EvoClawBench: Can Agents Learn Reusable Skills from Their Own Runs?

Anonymous ACL submission

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. 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. Experiments with OpenClaw and nanobot under local execution show that direct BASELINE performance is strongly runtime-dependent: OpenClaw remains below 20% across models, while nanobot ranges from 56.45% to 96.13%. Self-authored skills have mixed effects. nanobot `GPT-5.4` stays above 96% in all modes and `MiniMax-M2.7` improves from 90.97% to 94.50% under POSTSKILL, but nanobot `DeepSeek-V4-Pro` drops from 77.77% to 4.80% with PRESKILL and 0.99% with POSTSKILL. OpenClaw shows similarly non-monotonic behavior, with some skill runs near baseline and others collapsing. 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; 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 of doing so. This leaves a focused evaluation question: *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 procedural artifacts as ex-

Table 1: Comparison of EvoClawBench with existing benchmarks. "Skill Cond." indicates whether the benchmark includes agent skills. "Det. Verifier" indicates whether deterministic (non-LLM) verification is included. "Closed Loop" indicates 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**	✓	✓	✓	✓

ternal 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 related but 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 most directly tests the title question: whether an agent can distill reusable procedural memory from its own completed run. Both are compared against BASELINE under execution-only and end-to-end metrics, so a skill workflow must improve not only task score but also justify its additional token, cost, and time overhead.

Our paper makes the following contributions:

- **Benchmark.** We build EvoClawBench, a benchmark for evaluating whether LLM agents can learn reusable skills from controlled self-authored skill workflows and their own first-run evidence. It covers 100 tasks with repeated subproblems, fixtures, automated or hybrid graders, and runtime-adapter support.
- **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.
- **Empirical findings.** We provide a current cross-runtime result table showing that runtime scaffolding strongly affects absolute scores, and that self-authored skills do not produce monotonic gains. Focused subset checks make the cost tradeoff, skill-content controls, and hybrid-judge robustness explicit.
- **Open artifact.** We release the benchmark tasks, fixtures, graders, scripts, and reported result exports through an anonymous review repository at `https://anonymous.4open.science/r/EvoClawBench-9380/`.

2 Related Work

Agent and software engineering benchmarks. Repository-level benchmarks evaluate whether agents can modify realistic codebases and satisfy executable checks. SWE-Bench evaluates real GitHub issue resolution (Jimenez et al., 2024), while SWE-agent studies agent-computer interfaces for automated 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

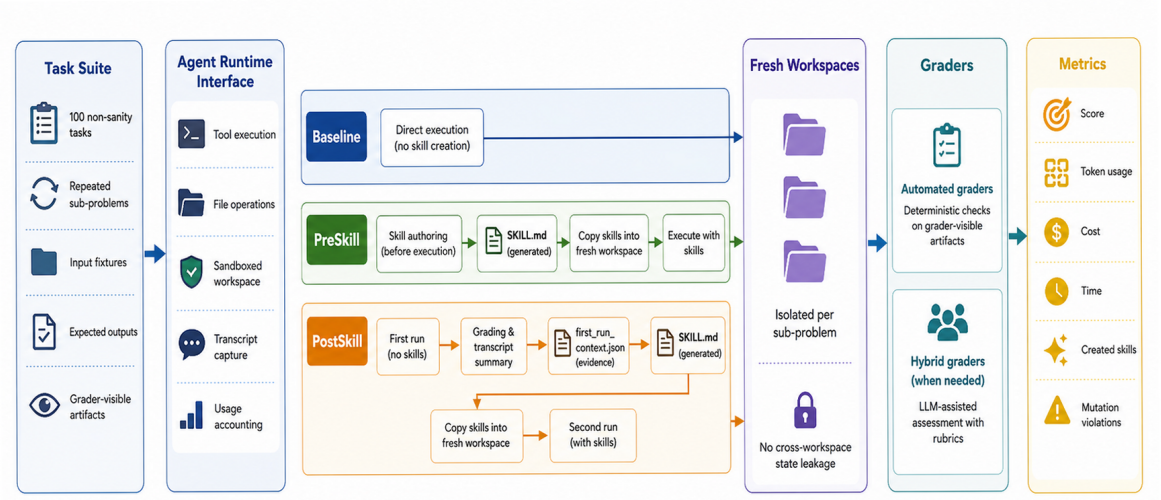


Figure 1: Overview for EvoClawBench. The task-suite panel summarizes the 100 non-sanity tasks in the official suite; Figure 2 breaks down their families and fixture formats. The pipeline routes tasks through a runtime-agnostic agent interface, compares BASELINE, PRESKILL, and POSTSKILL in fresh workspaces,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

benchmarks are useful for measuring task completion, but they do not isolate whether an agent can write reusable skills at runtime and then 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 and Yan, 2026; Wu and 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 is designed to expose whether agent-authored and own-run-derived skills improve task success after accounting for the full cost of creating and reusing skills.

3 EvoClawBench Construction

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

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

Each task run is executed in an isolated 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. For the reported experiments, we use local subprocess execution through the OpenClaw and nanobot runtimes, set `-environment local`, run 32 workers, and execute one benchmark run per task and mode. Because these runs do not use the Docker sandbox, no Docker resource limit is applied. The reported table should therefore be read as local runtime-adapter results rather than as a claim about all supported sandbox configurations.

3.3 Skill Authoring and Reuse

EvoClawBench represents reusable procedures with the standard `SKILL.md` format adopted by the supported runtimes. 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 from the authoring workspace and copies them into fresh execution workspaces for reuse. 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`. This guardrail fixes the experimental condition: PRESKILL and POSTSKILL execution may consult generated skills, but may not revise them while solving the task.

3.4 Runtimes

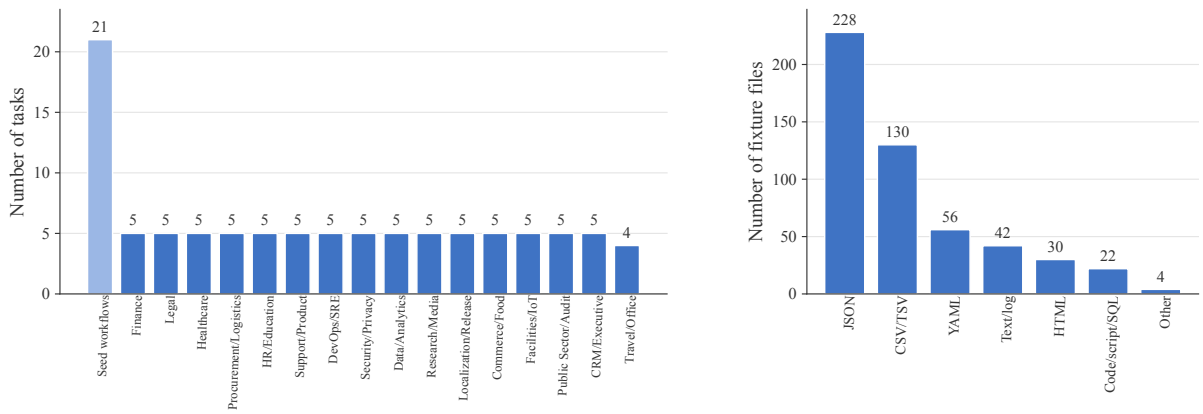
EvoClawBench currently supports OpenClaw and nanobot. OpenClaw is invoked through `openclaw agent -message`, while nanobot is invoked through `nanobot agent -workspace ... -config ... -message ...`. The runtime abstraction keeps task prompts, workspaces, grading, and metrics shared across claws, allowing 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 three strategies.

BASELINE. The agent receives the task prompt and solves it directly. The prompt prefix forbids



(a) Official tasks across benchmark families

(b) Repository-local fixture files across formats

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.

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 $S_b(t)$, $S_p(t)$, and $S_q(t)$ denote the mean task scores for BASELINE, PRESKILL, and POSTSKILL, respectively. The primary execution-only ratios against baseline are:

$$R_p(t) = \frac{S_p(t)}{S_b(t)}, \qquad R_q(t) = \frac{S_q(t)}{S_b(t)}. \tag{1}$$

When the baseline score is zero, positive candidate scores are reported as infinite improvement and zero-to-zero comparisons are reported as 1.0.

EvoClawBench reports two metric scopes. **Execution-only** compares only the task execution phases: direct baseline execution, preskill reuse execution, and postskill second execution. **End-to-end** includes all workflow cost: skill generation for PRESKILL, first execution and skill summary for POSTSKILL, and 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:

$$E_x = \frac{U_b}{U_x}. \tag{2}$$

Values above 1.0 therefore 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 generated skill improves the final task attempt, whereas end-to-end metrics test whether the full workflow justifies its authoring and summarization overhead. This distinction matters because a skill may preserve task score yet remain unattractive if it substantially increases token use, cost, or wall time.

5 Results

5.1 Experimental Setup

We evaluate five models on the hardened benchmark suite under both OpenClaw and nanobot using local subprocess execution. All runs use `uv run scripts/benchmark.py --mode all --workers 32 --environment local`, with `--runtime openclaw` or `--runtime nanobot`. Each task is run once per mode. Unless otherwise stated, results report the execution-only mean over the benchmark loader's 101 tasks, which include the

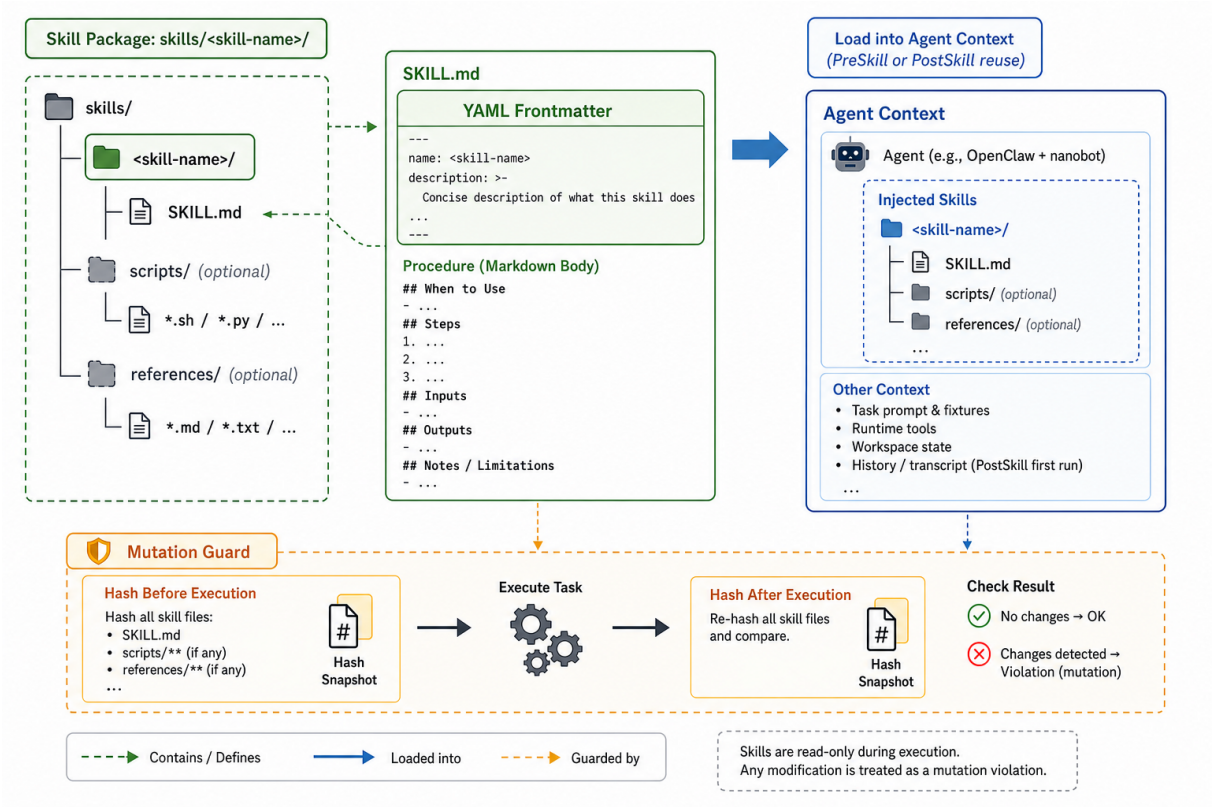


Figure 3: Structure of a reusable skill artifact in EvoClawBench. Generated skills use a `SKILL.md` document with optional supporting scripts or references, are injected into the agent context during reuse, and are protected by before/after mutation checks during execution.

100 official tasks plus `task_00_sanity`. The numerical results in Table 2 are computed from the repository-local benchmark outputs.

5.2 Evaluation Findings

Finding 1: Runtime scaffolding strongly changes the execution regime. Under OpenClaw, direct BASELINE performance remains below 20%, ranging from 18.63 for `GPT-5.4` to 19.99 for `DeepSeek-V4-Pro`. Under nanobot, direct BASELINE scores are substantially higher, ranging from 56.45 for `Qwen3.6-Plus` to 96.13 for `GPT-5.4`. This contrast shows why EvoClawBench reports runtime as part of the experimental condition rather than treating the model identifier alone as sufficient. It should not be read as a leaderboard claim that one runtime is inherently superior, because the runtime also determines prompt wrapping, workspace setup, tool invocation, and usage extraction. The hardened tasks are designed to resist shallow copying of exposed answer fields or schemas: agents must reconcile multi-record evidence, conflicts, revisions, and grader-visible output constraints. As a result, a generated skill must improve the specific model-runtime execution setting rather than merely preserve performance on an already saturated benchmark.

Finding 2: Skill timing is model-dependent rather than monotonic. `GPT-5.4` illustrates the runtime dependence sharply: OpenClaw falls from 18.63 under BASELINE to 1.14 under POSTSKILL, while nanobot stays above 96% in all three modes. `Qwen3.6-Plus` improves under nanobot PRESKILL but regresses under nanobot POSTSKILL, whereas its OpenClaw row remains near 19% for BASELINE and POSTSKILL. `GPT-5.4 mini` improves under OpenClaw skill workflows and under nanobot POSTSKILL, but its nanobot PRESKILL score is slightly below its nanobot BASELINE. These rows do not support a blanket claim that PRESKILL consistently outperforms POSTSKILL, or that either skill workflow reliably improves over direct execution across runtimes.

Finding 3: Skill workflows still add substantial overhead. Even when execution-only quality is similar, skill workflows include authoring or sum-

Table 2: Repository-local EvoClawBench results from current OpenClaw and nanobot three-mode outputs. Scores are execution-only mean task scores in percent over the 101-task loader suite; Skills lists PRESKILL / POSTSKILL created-skill counts.

Runtime	Model	BASELINE	PRESKILL	POSTSKILL	Skills
OpenClaw	`GPT-5.4`	18.63	18.07	1.14	22 / 0
OpenClaw	`Qwen3.6-Plus`	19.16	17.38	19.12	21 / 22
OpenClaw	`DeepSeek-V4-Pro`	19.99	15.06	0.00	21 / 0
OpenClaw	`MiniMax-M2.7`	18.88	17.00	19.10	26 / 20
OpenClaw	`GPT-5.4 mini`	19.11	19.42	19.36	22 / 20
Nanobot	`GPT-5.4`	96.13	96.73	96.17	101 / 100
Nanobot	`Qwen3.6-Plus`	56.45	59.90	54.33	70 / 39
Nanobot	`DeepSeek-V4-Pro`	77.77	4.80	0.99	27 / 0
Nanobot	`MiniMax-M2.7`	90.97	92.90	94.50	99 / 99
Nanobot	`GPT-5.4 mini`	88.71	87.41	89.77	100 / 82

marization phases. For `Qwen3.6-Plus`, end-to-end token-efficiency ratios are 0.38 for PRESKILL and 0.30 for POSTSKILL, meaning the full workflows use substantially more tokens than direct BASELINE execution. For `MiniMax-M2.7`, the corresponding ratios are 0.21 and 0.26; for `GPT-5.4 mini`, they are 0.32 and 0.28. For `GPT-5.4`, PRESKILL has an end-to-end token-efficiency ratio of 0.36; for `DeepSeek-V4-Pro`, the corresponding ratio is 0.40. The current evidence therefore suggests that runtime skill creation must be selective: skill creation needs to clear both a quality bar and an amortized-cost bar.

Finding 4: Skill counts are not sufficient to predict gains. The OpenClaw rows create between 21 and 26 PRESKILL skills, but their PRESKILL scores range from 15.06 to 19.42. The nanobot rows create many more PRESKILL skills, yet the score range remains wide. POSTSKILL skill counts are also not enough to explain outcomes: rows with many POSTSKILL skills can land below, near, or above their own BASELINE scores. This suggests that future evaluations should inspect generated skill content and reuse behavior, not only the number of produced skill directories. Most observed regressions are therefore better explained by the quality and fit of generated skills than by the raw number of skill directories. Figure 4 illustrates why this inspection matters for POSTSKILL. The summary phase receives compact first-run evidence rather than hidden state, but that evidence can still make a generated skill too specific to one execution context. When reused in a fresh workspace, such a skill may preserve an incidental assumption, route attention to the wrong fixture pattern, or skip checks that the second execution still requires. The qualitative case therefore complements the aggregate skill-count result: the question is not only whether a skill exists, but whether the procedure it encodes remains valid under reuse.

5.3 Subset Cost, Amortization, and Robustness Checks

To address cost and judge-sensitivity concerns directly, we ran focused subset experiments on the 12 hybrid tasks using OpenClaw and `GPT-5.4 mini`. Table 5 shows the end-to-end resource tradeoff. On this subset, both skill workflows slightly reduce score while increasing tokens, cost, and wall-clock time. The negative score-per-extra-token values therefore make the cost-quality tradeoff explicit rather than hiding it behind execution-only scores. The subset is not a replacement for the full-suite table; it is a targeted audit of the hybrid tasks where judge cost and scoring robustness are most salient.

Table 6 estimates how many repeated reuses would be needed before the authoring or summary overhead is amortized. For tokens and provider cost, neither PRESKILL nor POSTSKILL breaks even on this subset because the reuse executions are not cheaper than direct execution. Wall-clock time can break even after repeated reuse, but only after 14 PRESKILL executions or 9 POSTSKILL executions under the observed local run times. Thus the subset supports a cost-sensitive interpretation: skills may be worth keeping only when they are reused many times and preserve or im-

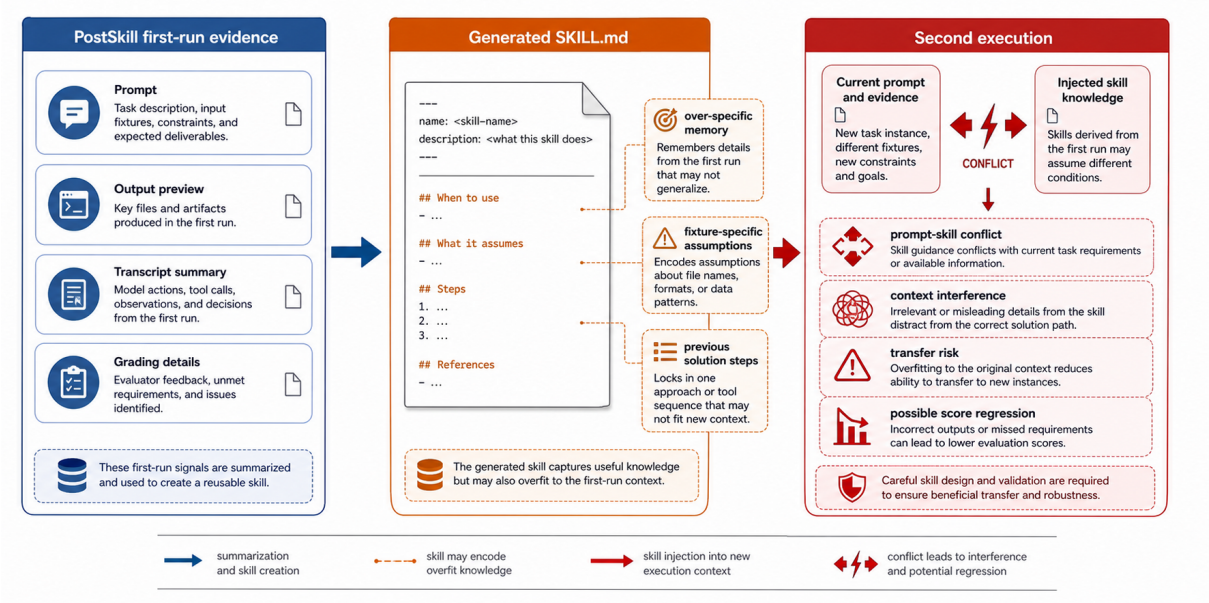


Figure 4: Qualitative failure mode for experience-derived skills. In POSTSKILL, first-run evidence can help produce reusable procedural memory, but it can also encode over-specific assumptions that interfere with the second execution context.

prove quality.

We also ran a 4-task ablation subset (`task_02`, `task_07`, `task_15`, and `task_21`) to separate skill content from reuse scaffolding. Table 7 shows that normal PRESKILL reuse improves the mean score by 4.05 points, but an empty-skill scaffold improves by 19.84 points, while an irrelevant skill improves by only 1.52 points. This pattern is not evidence that empty skills are generally useful; rather, it shows that the reuse prompt, context reset, and run-to-run variance can explain some gains that might otherwise be attributed to skill content. The POSTSKILL comparison is also cautious: normal POSTSKILL reuse is slightly above baseline, but discarding the generated skill after the summary phase falls below baseline. Because these ablations are single subset runs rather than repeated-seed estimates, we use them as diagnostic controls for attribution, not as a new headline result.

Finally, we regraded the 36 task-mode outputs from the 12 hybrid-task subset with alternate judges. Because hybrid scores combine deterministic checks with an LLM judgment, the regrade holds the automated sub-scores fixed and replaces only the LLM-judge component. One alternate judge shows high agreement with the original scores: Pearson correlation is 0.9970, mean absolute score delta is 0.0141, 35 of 36 pairs are within 0.05, and all 36 pairs are within 0.10. The largest disagreement is `task_18_dep_audit` in PRESKILL (+0.0880). As a second check, `GPT-5.4 mini` also preserves the overall trend, though less tightly: Pearson correlation is 0.9512, mean absolute score delta is 0.0609, and 30 of 36 pairs are within 0.10. The largest `GPT-5.4 mini` disagreement is `task_11_web_extraction` in BASELINE (-0.3972). These subset audits reduce the concern that the reported hybrid-task pattern is an artifact of a single judge, while the remaining disagreements still motivate auditing high-stakes leaderboard claims. They do not remove judge dependence as a limitation; instead, they show how future benchmark releases should report both original and alternate-judge agreement when hybrid grading affects a result.

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: neither the pre-execution PRESKILL control nor the own-run POSTSKILL condition is uniformly dominant, and Figure 4 shows how own-run summaries can overfit a second execution context. Future skill systems should use selective creation policies and validation before reuse, while benchmark reports should include provenance, mutation checks, and end-to-end resources.

Limitations

Transfer scope. The v1 protocol evaluates POSTSKILL by rerunning the same task with the same fixtures, which measures within-task skill distillation but does not yet test transfer to unseen related tasks. This means the reported results should not be interpreted as evidence that generated skills generalize across new tasks, users, or domains.

Experimental coverage. The task suite covers many practical office, data, coding, DevOps, security, and domain-document workflows, but it is still smaller than large public agent benchmarks. The current reported table covers two runtimes under one local execution configuration with one run per task and mode. Repeated seeds, additional models, confidence intervals, Docker resource limits, and more runtime replications remain future work.

Grading dependence. Hybrid grading tasks may depend on the selected judge model, although automated checks are used whenever feasible. Future result files should store the judge model explicitly because absolute scores may change under a different judge even if the task prompts and artifacts are unchanged.

Runtime scope. The current implementation and reported experiments cover OpenClaw and nanobot. Broader runtime coverage is still needed before claiming conclusions across a wider set of agent runtimes.

Ethical Considerations

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.

Artifacts and Data 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/`. The project metadata in `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.

References

Saaket Agashe, Kyle Wong, Vincent Tu, Jiachen Yang, Ang Li, and Xin Eric Wang. 2025. Agent s2: A compositional generalist-specialist framework for computer use agents. *Preprint*, arXiv:2504.00906.

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

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 Mdry. 2025. Mle-bench: Evaluating machine learning agents on machine learning engineering.

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

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. 2026. Memp:

Exploring agent procedural memory. *Preprint*, arXiv: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. 2024. Magentic-one: A generalist multi-agent system for solving complex tasks. *Preprint*, arXiv:2411.04468.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pages 54107–54157.

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, and 22 others. 2026. Skillsbench: Benchmarking how well agent skills work across diverse tasks. *Preprint*, arXiv:2602.12670.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, and 1 others. 2024. Agentbench: Evaluating llms as agents. 2024:52989–53046.

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, and 66 others. 2026. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *Preprint*, arXiv:2601.11868.

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

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. 2026. Trace2skill: Distill trajectory-local lessons into transferable agent skills. *Preprint*, arXiv: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. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis.

Cursor Research, Aaron Chan, Ahmed Shalaby, Alexander Wettig, Aman Sanger, Andrew Zhai, Anurag Ajay, Ashvin Nair, Charlie Snell, Chen Lu, and 1 others. 2026. Composer 2 technical report. *arXiv preprint arXiv:2603.24477*.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. 2024. 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)*, pages 16022–16076.

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

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

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, and 5 others. 2025. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations*, volume 2025, pages 65882–65919.

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

Haoyang Xie, Xinyuan Wang, Yancheng Wang, Puda Zhao, and Feng Ju. 2026. From history to state: Constant-context skill learning for llm agents. *Preprint*, arXiv: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. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments.

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, and 2 others. 2025. TheAgentCompany: Benchmarking LLM agents on consequential real world tasks.

Renjun Xu and Yang Yan. 2026. Agent skills for large language models: Architecture, acquisition, security, and the path forward.

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

Min Yang, Jinghua Piao, Xu Xia, Xiaochong Lan, Jiaju Chen, Yongshun Gong, and Yong Li. 2026a. Skillmaster: Toward autonomous skill mastery in llm agents. *Preprint*, arXiv: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. 2026b. Autoskill: Experience-driven lifelong learning via skill self-evolution. *Preprint*, arXiv:2603.01145.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models.

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

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, and 1 others. 2024. Webarena: A realistic web environment for building autonomous agents. 2024:15585–15606.

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`.

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 3: 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 4: 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 it is instructed not to redo the task and not to write final task outputs under `outputs/`. Skills produced in the summary workspace are copied into a fresh second execution workspace, where mutation checks are applied before and after execution.

C Subset Audit Details

The following tables provide the detailed subset evidence summarized in Section 5.3.

Table 5: End-to-end cost on the 12-task hybrid subset. Scores are percentages. "Delta/1M tok." is score-point change per one million extra tokens relative to BASELINE.

Mode	Score	Delta	Tokens	USD	Time h	Requests	Delta/1M tok.
BASELINE	80.96	0.00	1.94M	1.262	0.305	84	–
PRESKILL	79.77	-1.19	7.07M	3.745	0.689	270	-0.232
POSTSKILL	79.21	-1.75	7.71M	4.128	0.833	306	-0.303

Table 6: Amortized break-even reuse count on the same subset. "Never" means the reuse execution is not cheaper than direct BASELINE execution for that resource.

Mode	Tokens	USD	Time
PRESKILL	never	never	14
POSTSKILL	never	never	9

Table 7: Skill-content ablation on a 4-task representative subset. Scores and deltas are percentages; resources 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 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 8: 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_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

E 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 9: 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_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	Name	Prompt focus	Grader-visible output surface	Checks
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_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	Name	Prompt focus	Grader-visible output surface	Checks
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_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	Name	Prompt focus	Grader-visible output surface	Checks
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

F 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 10: 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_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	Name	Grade	Check surface	Representative criteria
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_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	Name	Grade	Check surface	Representative criteria
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_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_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	Name	Grade	Check surface	Representative criteria
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_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	Name	Grade	Check surface	Representative criteria
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_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_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	Name	Grade	Check surface	Representative criteria
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_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	Name	Grade	Check surface	Representative criteria
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.

G Task Families and Fixture Mechanics

The current suite combines the original 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 still mapping them to distinct artifacts.

Table 11: 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.

Family	Tasks	Sub-problems	Fixture and output mechanics
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.
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.

H Generated Family Design Matrix

Table 12 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 12: 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.

I 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 13: 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.

J Representative Output Schemas

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

Table 14: 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
Travel/Office	task_97: Calendar Conflict Resolution	conflicts, exceptions, action_items, rule_labels, resolved

K Suite Distributions

Table 15: Grading types.

Type	Tasks
automated	88
hybrid	12

Table 16: Sub-problem counts.

Sub-problems	Tasks
5	98
6	2

Table 17: 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

L Skill Artifact Structure

Generated skill directories are discovered under `skills/`. The seeded `skills/skill-creator` bundle is excluded from created-skill metrics. A generated skill must include `SKILL.md`; optional scripts and

references are counted when present. Before reuse execution, non-seeded skill files are hashed. The same files are hashed after execution, and any add, delete, or content change is recorded as a mutation violation for that phase.

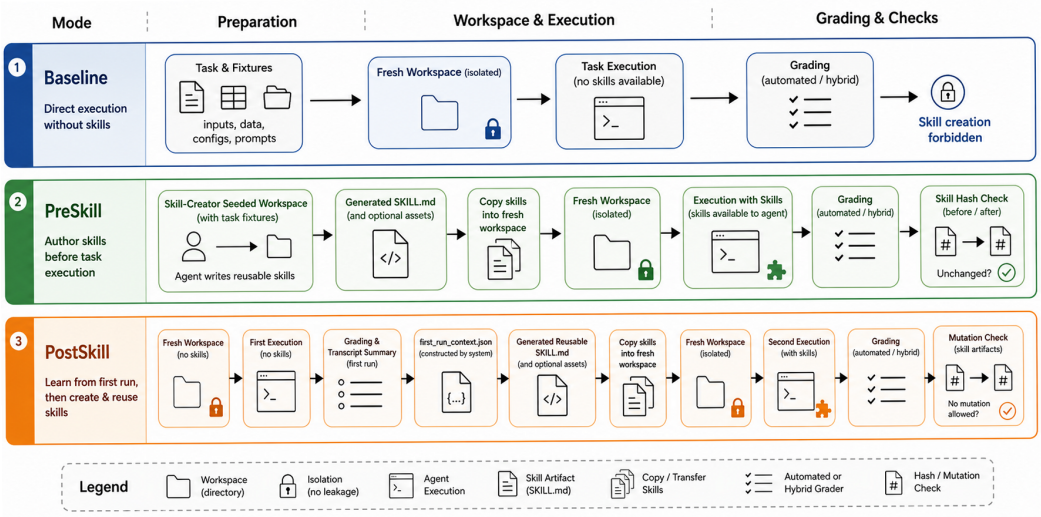


Figure 5: Three-mode evaluation protocol. BASELINE directly executes a task without skill creation; PRESKILL first creates task-specific skills and then reuses them in a fresh workspace; POSTSKILL summarizes reusable skills from first-run evidence before a second execution. Skill reuse phases are checked for mutation violations.

M 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 18: 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.

N 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. Runs should be reported consistently across BASELINE, PRESKILL, and POSTSKILL so that score, cost, and skill-reuse comparisons refer to the same benchmark protocol.