STUDYING WITHOUT A SYLLABUS: TASK-AGNOSTIC ENVIRONMENT PREPROCESSING

Anonymous authors
Paper under double-blind review

ABSTRACT

Before an LLM agent tackles tasks in a new environment, the environment's corpora and tools can be inspected to construct reusable resources such as indices, scripts, or procedural guidance. Most automated adaptation methods rely on task examples, trajectories, or evaluation feedback to decide what to build. Existing task-agnostic approaches avoid this supervision but commit in advance to strategies designed for particular environment types. We ask whether, before test time and without knowledge of the downstream task distribution, a meta-agent can instead study an unfamiliar environment without a syllabus and choose how to prepare it. We formalize task-agnostic environment preprocessing and compare unaided and archive-equipped META-AGENTs with fixed synthetic-practice and corpus-processing methods across six heterogeneous benchmarks. Studying significantly improves downstream performance in 21 of 24 method–benchmark comparisons. A META-AGENT variant leads on five benchmarks, while fixed corpus processing leads on the most corpus-heavy benchmark. The same artifacts provide less consistent gains when used by a different task-solving agent. Studied artifacts can also reduce the sampling needed to reach a given score, showing how reusable preparation can shift computation from repeated test-time attempts to pre-task study.

1 INTRODUCTION

LLM agent performance depends substantially on choices beyond the underlying model: prompts, tools, context and memory management, and the organization of corpora, databases, and external services (Le Sellier De Chezelles et al., 2024; Weng, 2026; Yang et al., 2024; Yao et al., 2026). Consider an agent that performs retrieval over a large filesystem. If files are unnamed and disorganized, the agent receives no clues about file content and must resort to expensive, exhaustive search. Consequently, practitioners routinely adapt components of the agent *harness* and *environment* to tailor them to the tasks the agent is expected to perform. This has led to increased interest in automated approaches to optimizing agents for novel tasks.

Existing automated adaptation methods typically use information about the expected test-time task distribution to guide modifications. Prompt optimizers such as DSPy (Khattab et al., 2024) and GEPA (Agrawal et al., 2026) use *task examples* and *evaluation feedback*. VeRO (Ursekar et al., 2026) and Meta-Harness (Lee et al., 2026) modify the entire harness as code using similar signals. Adaptive memory systems such as AWM (Wang et al., 2025), ACE (Zhang et al., 2026b), and Dynamic Cheatsheet (Suzgun et al., 2026) extract reusable knowledge and procedures from *task trajectories* with or without *labels*. Such supervisory signals may be unavailable when an agent first encounters a new environment, particularly when representative tasks are costly to obtain. This cold start motivates *task-agnostic adaptation* of agents and environments prior to test time.

Several lines of work have addressed the problem of agent and environment adaptation before test time. One line substitutes knowledge of the true task distribution with self-generated exploratory practice. SPICE (Liu et al., 2025) uses corpus-grounded self-play to generate a curriculum for parameter adaptation, while PREPING (Choi et al., 2026b) generates synthetic tasks in tool environments and distills their trajectories into a procedural playbook. In corpus environments, retrieval structures can be prepared offline to make document collections more amenable to search by agents: RAPTOR (Sarthi et al., 2024) recursively clusters and summarizes documents, GraphRAG (Edge et al., 2024) constructs

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

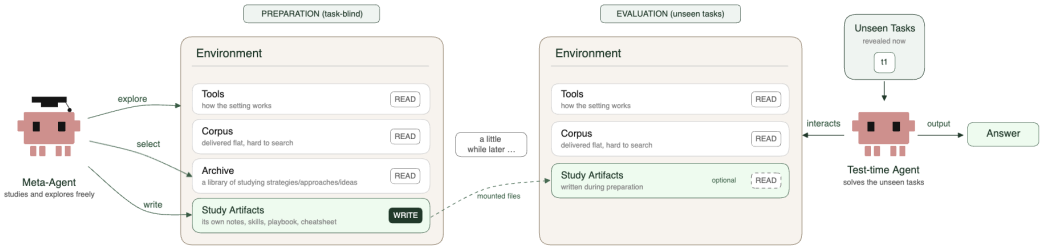


Figure 1: An instance of our META-AGENT W/ ARCHIVE pipeline. The environment E is a sandboxed container with a filesystem. The base environment containing benchmark-specific artifacts such as tools and corpora is shared across study and test time. Before test time, the meta-agent explores E using strategies from a mounted archive. It writes study artifacts into a volume that is later mounted into the test-time agent's sandbox. The test-time agent uses the study artifacts and the pre-existing benchmark-specific artifacts to solve unseen tasks.

a knowledge graph and community summaries, and CORPUS2SKILL (Sun et al., 2026) compiles a corpus into a navigable skill hierarchy. Closest to our framing, Machine Studying (Li et al., 2026) treats studying as an explicit pre-task process and evaluates how well an agent can construct reusable context from a corpus without downstream tasks. Each of these methods commits to a preprocessing strategy depending on the environment's structure. With many methods available and suited to different environment types, we ask whether an *open-ended studying system* can choose how to preprocess an environment as a function of what it contains.

Specifically, we ask whether a META-AGENT can explore an environment without knowledge of the downstream task distribution and produce artifacts that improve a frozen solver at test time. The studier may inspect the environment and use feedback from synthetic practice, but receives no held-out target tasks, trajectories, verifier outputs, or evaluation feedback before its artifacts are frozen. It may add arbitrary file-based artifacts—including indices, knowledge bases, scripts, tools, or guidance—through the solver's existing harness interfaces; the solver's weights and harness otherwise remain fixed. We compare two META-AGENT variants with the fixed task-agnostic workflows PREPING and CORPUS2SKILL.

Across six benchmarks spanning 36 unique environments, 21 of 24 studying method–benchmark comparisons significantly improve over NO STUDY. A META-AGENT variant achieves the highest downstream Avg@3 reward on five of six benchmarks, while the specialized CORPUS2SKILL pipeline leads on BCP-G. Comparisons with a task-blind selector suggest that the archive-equipped agent's value is not explained by selecting a single fixed workflow alone. When the frozen artifacts are transferred to a different solver agent, their benefits become less consistent, indicating that artifact utility depends partly on the solver that consumes them.

We make four contributions:

1. We formalize **task-agnostic environment preprocessing**, where systems construct reusable artifacts without held-out target tasks, trajectories, or evaluation feedback.
2. We introduce **open-ended studying META-AGENTS** that choose how to explore an environment and what to create, with unaided and archive-assisted variants.
3. We show that **studying generally improves downstream performance** and that open-ended studying is competitive across heterogeneous environments.
4. We analyze study behavior, test-time sampling, and cross-solver artifact transfer.

2 RELATED WORK

Task-informed adaptation of agent harnesses. Downstream task information can guide both what an agent system should change and whether the change helped. DSPy (Khattab et al., 2024), GEPA (Agrawal et al., 2026), and PromptBreeder (Fernando et al., 2024) optimize instructions or demonstrations against task examples and evaluation signals, while VeRO (Ursekar et al., 2026), Meta-Harness (Lee et al., 2026), and Meta Agent Search (Hu et al., 2025) optimize executable

Method	BCP-G	OfficeQA	Harvey LAB	DABStep	APEX-Agents	AppWorld
NO STUDY	0.197 ± 0.009	0.549 ± 0.020	0.260 ± 0.008	0.353 ± 0.010	0.183 ± 0.005	0.576 ± 0.016
LLM selector	0.471 ± 0.015↑	0.535 ± 0.028	0.227 ± 0.006↓	0.384 ± 0.031↑	0.221 ± 0.001↑	0.603 ± 0.028↑
PREPING	0.285 ± 0.012↑	0.609 ± 0.020↑	0.299 ± 0.013↑	$\underline{0.464 \pm 0.060}$↑	$\underline{0.231 \pm 0.010}$↑	0.645 ± 0.036↑
CORPUS2SKILL	$\mathbf{0.471 \pm 0.015}$↑	0.535 ± 0.028	0.227 ± 0.006↓	0.324 ± 0.009↓	0.221 ± 0.001↑	0.603 ± 0.028↑
META-AGENT W/O ARCHIVE	0.247 ± 0.016↑	$\mathbf{0.633 \pm 0.014}$↑	$\underline{0.361 \pm 0.007}$↑	0.400 ± 0.006↑	0.224 ± 0.004↑	$\mathbf{0.688 \pm 0.057}$↑
META-AGENT W/ ARCHIVE	$\underline{0.286 \pm 0.021}$↑	$\underline{0.615 \pm 0.004}$↑	$\mathbf{0.366 \pm 0.012}$↑	$\mathbf{0.529 \pm 0.097}$↑	$\mathbf{0.233 \pm 0.011}$↑	$\underline{0.665 \pm 0.042}$↑

Table 1: **Avg@3 downstream reward.** Studying-method SDs measure spread across three study runs. The NO STUDY SD is $s/\sqrt{3}$. The selector SD uses total variance across selections and reused study arms. Its row reuses existing arms and is excluded from best-value styling. ↑ and ↓ mark significant gains and losses relative to NO STUDY under uncorrected 95% paired task bootstrap intervals (Table 7). Best and second-best studying methods are bold and underlined.

harnesses or code-defined agents. Memory and context systems, including AWM (Wang et al., 2025), Memp (Fang et al., 2026), ACE (Zhang et al., 2026b), Dynamic Cheatsheet (Suzgun et al., 2026), Reflexion (Shinn et al., 2023), ExpeL (Zhao et al., 2024), MemGPT (Packer et al., 2023), A-MEM (Xu et al., 2025), and Mem0 (Chhikara et al., 2025), instead distill knowledge, procedures, or code from observed interactions. Other work relocates this processing within the interaction lifecycle: PROACT (Hu et al., 2026) acts between user turns, IDLESPEC (Choi et al., 2026a) during tool-call latency, and AUTO-DREAMER (Ye et al., 2026a) after experience has accumulated. These methods use task examples, trajectories, or feedback to decide what to change. Our setting asks what to construct before such task information is available.

Task-agnostic adaptation of agent harnesses and environments. Without task information, a system must derive its objective from what is available before use. Sleep-time Compute (Lin et al., 2025) studies how preprocessing a supplied context can displace computation after an unknown query arrives. Machine Studying (Li et al., 2026) fixes the input modality to a corpus and the candidate interventions to self-supervised training, synthetic-data fine-tuning, and amortized note construction. Explore-then-Act (Ye et al., 2026b) trains an exploration policy with environment-specific checkpoint coverage alongside task rewards, then summarizes a goal-free rollout for later task execution. Reward-free self-evolution (Zhang et al., 2026a) trains a web exploration policy by selecting trajectories according to the downstream utility of their Markdown world knowledge, then explores without rewards at inference. Retrieval systems assume that useful preprocessing takes the form of queryable corpus structure, from BM25 (Robertson & Zaragoza, 2009), DPR (Karpukhin et al., 2020), and retrieval-augmented generation (Lewis et al., 2020) to the structures built by RAPTOR (Sarthi et al., 2024), GraphRAG (Edge et al., 2024), CORPUS2SKILL (Sun et al., 2026), and PANINI (Rajesh et al., 2026). Practice-based systems use self-generated interaction to acquire executable skills in Voyager (Wang et al., 2024), synthesize adaptation data in SPICE (Liu et al., 2025) and Learn-by-interact (Su et al., 2025), or produce a procedural playbook in PREPING (Choi et al., 2026b). PREPING's guided-exploration baseline is the nearest operational analogue to META-AGENT W/O ARCHIVE because both explore before target tasks and record reusable guidance. We place these commitments under a common protocol and test whether a study agent can choose and combine preprocessing workflows based on the environment.

3 TASK-AGNOSTIC ENVIRONMENT PREPROCESSING

In our setup, an agent π operates in an environment E. An agent $\pi = (m, H)$ is a tuple of an LLM m and a harness H. The environment E is a shared runtime and collection of resources required to complete a set of tasks. Practically, E is implemented as an isolated Docker sandbox compatible with the Harbor framework (Harbor Framework Team, 2026). The harness H represents the program that invokes the model within the environment. H maintains context, registers and invokes tools, and drives the interaction loop between the model and environment.

Each environment E admits a space $\mathcal{T}_E$ of feasible downstream tasks. Each task $t = (x_t, r_t) \in \mathcal{T}_E$ consists of a prompt x_t and a reward function r_t that scores the output of an agent prompted with x_t. A benchmark provides a finite held-out test set. We write D_E for the uniform empirical distribution over that set. Let π_{solver} denote a frozen downstream agent. Its aggregate reward on an environment

E' and test distribution D_E is then

$$R_{\pi_{\text{solver}}}(E', D_E) = \mathbb{E}_{t\sim D_E}\left[r_t(\pi_{\text{solver}}(x_t, E'))\right], \tag{1}$$

where E' is either the original environment E or a version of it modified before test time.

The goal of a task-agnostic *studying system* S is to improve $R_{\pi_{\text{solver}}}(E, D_E)$ without observing D_E: S does not receive samples from D_E or signals derived from held-out target evaluation. In our work, S uses the frozen solver and the original environment to output a modified environment E_{studied}, i.e. $S\colon \Pi \times \mathcal{E} \to \mathcal{E}$, where Π and $\mathcal{E}$ are the spaces of agents and environments. We hold H fixed, although the interventions use its existing interfaces. S may add file-based artifacts such as indices, knowledge bases, scripts, or executable tools while retaining the original resources in E. Some methods may also deliver artifacts through the prompt or restrict how the solver accesses original resources. Appendix A.3.1 details these delivery mechanisms. We seek a system S that maximizes the expected aggregate reward of the frozen solver across environments.

$$S^\star = \arg\max_S \ \mathbb{E}_{E\sim p}[R_{\pi_{\text{solver}}}(S(\pi_{\text{solver}}, E),\ D_E)] \tag{2}$$

There is no a priori restriction on the structure or logic of S. It may be a filesystem preprocessing pipeline as in CORPUS2SKILL, a fixed orchestration of specialized agents as in PREPING, or an open-ended agentic process like our META-AGENT variants.

4 METHODS

In practice, what S should do depends on the environment. A large corpus may benefit from indexing and summarization, whereas a broad tool surface may require skills or instructions distilled from synthetic practice or self-play. Appendix Figure 3 describes the breadth of environments in our benchmark suite using counts of files and tools exposed to the solver.

One of our aims is to compare *open-ended* study strategies with *fixed* ones. We call a studying system open-ended when its studying procedure and artifact type are not fixed in advance. We consider two representative *fixed* baseline systems with differing preprocessing procedures, each designed for a different environmental niche. In contrast, our two META-AGENT variants use a study agent π_{study} that can explore an environment and modify it in an *open-ended* way. We describe these methods here. Appendix A.3.1 reports additional implementation details.

4.1 BASELINES: FIXED STUDYING SYSTEMS

PREPING. PREPING (Choi et al., 2026b) performs synthetic practice organized into cycles to generate a procedural playbook for a frozen solver. In each cycle, a *Proposer* generates a batch of tasks based on the environment, a practice *Solver* attempts them in the target harness, and a *Validator* scores feasibility and completion. A reflector and curator then distill lessons from successes and failures into a playbook containing strategies, pitfalls, interface notes, and reusable code. The number of cycles, tasks per cycle, and per-task execution limit determine the amount of synthetic practice and study cost. At task time, playbook entries are retrieved for each task instruction and prepended to it. Although PREPING *uses* agents, their roles, invocation order, and output format are fixed.

CORPUS2SKILL. CORPUS2SKILL (Sun et al., 2026) restructures a document corpus into a navigable directory of skills. Its pipeline summarizes and embeds documents, clusters them into a labeled hierarchy, and constructs indexes that map topics to source documents. The pipeline follows a fixed end-to-end procedure without adapting its strategy in response to runtime feedback. The amount of source content processed and the resulting hierarchy are controlled by a document-length limit, whether document summaries are generated, the hierarchy branching ratio, the maximum number of top-level clusters, and the minimum cluster size. These choices affect study cost. At task time, CORPUS2SKILL disables direct corpus search and enforces skill navigation. Unlike PREPING, this system compiles declarative content without generating or executing synthetic tasks.

4.2 OPEN-ENDED STUDYING SYSTEMS

META-AGENTS. Both META-AGENT variants instantiate a study agent π_{study} that explores the environment, chooses a studying procedure, and constructs file-based artifacts it expects to help

a future solver. META-AGENT W/O ARCHIVE is prompted to study without a supplied workflow and write its artifact set under a specified directory. META-AGENT W/ ARCHIVE receives a seed archive containing PREPING, CORPUS2SKILL, and the META-AGENT W/O ARCHIVE workflow as executable skills with descriptions. The skills are mounted in the sandbox and accessible through standard shell tools as shown in Figure 1. The agent may invoke, combine, or ignore their outputs when constructing the final artifacts. Its final artifact set includes a `prefix.md` file that is injected into the solver's system prompt.

4.3 WORKFLOW-SELECTION BASELINE

Task-blind LLM selector. To distinguish archive-assisted META-AGENT studying from one-shot workflow selection, we evaluate a non-agentic LLM selector. It receives a text description of the environment and the three archived workflows, but no target-task information or tool access, and chooses exactly one workflow or NO STUDY. Unlike META-AGENT W/ ARCHIVE, it cannot explore the environment, execute and compare multiple workflows, or combine and revise their outputs. Appendix A.2.2 provides the full protocol.

Method	BCP-G	OfficeQA	Harvey LAB	DABStep	APEX-Agents	AppWorld
NO STUDY	**0.921 ± 0.001**	0.783 ± 0.005	0.561 ± 0.003	0.557 ± 0.025	**0.514 ± 0.002**	0.812 ± 0.017
PREPING	0.874 ± 0.013↓	0.774 ± 0.018	**0.592 ± 0.005**↑	0.542 ± 0.018	0.498 ± 0.006	0.818 ± 0.005
CORPUS2SKILL	0.851 ± 0.007↓	0.778 ± 0.002	0.553 ± 0.005	0.495 ± 0.013↓	0.485 ± 0.007↓	0.847 ± 0.010↑
META-AGENT W/O ARCHIVE	0.917 ± 0.006	**0.785 ± 0.009**	0.582 ± 0.003	0.612 ± 0.039↑	0.507 ± 0.004	**0.875 ± 0.006**↑
META-AGENT W/ ARCHIVE	0.847 ± 0.017↓	0.785 ± 0.013	0.591 ± 0.008↑	**0.625 ± 0.051**↑	0.501 ± 0.005	0.849 ± 0.015↑

Table 2: **Avg@3 reward after transfer to OpenCode with GLM-5.3-Flash.** Frozen artifacts are reused without further study. Studying-method SDs measure spread across three study runs. The NO STUDY SD uses the three repetition-level benchmark means. ↑ and ↓ mark significant gains and losses relative to NO STUDY under uncorrected 95% paired task bootstrap intervals (Appendix Table 6). Best values are bold and second-best values are underlined.

5 EXPERIMENTAL SETUP

We conduct two experiments across the same six benchmarks. The primary experiment compares four studying methods **PREPING**, **CORPUS2SKILL**, **META-AGENT W/O ARCHIVE**, and **META-AGENT W/ ARCHIVE** against NO STUDY using a frozen Claude Code solver with Claude Haiku 4.5. The transfer experiment reuses the resulting artifacts without further study and evaluates them with GLM-5.3-Flash through OpenCode.

Benchmarks. We evaluate on six agentic benchmarks spanning diverse environment types. **BCP-Grep (BCP-G)** adapts BrowseComp-Plus (Chen et al., 2026). It retains the 830 questions and fixed 100,195-document corpus but exposes each document as a file for shell-based search rather than through the original BM25 retriever. **OfficeQA** (Opsahl-Ong et al., 2026) and **Harvey LAB** (Grupen et al., 2026) also expose document collections. OfficeQA contributes 246 tasks. For Harvey LAB, we use only its 250 firm-knowledge tasks, which share the same fictional firm's document-management system. **DABStep** (Egg et al., 2025) contributes 450 tasks over seven files that combine task data, reference data needed to compute answers, and protocol instructions for processing that data using Python. **APEX-Agents** (Vidgen et al., 2026) contributes 452 tasks grouped into 31 worlds, each with its own shared filesystem and tools. We study each world independently. The reported score therefore aggregates 31 studied environments. **AppWorld** (Trivedi et al., 2024) contributes 168 tasks and exposes a multi-application tool interface whose database state changes across target tasks. Figure 3 and Table 3 report the observed corpora and tools.

Solver Agents. The primary solver uses Claude Haiku 4.5 at temperature 1.0 through Claude Code (Anthropic, 2026). The transfer solver uses GLM-5.3-Flash through OpenCode 1.18.18.

Study Method Harnesses and Models. We use Claude Code as the coding harness for all agentic components in every studying methods. Claude Opus 4.8 performs META-AGENT study and PREPING proposal, validation, reflection, and curation. Claude Haiku 4.5 executes PREPING synthetic tasks, every primary downstream task, and CORPUS2SKILL document-card generation.

Claude Sonnet 4.6 performs CORPUS2SKILL cluster summarization, labeling, repartitioning, and entity extraction. PREPING and CORPUS2SKILL model assignments match the methods' released defaults. We similarly retain `Qwen3-Embedding-8B` for CORPUS2SKILL embeddings and `text-embedding-3-small` for PREPING playbook retrieval.

We run PREPING for five cycles of 10 synthetic tasks per environment. The original work runs 10 cycles, but its coverage and entropy curves show diminishing gains after roughly five (Choi et al., 2026b). For CORPUS2SKILL, we use the published default configuration where possible, with benchmark-specific adjustments to hierarchy branching, document length, and tree compaction. Additional implementation details and prompts appear in Appendices A.3.1 and A.3.2.

Repetitions and rollout metrics. Each studying system is run independently $N_{\text{study}} = 3$ times per environment, producing three artifact sets. For APEX-Agents, this means three study runs for each of its 31 world environments. For every artifact set, downstream evaluation is independently repeated $N_{\text{eval}} = 3$ times. We reserve K for the number of those task-time repeats aggregated by Avg@K or Best@K. The primary tables use $K = 3$. With D_E defined as in §3, let $r_{s,t,j}$ be the reward from rollout j on task t using the artifact produced by study iteration s. We report

$$\begin{aligned} \text{Avg@}K &= \frac{1}{N_{\text{study}} K} \sum\nolimits_{s=1}^{N_{\text{study}}} \mathbb{E}_{t \sim D_E} \left[\sum_{j=1}^{K} r_{s,t,j} \right], \\ \text{Best@}K &= \frac{1}{N_{\text{study}}} \sum\nolimits_{s=1}^{N_{\text{study}}} \mathbb{E}_{t \sim D_E} \left[\max_{1 \le j \le K} r_{s,t,j} \right] \end{aligned} \tag{3}$$

Avg@K averages all K rollout rewards per task, whereas Best@K selects the best of the K rollouts per task. Both then average over tasks and study iterations. We report the sample standard deviation across the N_{study} study-run-specific task macro-averages, so uncertainty reflects variation across study runs rather than individual tasks or rollouts.

NO STUDY has no artifact-set dimension and is evaluated with $N_{\text{eval}} = 10$ independent task-time repetitions. For $K = 3$, both Avg@3 and Best@3 are computed over all $\binom{10}{3} = 120$ subsets of three repetitions. Avg@3 averages the subset means, whereas Best@3 averages the subset maxima. For Avg@3, the reported SD is $s/\sqrt{3}$, where s is the sample SD of the ten per-repetition benchmark means. For Best@3, the reported SD corrects the spread across the 120 subset-level benchmark estimates for overlap among subsets. Both SDs describe variation in a three-repetition estimate rather than variance across study runs.

Metrics and ranking. For each target task, we use the benchmark's original evaluation measure except on Harvey LAB. Its strict all-criteria-pass reward was uniformly low for Claude Haiku 4.5 across methods in preliminary experiments, so we report the dense rubric criterion pass rate. The primary metrics are study cost C_{study}, downstream reward r, and downstream inference cost $C_{\text{inference}}$, with costs in API dollars. Study cost includes generation, control, embeddings, and compilation. We assess differences from NO STUDY using uncorrected 95% paired task-bootstrap intervals. We call a method the lead when it has the highest unrounded mean among the four studying methods. We call methods tied only when their unrounded means are equal. We score each selector choice using the corresponding existing method evaluation rather than rerunning the workflow. The selector is therefore not an independently evaluated method and is excluded from studying-method rankings.

Study cost. Tables 1 and 4 report full-configuration performance without a shared monetary study budget. Study costs vary with the method and environment and with the archive workflows selected by META-AGENT W/ ARCHIVE. Table 10 reports these costs by benchmark.

6 RESULTS

RQ1. How do studying methods compare across environments? Studying generally improves downstream performance. Across the four independently run studying methods and six benchmarks, 21 of 24 Avg@3 comparisons show significant gains over NO STUDY. PREPING and both META-AGENT variants improve on every benchmark. The three exceptions involve CORPUS2SKILL, which is indistinguishable from NO STUDY on OfficeQA and significantly worse on Harvey LAB and

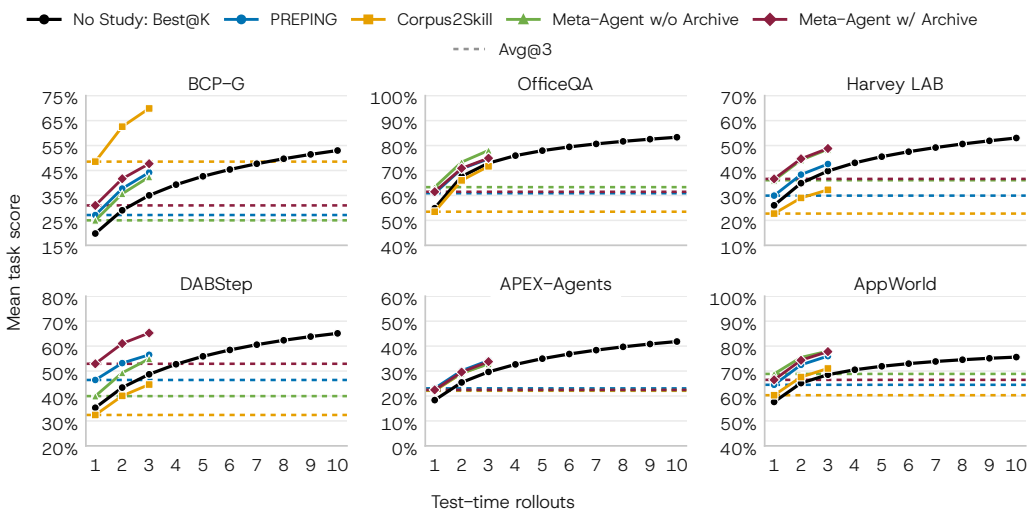


Figure 2: **Test-time scaling with and without study.** NO STUDY Best@K curves are compared with Best@K curves and Avg@3 reference levels for each study method. Best@K is an oracle upper bound. Harvey LAB uses dense verifier pass rate.

DABStep. A META-AGENT variant leads five of six benchmarks: META-AGENT W/O ARCHIVE on OfficeQA and AppWorld, and META-AGENT W/ ARCHIVE on Harvey LAB, DABStep, and APEX-Agents. CORPUS2SKILL leads on BCP-G. Thus, studying is broadly useful, although a specialized fixed strategy can fail outside its niche. These are unconstrained rather than cost-matched comparisons. Tables 4, 7, 9, and 10 report complementary Best@3 results, paired intervals, and costs.

RQ2. What is the value of agentic access to a workflow archive? The workflow archive provides clear gains in some environments, and its utility is not reducible to choosing a single workflow. We assess this through three comparisons. First, relative to META-AGENT W/O ARCHIVE, META-AGENT W/ ARCHIVE yields significant Avg@3 gains on BCP-G and DABStep; the iteration-resampled confidence intervals include zero on the other four benchmarks (Table 8). Second, we compare META-AGENT W/ ARCHIVE against a benchmark-level oracle that retrospectively selects the best of the three archived workflows when each is run independently. META-AGENT W/ ARCHIVE exceeds this reference on Harvey LAB, DABStep, and APEX-Agents. On OfficeQA and AppWorld, it ranks second behind META-AGENT W/O ARCHIVE, with no clear difference under iteration-resampled confidence intervals (Table 8). BCP-G is the only substantial miss: the agent does not retain CORPUS2SKILL output, although CORPUS2SKILL is the strongest workflow there. Third, we compare META-AGENT W/ ARCHIVE with the task-blind LLM selector to test whether one-shot workflow selection captures the value of agentic archive use. The selector performs best on BCP-G, while META-AGENT W/ ARCHIVE outperforms it on the other five benchmarks. These comparisons suggest that exploring the environment and executing, inspecting, or combining workflow outputs can add value beyond selection alone, although they do not isolate the contribution of each capability.

RQ3. Do studied artifacts remain useful when consumed by a different solver agent? Artifact utility depends on the consuming solver agent. We reuse the frozen artifacts with GLM-5.3-Flash through OpenCode, whose NO STUDY baseline exceeds that of Claude Haiku 4.5 on all six benchmarks. Only 7 of 24 method–benchmark comparisons significantly improve over NO STUDY, compared with 21 in the primary experiment, while five significantly underperform it (Tables 2 and 6). META-AGENT W/O ARCHIVE transfers most robustly, retaining significant gains on DABStep and AppWorld without a significant loss on any benchmark. The transfer changes both the underlying model and its harness, so we attribute these differences to the solver agent as a whole and cannot isolate the contribution of either component.

BCP-G illustrates one possible source of solver dependence. Claude Haiku 4.5 reaches a gold document in only 22.5% of NO STUDY rollouts compared to 57.2% in CORPUS2SKILL. GLM-5.3-Flash reaches a gold document in 95% of NO STUDY rollouts, and its reward falls when using

the transferred CORPUS2SKILL artifacts. On Harvey LAB, the META-AGENT artifacts index the environment's 266 legal matters and reduce tool calls by one third for both solvers; PREPING and META-AGENT W/ ARCHIVE retain significant gains after transfer. These cases are consistent with artifact utility depending on what the consuming solver can already recover reliably, but they do not establish this mechanism causally.

RQ4. How does test-time compute scale with and without study? Studying can reduce the test-time sampling needed to reach a score in Figure 2. Each strongest studied Best@1 reference is the $K = 1$ point on a colored solid curve, while the colored dotted line for that method shows its Avg@3 level. Best@1 with artifacts uses one rollout and no oracle selection. To exceed the strongest studied Best@1 score, NO STUDY needs 8 rollouts on BCP-G, 2 on OfficeQA, 3 on Harvey LAB, 5 on DABStep, 2 on APEX-Agents, and 4 on AppWorld. Its Best@K assumes oracle selection, making these crossover points optimistic. At average per-rollout costs from Appendix Table 9, NO STUDY costs 1.6–5.5$\times$ as much task-time inference at these points. Total savings also require enough artifact reuse to amortize study cost. Dividing study cost by the per-task saving and rounding up to whole tasks gives break-even points of 3 tasks on OfficeQA, 4 on AppWorld, 37 on BCP-G, 40 on DABStep, 70 on Harvey LAB, and about 3,700 on APEX-Agents, well beyond its 452 tasks. Appendix A.2.3 gives the calculation details.

7 INTERPRETABILITY

We examine how systems study and when their artifacts help or hinder the downstream solver. Appendix B.3 gives the artifact taxonomy, Appendix B.4 gives artifact examples, and Appendix C.1 reports observed artifact use. Appendices B and C provide the full protocols, tables, and figures.

7.1 STUDYING PROCESS

A natural hypothesis is that broader exploration helps a studying system distill an environment. We test this by relating downstream reward to counts of distinct files accessed and benchmark-specific tools invoked during study. Appendix B.1 defines the measurement protocol and its limitations.

Broader exploration does not reliably predict quality. Figures 5 and 6 show inconsistent relationships with downstream reward. On OfficeQA, CORPUS2SKILL processes all 697 files yet trails methods that inspect fewer. Harvey LAB forms the clearest split: both meta-agents discover its sole benchmark-specific tool, `labread`, and score 0.35–0.38, while methods that do not invoke it score 0.22–0.31.

Exploration strategies vary by environment. We classify META-AGENT study traces to identify recurring behaviors. Appendix B.3 provides the behavior and artifact taxonomies and the extraction protocol. Table 12 shows that inventory, unavailable-resource tracking, and cost-conscious extraction are common across environments. Parallel delegation is observed only in Harvey LAB and APEX-Agents. The artifact taxonomy in Table 13 shows a shared core of navigation structures, method recipes, and environment constants.

Archive use is environment-sensitive but imperfect. The archive-assisted meta-agent can combine artifacts from META-AGENT W/O ARCHIVE, PREPING, and CORPUS2SKILL. The resulting composition varies across environments. META-AGENT W/O ARCHIVE and PREPING outputs appear in every BCP-G, OfficeQA, and DABStep artifact set, whereas CORPUS2SKILL appears in every AppWorld artifact set and 86 of 93 APEX-Agents artifact sets in Table 11. On BCP-G, the META-AGENT omits CORPUS2SKILL even though it is the strongest standalone method.

7.2 SOLVER PROCESS

Benefits concentrate in particular task strata. On DABStep, META-AGENT W/ ARCHIVE improves reward by 0.206 on hard tasks but only 0.021 on easy tasks, where NO STUDY already scores 0.785 in Table 15. On BCP-G, CORPUS2SKILL improves all ten topics, and its lift rises from 0.232 with one gold document to 0.307 with four or more in Tables 19 and 18. Three methods obtain their largest AppWorld gain on two-application tasks, while META-AGENT W/O ARCHIVE peaks on one-application tasks in Table 17. Every method obtains its largest APEX-Agents domain gain in

Law in Table 16, and Appendix Figure 4 shows the world-level variation. These patterns do not define a universal notion of task difficulty.

Observed use does not guarantee benefit. Appendix C.1 reports within-task contrasts between trials with and without observed artifact use. Appendix C.3 gives artifact examples with clear mechanisms. In DABStep, an artifact supplies a missing fraud boundary that helps the solver. In APEX-Agents, an incomplete artifact directs the solver away from task-specific files containing the required inputs.

8 Discussion and Limitations

Our results provide evidence that task-agnostic, environment-conditioned studying is generally beneficial: 21 of 24 method–benchmark comparisons improve over No Study. *Dynamically* selecting how to study is robust but not universally optimal. A Meta-Agent variant leads on five of six benchmarks, while the specialized Corpus2Skill pipeline is strongest on BCP-G. A promising direction is therefore to improve how the agent selects a studying strategy. A broader procedure archive or demonstrations of successful environment–strategy matches could improve selection in context. Across repeated deployments, downstream evaluation feedback could instead be used to update the Meta-Agent's strategy-selection policy – via the weights or harness – turning experience across environments into expertise, in a form of meta-learning.

Studying can reduce the test-time sampling needed to reach a given performance level: one studied rollout reaches scores that require two to eight No Study rollouts under oracle selection. Because the artifacts can be reused, this advantage can amortize study cost across future tasks. It is not guaranteed, however: qualitative examples show that artifacts can sometimes misdirect the solver.

The transfer experiment qualifies these findings by showing that artifact utility depends on the solver agent. However, the transfer changes the underlying model and harness together, so we cannot isolate which component drives the observed differences. We also hold the studying agent's model and harness fixed, leaving the sensitivity of artifact construction to the studying agent unresolved.

Our conclusions are further limited by evaluation scope and cost accounting. We test one workflow archive, and the LLM selector reuses existing method evaluations rather than executing its selections independently. Moreover, the comparison changes several capabilities at once, so it does not isolate the contributions of environment exploration, workflow selection and execution, or output combination. We likewise cannot identify the value of individual archive entries or of expanding the archive. Although our benchmarks span different corpus sizes, tool surfaces, and domains, six benchmarks cannot cover every axis of environmental variation.

Finally, the primary comparison is neither cost matched nor model matched, limiting how directly performance differences can be attributed to method design. Compute-matched evaluation is important for comparing these systems, but defining equivalent compute across heterogeneous pipelines is inherently difficult. Methods use different models, sometimes several within one pipeline, and can allocate additional compute among different components. Even within Corpus2Skill, for example, compute could be allocated to summarization, embedding, or broader corpus processing. Matching dollar expenditure would not control these choices and would remain sensitive to provider pricing, while model-independent measures such as FLOPs are difficult to obtain for closed models. We therefore report full-configuration performance alongside measured costs and leave controlled comparisons among compute-allocation policies to future work.

9 Conclusion

We study task-agnostic environment preprocessing where artifacts are frozen before target tasks or target evaluation feedback are available. We compare fixed studying workflows with open-ended Meta-Agent variants. A Meta-Agent variant leads on five of six benchmarks under the primary solver. When the same frozen artifacts are reused by another solver agent, seven method and benchmark comparisons retain significant gains over No Study and five show significant losses, providing evidence of conditional portability across solver agents. Studied artifacts also reduce the test-time sampling needed to reach a given score. Together, these results show the promise of reusable pre-task artifacts while emphasizing that their value depends on the environment, the selected workflow, and the solver that consumes them.

AI USE STATEMENT

AI was used in the following manner for this paper. During the ideation phase, searching for literature was done through AI searching along with Semantic Scholar. Additionally in this phase AI was consulted at several points for validating the soundness of ideas. Coding agents were used throughout the implementation with the produced code carefully checked by the authors for correctness. In the writing phase, AI was used for drafting the initial sections of the paper to give the authors the ability to sketch out the direction of the final paper. From there, the authors only used this draft as a guideline for content length and structure for their full writing. AI was then used to polish this full human writing and to generate the code for the figures, which the authors again ensured correctly matched the results and did not hallucinate. Every aspect of AI use as used in this project, has (1) been disclosed in the AI Usage checklist and (2) has been carefully supervised by the authors to ensure no hallucination. The specific AI used was Claude through Claude Code and GPT models through Codex.

ETHICS STATEMENT

Task-agnostic environment preprocessing may reduce the expert effort required to adapt agents to specialized environments. Studying systems also explore environments before target tasks are known and persist what they learn for later use, creating four risks. **(1) Exploration side effects.** Study may invoke costly or irreversible actions and should use sandboxes, least-privilege access, and approval gates. **(2) Sensitive artifacts.** Artifacts may preserve private data, environment structure, or credentials. They should inherit source access controls and retention policies. **(3) Misleading guidance.** Incomplete or stale artifacts can harm downstream performance, as our qualitative analysis illustrates. Provenance and human review can mitigate this risk. **(4) Archive misuse.** Reusable workflows can facilitate exploration in harmful environments. Archive entries should therefore specify permitted scopes and side-effect assumptions, and their execution should be logged. This work uses existing benchmarks and API models and does not involve human subjects. It does not introduce a pretrained model or a newly collected human-subject dataset.

REPRODUCIBILITY STATEMENT

Section 5 and Appendix A.3.1 document the benchmark adaptations, model roles, harness and decoding settings, study and target-task repetitions, artifact-delivery boundary, and metric aggregation. Representative implementation prompts appear in Appendix A.3.2, and the additional per-benchmark and interpretability analyses are reported throughout the appendices. The authors plan to release the studying and evaluation harness, workflow archive, configurations, and generated artifacts.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=RQm2KQTM5r.

Anthropic. Claude code documentation. https://code.claude.com/docs/en/overview, 2026. Accessed: 2026-09-01.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Sahel Sharifymoghaddam, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Hosna Oyarhoseini, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. BrowseComp-plus: A fair and disentangled evaluation benchmark for deep search agents. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22349–22370, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1023. URL https://aclanthology.org/2026.acl-long.1023/.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Daewon Choi, Kyunghyun Park, Woomin Song, Saket Dingliwal, Sai Muralidhar Jayanthi, Jinwoo Shin, and Aram Galstyan. Idlespec: Exploiting idle time via speculative planning for llm agents. *arXiv preprint arXiv:2605.22154*, 2026a.

Yumin Choi, Sangwoo Park, Minki Kang, Jinheon Baek, and Sung Ju Hwang. Preping: Building agent memory without tasks. *arXiv preprint arXiv:2605.13880*, 2026b.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Alex Egg, Martin Iglesias Goyanes, Friso Kingma, Andreu Mora, Leandro von Werra, and Thomas Wolf. Dabstep: Data agent benchmark for multi-step reasoning. *arXiv preprint arXiv:2506.23719*, 2025.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17490–17502, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.866. URL https://aclanthology.org/2026.findings-acl.866/.

Chrisantha Fernando, Dylan Sunil Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 13481–13544. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/fernando24a.html.

Niko Grupen, Gabe Pereyra, and Julio Pereyra. Harvey's legal agent benchmark (LAB). Harvey. https://github.com/harveyai/harvey-labs, 2026.

Harbor Framework Team. Harbor: A framework for evaluating and optimizing agents and models in container environments. Zenodo, 2026. URL https://doi.org/10.5281/zenodo.20953922.

Haoyi Hu, Qirong Lyu, Xianghan Kong, Weiwen Liu, Jianghao Lin, Zixuan Guo, Yan Xu, Yasheng Wang, Weinan Zhang, and Yong Yu. Anticipate and learn: Unleashing idle-time compute in proactive agents. *arXiv preprint arXiv:2605.25971*, 2026.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=t9U3LW7JVX.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan A, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=sY5N0zY5Od.

Thibault Le Sellier De Chezelles, Maxime Gasse, Alexandre Drouin, Massimo Caccia, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Quentin Cappart, Graham Neubig, Ruslan Salakhutdinov, Nicolas Chapados, and Alexandre Lacoste. The BrowserGym ecosystem for web agent research. *arXiv preprint arXiv:2412.05467*, 2024. doi: 10.48550/arXiv.2412.05467.

Yoonho Lee, Roshen Sanjay Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=tmbOUyFx3R.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.

Jacob Xiaochen Li, Rick Battle, and Omar Khattab. Machine studying: A system-level reframing of continual adaptation from declarative corpora. In *Continual Adaptation at Scale: Towards Sustainable AI Workshop*, 2026. URL https://openreview.net/forum?id=e7duOnFgL7.

Kevin Lin, Charlie Snell, Yu Wang, Charles Packer, Sarah Wooders, Ion Stoica, and Joseph E Gonzalez. Sleep-time compute: Beyond inference scaling at test-time. *arXiv preprint arXiv:2504.13171*, 2025.

Bo Liu, Chuanyang Jin, Seungone Kim, Weizhe Yuan, Wenting Zhao, Ilia Kulikov, Xian Li, Sainbayar Sukhbaatar, Jack Lanchantin, and Jason Weston. Spice: Self-play in corpus environments improves reasoning. *arXiv preprint arXiv:2510.24684*, 2025.

Krista Opsahl-Ong, Arnav Singhvi, Jasmine Collins, Ivan Zhou, Cindy Wang, Ashutosh Baheti, Owen Oertell, Jacob Portes, Sam Havens, Erich Elsen, et al. Officeqa pro: An enterprise benchmark for end-to-end grounded reasoning. *arXiv preprint arXiv:2603.08655*, 2026.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Shreyas Rajesh, Pavan S Holur, Mehmet Yigit Turali, Chenda Duan, and Vwani Roychowdhury. Panini: Continual learning in token space via structured memory. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=BNGK86bQRr.

Stephen E. Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D Manning. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=GN921JHCRw.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=vAElhFcKW6.

Hongjin Su, Ruoxi Sun, Jinsung Yoon, Pengcheng Yin, Tao Yu, and Sercan O. Arik. Learn-by-interact: A data-centric framework for self-adaptive agents in realistic environments. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=3UKOzGWCVY.

Yiqun Sun, Pengfei Wei, and Lawrence B. Hsieh. Don't retrieve, navigate: Distilling enterprise knowledge into navigable agent skills for QA and RAG. In *First Workshop on Agent Skills*, 2026. URL https://openreview.net/forum?id=iFPjQzjIy8.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7080–7106, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-380-7. doi: 10.18653/v1/2026.eacl-long.333. URL https://aclanthology.org/2026.eacl-long.333/.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.850. URL https://aclanthology.org/2024.acl-long.850/.

Varun Ursekar, Apaar Shanker, Veronica Chatrath, Yuan Xue, and Samuel Marc Denton. VeRO: A harness for agents to optimize agents. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=zQzmwG2Nue.

Bertie Vidgen, Austin Mann, Abby Fennelly, John Wright Stanly, Lucas Rothman, Marco Burstein, Julien Benchek, David Ostrofsky, Anirudh Ravichandran, Debnil Sur, et al. Apex-agents. *arXiv preprint arXiv:2601.14242*, 2026.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=ehfRiF0R3a.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/wang25bx.html.

Lilian Weng. Harness engineering for self-improvement. *lilianweng.github.io*, July 2026. URL https://lilianweng.github.io/posts/2026-07-04-harness/.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for LLM agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=FiM0M8gcct.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. doi: 10.48550/arXiv.2405.15793. arXiv:2405.15793.

Yilun Yao, Xinyu Tan, Chao-Hsuan Liu, Yaoming Li, Zhengyang Wang, Wenhan Yu, Zhewen Tan, Yuxuan Tian, Guangxiang Zhao, Lin Sun, Xiangzheng Zhang, and Tong Yang. Harness-bench: Measuring harness effects across models in realistic agent workflows, 2026. URL https://arxiv.org/abs/2605.27922.

Chongrui Ye, Yuxiang Liu, Yu Wang, Haofei Yu, Yining Zhao, Ge Liu, Julian McAuley, and Jiaxuan You. Auto-dreamer: Learning offline memory consolidation for language agents. *arXiv preprint arXiv:2605.20616*, 2026a.

Ziang Ye, Wentao Shi, Yuxin Liu, Yu Wang, Zhengzhou Cai, Yaorui Shi, Qi Gu, Xunliang Cai, and Fuli Feng. Look before you leap: Autonomous exploration for llm agents. *arXiv preprint arXiv:2605.16143*, 2026b.

Qifan Zhang, Dongyang Ma, Tianqing Fang, Jia Li, Jing Tang, Nuo Chen, Haitao Mi, and Yan Wang. Training llm agents for spontaneous, reward-free self-evolution via world knowledge exploration. *arXiv preprint arXiv:2604.18131*, 2026a.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=eC4ygDs02R.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

A SUPPORTING MATERIALS

This section provides supporting material for our experiment setup, main results, and analysis.

A.1 BENCHMARK AND ENVIRONMENT DETAILS

Figure 3 and Table 3 provide complementary views of the files and tool entry points in each benchmark.

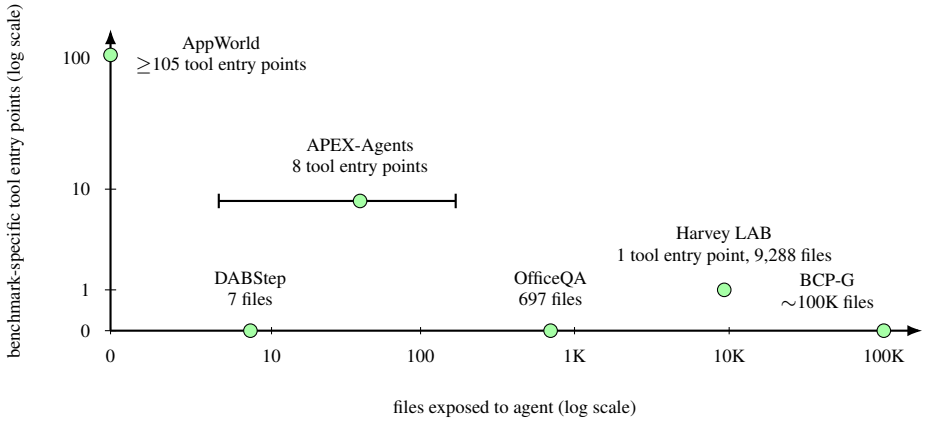


Figure 3: **File and tool-entry-point counts across benchmark environments.** Axes show $\log_{10}(1+n)$ so environments with 0 benchmark-specific tool entry points or real files remain visible. APEX-Agents' horizontal interval is its per-world file range (4–170, median ∼40). AppWorld exposes at least 105 distinct tool entry points across observed traces, while its task-specific state lives in app databases rather than real files. Generic shell and Python operations are not counted.

Benchmark	Benchmark-specific tool entry points	Files exposed to the agent
BCP-G (Chen et al., 2026)	None	100,195 text files
OfficeQA (Opsahl-Ong et al., 2026)	None	697 Treasury Bulletin text files
Harvey LAB (Grupen et al., 2026)	1 CLI tool entry point for `labread`	9,288 documents in 266 matter folders (DOCX, XLSX, PPTX, EML)
DABStep (Egg et al., 2025)	None	7 files requiring Python access, including task data, reference data needed to compute answers, and protocol instructions
APEX-Agents (Vidgen et al., 2026)	8 distinct observed tool entry points across 9 MCP servers	Per-world file-system median ∼40, range 4–170, plus `/.apps_data`, estimated across 20 worlds
AppWorld (Trivedi et al., 2024)	At least 105 distinct tool entry points across 8 apps in 150 traces	No mounted file corpus. Task-specific state is stored in app databases, with files accessed through the virtual `file_system` app

Table 3: Files and benchmark-specific tool entry points available in each benchmark environment.

A.2 ADDITIONAL RESULTS

This section reports additional performance and cost results that accompany the main analysis.

Method	BCP-G	OfficeQA	Harvey LAB	DABStep	APEX-Agents	AppWorld
NO STUDY	0.350 ± 0.016	0.730 ± 0.021	0.398 ± 0.013	0.487 ± 0.018	0.297 ± 0.010	0.685 ± 0.016
LLM selector	0.680 ± 0.017↑	0.718 ± 0.008	0.322 ± 0.010↓	0.528 ± 0.043↑	0.331 ± 0.005↑	0.710 ± 0.018
PREPING	0.443 ± 0.013↑	0.747 ± 0.016	0.426 ± 0.017↑	$\underline{0.565 \pm 0.042}$↑	$\underline{0.342 \pm 0.010}$↑	0.760 ± 0.040↑
CORPUS2SKILL	$\mathbf{0.680 \pm 0.017}$↑	0.718 ± 0.008	0.322 ± 0.010↓	0.446 ± 0.005↓	0.331 ± 0.005↑	0.710 ± 0.018
META-AGENT W/O ARCHIVE	0.410 ± 0.019↑	$\mathbf{0.780 \pm 0.016}$↑	$\underline{0.485 \pm 0.009}$↑	0.549 ± 0.012↑	0.329 ± 0.004↑	$\mathbf{0.780 \pm 0.059}$↑
META-AGENT W/ ARCHIVE	$\underline{0.451 \pm 0.023}$↑	$\underline{0.749 \pm 0.006}$	$\mathbf{0.488 \pm 0.008}$↑	$\mathbf{0.653 \pm 0.088}$↑	$\mathbf{0.343 \pm 0.017}$↑	$\underline{0.778 \pm 0.033}$↑

Table 4: **Best@3 downstream reward.** Studying-method SDs measure spread across three study runs. The NO STUDY SD is the overlap-corrected spread across three-rollout subsets. The selector SD uses total variance across selections and reused study arms. Its row reuses existing arms and is excluded from best-value styling. ↑ and ↓ mark significant gains and losses relative to NO STUDY under uncorrected 95% paired task bootstrap intervals (Table 7). Best and second-best studying methods are bold and underlined.

A.2.1 SOLVER TRANSFER TO OPENCODE WITH GLM-5.3-FLASH

The main-text Avg@3 results appear in Table 2. Table 5 reports the complementary Best@3 results.

Method	BCP-G	OfficeQA	Harvey LAB	DABStep	APEX-Agents	AppWorld
NO STUDY	$\mathbf{0.965 \pm 0.001}$	0.846 ± 0.005	0.661 ± 0.003	0.691 ± 0.025	$\mathbf{0.627 \pm 0.002}$	0.887 ± 0.017
PREPING	0.949 ± 0.003↓	0.841 ± 0.025	$\mathbf{0.690 \pm 0.013}$↑	0.662 ± 0.028	0.601 ± 0.006↓	0.871 ± 0.009
CORPUS2SKILL	0.939 ± 0.003↓	$\underline{0.855 \pm 0.008}$	0.657 ± 0.002	0.605 ± 0.012↓	0.599 ± 0.013↓	0.889 ± 0.003
META-AGENT W/O ARCHIVE	$\underline{0.965 \pm 0.006}$	$\mathbf{0.856 \pm 0.005}$	0.671 ± 0.008	$\underline{0.727 \pm 0.072}$↑	0.607 ± 0.004	$\mathbf{0.937 \pm 0.019}$↑
META-AGENT W/ ARCHIVE	0.931 ± 0.011↓	0.854 ± 0.012	$\underline{0.680 \pm 0.002}$	$\mathbf{0.750 \pm 0.044}$↑	$\underline{0.612 \pm 0.015}$	$\underline{0.897 \pm 0.009}$

Table 5: **Best@3 downstream reward with OpenCode and GLM-5.3-Flash.** The frozen study artifacts are reused from the primary experiment. Studying-method SDs measure spread across three study runs. The NO STUDY SD uses the three repetition-level benchmark means. ↑ and ↓ mark significant gains and losses relative to NO STUDY under uncorrected 95% paired task bootstrap intervals (Table 6). Best values are bold and second-best values are underlined.

Table 6 shows paired significance tests for the differences between each studying method and NO STUDY in the GLM-5.3-Flash experiments with 95% confidence intervals from 50,000 paired task bootstrap samples. Table 8 further isolates the archive effect by pairing META-AGENT W/ ARCHIVE against META-AGENT W/O ARCHIVE on the same tasks. Because each arm rests on three independent study iterations, it also reports an interval from a bootstrap that resamples the three iterations of each arm together with the tasks. The task-only intervals treat the studied artifacts as fixed and describe uncertainty over the evaluation tasks alone, whereas the iteration-resampled intervals also reflect variation across study runs.

Benchmark	Method	Avg@3 difference	Avg@3 95% CI	Best@3 difference	Best@3 95% CI
BCP-G	PREPING	−0.047	[−0.059, −0.035]	−0.016	[−0.026, −0.006]
BCP-G	CORPUS2SKILL	−0.070	[−0.086, −0.055]	−0.026	[−0.040, −0.013]
BCP-G	META-AGENT W/O ARCHIVE	−0.004	[−0.013, +0.005]	0.000	[−0.009, +0.009]
BCP-G	META-AGENT W/ ARCHIVE	−0.074	[−0.088, −0.060]	−0.034	[−0.046, −0.023]
OfficeQA	PREPING	−0.009	[−0.031, +0.011]	−0.004	[−0.027, +0.019]
OfficeQA	CORPUS2SKILL	−0.005	[−0.026, +0.016]	+0.009	[−0.011, +0.030]
OfficeQA	META-AGENT W/O ARCHIVE	+0.002	[−0.018, +0.023]	+0.011	[−0.008, +0.030]
OfficeQA	META-AGENT W/ ARCHIVE	+0.001	[−0.019, +0.022]	+0.008	[−0.014, +0.030]
Harvey LAB	PREPING	+0.031	[+0.015, +0.048]	+0.029	[+0.010, +0.049]
Harvey LAB	CORPUS2SKILL	−0.008	[−0.029, +0.013]	−0.004	[−0.029, +0.022]
Harvey LAB	META-AGENT W/O ARCHIVE	+0.021	[−0.000, +0.042]	+0.010	[−0.012, +0.034]
Harvey LAB	META-AGENT W/ ARCHIVE	+0.030	[+0.013, +0.049]	+0.018	[−0.002, +0.040]
DABStep	PREPING	−0.015	[−0.038, +0.008]	−0.029	[−0.059, +0.002]
DABStep	CORPUS2SKILL	−0.062	[−0.085, −0.040]	−0.086	[−0.119, −0.053]
DABStep	META-AGENT W/O ARCHIVE	+0.055	[+0.031, +0.080]	+0.036	[+0.003, +0.070]
DABStep	META-AGENT W/ ARCHIVE	+0.068	[+0.044, +0.091]	+0.059	[+0.027, +0.091]
APEX-Agents	PREPING	−0.017	[−0.035, +0.001]	−0.025	[−0.050, −0.001]
APEX-Agents	CORPUS2SKILL	−0.030	[−0.049, −0.011]	−0.028	[−0.051, −0.005]
APEX-Agents	META-AGENT W/O ARCHIVE	−0.008	[−0.026, +0.010]	−0.019	[−0.044, +0.005]
APEX-Agents	META-AGENT W/ ARCHIVE	−0.013	[−0.030, +0.003]	−0.015	[−0.036, +0.006]
AppWorld	PREPING	+0.007	[−0.021, +0.036]	−0.016	[−0.046, +0.012]
AppWorld	CORPUS2SKILL	+0.036	[+0.007, +0.066]	+0.002	[−0.030, +0.034]
AppWorld	META-AGENT W/O ARCHIVE	+0.064	[+0.036, +0.093]	+0.050	[+0.018, +0.085]
AppWorld	META-AGENT W/ ARCHIVE	+0.037	[+0.012, +0.064]	+0.010	[−0.016, +0.038]

Table 6: **Paired task differences from NO STUDY after solver transfer.** Differences use task-matched Avg@3 and Best@3 scores computed from unrounded task-level values. Confidence intervals are uncorrected percentile intervals from 50,000 paired task bootstrap samples.

Benchmark	Method	Avg@3 difference	Avg@3 95% CI	Best@3 difference	Best@3 95% CI
BCP-G	PREPING	+0.088	[+0.075, +0.100]	+0.092	[+0.073, +0.112]
BCP-G	CORPUS2SKILL	+0.273	[+0.251, +0.295]	+0.329	[+0.301, +0.357]
BCP-G	META-AGENT W/O ARCHIVE	+0.050	[+0.038, +0.061]	+0.060	[+0.041, +0.080]
BCP-G	META-AGENT W/ ARCHIVE	+0.089	[+0.077, +0.101]	+0.101	[+0.083, +0.119]
OfficeQA	PREPING	+0.060	[+0.036, +0.084]	+0.017	[−0.012, +0.046]
OfficeQA	CORPUS2SKILL	−0.014	[−0.037, +0.010]	−0.011	[−0.038, +0.016]
OfficeQA	META-AGENT W/O ARCHIVE	+0.084	[+0.061, +0.108]	+0.051	[+0.024, +0.078]
OfficeQA	META-AGENT W/ ARCHIVE	+0.066	[+0.043, +0.089]	+0.020	[−0.005, +0.044]
Harvey LAB	PREPING	+0.039	[+0.025, +0.053]	+0.028	[+0.009, +0.047]
Harvey LAB	CORPUS2SKILL	−0.033	[−0.053, −0.013]	−0.076	[−0.102, −0.050]
Harvey LAB	META-AGENT W/O ARCHIVE	+0.100	[+0.080, +0.121]	+0.087	[+0.062, +0.112]
Harvey LAB	META-AGENT W/ ARCHIVE	+0.106	[+0.086, +0.126]	+0.090	[+0.066, +0.115]
DABStep	PREPING	+0.111	[+0.092, +0.131]	+0.078	[+0.055, +0.101]
DABStep	CORPUS2SKILL	−0.029	[−0.045, −0.013]	−0.041	[−0.063, −0.019]
DABStep	META-AGENT W/O ARCHIVE	+0.047	[+0.033, +0.060]	+0.062	[+0.039, +0.086]
DABStep	META-AGENT W/ ARCHIVE	+0.176	[+0.154, +0.199]	+0.166	[+0.140, +0.192]
APEX-Agents	PREPING	+0.047	[+0.034, +0.060]	+0.045	[+0.027, +0.064]
APEX-Agents	CORPUS2SKILL	+0.037	[+0.021, +0.054]	+0.034	[+0.011, +0.057]
APEX-Agents	META-AGENT W/O ARCHIVE	+0.040	[+0.025, +0.056]	+0.032	[+0.011, +0.053]
APEX-Agents	META-AGENT W/ ARCHIVE	+0.050	[+0.034, +0.067]	+0.046	[+0.024, +0.069]
AppWorld	PREPING	+0.069	[+0.029, +0.109]	+0.075	[+0.032, +0.120]
AppWorld	CORPUS2SKILL	+0.027	[+0.000, +0.052]	+0.025	[−0.007, +0.057]
AppWorld	META-AGENT W/O ARCHIVE	+0.112	[+0.080, +0.146]	+0.094	[+0.057, +0.134]
AppWorld	META-AGENT W/ ARCHIVE	+0.088	[+0.057, +0.120]	+0.092	[+0.054, +0.132]

Table 7: **Paired task differences from NO STUDY.** Differences use task-matched Avg@3 and Best@3 scores. Confidence intervals are uncorrected percentile intervals from 50,000 paired task bootstrap samples. The unrounded lower endpoint for the AppWorld CORPUS2SKILL Avg@3 comparison is 0.0003.

	Avg@3			Best@3		
Benchmark	Difference	Task 95% CI	Iteration 95% CI	Difference	Task 95% CI	Iteration 95% CI
BCP-G	+0.039	[+0.028, +0.050]	[+0.012, +0.069]	+0.041	[+0.023, +0.059]	[+0.005, +0.076]
OfficeQA	−0.019	[−0.039, +0.002]	[−0.048, +0.010]	−0.031	[−0.058, −0.005]	[−0.069, +0.007]
Harvey LAB	+0.005	[−0.011, +0.021]	[−0.018, +0.029]	+0.003	[−0.018, +0.024]	[−0.026, +0.032]
DABStep	+0.130	[+0.112, +0.148]	[+0.034, +0.219]	+0.104	[+0.081, +0.127]	[+0.026, +0.196]
APEX-Agents	+0.010	[−0.002, +0.021]	[−0.008, +0.028]	+0.014	[−0.004, +0.033]	[−0.013, +0.041]
AppWorld	−0.024	[−0.054, +0.007]	[−0.097, +0.052]	−0.002	[−0.040, +0.038]	[−0.077, +0.077]

Table 8: **Paired task differences of META-AGENT W/ ARCHIVE from META-AGENT W/O ARCHIVE.** Positive values favor archive access. Differences use task-matched Avg@3 and Best@3 scores averaged over the three study iterations of each arm. Task intervals are percentile intervals from 50,000 paired task bootstrap samples with study iterations held fixed. Iteration intervals are percentile intervals from 10,000 samples that resample the three study iterations of each arm with replacement together with the tasks.

A.2.2 SINGLE-WORKFLOW LLM SELECTOR

We test whether a non-agentic LLM selector can replace the archive-equipped META-AGENT. Claude Opus 4.8 receives the controller's task-agnostic environment preprocessing goal, the task-neutral projection of one environment group, descriptions of the native tools, and the verbatim descriptions of the three archived workflows. Benchmark-identifying names are redacted. The prompt withholds downstream task instructions, verifiers, metrics, answer-bearing files, and baseline traces. The selector cannot invoke tools or inspect the environment beyond the supplied projection. It must choose exactly one of PREPING, CORPUS2SKILL, NO STUDY, or META-AGENT W/O ARCHIVE. The selector system and user prompts appear in Appendix A.3.2.

We draw five independent selections at temperature 1.0 for each of the 36 environment groups, giving 180 calls. The selector chooses CORPUS2SKILL in 176 of 180 calls. On DABStep, it chooses CORPUS2SKILL once and META-AGENT W/O ARCHIVE four times. It never chooses PREPING or NO STUDY. We score each choice using the corresponding arm already evaluated in the main experiment, so the selector rows in Tables 1 and 4 contain no new solver rollouts. The reported SD applies the law of total variance across selections and study runs. We omit the selector from transfer tables because those tables evaluate artifact portability rather than repeat a mixture of already shown transferred arms.

Method	**BCP-G**	**OfficeQA**	**Harvey LAB**	**DABStep**	**APEX-Agents**	**AppWorld**
NO STUDY	0.540 ± 0.011	0.272 ± 0.004	0.239 ± 0.030	0.125 ± 0.008	0.236 ± 0.009	0.197 ± 0.007
PREPING	0.645 ± 0.070	0.246 ± 0.007	0.243 ± 0.052	0.135 ± 0.011	0.290 ± 0.017	0.262 ± 0.067
CORPUS2SKILL	0.783 ± 0.053	0.450 ± 0.066	0.309 ± 0.011	0.129 ± 0.001	0.279 ± 0.016	0.212 ± 0.011
META-AGENT W/O ARCHIVE	0.620 ± 0.036	0.248 ± 0.007	0.258 ± 0.007	0.139 ± 0.004	0.248 ± 0.014	0.206 ± 0.023
META-AGENT W/ ARCHIVE	0.751 ± 0.122	0.255 ± 0.014	0.265 ± 0.029	0.132 ± 0.016	0.256 ± 0.027	0.184 ± 0.012

Table 9: **Average inference cost per rollout.** Values are mean $\pm$ sample SD in USD. Studying-method statistics are across three independent artifact iterations. NO STUDY statistics are across 10 full task-time repetitions. Unpriced rollouts are excluded rather than imputed.

A.2.3 BREAK-EVEN REUSE COUNT

For the studied method that sets each Best@1 reference in Figure 2, the study cost is recovered after $M^\star = C_{\text{study}}/(Kc_{\text{NS}} - c_{\text{studied}})$ reused tasks, where K is the NO STUDY crossover and c is the mean inference cost per rollout from Table 9. With the study costs in Table 10, $M^\star$ is 36.1 for BCP-G (CORPUS2SKILL), 2.6 for OfficeQA (META-AGENT W/O ARCHIVE), 69.0 for Harvey LAB (META-AGENT W/ ARCHIVE), 39.5 for DABStep (META-AGENT W/ ARCHIVE), 3746 for APEX-Agents (PREPING), and 3.7 for AppWorld (META-AGENT W/O ARCHIVE). Because Best@K assumes oracle selection, the true NO STUDY cost to reach each score is higher, so these counts are upper bounds. The main text rounds these counts up to whole tasks. They use mean costs and do not propagate uncertainty. At the higher Best@3 target for studied agents, NO STUDY

Method	BCP-G	OfficeQA	Harvey LAB	DABStep	APEX-Agents	AppWorld
NO STUDY	–	–	–	–	–	–
PREPING	24.35 ± 3.90	15.26 ± 10.43	17.57 ± 12.37	11.71 ± 8.26	681.79 ± 7.63	23.97 ± 8.61
CORPUS2SKILL	127.65 ± 110.56	1.42 ± 0.02	1.15 ± 0.02	0.01 ± 0.00	$17.49 \pm 0.02^{\ddagger}$	1.04 ± 0.05
META-AGENT W/O ARCHIVE	0.68 ± 0.19	0.76 ± 0.09	7.36 ± 4.10	0.59 ± 0.09	66.96 ± 8.37	2.18 ± 0.43
META-AGENT W/ ARCHIVE	97.87 ± 112.73	15.99 ± 9.67	31.21 ± 22.94	19.47 ± 0.65	584.21 ± 198.78	13.83 ± 19.72

Table 10: **Average pre-task cost.** Values are mean ± sample SD in USD across three independent artifact-construction iterations. APEX-Agents costs are summed over its 31 worlds. NO STUDY performs no pre-task computation and is shown as dashes. The APEX-Agents CORPUS2SKILL cell, marked ‡, uses the two metered iterations because the original iteration did not retain cost telemetry.

requires 6 rollouts on OfficeQA, 7 on Harvey LAB, and 5 on APEX-Agents, and does not reach the strongest studied score within 10 rollouts on BCP-G, DABStep, or AppWorld.

Figure 4 disaggregates the META-AGENT W/ ARCHIVE Avg@3 lift over NO STUDY across individual APEX-Agents worlds, grouped by domain.

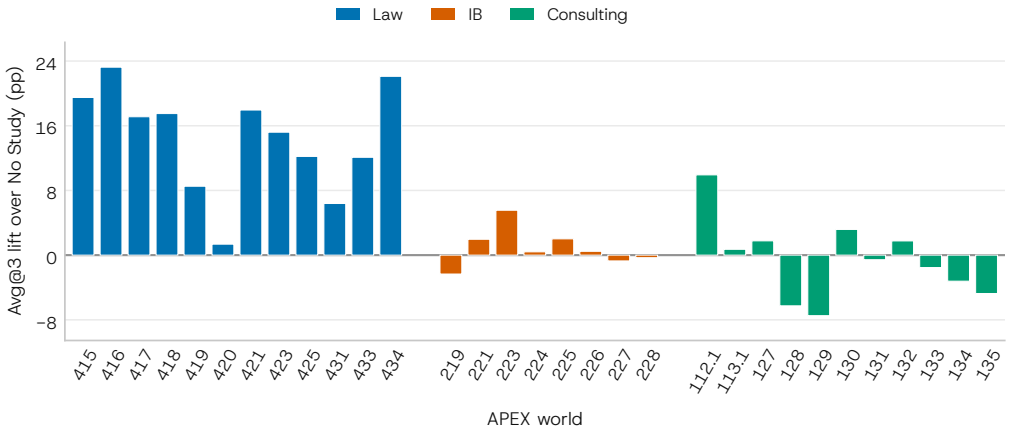


Figure 4: **META-AGENT W/ ARCHIVE lift by APEX-Agents world.** Each bar shows the signed percentage-point difference between META-AGENT W/ ARCHIVE Avg@3 and the NO STUDY Avg@3 baseline for one APEX-Agents world. Within each of three independent artifact iterations, the studied score initially averages the three evaluation rewards for each task and then macro-averages tasks. The plotted score averages those three artifact-iteration estimates. For NO STUDY, we compute the three-repetition mean for each of the $\binom{10}{3}$ subsets and average those subset estimates. This point estimate is algebraically equal to the 10-repetition mean. Bar color denotes domain. World scores macro-average the tasks in that world.

A.3 Additional Implementation Details

A.3.1 Studying method configuration

This section reports implementation choices not specified in the main text.

PREPING. We follow the PREPING native settings closely in our experiments. The proposer may receive the `task_history` of earlier synthetic tasks. The synthetic-task pipeline may use generated ground truth, ground-truth code, unit tests, and validator scores. We use a feasibility threshold of 5/5 and treat completion scores of at least 4/5 as successful. Proposal and validation temperatures are 0.7 and 0. Only validator-approved feasible trajectories are reflected on and curated into the playbook. The standalone arm retrieves up to 40 playbook entries separately for each target instruction. When PREPING is selected inside META-AGENT W/ ARCHIVE, the final bundle instead contains one static playbook that never retrieves against a target instruction.

CORPUS2SKILL. We use the published defaults except for benchmark-specific adjustments to hierarchy branching, document length, and tree compaction. We increase the branching ratio from 10 to 40 on OfficeQA and Harvey LAB. We use a 4,000-character document limit and compact tree representation on DABStep, OfficeQA, and Harvey LAB. The remaining benchmarks retain the default 8,000-character limit and non-compact representation. At target time, this arm disables direct corpus search, requires at least two Skill calls and one successful `get_document` call, and permits full-document access only through identifiers found in the compiled hierarchy.

META-AGENT interventions. META-AGENT W/O ARCHIVE receives a task-neutral preamble and may write only to the single `/metanoarchive` output directory. META-AGENT W/ ARCHIVE injects the finalized `prefix.md` into the system prompt and mounts a bundle with the fixed layout shown in Appendix A.3.2. It may retain at most one successful output from each workflow family and may run synthetic probes through the pinned target solver during study. AppWorld starts target tasks from task-specific database states even though the study-time interface schema is shared.

Artifact delivery boundary. NO STUDY makes no environment or prompt change. Each studying method exposes only its frozen output through its native interface. The user-provided target instruction text is preserved in every arm, but the surrounding intervention differs. PREPING prepends retrieved playbook entries, CORPUS2SKILL adds navigation instructions and restricts corpus access, META-AGENT W/O ARCHIVE adds a preamble and mounted files, and META-AGENT W/ ARCHIVE injects `prefix.md` into the system prompt and mounts its bundle.

A.3.2 Representative implementation prompts

The listings below show one representative prompt for each shared method component needed to interpret the intervention. Benchmark-specific environment clauses, proposer and validator variants, and submission suffixes are omitted here. The implementation and released run records contain the complete rendered prompts. Punctuation and formatting are normalized to the manuscript style.

PREPING task proposer.

```
You are a synthetic task generator for an AI agent to construct the memory in the environment.

Your goal is to generate diverse, realistic, and challenging task instructions that an AI
↪  assistant would need to complete. These tasks should reflect real-world user scenarios.

## API Documentation
{api_docs}

## Guidelines
1. **Feasibility**: Generate tasks that are actually executable with the given APIs
2. **Naturalness**: Write as a real user would phrase them, not as technical API calls
3. **Verifiability**: Instructions must be precise enough to allow for an objective judgment
↪  of success or failure.
   - The agent will submit its answer as a single text string. Success is judged by comparing
   ↪  that string to the expected value.
   - For this to be verifiable, each QUERY must require **exactly one type of answer** (e.g.
   ↪  one number, or one name/title, or one list.)
```

```
   - Do not ask for two or more distinct values in one task (e.g. both a count and a list, or
   ↪  both title and artist). Otherwise the expected format of the submitted text is
   ↪  ambiguous and cannot be reliably checked.
4. **Entity Use**:
   - Prefer realistic, natural task phrasing over rigid placeholder patterns.
   - If **environment information is provided**, treat it as optional grounding support:
     - You may use discovered names, titles, folders, contacts, and other entities when they
     ↪  help make a task concrete.
     - You do not need to force those entities into every task.
   - If **environment information is empty or missing**, you may use generic user-centric
   ↪  references such as:
     - "my favorite playlist", "my song library", "my playlists", "my album library", "artists
     ↪  I follow"

## Task Category Guidelines
Generate tasks across the following categories:

### 1. QUERY Tasks (Information Retrieval)
Tasks that require finding and returning specific information. The answer should be objective
↪  and verifiable (number, name, comma-separated list, yes/no).
Include concrete details with specific output format such as numbers, dates, names, locations,
↪  thresholds, item lists, etc.
Avoid vague terms such as 'check', 'review', 'show' or 'ensure' unless they are accompanied by
↪  measurable criteria.

### 2. ACTION Tasks (State Modification)
Tasks that require modifying the state of one or more apps. No explicit answer required -
↪  success is determined by state changes.

{environment_info_section}
{task_history_section}

## Output Format
Return a JSON array of task instructions:
```json
[
    {{
        "category": "QUERY|ACTION",
        "involved_apps": ["app1", "app2"],
        "involved_apis": ["app.api1", "app.api2"],
        "instruction": "Natural language task instruction"
    }}
]
```

Generate {num_tasks} diverse task instructions:
```

PREPING trajectory validator.

```
You are an expert evaluator of agent task execution.

Evaluate the following Task and Trajectory using a 5-point Likert scale for each criterion.

## Evaluation Criteria

### 1. Task Feasibility
Evaluate whether the task is executable and the entities are grounded in the current
↪  environment.
- Do not judge feasibility by API availability alone.
- Verify that every required target entity in the instruction (person, contact, request, file,
↪  thread, project, etc.) is present and can be uniquely identified from the trajectory
↪  evidence.
- If the entity cannot be identified after reasonable paginated search using relevant APIs,
↪  the task is infeasible.

- **5 (Excellent)**: All required entities/constraints are explicitly present and uniquely
↪  identified. Clearly executable as stated.
- **4 (Good)**: Entities are present with minor ambiguity resolvable by a simple lookup. No
↪  core entity missing.
- **3 (Acceptable)**: Plausible, but key grounding is uncertain or search is insufficient.
- **2 (Poor)**: Likely infeasible. Required entity not found or not uniquely identifiable
↪  after reasonable search.
- **1 (Unacceptable)**: Infeasible/contradictory or relies on non-existent
↪  entities/conditions.

### 2. Task Completion
Judge whether the task instruction was successfully completed.

- Evaluate all task instructions are satisfied based on the trajectory, including all
↪  constraints.
- Treat a call to `apis.supervisor.complete_task(...)` as a required completion step, not as
↪  evidence that the task was successful correctly.
```

```
- If the task asked for action, do not treat the task as successful unless the trajectory includes a call to `apis.supervisor.complete_task()` with no answer argument (or with `None`).
- If the task asked for information, success requires a call to `apis.supervisor.complete_task(answer=<answer>)`, and the answer should contain only the exact requested value: a number (int or float), a direct string value, a plain comma-separated list, or `yes`/`no`.
- Do not treat verbose answers, action summaries, explanations, full sentences, or answers with extra words, units, or symbols as correct unless explicitly requested. e.g. `10` is acceptable, but `The answer is 10 songs.` is not. Do not require an answer for action-only tasks.

- **5 (Excellent)**: Every requirement AND every constraint is satisfied, and the trajectory shows clear completion for each required outcome.
- **4 (Good)**: All requirements/constraints appear satisfied with minor ambiguity, and nothing important is missing.
- **3 (Acceptable)**: Some progress, but at least one requirement/constraint is missing, ambiguous, or only asserted without support in the trajectory steps (surface-level success). This includes cases where an object/ID exists but correctness constraints (privacy, filters, counts, formatting, attachment actually uploaded, etc.) are not confirmed.
- **2 (Poor)**: Minor progress only. The main outcome is not achieved or an error/failed response prevents completion.
- **1 (Unacceptable)**: No meaningful progress toward the task goal.

## Task Instruction
{task_instruction}

## Trajectory
{trajectory}

Respond in JSON format:
```json
{{
  "feasibility_reason": "1-2 sentence explanation for feasibility score",
  "feasibility_score": 1-5,

  "task_completion_reason": "1-2 sentence explanation for task completion score",
  "task_completion_score": 1-5
}}
```
```

PREPING reflector.

```
You are an expert agent and educator. Your job is to diagnose the current trajectory: identify what went wrong (or could be better), API usage, and ground truth when applicable.

Instructions:
- Carefully analyze the model's reasoning trace to identify where it went wrong
- Take the environment feedback into account, comparing the predicted answer with the (optional) ground truth to understand the gap
- Identify specific conceptual errors, calculation mistakes, or misapplied strategies
- Provide actionable insights that could help the model avoid this mistake in the future
- Identify root causes: wrong source of truth, bad filters (timeframe/direction/identity), formatting issues, or missing authentication and how to correct them.
- Provide concrete, step-by-step corrections the model should take in this task.
- Be specific about what the model should have done differently
- You will receive bulletpoints that are part of playbook that's used by the generator to answer the question.
- You need to analyze these bulletpoints, and give the tag for each bulletpoint, tag can be ['helpful', 'harmful', 'neutral'] (for the generator to generate the correct answer)
- Explicitly curate from the environment feedback the output format/schema of APIs used when unclear or mismatched with expectations (e.g., apis.blah.show_contents() returns a list of content_ids (strings), not content objects)

Inputs:

Task Instruction:
{{task_description}}

{{ground_truth_result}}

{{ground_truth_code}}
{{unit_test_results}}
PrePing playbook (playbook that's used by model for code generation):
PLAYBOOK_START
{{playbook}}
PLAYBOOK_END

Agent-Environment Trajectory (including reasonings actions, and final observation):
{{trajectory}}
```

```
Outputs: Your output should be a json object, which contains the following fields
- reasoning: your chain of thought / reasoning / thinking process, detailed analysis and
↪  calculations
- error_identification: what specifically went wrong in the reasoning?
- root_cause_analysis: why did this error occur? What concept was misunderstood?
- correct_approach: what should the model have done instead?
- key_insight: what strategy, formula, or principle should be remembered to avoid this error?
- bullet_tags: a dictionary mapping each bullet_id (the {section}-{number} prefix shown in the
↪  playbook) to its tag ('helpful', 'harmful', or 'neutral')
Answer in this exact JSON format:

{
"reasoning": "[Your chain of thought / reasoning / thinking process, detailed analysis and
↪  calculations]",
"error_identification": "[What specifically went wrong in the reasoning?]",
"root_cause_analysis": "[Why did this error occur? What concept was misunderstood?]",
"correct_approach": "[What should the model have done instead?]",
"key_insight": "[What strategy, formula, or principle should be remembered to avoid this
↪  error?]",
"bullet_tags": {"bullet_id_1": "helpful", "bullet_id_2": "harmful", "bullet_id_3": "neutral"}
}
```

PREPING curator.

```
You are a master curator of knowledge. Your job is to identify what new insights should be
↪  added to an existing playbook based on a reflection from a previous attempt.

Context:
- The playbook you created will be used to help answering similar questions.

Instructions:
- Review the existing playbook and the reflection from the previous attempt
- Identify ONLY the NEW insights, strategies, or mistakes that are MISSING from the current
↪  playbook
- Avoid redundancy - if similar advice already exists, only add new content that is a perfect
↪  complement to the existing playbook
- Do NOT regenerate the entire playbook - only provide the additions needed
- Focus on quality over quantity - a focused, well-organized playbook is better than an
↪  exhaustive one
- Format your response as a PURE JSON object with specific sections
- For any operation if no new content to add, return an empty list for the operations field
- Be concise and specific - each addition should be actionable
- For coding tasks, explicitly curate from the reflections the output format/schema of APIs
↪  used when unclear or mismatched with expectations (e.g., apis.blah.show_contents() returns
↪  a list of content_ids (strings), not content objects)

Task Instruction:
{question_context}

Current Playbook:
{current_playbook}

Agent-Environment Trajectory (actions and outputs from the attempt):
{trajectory}

Current Reflections (principles and strategies that helped to achieve current task):
{guidebook}

Your Task:
Output ONLY a valid JSON object with these exact fields:
- reasoning: your chain of thought / reasoning / thinking process, detailed analysis and
↪  calculations
- operations: a list of operations to be performed on the playbook
- type: the type of operation to be performed
- section: the section to add the bullet to (one of: strategies, code_snippets, pitfalls,
↪  apis)
- content: the new content of the bullet

Available Operations:
1. ADD: Create new bullet points with fresh IDs
- section: the section to add the new bullet to
- content: the new content of the bullet. Note: no need to include the bullet_id in the
↪  content like '[ctx-00263] helpful=1 harmful=0 ::', the bullet_id will be added by the
↪  system.

RESPONSE FORMAT - Output ONLY this JSON structure (no markdown, no code blocks):
{{
"reasoning": "...",
"operations": [
    {{ "type": "ADD", "section": "...", "content": "..." }}
]
```

```
}}
```

META-AGENT W/O ARCHIVE prompt, represented by BCP-G.

```
ENVIRONMENT. BCP-Grep study. The material you are studying is a fixed collection of plaintext
↪  documents under `/workspace/corpus/`. Each `<docid>.txt` file starts with its `DOCID` and
↪  `URL`, followed by the complete document text. Use Claude Code's native `Grep` across
↪  `/workspace/corpus`, then `Read` matching files. Use `Glob` when filename discovery is
↪  useful. There is no web access. You may write ONLY under `/metanoarchive/`.

You are studying for a test that you have no prior information about other than that it will
↪  be based on the environment you have been placed in, the documents it contains, and the
↪  tools available to you for finding and reading them. You are not allowed to search the web
↪  at any point or to go outside this environment. You are given a limited budget in which
↪  you may explore as you wish. Inspect the corpus to learn where different kinds of
↪  information live and exercise the available tools to learn what they do and how they
↪  behave. Your goal is to leave behind a minimal set of artifacts in the /metanoarchive/
↪  directory that inform a future, budget-constrained version of yourself as much as possible
↪  without overwhelming it. Maximize useful signal per token and add nothing that is merely
↪  noise. A test-time agent will have the same access you do, so its budget is wasted on
↪  exactly two things. They are finding where relevant information lives in an unfamiliar
↪  corpus and figuring out how to operate unfamiliar tools correctly. Spend your study budget
↪  mapping those two things and little else. Record where to look for a given kind of
↪  information and how to use each non-obvious tool, including its purpose, the few calls
↪  that matter, and any quirks or pitfalls you actually hit. Prefer pointers and short
↪  operating notes over copied content. Say where to look and how to act rather than
↪  reproducing material, which can go stale and invites blind reliance. Be ruthless about
↪  clutter. Do not record anything obvious from a document's title or a tool's name, anything
↪  a test-time agent could rediscover in a step or two, or general knowledge you already
↪  possess. If a line does not save meaningful test-time budget, delete it. At test time you
↪  will be given read access to whatever you created in /metanoarchive/ along with the full
↪  environment, but you will be on a limited inference budget, so the quality, brevity, and
↪  accessibility of what you leave behind may greatly affect your test-time performance.
↪  Proceed however you like, but the only write access you have is to the /metanoarchive/
↪  directory, and you may explore the corpus through its native filesystem tools and that
↪  directory.

GROUNDING REQUIREMENT: every fact, entity, claim, entry, table row, Q&A pair, map node, or
↪  note recorded in an artifact MUST cite the exact numeric document ID and file path, for
↪  example `[5412] /workspace/corpus/5412.txt`. A future agent must be able to open the cited
↪  file directly with `Read` instead of repeating discovery. For tool-operating notes, name
↪  the native tool, minimal call pattern, path, and any observed quirk or pitfall.
```

CORPUS2SKILL downstream navigation prompt.

```
You are navigating a closed, precompiled corpus through Claude Code skills and one exact-ID
↪  document tool.

For every task:

1. Invoke at least two plausible top-level skills with the Skill tool before settling on a
↪  branch.
2. Descend through the selected skills by reading their `SKILL.md` and nested `INDEX.md` files
↪  with Read. Use Glob to discover relevant index files and Grep only within the skill/index
↪  tree. Consult `entity_index.json` when named entities can disambiguate the route. If a
↪  branch is thin or unhelpful, cross-jump to a related or second plausible skill.
3. Select exact document IDs from leaf indexes before calling `get_document`. The document
↪  tool does not search or list the corpus.
4. Make at least one successful `get_document` call. Treat retrieved full documents, rather
↪  than skill summaries, index summaries, or entity mappings, as factual evidence for the
↪  answer.

Do not use corpus-search tools or open-web tools. Do not read, grep, parse, or run scripts
↪  against `documents.json`. Access full documents only through `get_document` with IDs found
↪  in leaf indexes.
```

META-AGENT W/ ARCHIVE Prompt

```
# Preprocessing goal

Prepare one frozen, task-blind bundle that gives the later target solver the best chance of
↪  solving
unknown tasks from this environment group. You may inspect only the neutral environment
↪  projection.
Real target instructions, verifier code, metric details, answer-bearing files, and real
↪  baseline
traces are unavailable and must remain unavailable.
```

```
You are the controller. Every synthetic probe and every eventual target run uses the pinned
target solver through the real harness. The elapsed deadline began before this session
↪  invocation.
Resuming never resets it. The gateway is the authority for the deadline, active work, costs,
↪  and
committed final state. If the main deadline fires, all exploration/workflow/probe tools are
↪  disabled
and active jobs are canceled. A short hard-bounded grace permits only working-note updates and
prefix/bundle consolidation from workflow jobs that completed before the deadline. It does not
permit new artifacts, probes, or workflow execution. Expiry commits `timeout_forced_null`.

## Durable work

`working-notes.md` in durable scratch is your authoritative diagnosis and decision record.
↪  Update it:

- after your initial diagnosis,
- after every meaningful probe or workflow result,
- before entering a long `await_jobs` call,
- whenever you change the candidate stack, and
- immediately before finalization.

Keep draft prefix text, your ordinary artifacts, and workflow checkpoints in durable scratch.
↪  A
session may be compacted or interrupted, so do not leave the only copy of a decision in
↪  conversation
history or transient filesystem state.

Use `update_working_notes` so each update is fsynced and recorded in the append-only ledger.

## Workflows and probes

The control skills describe MetaNoArchive, PREPING, Corpus2Skill, and synthetic probe
↪  invocation without
performance or routing advice. You may invoke, configure, combine, compare, or decline them.
↪  Select
at most one successful output from each workflow family. PREPING delivery is one complete
↪  static
playbook shared by the group. It must never retrieve against a real target instruction.
↪  Controller-authored
files are ordinary inert artifacts and never new Claude Code skills.

Gateway submissions return durable job IDs immediately. When work is outstanding, call
`await_jobs(job_ids, return_when)` once and let it block. Do not implement sleeps, polling
↪  loops,
repeated status prompts, or conversational check-ins. The wait wakes for a requested terminal
↪  job,
the 30-minute warning, the final deadline, or infrastructure requiring attention. Before a
↪  long
wait, update working notes.

## Final bundle

The admitted `/meta_archive` layout is fixed: `manifest.toml`, `prefix.md`, canonical workflow
↪  skills
under `skills/`, workflow artifacts under `artifacts/{metanoarchive,preping,corpus2skill}/`,
↪  your inert
artifacts under `artifacts/meta/`, and declared additive tools/data under `tools/`. Seed
↪  control skills
never transfer. Native solver tools and filesystem access remain available. Canonical prompt
↪  blocks
describe actual validated knobs and allocated deliverables. You may weave, reorder, shorten,
↪  extend,
or decline them, but every interface named in the final prefix must be truthful.

Call `finalize_bundle` to validate and lock the static bundle, cancel unselected work, and end
preprocessing. You may instead call `choose_null`. A deadline or unrecoverable crash forces
↪  null. All
null states receive the unmodified baseline handoff. Never write or infer a real task
↪  instruction.
```

Single-workflow LLM-selector system prompt.

```
# Preprocessing goal

Prepare one frozen, task-blind bundle that gives the later Claude Haiku 4.5 solver the best
↪  chance of solving
unknown tasks from this environment group. You may inspect only the neutral environment
↪  projection.
Real target instructions, verifier code, metric details, answer-bearing files, and real
↪  baseline
```

```
traces are unavailable and must remain unavailable.

Every eventual target run uses the pinned Claude Haiku 4.5 solver through the real Claude Code
harness. Native solver tools and filesystem access remain available in every case.

## Your decision

You are a non-agentic selector: you cannot explore the environment, run probes, or invoke
↪  tools. The
workflow you choose runs exactly once at its default knobs and is delivered through its own
↪  native
interface; nothing is combined, edited, or rewritten. Commit to exactly one of the following:

- `preping` -- run the canonical PREPING workflow at its default setting;
- `corpus2skill` -- run the canonical Corpus2Skill workflow at its default setting;
- `null` -- deliver the unmodified baseline environment with no study artifacts;
- `metanoarchive` -- run the canonical Meta-Agent w/o Archive workflow at its default setting.

## Output contract

Respond with a single JSON object and nothing else, no code fences, no prose before or after:

{"workflow": "<preping|corpus2skill|null|metanoarchive>", "rationale": "<at most 120 words>"}
```

Single-workflow LLM-selector user prompt template.

```
# Environment projection

## environment-summary.json

```json
{environment_summary_json}
```

Readable files below `environment/`

environment/{relative_file_path}

```
{file_contents}
```

The file section repeats for every readable file. When the build context contains no readable
files, it instead contains

```
(none, the build context is empty)
```

Available canonical workflows

control_skills/preping/SKILL.md

```markdown
{verbatim_preping_skill}
```

control_skills/corpus2skill/SKILL.md

```markdown
{verbatim_corpus2skill_skill}
```

control_skills/metanoarchive/SKILL.md

```markdown
{verbatim_meta_no_archive_skill}
```

Decision

Choose exactly one of `preping`, `corpus2skill`, `null`, or `metanoarchive` and answer with the
↪ JSON
object specified in the system prompt.
```

B STUDY TIME INTERPRETABILITY

This section details our methods for analyzing how the studying systems explored each environment, which behaviors they exhibited, and what they produced.

B.1 Environment Exploration Analysis

We measure environment exploration using the number of unique files read and benchmark-specific tool entry points invoked during study. Figures 5 and 6 relate these quantities to downstream reward for each benchmark, method, and artifact iteration.

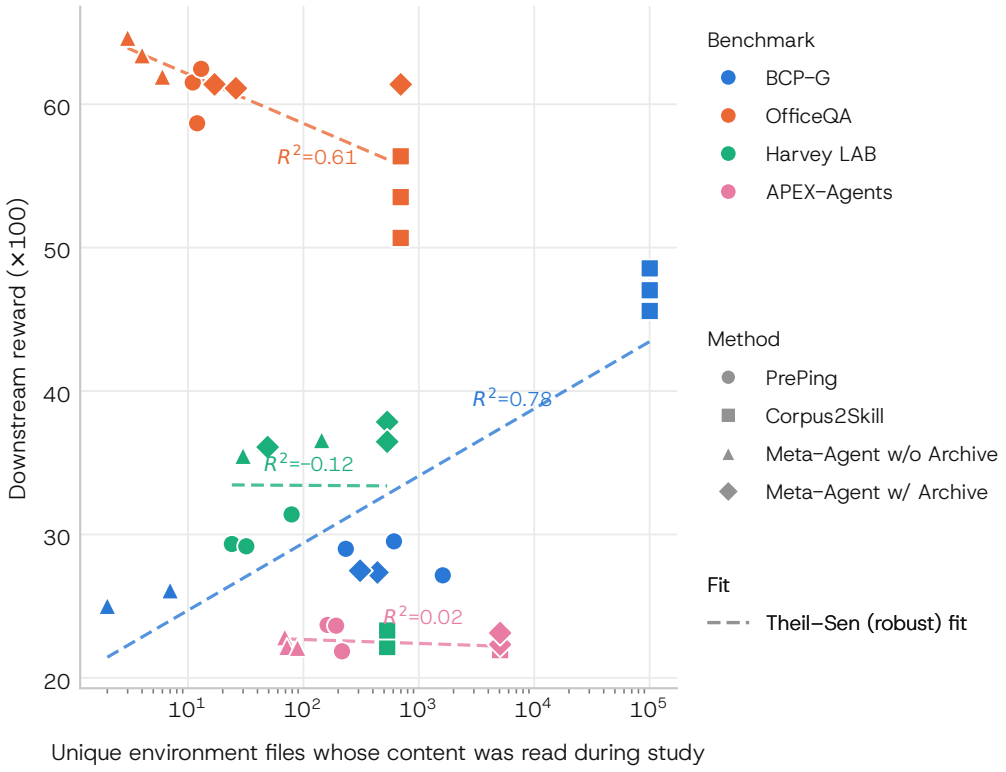


Figure 5: **File access and downstream reward.** Each point is one benchmark–method–artifact-iteration combination. The horizontal axis counts unique environment files accessed during the analyzed study attempt. The vertical axis is that artifact's task-macro Avg@3 reward. Dashed lines are descriptive per-benchmark Theil–Sen fits of reward on $\log_{10}$(files), pooled across methods and iterations. The annotated R^2 is computed against the robust line. DABStep is omitted because all methods access all seven files. AppWorld has no real file surface.

Measurement. For trace-based methods, we count a file when its contents appear in the study trace through a direct read or an attributed search result. Directory listings and filename mentions do not count. For Corpus2Skill, which has no interactive trace, we instead count the documents processed by its compiler. We count each benchmark-specific tool entry point invoked at least once, including failed calls. Shell and generic file operations are excluded.

Scope. Counts are aggregated over the available study traces for each benchmark, method, and artifact iteration. APEX-Agents additionally aggregates across its 31 worlds. DABStep is omitted from the file analysis because every method reads all seven files, and AppWorld because it exposes no real files. Trace-derived file counts are lower bounds because internal access and truncated results may not expose every file read.

Results. We fit a Theil–Sen line to the points for each benchmark. These fits are descriptive: exploration and method vary together, so they do not isolate the effect of reading an additional file or invoking an additional tool. File-count slopes are nearly flat for Harvey LAB and APEX-Agents, negative for OfficeQA, and positive for BCP-G. The stronger OfficeQA and BCP-G associations

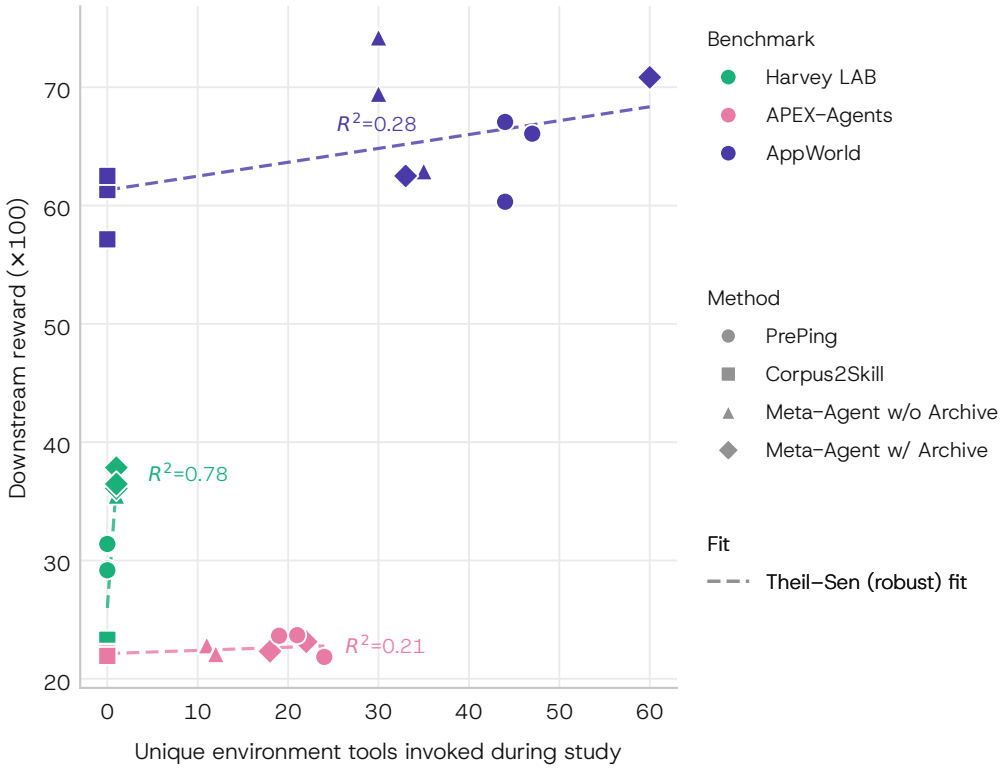


Figure 6: **Tool use and downstream reward.** Each point is one benchmark–method–artifact-iteration combination. The horizontal axis counts distinct benchmark-specific tool entry points invoked during the analyzed study attempt. The vertical axis is that artifact's task-macro Avg@3 reward. Dashed lines are descriptive per-benchmark Theil–Sen fits pooled across methods and iterations, with R^2 computed against the robust line. Harvey LAB has one benchmark-specific tool entry point, so its fit represents a method-confounded binary split rather than a continuous breadth relationship. BCP-G, OfficeQA, and DABStep are omitted because they expose no benchmark-specific tool entry points.

are driven largely by CORPUS2SKILL, which processes the entire corpus and therefore lies far from the trace-based methods on the file axis. Tool-count slopes are also nearly flat for APEX-Agents and AppWorld. Harvey LAB is the exception: both META-AGENT variants invoke `labread` and outperform methods that do not, although tool use and method identity are confounded. Overall, exploration breadth may matter, but it does not by itself determine downstream performance.

B.2 COMPOSITION OF ARCHIVE-ASSISTED ARTIFACTS

META-AGENT W/ ARCHIVE can invoke three studying procedures. They are META-AGENT W/O ARCHIVE, PREPING, and CORPUS2SKILL. It can inspect their outputs and combine the resulting artifacts for the solver. Table 11 reports how often artifacts from each procedure appeared in the final artifact set.

Results. Artifact composition varies across environments. META-AGENT W/O ARCHIVE and PREPING appear in every BCP-G, OfficeQA, and DABStep artifact set, whereas CORPUS2SKILL appears in none. In contrast, CORPUS2SKILL appears in every AppWorld artifact set, both available Harvey LAB sets, and 86 of 93 APEX-Agents sets. An output may be absent because its procedure failed or because META-AGENT W/ ARCHIVE did not retain it. The table therefore describes final artifact composition rather than invocation decisions. Composition is also imperfect because META-AGENT W/ ARCHIVE omits CORPUS2SKILL on BCP-G even though it is the strongest standalone method there.

Artifact source	BCP-G	OfficeQA	Harvey LAB	DABStep	APEX-Agents	AppWorld
META-AGENT W/O ARCHIVE	3/3	3/3	3/3	3/3	68/93	2/3
PREPING	3/3	3/3	2/3	3/3	34/93	1/3
CORPUS2SKILL	0/3	0/3	2/3†	0/3	86/93	3/3

Table 11: Sources of artifacts included by META-AGENT W/ ARCHIVE. Each cell reports the number of final artifact sets containing output from that procedure. Multiple sources may appear in one artifact set. Iterations are pooled, and APEX-Agents is also pooled across worlds. The APEX-Agents entry $68/93$ for META-AGENT W/O ARCHIVE counts final artifact sets that retained its output. It is not a denominator for analyzable study trajectories. † CORPUS2SKILL was unavailable in the first Harvey LAB iteration and was included in both iterations in which it was available.

B.3 TAXONOMIES OF META-AGENT STUDY BEHAVIOR AND ARTIFACTS

We analyze the META-AGENT variants in two ways: the behaviors visible in their study traces and the content of their final artifacts.

Study behavior taxonomy. We define nine multi-label categories for actions visible in a study trace:

- **Environment inventory before content.** Enumerating the environment's whole surface (file tree, corpus size, app list) with the agent's own listing or counting calls before the first content-bearing read.
- **Interface probing.** Issuing calls whose purpose is to learn the interface itself, including help probes, format tests, trying a tool to see whether it exists, or sweeping logins to confirm an authentication pattern, and recording the lesson.
- **Documentation verification.** Explicitly testing a claim made by shipped documentation or metadata against observed data and recording the contradiction or confirmation.
- **Structured sampling or hypothesis testing.** Sampling by named strata (per decade, ID range, app, or document type), or stating a hypothesis about the environment's organization and running a call to confirm or refute it, including bisecting a discovered boundary.
- **Recording unavailable resources.** Recording something as absent, empty, broken, or impossible so that the downstream agent does not spend time attempting to use it.
- **Budget-conscious extraction.** Making an explicit economizing decision: using cheap shell extraction before LLM reading, storing pointers instead of copied content, avoiding duplication of information already available to the solver, or timing operations to assess their cost.
- **Parallel sub-agent delegation.** Delegating exploration to parallel sub-agents with an explicit work partition and a specified output format or budget rule.
- **Artifact verification.** A dedicated pass checking claims destined for or already in the artifact against the environment: auditing cited paths, recomputing a recorded value, testing the artifact with mock questions, or a corrective edit after a re-check.
- **Test anticipation and rehearsal.** Predicting what the test will ask from a concrete task-bearing surface (inboxes, chat threads, an active sample task) and acting on the inference, including rehearsing a realistic task end-to-end.

Trace labeling. Claude Sonnet 4.6 labels every analyzable trace at temperature 0 against the fixed category definitions. "w/o" denotes direct META-AGENT W/O ARCHIVE study, while "w/" denotes the META-AGENT W/O ARCHIVE workflow invoked by META-AGENT W/ ARCHIVE. Each positive label requires a verbatim evidence quote, which we mechanically verify appears in the trace.

Behavior findings. Environment inventory, recording unavailable resources, and budget-conscious extraction appear in nearly all applicable traces. Test anticipation is concentrated in APEX-Agents and AppWorld, where histories and examples provide clues about likely downstream tasks. Parallel

	BCP-G		OfficeQA		Harvey LAB		DABStep		APEX-Agents		AppWorld	
Behavior	w/o	w/	w/o	w/	w/o	w/	w/o	w/	w/o	w/	w/o	w/
Environment inventory before content	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	93/93	73/73	n/a	n/a
Interface probing	3/3	3/3	2/3	2/3	3/3	3/3	0/3	0/3	84/93	68/73	3/3	1/2
Documentation verification	2/3	0/3	0/3	0/3	1/3	3/3	2/3	3/3	45/93	43/73	0/3	0/2
Structured sampling / hypothesis testing	2/3	3/3	3/3	2/3	2/3	3/3	2/3	2/3	63/93	58/73	2/3	1/2
Recording unavailable resources	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	89/93	70/73	2/3	2/2
Budget-conscious extraction	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	91/93	71/73	3/3	2/2
Parallel sub-agent delegation	0/3	0/3	0/3	0/3	3/3	2/3	0/3	0/3	14/93	10/73	0/3	0/2
Artifact verification	0/3	1/3	1/3	2/3	2/3	2/3	0/3	1/3	35/93	23/73	1/3	1/2
Test anticipation and rehearsal	0/3	0/3	0/3	1/3	0/3	0/3	0/3	1/3	79/93	66/73	3/3	2/2

Table 12: Study behaviors observed in available trajectories. Each cell reports trajectories exhibiting the behavior out of all analyzable trajectories. For archive-assisted APEX-Agents, 73 of the 93 world-iteration trajectories were analyzable. Eighteen iteration-2 trajectories were forced null by the infrastructure-cost cascade, and the controller declined the META-AGENT W/O ARCHIVE workflow in the remaining two. AppWorld inventory is not applicable because its tool schemas are supplied rather than discovered through trace actions.

sub-agent delegation appears only in Harvey LAB and APEX-Agents. The META-AGENT variants therefore share a common diagnostic core while adapting some exploration strategies to the environment.

Artifact taxonomy. We define twelve multi-label categories for content appearing anywhere in a study run's final artifact set.

- **Navigation architecture.** Routing indexes, need-to-path or metric-to-source tables, file or app maps, or escalation recipes into deeper artifact tiers.
- **Method recipes.** Concrete commands, code snippets, or step-by-step procedures for recurring task shapes.
- **Environment constants.** Verified formats, units, formulas, schema semantics, corpus counts, or identifier conventions, as distinct from task-answer values.
- **Located values.** Specific data values, computed results, verdicts, or recommendations recorded with their location or provenance.
- **Test anticipation.** Explicit modeling of likely test asks, including predicted prompts and pre-answered probes. Generic descriptions of the task genre do not count.
- **Trap and contradiction documentation.** Seeded errors, dirty data, conflicting versions or figures, version-supersession doctrine, or adjudication guidance.
- **Verification doctrine.** Verify-live rules, anti-fabrication warnings, or re-derive-before-quoting requirements.
- **Format and grading guidance.** Output-format laws, tolerance or precision semantics, or grading mechanics translated into solver behavior. In-corpus scoring worksheets do not count.
- **Negative maps and budget economics.** What is absent, empty, broken, or not worth reading, and measured cost models of operations.
- **Environment corrections.** Overriding the environment's own documentation or naive expectations, including data repair and tool-capability matrices.
- **Identity and social decoding.** Persona orientation, role or alias maps, or name-collision warnings.
- **Honesty disclosures.** Staleness, coverage, or truncation caveats about the artifact itself, declared unknowns, or provenance labels.

Artifact labeling. An LLM coder (`claude-fable-5`) labels each of the 216 artifact sets against the fixed category definitions. Corpus dumps inside CORPUS2SKILL outputs are excluded, and compiled skill trees are skimmed as machine-generated output. Each positive label requires a verbatim evidence quote. All 2,199 positive labels passed a mechanical check that the quote appears

in the cited file. A second, independent LLM coder re-labeled a stratified sample of 24 artifact sets, covering 288 category decisions. Agreement was 95.5% (275 of 288). Two disagreements affecting small-sample cells were adjudicated against the codebook. We treat the resulting counts as descriptive.

	BCP-G		OfficeQA		Harvey LAB		DABStep		APEX-Agents		AppWorld	
Artifact category	w/o	w/	w/o	w/	w/o	w/	w/o	w/	w/o	w/	w/o	w/
Navigation architecture	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	93/93	93/93	3/3	3/3
Method recipes	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	93/93	92/93	3/3	3/3
Environment constants	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	93/93	93/93	3/3	3/3
Located values	0/3	3/3	0/3	2/3	3/3	3/3	3/3	2/3	93/93	93/93	0/3	0/3
Test anticipation	1/3	1/3	2/3	3/3	2/3	1/3	3/3	3/3	60/93	61/93	3/3	1/3
Trap & contradiction documentation	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	92/93	83/93	0/3	1/3
Verification doctrine	1/3	3/3	3/3	3/3	3/3	3/3	2/3	3/3	34/93	93/93	1/3	3/3
Format & grading guidance	2/3	3/3	3/3	3/3	0/3	3/3	3/3	3/3	9/93	58/93	3/3	3/3
Negative maps & budget economics	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	93/93	75/93	3/3	2/3
Environment corrections	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	92/93	74/93	3/3	3/3
Identity & social decoding	0/3	0/3	0/3	0/3	3/3	3/3	0/3	0/3	92/93	93/93	2/3	3/3
Honesty disclosures	1/3	3/3	3/3	3/3	3/3	3/3	1/3	3/3	59/93	91/93	2/3	3/3

Table 13: Artifact content produced by the META-AGENT variants. Each cell reports artifact sets exhibiting the category out of all artifact sets for that variant. META-AGENT W/O ARCHIVE and META-AGENT W/ ARCHIVE denote the variants without and with the workflow archive, respectively. Iterations are pooled, and APEX-Agents is additionally pooled across worlds.

Artifact findings. Table 13 shows that navigation structures, method recipes, and environment constants appear in nearly every artifact set from both variants. Other content reflects the environment. Identity and social decoding is concentrated in Harvey LAB and APEX-Agents, while located values are absent from AppWorld, whose state is randomized across tasks. Archive-assisted artifacts also contain more operating guidance on APEX-Agents. Verification guidance increases from 34 of 93 unaided artifacts to all 93 archive-assisted artifacts, format and grading guidance from 9 to 58, and honesty disclosures from 59 to 91. These descriptive patterns do not establish which artifact properties caused changes in downstream reward.

B.4 QUALITATIVE ARTIFACT EXAMPLES

These excerpts illustrate two forms of environment-specific information captured during study: corrections to domain rules and consolidation of context scattered across the environment. Both were produced by META-AGENT W/ ARCHIVE in iteration one. Typography is normalized, and bracketed ellipses mark omitted text.

DABStep: correcting documentation and recovering computation rules. The artifact corrects the supplied documentation and records schema and fee-computation rules that cannot be inferred safely from field names alone. The excerpt comes from the `payment-data` environment group at `/meta_archive/artifacts/metanoarchive/study/README.md`.

> `payments-readme.md lists card_scheme as [MasterCard, Visa, Amex, Other]. WRONG. Real schemes = NexPay, GlobalCard, SwiftCharge, TransactPlus.`
>
> `[...] fee = fixed_amount + rate * transaction_value / 10000.`
>
> `A rule applies to a txn only if EVERY field matches. Rule field null/empty-list = wildcard (matches all).`
>
> `monthly_volume and monthly_fraud_level are not stored. They must be COMPUTED per merchant per calendar month from payments, then matched to the rule's range string.`

APEX-Agents: reconstructing identities and decision rationale. The artifact links inconsistent names across communication channels and preserves the rationale behind a project decision. The excerpt comes from `investment-banking-world-221` at `/meta_archive/artifacts/metanoarchive/study/02_people_and_narrative.md`.

```
Ola / "Tarek" Amethyst:  Deal structuring, M&A & pro forma
analysis (chat msg 71 addresses "Tarek" for M&A slides =
same person as "Ola").
Huajian / Luca Amethyst:  Pitch deck.  "Luca" does the slide
work in chat.
Names drift:  Ola ≈ Tarek, and Huajian ≈ Luca.  The email
From names and chat names don't perfectly align.  Treat by
ROLE, not name.
[...]  Final target = TPVG [...]  despite Luca flagging TPVG
is venture/growth-focused, outside BBDC's senior-secured
profile.  Justification:  diversification + tech/venture
exposure.
No DCF for BBDC. Valuation = Trading Comps only.  Merger
model = accretion/dilution.
```

The examples show how study artifacts can make both domain rules and dispersed contextual information directly available to the solver.

C Test-Time Interpretability

C.1 Observed Artifact Use

Determining whether an artifact influenced a rollout is inherently difficult. Artifacts may be accessed through explicit file reads or supplied through prompts, and a solver may act on injected guidance without visibly retrieving it. We therefore measure *observed artifact use*, not causal influence.

Trace labeling. We use GPT-5.6-Luna to classify task-time traces. A trace counts as observed use when the solver deliberately retrieves artifact content or clearly acts on distinctive guidance from it. Mere exposure, directory listings, failed retrievals, and generic behavior do not count. The judge receives the trajectory and mechanical annotations of artifact-related actions, but not its reward, corresponding No Study traces, or verifier output. Figures 7 and 8 provide the complete rubric.

Because artifact delivery differs across methods, "no observed use" does not establish that an artifact had no influence, and use rates should not be compared across methods.

Within-task comparison. For each task, method, and artifact iteration represented by both observed-use and no-observed-use trials, we compute the difference in mean reward between the two groups and average these differences across tasks. We omit cells with fewer than 10 eligible task–iterations. This controls for differences between tasks but remains observational because artifact use may follow events within a rollout. Table 14 reports these comparisons.

Method	BCP-G	OfficeQA	Harvey LAB	DABStep	APEX-Agents	AppWorld
PREPING	−0.054	+0.000	+0.005	+0.071	−0.005	−0.009
Corpus2Skill	–	–	–	−0.032	+0.020	−0.006
Meta-Agent w/o Archive	+0.015	–	+0.029	+0.112	+0.038	–
Meta-Agent w/ Archive	−0.044	−0.005	–	+0.080	−0.010	−0.042

Table 14: Within-task reward difference between observed-use and no-observed-use trials. Dashes indicate fewer than 10 contributing task–iterations. Full coverage counts will be included in the released analysis tables.

Observed use is not uniformly associated with higher reward. Among the within-task contrasts, the clearest positive associations occur on DABStep for PREPING and both Meta-Agent variants. The BCP-G contrasts for PREPING and Meta-Agent w/ Archive are negative. These patterns are descriptive and do not isolate the causal effect of artifact use.

C.2 Lift by Task Stratum

We next ask which kinds of tasks benefit from studying. We use benchmark-provided difficulty labels for OfficeQA (Opsahl-Ong et al., 2026) and DABStep (Egg et al., 2025), difficulty and application-

```
You are auditing whether a test-time solver agent USED a study artifact that was placed in its
environment. You will see the solver's trace and mechanical annotations. You are NOT judging
whether the artifact helped -- only whether the agent used it.

Output exactly one JSON object:
{"used": true|false,
 "channel": "<how the artifact was engaged, such as 'read /metanoarchive files', 'compiled
            get_document', 'playbook citation', or 'none'>",
 "evidence": ["2-5 bullets citing step ids and short quotes"],
 "confidence": 0.0-1.0}

USED (true) means at least one of:
- The agent, by its own tool call, brought artifact CONTENT into its context: a
  Read/cat/grep/head of a file under an artifact mount, a Skill load, or a SUCCESSFUL
  get_document call -- and the retrieved content is more than a bare directory listing.
- The agent's OWN authored text (message, reasoning, or tool arguments -- never user-role text
  or tool observations) engages injected artifact content: it cites a playbook ID (e.g.
  [strategies-NNN]/[pitfalls-NNN]), explicitly invokes the artifact ("According to the
  playbook, I should ..." / "Based on the playbook, I expect ...") and then acts on the stated
  plan -- this counts even when the plan's individual tactics are generic, because the agent
  ↪  is
  visibly steering by the artifact -- or reproduces a distinctive artifact-only fact, rule, or
  ORDERED multi-step recipe at a point where it shapes what the agent does next. Compare the
  agent's behavior against the injected playbook text you can see in the first user message: a
  specific fact or rule stated in the playbook and applied by the agent counts even without a
  citation. A single generic tactic (grep, read-the-docs) that any competent agent might use
  does not.

NOT used (false):
- Exposure alone: injected playbook/prefix text sitting in the prompt, or artifact text echoed
  back in observations, with no agent-authored engagement.
- A coerced or trivial touch: an `ls` of the mount, a mandated Skill call whose output the
  agent never draws on, or get_document calls that ALL errored.
- Generic diligence that matches artifact advice only thematically, with no citation and no
  artifact-only fact (model-default behavior).

Hard rules:
1. Only agent-authored text counts for citations/provenance. Artifact text arrives in
↪  user-role
   messages and large read observations -- matches there are exposure, not use.
2. Echo traps: /logs/agent dumps, tool-result re-reads, and embedded subagent transcripts do
   not count as the agent engaging the artifact.
3. Reading an artifact file and then visibly ignoring it still counts as used (content entered
   context by the agent's own action). Merely listing filenames does not.
4. When genuinely uncertain, set used=false with low confidence rather than guessing true.
```

Figure 7: Artifact-use judge rubric (part one of two): core criteria. Each prompt also includes the relevant method gate and benchmark note from Figure 8, mechanical annotations, and the rendered trace. Method gates take precedence over the core criteria.

count labels for AppWorld (Trivedi et al., 2024), domain labels for APEX-Agents (Vidgen et al., 2026), and topic and gold-document counts for BCP-G (Chen et al., 2026). Exact metadata sources and mappings will be included in the released analysis.

For each task stratum $\mathcal{G}$, we report

$$\frac{1}{|\mathcal{G}|} \sum_{i \in \mathcal{G}} \left(r_i - \bar{r}^{\mathrm{NS}}_{t(i)} \right), \tag{4}$$

where i indexes one studied rollout in stratum $\mathcal{G}$, r_i is its reward, and $t(i)$ is its task. We pool the three artifact iterations and three evaluation repetitions with equal weight. Every task therefore receives the same total weight. The quantity $\bar{r}^{\mathrm{NS}}_{t(i)}$ is the mean reward across all ten NO STUDY repetitions for the same task. This analysis is exploratory, and the comparisons are not adjusted for the number of strata examined.

Difficulty. Difficulty does not produce a universal pattern. On DABStep, the positive estimates for PREPING and both META-AGENT variants are substantially larger on hard tasks than on easy ones. META-AGENT W/ ARCHIVE improves reward by 0.206 on hard tasks but only 0.021 on easy tasks, where NO STUDY already scores 0.785. OfficeQA shows little separation between difficulty levels. On AppWorld, both META-AGENT variants have positive but non-monotonic gains, with their largest estimates at intermediate difficulty.

```
METHOD c2s: the corpus is compiled into navigable skills (Skill tool, INDEX/SKILL.md files,
↪  MCP
corpus-documents get_document, mounted under /opt/corpus2skill or similar). On most benchmarks
a system prompt MANDATES >=2 Skill calls and >=1 get_document. A mandated call still counts as
used ONLY if artifact content actually entered context (successful get_document, or
↪  index/skill
text visibly consulted beyond the coerced minimum). All-error get_document with no
skill-content engagement is not use.

METHOD META-AGENT W/O ARCHIVE. Small markdown files are mounted read-only at /metanoarchive or
↪  /context. The
task preamble tells the agent it may inspect them. Used = the agent Read/cat/grep'd artifact
file CONTENT (tool args show the path). A bare `ls` of the mount, or the mount merely being
named in prompts, is not use.

METHOD preping: a playbook is pasted into the FIRST USER MESSAGE (PLAYBOOK_BEGIN/END, bullets
tagged [strategies-NNN]/[pitfalls-NNN]/[apis-NNN]/[code_snippets-NNN]). Exposure is 100% by
construction. Only agent-authored engagement counts. Used = a playbook-ID citation in agent
text, OR agent text/code reproducing a distinctive playbook-only fact or recipe (a specific
file/table/endpoint the playbook names, a code snippet's distinctive shape) at a point that
shapes its next action. Generic strategies the model would use anyway are not use.

METHOD META-AGENT W/ ARCHIVE: a study bundle is delivered two ways at once -- (a) an
↪  orientation prefix
injected into the SYSTEM PROMPT (not visible in the trace, and it may embed a playbook with
[strategies-NNN]-style tags and names artifact file locations), and (b) artifact files mounted
under /meta_archive/... (e.g. /meta_archive/artifacts/metanoarchive/study/,
/meta_archive/artifacts/preping/playbook/). Used = the agent reads /meta_archive file content,
OR its own text cites a playbook ID or reproduces a distinctive bundle-only fact or recipe
(e.g. a corpus-era/table-format rule, a named source file, a prescribed search recipe) at a
decision point. Because the prefix is invisible here, weigh the mechanical annotations and
distinctive-content evidence carefully. Generic competence is not use.

Benchmark appworld: the agent drives simulated apps via mcp__appworld__* tools. c2s here has
↪  no
system mandate. Any skill/get_document access was a free choice. Meta-Agent w/o Archive files
↪  live under
/metanoarchive (e.g. 00_START_HERE.md, per-app API notes).

Benchmark APEX-Agents: finance/law/consulting tasks over /filesystem and /.apps_data. c2s
↪  get_document
is often broken ('not connected'/'No such tool'). All-error access is not use. The Meta-Agent
↪  w/o Archive output
is a per-world markdown pack. The preping playbook often names exact workbook paths, tab
indices, cell ranges, or figures. An agent going straight to a playbook-named path/tab or
quoting playbook-supplied figures is using the playbook even without a citation.

Benchmark bcp-grep (BrowseComp over plaintext /workspace/corpus with native Grep/Glob/Read).
The Meta-Agent w/o Archive README carries search mechanics (rg flags, triage recipes). c2s
↪  replaces search
with skills+get_document, so c2s corpus access is itself artifact use when content is
retrieved. The preping playbook prescribes an ORDERED candidate-list intersection recipe
(corpus-wide grep -rl for one clue, intersect with a second clue via xargs grep -l, then read
the intersection) -- an agent executing that ordered recipe is using the playbook even without
a citation. A lone grep is not.

Benchmark dabstep: data analysis over /app/data (payments.csv + docs). c2s compiles the same
files the agent can read directly. Only reads of the COMPILED copies or successful
↪  get_document
count as artifact use, reads of /app/data originals do not. Known distinctive playbook-only
rule: treating an EMPTY LIST in fees.json fields (account_type, aci) as a wildcard applying to
all values -- the corpus manual only covers null. An agent stating or applying
empty-list-as-wildcard is using the playbook.

Benchmark harvey_lab: legal research over a 9,288-doc DMS at /dms (labread CLI for binary
docs). Meta-Agent w/o Archive output = 00-START-HERE.md + matter-index.md under the artifact
↪  mount. Reads of /dms
originals are corpus access, not artifact use.

Benchmark officeqa: numeric answers from 697 Treasury Bulletin text files at /app/corpus.
Artifact mounts hold corpus maps and table-parsing recipes. Reads of /app/corpus originals are
corpus access, not artifact use.
```

Figure 8: Artifact-use judge rubric (part two of two): method gates and benchmark notes. Each prompt includes one of each. The CORPUS2SKILL gate excludes mandated calls that do not bring artifact content into context.

APEX-Agents gains concentrate in Law. All four methods have their largest domain-level lift on Law, ranging from +0.091 to +0.139. Estimates are smaller in Investment Banking, ranging from +0.004 to +0.017, and mixed in Management Consulting, ranging from −0.024 to +0.028.

Method	OfficeQA		DABStep		AppWorld		
	Easy	Hard	Easy	Hard	Difficulty 1	Difficulty 2	Difficulty 3
NO STUDY reward	0.696	0.424	0.785	0.271	0.746	0.458	0.513
PREPING	+0.070	+0.051	−0.005	+0.134	−0.017	+0.190	+0.053
CORPUS2SKILL	−0.011	−0.016	+0.007	−0.036	−0.038	+0.056	+0.064
META-AGENT W/O ARCHIVE	+0.085	+0.084	+0.016	+0.052	+0.087	+0.141	+0.113
META-AGENT W/ ARCHIVE	+0.080	+0.054	+0.021	+0.206	+0.059	+0.146	+0.071

Table 15: Task-matched reward lift over NO STUDY by dataset difficulty label. The first row gives the NO STUDY reward level of each stratum. Full counts and intervals will be included in the released analysis. AppWorld uses its three-level difficulty scale.

Method	Investment Banking	Law	Management Consulting
NO STUDY reward	0.074	0.246	0.212
PREPING	+0.017	+0.091	+0.028
CORPUS2SKILL	+0.009	+0.122	−0.024
META-AGENT W/O ARCHIVE	+0.004	+0.113	−0.003
META-AGENT W/ ARCHIVE	+0.010	+0.139	−0.006

Table 16: Task-matched reward lift over NO STUDY on APEX-Agents by task domain. The first row gives the NO STUDY reward level of each domain. Full counts and intervals will be included in the released analysis.

BCP-G retrieval strata. CORPUS2SKILL has positive lift across all ten topic labels (Table 19), indicating that its overall gain is not driven by a single topic. Its lift also rises from +0.232 with one gold document to +0.307 with four or more (Table 18). This pattern is consistent with corpus compilation being more useful when tasks require locating more supporting documents, although the analysis does not isolate retrieval burden from other task differences.

AppWorld application count. PREPING, CORPUS2SKILL, and META-AGENT W/ ARCHIVE have their largest estimated lift on tasks involving two applications, with gains from +0.094 to +0.127. META-AGENT W/O ARCHIVE peaks on one-application tasks at +0.125, narrowly above its +0.119 lift for two applications (Table 17).

Together, these results do not reveal a universal notion of task difficulty. Studying helps when the information captured during study matches the structure required by downstream tasks.

Method	1 app	2 apps	3 apps
NO STUDY reward	0.690	0.459	0.573
PREPING	+0.030	+0.127	−0.018
CORPUS2SKILL	−0.032	+0.094	−0.003
META-AGENT W/O ARCHIVE	+0.125	+0.119	+0.012
META-AGENT W/ ARCHIVE	+0.071	+0.118	+0.034

Table 17: Task-matched reward lift over NO STUDY on AppWorld by the number of applications required. The first row gives the NO STUDY reward level of each stratum. Full counts and intervals will be included in the released analysis.

C.3 HELPFUL AND HARMFUL ARTIFACT USE

Two matched examples illustrate how artifacts can help or hinder the solver. We selected cases with a clear trace-level mechanism and consistent outcome contrast across available repetitions. Each compares a META-AGENT W/ ARCHIVE rollout with a NO STUDY rollout on the same task and downstream model. The examples are not representative samples, are not paired by random seed, and should not be interpreted causally.

Method	1 gold document	2–3 gold documents	4+ gold documents
NO STUDY reward	0.219	0.185	0.189
PREPING	+0.091	+0.082	+0.090
CORPUS2SKILL	+0.232	+0.281	+0.307
META-AGENT W/O ARCHIVE	+0.055	+0.050	+0.044
META-AGENT W/ ARCHIVE	+0.098	+0.090	+0.078

Table 18: Task-matched reward lift over NO STUDY on BCP-G by the number of gold documents. The first row gives the NO STUDY reward level of each stratum. Full counts and intervals will be included in the released analysis.

	Topic									
Method	Art	Geography	History	Music	Politics	Science & tech.	Sports	TV & movies	Video games	Other
NO STUDY reward	0.221	0.181	0.172	0.173	0.247	0.329	0.147	0.130	0.152	0.219
PREPING	+0.085	+0.081	+0.087	+0.087	+0.090	+0.097	+0.087	+0.057	+0.066	+0.132
CORPUS2SKILL	+0.252	+0.335	+0.240	+0.322	+0.191	+0.223	+0.398	+0.248	+0.375	+0.227
META-AGENT W/O ARCHIVE	+0.062	+0.042	+0.043	+0.028	+0.048	+0.071	+0.053	+0.031	+0.028	+0.073
META-AGENT W/ ARCHIVE	+0.070	+0.067	+0.095	+0.081	+0.111	+0.136	+0.076	+0.059	+0.075	+0.113

Table 19: Task-matched reward lift over NO STUDY on BCP-G by BrowseComp topic label. The first row gives the NO STUDY reward level of each topic. Full counts and intervals will be included in the released analysis.

DABStep: recovering the correct fraud boundary. For task `1701`, the artifact records the dataset's fraud-level buckets:

```
monthly_fraud_level buckets:  <7.2%, 7.2%-7.7%, 7.7%-8.3%,
>8.3%.
= fraud volume / total volume that month.
```

The archive-assisted solver reads this guidance and applies the correct upper bucket:

```
Read /meta_archive/artifacts/metanoarchive/study/fee_
matching.md
April 2023 fraud rate:  9.92%
Fraud level bucket:  >8.3%
```

The matched NO STUDY solver computes the same fraud rate but invents a different boundary:

```
elif fraud_rate <= 9:  fraud_bracket = '8%-9%'
else:  fraud_bracket = '>9%'
monthly_fraud_level = '>9%'
```

The incorrect bucket propagates into the final Fee-ID set. NO STUDY fails all 10 evaluations on this task, while the artifact iteration shown here succeeds in all three of its evaluations.

APEX-Agents: an artifact creates a search tunnel. For task `world-129-pj-01`, the artifact directs the solver toward general ServiceNow pricing documents:

```
These documents collectively cover the competitive pricing,
packaging, and market positioning landscape for workflow
automation platforms ...  [including] ServiceNow.
Competitions_Price_Benchmarking.xlsx, ...  ServiceNow ITOM
HRSD SecOps modules.
```

The solver follows this route and substitutes unsupported assumptions for the task inputs:

```
The skills that seem most relevant are:
skill-02-workflow-automation-pricing ...
Assumptions for calculation:  ...  Total ServiceNow
customers:  1,000 ...  Average users per customer:  150
```

```
...  Module adoption varies by customer segment.
ServiceNow Total Annual Revenue Forecast:  $146.2m
```

The artifact does not index the two root-level images containing the required subscriber counts and price multipliers. The matched NO STUDY solver discovers those files through a directory listing and obtains the correct total of $1,438.2 million. NO STUDY succeeds in 6 of 10 evaluations on this task, whereas META-AGENT W/ ARCHIVE fails in all nine evaluations across the three artifact iterations. This example shows how a plausible but incomplete artifact can narrow the solver's search and hide relevant evidence.