summary
Commerce/Food	task_77: Ecommerce Catalog Normalization	normalized_skus, refund_ids, inspection_score, policy_violations, recommended_action
Data/Analytics	task_62: SQL Schema Migration Review	row_count, quality_failures, metric_delta, rule_ids, experiment_winner
DevOps/SRE	task_52: Kubernetes Policy Review	policy_violations, required_changes, slo_status, backup_passed, risk_score
Facilities/IoT	task_82: Facilities Maintenance Prioritization	priority_assets, maintenance_due, anomaly_ids, diagnostic_codes, dispatch_required
Finance	task_22: Finance Ledger Reconciliation	ledger_total, exception_ids, currencies, tax_total, balanced
HR/Education	task_42: Employee Onboarding Checklist	completion_rate, missing_items, calibrated_scores, assigned_track, intervention_ids
Healthcare	task_32: Appointment Referral Triage	eligible_ids, routing_queue, followup_ids, stockout_ids, urgent_count
Legal	task_27: Contract Clause Extraction	clause_labels, missing_fields, high_risk_terms, parties, effective_date
Localization/Release	task_72: Localization Placeholder Qa	placeholder_errors, glossary_violations, release_sections, flag_actions, publish_ready
Procurement/Logistics	task_37: Procurement Bid Scoring	selected_vendor, match_exceptions, late_shipments, risk_suppliers, savings_estimate
Public Sector/Audit	task_87: Civic Service Request Routing	routing_queue, redactions_required, missing_evidence, risk_summary, compliant
Research/Media	task_67: Research Claim Evidence	supported_claims, unsupported_claims, source_count, duplicate_citations, confidence
Security/Privacy	task_57: Security Alert Correlation	incident_ids, redacted_fields, raw_pii_present, approved_exceptions, severity_counts
Support/Product	task_47: Support Ticket Escalation	priority_queue, duplicate_groups, themes, sla_breaches, reply_template

Family	Representative task	Required output fields
Travel/Office	task_97: Calendar Conflict Resolution	conflicts, exceptions, action_items, rule_labels, resolved

O SUITE DISTRIBUTIONS

Table 19: Grading types.

Type	Tasks
automated	88
hybrid	12

Table 20: Sub-problem counts.

Sub-problems	Tasks
5	98
6	2

Table 21: Fixture extensions.

Ext.	Files	Ext.	Files	Ext.	Files
json	228	csv	128	yaml	56
txt	37	html	30	py	9
log	5	sh	5	sql	5
xml	3	tsv	2	go	1
js	1	mod	1	none	1

P RESULT JSON SCHEMA

The aggregate output JSON contains `baseline_results`, `preskill_results`, `postskill_results`, and `metrics`. The table below summarizes the fields used for benchmark reporting.

Table 22: Result JSON fields used for benchmark reporting.

Object	Fields	Use in paper
top-level	version, timestamp, benchmark, model, runtime, run_id, judge model	Identifies the benchmark run, runtime configuration, and grading configuration.
top-level	mode, environment, workers, runs_per_task	Execution controls for a benchmark run.
results	baseline_results	One graded result object per loaded task for direct execution.
results	preskill_results	One graded result object per loaded task after preskill skill creation and reuse execution.
results	postskill_results	One graded result object per loaded task after first execution, summary, and second execution.
metrics.execution_only	mean_scores, ratios_vs_baseline, usage, efficiency_vs_baseline	Final execution phase quality and resource comparisons.
metrics.end_to_end	mean_scores, ratios_vs_baseline, usage, efficiency_vs_baseline	Full workflow quality and overhead comparisons.
metrics.created_skills	preskill_count, postskill_count, preskill, postskill	Skill counts and per-skill metadata collected from workspaces.
metrics.skill_quality	preskill, postskill	Heuristic quality summaries for generated skills.
metrics.skill_mutation_violations	preskill, postskill	Counts reuse executions that added, removed, or edited skill files.

Q RESULT VALIDITY CHECKLIST

Before adding a result row, record the model, runtime, execution mode, environment, worker count, judge model, loaded task count, official task count, mean scores for all modes, created-skill counts, and mutation checks. Audit every final execution phase, and the POSTSKILL first execution, for timeouts, fully empty assistant transcripts, and runtime fallback replies; report affected cells explicitly instead of averaging them with completed executions. Runs should be reported consistently across BASELINE, PRESKILL, and POSTSKILL so that score, cost, and skill-reuse comparisons refer to the same benchmark protocol, task basis, and execution audit.