NOT ALL SKILLS HELP: MEASURING AND REPAIRING AGENT SKILL LIBRARIES

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM agents can improve without weight updates by accumulating natural-language skills, but a plausible instruction need not help the task on which it is used. We propose **ASSAY**, a framework that connects skill-library repair and task-specific selection through shared execution measurements. Randomized masking estimates each skill's marginal causal effect under sampled skill contexts across development tasks. The resulting task-indexed profiles guide conditional rewrites offline and are combined over nearby development tasks to select skills for each test instruction. Task similarity retrieves development evidence; execution-based scores determine which eligible skills to suppress. On GPT-5.1 / AppWorld *test_normal*, ASSAY achieves 77.4% completion versus 70.8% for ACE with the same templates. On the same restructured library, task-conditioned masking outperforms count-matched random and global-score masking by 7.4 and 4.2 percentage points, respectively. Independently regenerating attribution yields 76.8%. Across seven models and four providers, the complete pipeline improves over the respective baselines on both AppWorld splits and for five of seven models on τ-bench retail. GPT-5.1 improves from 49.9% with ACE to 66.4% on *test_challenge*. These results establish the practical value of connecting skill generation to empirical curation and task-specific use.

1 INTRODUCTION

The past two years have seen rapid progress in LLM agents that improve without weight updates. The recipe is simple: let the agent attempt tasks, distill successful trajectories into natural-language *skills* (short rules, heuristics, procedural templates), and inject them into the context window for future tasks (Shinn et al., 2023; Zhao et al., 2024; Zhang et al., 2025; Gupta et al., 2025). On interactive benchmarks such as AppWorld (Trivedi et al., 2024), skill-based methods have produced double-digit gains, rivaling methods that fine-tune model weights. A central question remains: does a retained skill actually help on the task where it is used? Execution feedback and LLM judgment can produce useful instructions, but observing a successful trajectory with a skill present does not isolate the effect of including that skill.

Skill accumulation can have a downside: an instruction useful in its original context can become overhead when its application conditions no longer hold. On AppWorld, the GPT-5.1 skill library used by ACE reduces *test_challenge* completion from 52.5% to 49.9%, while ASSAY reaches 66.4%. Figure 1 illustrates masking applied directly to the upstream library: the full-library run follows off-topic rules, including a Spotify playback verification rule, and misses a dimensional constraint in an Amazon purchasing task; per-task masking suppresses two selected rules and the agent succeeds. A skill library records what an agent has learned; the skill text alone does not establish when applying it improves execution. Curation therefore concerns both an instruction's application conditions and its inclusion in the current task.

We address both decisions through task-indexed execution evidence. Randomized masking (Covert et al., 2021) on held-out tasks estimates how including each skill changes completion under sampled skill contexts. Rather than reducing each skill to one library-wide score, we retain its profile across development tasks. We present **ASSAY**, which uses these measurements in three stages: attribute skill effects, restructure the library offline, and personalise the retained context for each test task.

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.

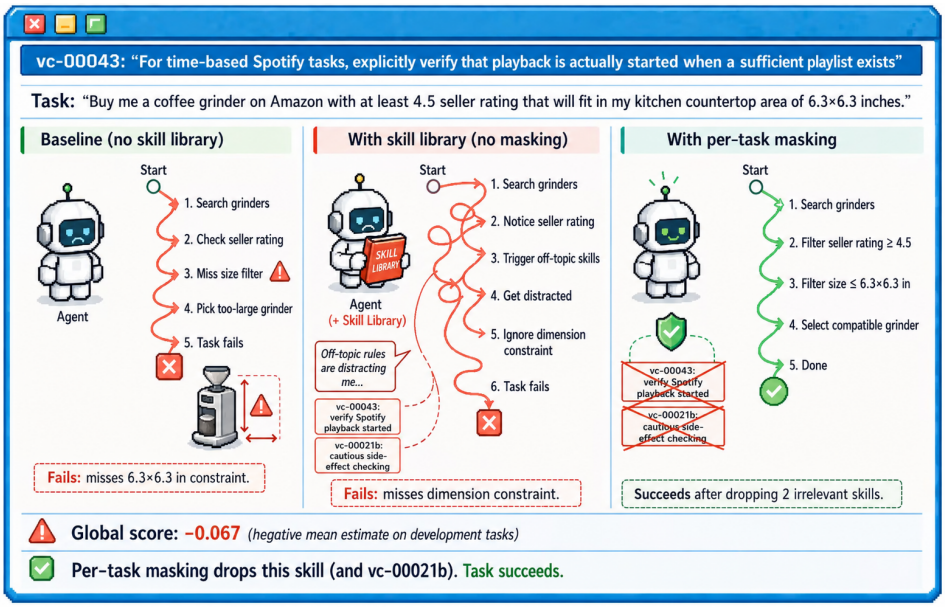

Figure 1: **Task-conditioned masking recovers a failed execution.** The task is to purchase a coffee grinder on Amazon that fits a 6.3×6.3-inch countertop and has a seller rating ≥4.5. *Left*: without any skill library, the agent fails to enforce the size constraint. *Centre*: with the upstream skill library, off-topic rules, including a Spotify playback verification rule (`vc-00043`) and a side-effect checking rule (`vc-00021b`), distract the agent from the dimensional requirement. *Right*: masking applied directly to this library suppresses the two skills and the agent succeeds. The estimated causal effect of `vc-00043` averages −0.067 across development tasks (bottom).

A shared attribution matrix links library revision to deployment selection. Offline, its task profiles guide conditional rewrites, and development executions decide whether to accept them. At deployment, task similarity retrieves development evidence; a weighted combination of the measured effects determines which eligible skills to suppress. Shared masks estimate all skills within a single KM-rollout attribution experiment, and inference requires no test-task reward query.

Our evaluation spans seven models, four providers, and two benchmarks. The complete pipeline improves over the respective baselines on both AppWorld splits for every model and on τ-bench retail for five of seven. Controlled comparisons on GPT-5.1 / AppWorld *test_normal* show gains beyond identical templates and, with removal counts held fixed, beyond random masking and task-independent attribution ranking. Independent re-attribution, development-set resampling, and increased mask budgets test sensitivity to how this evidence is acquired.

2 ASSAY: ATTRIBUTION-GUIDED SKILL CURATION

ASSAY represents each skill by its instruction text and a profile of measured effects across development tasks. These profiles form the attribution matrix $\mathbf{C} \in \mathbb{R}^{N \times M}$: its rows guide instruction edits, and weighted task columns yield deployment scores. We first measure this matrix (§2.1), then use it to restructure the shared library (§2.2) and select eligible skills for each test instruction (§2.3; Figure 2).

Preliminaries. We assume access to a skill library $\mathcal{S} = \{s_1, \ldots, s_N\}$ produced by an existing curation pipeline and a held-out development set $\mathcal{D} = \{d_1, \ldots, d_M\}$ disjoint from both training and test splits. During upstream curation, we apply difficulty-aware ordering: each training task is run twice with the bare agent to estimate difficulty, and tasks are sorted hardest-first so that the curator encounters high-signal failure cases early. The resulting library contains skills automatically distilled from training trajectories.

In addition to these learned skills, we append five hand-written operational templates (prefix `tpl-`) that capture common procedural patterns such as pagination, data validation, and cross-app identity

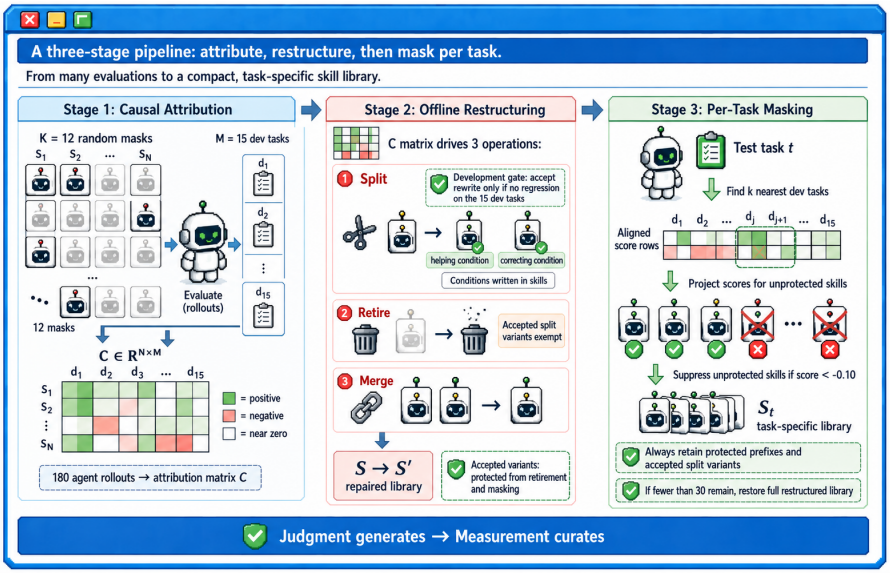

Figure 2: **Framework overview.** *Stage 1*: randomized masking produces a causal attribution matrix $\mathbf{C} \in \mathbb{R}^{N \times M}$; each cell estimates the effect of including a skill on a given task (green: positive; red: negative). *Stage 2*: split, retire, and merge restructure the library. A development gate accepts conditional rewrites; accepted variants are exempt from retirement and deployment masking. *Stage 3*: task-specific scores filter unprotected skills; protected prefixes and surviving accepted split variants are retained, with a fallback to the full restructured library.

resolution. These templates are derived from each benchmark's public documentation and training-split failure analysis, and are marked as protected: they participate in attribution trials but are exempt from subsequent modification and deployment-time masking (full text in Appendix B).

2.1 Measuring Causal Effects

A skill library is a collection of natural-language instructions that jointly shape the agent's behaviour. The challenge in evaluating any single skill is that its effect depends on which other skills are co-present: a verification rule may be harmless alone but harmful alongside a pagination rule that already performs the same check. To measure skill effects across co-occurring subsets, we turn to a randomized experiment.

Randomized masking protocol. For each of K independent trials, we construct a random mask $m_k \subseteq \mathcal{S}$ by including each skill independently with probability f (Bernoulli sampling). Let $\mathbf{1}[s_j \in m_k]$ denote the inclusion indicator for skill s_j in mask m_k, and let $o_k(d_i) \in \{0, 1\}$ denote the binary outcome on development task d_i when the agent operates under mask m_k. Define the sets of masks that include and exclude skill s_j as $\mathcal{M}_j^+ = \{k : s_j \in m_k\}$ and $\mathcal{M}_j^- = \{k : s_j \notin m_k\}$, respectively. The causal score of skill s_j on task d_i is the difference-in-means estimator:

$$\mathbf{C}[j, i] = \frac{1}{|\mathcal{M}_j^+|} \sum_{k \in \mathcal{M}_j^+} o_k(d_i) - \frac{1}{|\mathcal{M}_j^-|} \sum_{k \in \mathcal{M}_j^-} o_k(d_i). \tag{1}$$

For nonempty inclusion and exclusion groups, independent Bernoulli trials under a fixed agent protocol imply that $\mathbf{C}[j, i]$ is an unbiased estimator of the average treatment effect (ATE) of including skill s_j on task d_i, marginalised over the Bernoulli co-occurrence distribution at $f = 0.4$. Interactions contribute to this marginal effect through the sampled subsets; the estimator does not require additive skill effects. Conditional on the inclusion and exclusion group sizes, the variance is bounded by

$$\mathrm{Var}\big(\mathbf{C}[j, i] \mid |\mathcal{M}_j^+|, |\mathcal{M}_j^-|\big) \leq \frac{1}{4}\left(\frac{1}{|\mathcal{M}_j^+|} + \frac{1}{|\mathcal{M}_j^-|}\right), \tag{2}$$

since $o_k(d_i) \in \{0, 1\}$ implies $\mathrm{Var}(o_k(d_i)) \leq \frac{1}{4}$. The same K masks supply inclusion/exclusion comparisons for every skill on each task, making the attribution budget KM rollouts. Deployment decisions use neighbor-weighted combinations of development-task estimates (Eq. 6), with their utility evaluated through matched selection controls (§3.2). All upstream skills participate in attribution; prefix-protected skills remain exempt from downstream editing and deployment-time masking (§2.3). Hyperparameters are reported in §3; attribution diagnostics are in Appendix I.

Row statistics. The *global causal score* of skill s_j is the row mean:

$$\bar{C}(j) \;=\; \frac{1}{M} \sum_{i=1}^{M} \mathbf{C}[j, i], \tag{3}$$

capturing the average marginal contribution of s_j across all development tasks. The *estimated heterogeneity score* of skill s_j is the row range:

$$H(s_j) \;=\; \max_i \mathbf{C}[j, i] \;-\; \min_i \mathbf{C}[j, i], \tag{4}$$

summarising variation in the estimated effects across tasks.

Definition 1 (Heterogeneity candidate)**.** *A skill s_j is a candidate for conditional rewriting at threshold τ if $H(s_j) \geq \tau$. The range identifies variation to inspect; sign reversals additionally require estimates on both sides of zero.*

A skill with positive and negative task effects can have near-zero $\bar{C}(j)$ even when its application conditions matter. The full task profile preserves information that the global mean compresses. We use H to propose rewrites, with acceptance determined by the development gate below. The full attribution is computed separately for each base model and benchmark, because the causal structure of a skill library depends on the model that interprets it.

2.2 Offline Library Restructuring

Offline restructuring uses the task profiles to propose changes to the shared library. High-range profiles suggest application conditions for *splitting*; row means guide *retirement*; embedding similarity identifies near-duplicates for *merging*. We apply these operations in that order, evaluating proposed splits through a development gate.

Step 1: Split. For each skill s_j with $H(s_j) \geq \tau_{\text{split}}$, we use the base LLM to rewrite s_j into two conditional variants, each with an explicit trigger condition specifying when it should apply. The rewriting is guided by the causal score vector $(\mathbf{C}[j, 1], \ldots, \mathbf{C}[j, M])$: one preserves behavior associated with positive parent estimates, while the other revises behavior associated with negative estimates. The estimated task profiles guide rewrite proposals; each proposed pair is then evaluated through a *development gate* requiring the restructured library to match or exceed the original on every one of the M attribution tasks. If it does not, the original skill is kept. Splitting runs first because positive and negative task effects can yield $\bar{C}(j) \approx 0$; conditional rewriting can preserve useful instructions before the retirement rule is applied. At most $\tau_{\text{max_split}}$ candidates are processed. For example, a blanket pagination rule becomes conditional: enumerate all pages for browsing tasks, but focus on the specified item for targeted purchases (Appendix G). Accepted variants receive the parent's positive entries and sign-flipped negative entries, respectively, with zeros elsewhere; these assigned scores are not re-measured effects. Accepted variants are exempt from retirement and per-task masking. Downstream, $\mathbf{C}$ denotes the aligned measured and inherited rows (Appendix G.1).

Step 2: Retire. After splitting, retirement removes skills with $|\bar{C}(j)| < \tau_{\text{retire}}$, except prefix-protected skills and accepted split variants.

Step 3: Merge. Splitting may introduce near-duplicate variants. Remaining skills are embedded; pairs exceeding cosine similarity τ_{merge} are clustered, and the highest-scoring member of each cluster (by $\bar{C}$) is retained. Merging removes near-duplicates from the library resulting from splitting and retirement.

2.3 Per-Task Causal Masking

Offline restructuring determines the shared library $\mathcal{S}'$; per-task masking determines which eligible instructions a particular execution sees. For each test instruction, we retrieve nearby development

tasks and combine their measured skill effects. Selection therefore depends on execution evidence associated with those tasks, rather than directly ranking skill text by semantic similarity.

Task-specific effect prediction. We predict a skill's effect on a new task by transferring the estimates from similar development tasks. Given a test task t with instruction embedding $\mathbf{e}_t \in \mathbb{R}^d$, we identify the k nearest development tasks $\mathcal{N}(t) \subset \mathcal{D}$ by cosine similarity and compute attention weights via a temperature-scaled softmax:

$$w_i(t) \;=\; \frac{\exp\big(\tau \cdot \cos(\mathbf{e}_t,\, \mathbf{e}_{d_i})\big)}{\displaystyle\sum_{i' \in \mathcal{N}(t)} \exp\big(\tau \cdot \cos(\mathbf{e}_t,\, \mathbf{e}_{d_{i'}})\big)}, \qquad i \in \mathcal{N}(t), \tag{5}$$

where τ is an inverse-temperature parameter that concentrates weight on the closest neighbours. For an unprotected skill s_j, the task-specific prediction projects its measured score row onto nearby development tasks:

$$\hat{C}(s_j,\, t) \;=\; \sum_{i \in \mathcal{N}(t)} w_i(t) \cdot \mathbf{C}[j, i] \;=\; \mathbf{C}[j, :]\, \mathbf{w}(t), \tag{6}$$

where $\mathbf{w}(t) \in \mathbb{R}^M$ is the weight vector (zero outside $\mathcal{N}(t)$). Each test task induces a weighted combination of development-task columns, yielding task-specific scores for the skills eligible for masking. Prediction uses the test instruction and development evidence, without querying test-task outcomes.

Conservative masking rule. We retain protected skills and apply a negative threshold to the remaining task-specific scores:

$$\mathcal{S}_t \;=\; \big\{ s_j \in \mathcal{S}' : \hat{C}(s_j, t) \geq \tau_{\text{mask}} \;\text{ or }\; s_j \in \mathcal{S}_{\text{protected}} \big\}, \tag{7}$$

Here $\mathcal{S}_{\text{protected}} \subseteq \mathcal{S}'$ contains skills with protected prefixes (`tpl-`, `shr-`, `api-`) and surviving split variants accepted by the development gate. If fewer than 30 skills remain, the full restructured library $\mathcal{S}'$ is restored. The asymmetry of Eq. (7) is deliberate: the threshold $\tau_{\text{mask}} = -0.10$ retains near-zero predictions, while protection preserves operational templates and accepted conditional rewrites. Algorithm 1 summarises this inference procedure.

Algorithm 1 PER-TASK CAUSAL MASKING (inference time)

Input: Test task t; restructured library $\mathcal{S}'$; aligned score array $\mathbf{C}$; protected set $\mathcal{S}_{\text{protected}}$; development embeddings; k, τ, τ_{mask}, $\tau_{\min}$.
Output: Task-specific skill library $\mathcal{S}_t \subseteq \mathcal{S}'$.

1. Compute embedding $\mathbf{e}_t$ of the task instruction.
2. Find the k nearest development tasks $\mathcal{N}(t)$ by cosine similarity.
3. Compute weights $\mathbf{w}(t)$ by Eq. (5).
4. For each unprotected s_j, compute $\hat{C}(s_j, t)$ by Eq. (6).
5. Form $\mathcal{S}_t$ by retaining protected skills and scores $\geq \tau_{\text{mask}}$ (Eq. (7)).
6. If $|\mathcal{S}_t| < \tau_{\min}$, restore $\mathcal{S}_t \leftarrow \mathcal{S}'$.
7. Return $\mathcal{S}_t$.

3 EXPERIMENTS

We evaluate the complete pipeline across seven models, four providers, and two agent architectures. Controlled experiments on GPT-5.1 / AppWorld *test_normal* examine the selection mechanism and robustness; broader comparisons assess task completion under the same pipeline and hyperparameters.

3.1 EXPERIMENTAL SETUP

Benchmarks. AppWorld (Trivedi et al., 2024) simulates nine consumer applications (email, calendar, Venmo, Spotify, Amazon, etc.) in which the agent composes multi-step Python API calls via a code

Table 1: **AppWorld results** (TGC, %). Δ is computed relative to bare ReAct. **Bold**: best displayed method within each model block. Red: degrades over ReAct. For DeepSeek, V3.2 identifies our ASSAY and ACE reproduction; ReAct is a reported reference. The GPT-4o ReAct row uses an earlier snapshot (Appendix E).

		test_normal (168)		*test_challenge* (417)	
Model	**Method**	TGC	Δ	TGC	Δ
GPT-5.1	ReAct	61.9		52.5	
	ACE	67.3	+5.4	49.9	−2.6
	Ours	**77.4**	+15.5	**66.4**	+13.9
DeepSeek-V3.2	ReAct (reference)	69.1		47.0	
	ACE (reproduced)	78.0	+8.9	63.1	+16.1
	Ours	**83.3**	+14.2	**69.3**	+22.3
GPT-4.1	ReAct	66.7		50.4	
	CUGA	73.2	+6.5	57.6	+7.2
	Ours	**75.6**	+8.9	**64.0**	+13.6
GPT-4o	ReAct	48.8		30.2	
	Gupta et al.	68.5	+19.7	38.9	+8.7
	Ours	**71.4**	+22.6	**41.0**	+10.8
GPT-5.4	ReAct	82.1		81.1	
	Ours	**88.7**	+6.6	**85.4**	+4.3
Sonnet 4.5	ReAct	83.9		70.3	
	Ours	**89.3**	+5.4	**75.3**	+5.0
Gemini 2.5	ReAct	72.6		49.4	
	Ours	**81.0**	+8.4	**54.9**	+5.5
Historical reference (Qwen3-14B, wt-tuned)*		*86.9*		*67.6*	

*No accompanying publication; see footnote in §3.

REPL. We evaluate on two official test splits: *test_normal* (168 tasks) and *test_challenge* (417 tasks), both spanning difficulty levels 1–3. *test_challenge* includes tasks involving designated held-out applications. The primary metric is Task Goal Completion (TGC), requiring all task evaluation tests to pass. Appendix H distinguishes it from Scenario Goal Completion (SGC). The agent is a ReAct agent (Yao et al., 2022) with a 40-step budget. **τ-bench** (Yao et al., 2024) simulates retail customer service through function-calling tools. The agent converses with an LLM-simulated customer (GPT-4o, temperature 0) and must satisfy requests while adhering to a policy document. We evaluate on the retail domain (115 tasks) with exact database-state match; partial credit is not awarded. The agent is the benchmark's native `ToolCallingAgent` with a 30-turn budget. In both benchmarks, the skill library is injected as part of the system prompt and all agent calls use temperature 0.

Models and data. We evaluate seven base models: GPT-5.4, GPT-5.1, GPT-4.1, and GPT-4o (OpenAI); DeepSeek-V3.2 (DeepSeek, open-weight); Claude Sonnet 4.5 (Anthropic); and Gemini 2.5 Pro (Google). Model identifiers and decoding settings are provided in Appendix E. Each model runs the complete pipeline independently; the attribution matrix $\mathbf{C}$ is recomputed per model because the causal structure of a skill library depends on the model that interprets it (§2.1). Data is partitioned into strictly disjoint sets: AppWorld uses 90 training, 15 development (of 57), and 168+417 test tasks; τ-bench uses 500 training, 15 development (of 20), and 115 test.

Baselines. For AppWorld, the available skill-based comparisons are ACE (Zhang et al., 2025) for GPT-5.1 and our ACE reproduction with DeepSeek-V3.2, CUGA (Marreed et al., 2025) for GPT-4.1, and Gupta et al. (2025) for GPT-4o. We additionally report a historical weight-tuned AppWorld leaderboard reference.[1] On τ-bench, we compare against each model's unaugmented baseline, under

[1]Alibaba Cloud ApsaraLab, AppWorld leaderboard submission (February 2026, Qwen3-14B, weight-tuned). No accompanying publication is available as of this writing.

the reported benchmark protocol. For GPT-5.4, Claude Sonnet 4.5, and Gemini 2.5 Pro, we compare against unaugmented ReAct. The GPT-4o ReAct reference uses an earlier model snapshot, as detailed in Appendix E.

Attribution parameters. Attribution uses $K{=}12$ random masks with inclusion probability $f{=}0.4$ on $M{=}15$ development tasks: 180 rollouts per model–benchmark configuration, shared across all skills. Per-task masking combines $k{=}8$ nearest development tasks. Appendix Table 3 lists all hyperparameters; no per-cell tuning is performed. Appendix E separates attribution cost from upstream curation and gate evaluation; §3.2 varies attribution masks, development tasks, and budget.

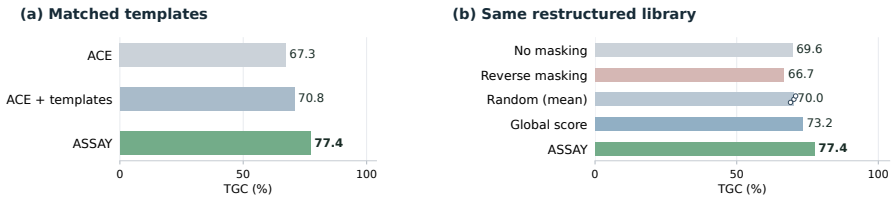

Figure 3: **Controlled comparisons** (GPT-5.1 / AppWorld *test_normal*). (a) Identical templates. (b) Same restructured library; random and global-score controls match ASSAY's per-task removal counts. Dots show the three random seeds; the bar shows their mean. Protocols are in Appendix F.

3.2 Controlled Evaluation and Robustness

All comparisons in this subsection and the following ablation use GPT-5.1 / AppWorld *test_normal* (168 tasks).

Template-matched comparison. Under the same prompt budget and evaluation protocol, ACE with the same five templates reaches 70.8%, versus 77.4% for ASSAY (Figure 3a; Appendix Table 5).

Independent re-attribution. Regenerating all $K = 12$ masks (seed 7) and rerunning the 180 development rollouts and downstream pipeline solves 129/168 tasks versus 130/168 originally (paired outcomes and diagnostics in Appendix I.1).

Development-set and attribution-budget sensitivity. Resampling three sets of 15 development tasks and rerunning the full pipeline yields 75.6–78.0% TGC (mean 76.6%). Across these sets, ASSAY solves 5–7 more tasks than count-matched global-score masking and exceeds random masking by 6.4–7.4 points. With the original development set fixed, independently generated budgets of $K = 12, 24, 48$ masks yield 130, 131, and 128 successes. These experiments support the selection advantage across the sampled development sets and the practical use of $K = 12$ in this configuration (Appendices I.2–I.3).

3.3 Ablation Study

On the same offline-restructured library, per-task masking raises GPT-5.1 / AppWorld *test_normal* completion from 69.6% (117/168) to 77.4% (130/168), isolating a 7.8-point increment at inference (Figure 3b; Appendix Table 6).

Reverse masking instead removes the five highest-scoring eligible skills and reduces TGC to 66.7%, 2.9 points below the same no-masking configuration. Appendix I.6 specifies the protocol.

Matched selection controls. Holding the restructured library and per-task removal counts fixed, random masking achieves 70.0% TGC (three-seed mean) and masking by the lowest global scores $\bar{C}(j)$ achieves 73.2% (123/168). ASSAY reaches 77.4%, exceeding them by 7.4 and 4.2 points (Figure 3b). The comparison separates the value of score-guided selection from the further gain obtained by retaining the task dimension. Appendix Table 7 reports every random-seed result.

3.4 Main Results

AppWorld. Table 1 shows gains over the respective baselines for all seven models on both splits. On *test_normal*, GPT-5.1 improves from 67.3% to 77.4% and DeepSeek-V3.2 from 78.0% to 83.3%,

each starting from the same upstream learned library as its ACE comparison. GPT-4.1 and GPT-4o improve over CUGA and Gupta et al. (2025); the other three models improve over unaugmented ReAct. On *test_challenge*, GPT-5.1's upstream library reduces completion from 52.5% to 49.9%, while ASSAY reaches 66.4%. DeepSeek-V3.2 reaches 69.3%. These results evaluate the complete pipeline across models; the preceding controls isolate selection on GPT-5.1 / *test_normal*.

Table 2: **τ-bench retail results** (%). Δ is the gain in percentage points. GPT-4.1 and GPT-4o baselines are reported public-leaderboard results.

Model	Unaugmented	ASSAY	Δ
GPT-5.4	73.9	**80.9**	+7.0
GPT-4.1	68.0	**73.9**	+5.9
Gemini 2.5 Pro	65.2	**73.9**	+8.7
Sonnet 4.5	73.0	73.0	0.0
DeepSeek-V3.2	63.5	**66.1**	+2.6
GPT-5.1	62.6	62.6	0.0
GPT-4o	60.4	**62.6**	+2.2

τ-bench. Table 2 evaluates whether the framework transfers to a qualitatively different setting: conversational rather than code-based, function-calling rather than REPL, and with a simulated human in the loop.

Five of seven models improve without weight modification (Table 2). GPT-5.1 and Sonnet 4.5 retain their unaugmented scores, although both improve on AppWorld. The benefit therefore varies across model–environment combinations (Appendix A.3).

3.5 Task-Dependent Attribution Profiles

Two upstream skills illustrate the information retained in a task profile. The displayed estimates and their task contexts show what a near-zero row mean compresses.

Example 1: Contact Validation Rule (`vc-00054`)

Skill. "Validate that note names map unambiguously to contacts before creating Venmo requests."
Global score. $\bar{C} = -0.03$ (near zero).
Positive estimate ($+0.50$): shared-expense reconciliation, where name validation addresses potential ambiguities.
Negative estimate (-0.67): single-app execution, where the instruction introduces additional contact cross-referencing.

Example 2: Response Checking Rule (`vc-00021a`)

Skill. "For repeated side-effecting API calls, capture and check each response."
Global score. $\bar{C} = +0.05$ (near zero).
Positive estimate ($+0.71$): side-effecting API calls, where response checking can expose silent failures.
Negative estimate (-0.23): read-only execution, where the instruction adds response-checking steps.

These profiles supply task contexts to inspect when specifying application conditions. At inference, Eq. 6 combines such task-level estimates into a selection score for each eligible skill (Appendix D).

4 Related Work

Skill generation and curation. A growing line of work improves agent performance by accumulating experience into the agent's context rather than its weights. The methods differ in mechanism:

Reflexion (Shinn et al., 2023) maintains verbal self-reflections in an episodic memory buffer, ExpeL (Zhao et al., 2024) extracts reusable insights by comparing successful and failed trajectories, ACE (Zhang et al., 2025) grows evolving playbooks through a modular generate-reflect-curate loop, and CUGA (Marreed et al., 2025) adopts a hierarchical planner-executor architecture with context enrichment. More recently, SkillNet (Liang et al., 2026) provides end-to-end tooling to create, evaluate, and connect skills within a unified ontology, and CoEvoSkills (Zhang et al., 2026) co-evolves a skill generator and surrogate verifier without access to ground-truth tests. Our work treats the output of any such pipeline as raw material for a second, empirical curation stage.

Skill retrieval and application. At test time, skill selection must decide which parts of the accumulated experience apply. Retrieval-augmented generation (Lewis et al., 2020) and its agent-oriented variants (Zhang et al., 2024) use relevance signals to select context. Gupta et al. (2025) extend BERTScore-Recall to set-level demonstration selection for in-context learning in agentic tasks, and Su et al. (2026) study skill retrieval augmentation at scale as agent skill libraries grow to thousands of entries. Relevance and execution utility can differ: a skill about "verifying list contents before iteration" can be semantically close to a single-record lookup yet introduce unnecessary verification steps. Per-task filtering uses task similarity to transfer measured effects: the final selection signal is predicted causal effect, not task–skill similarity alone.

Skill optimization beyond judgment. Several methods go beyond single-task judgment to optimize skill libraries. Voyager (Wang et al., 2024) builds an expanding library of executable skills. SkillRL (Xia et al., 2026a) uses reinforcement learning to recursively refine skills through failure analysis, treating the library as a dynamic component co-evolving with the agent policy. EvolveR (Wu et al., 2025) closes an experience-driven evolution loop, and Agentic Memory (Yu et al., 2026) optimizes memory management with GRPO. SkillClaw (Ma et al., 2026) enables collective skill evolution by aggregating trajectories across users to identify recurring behavioral patterns, and GraSP (Xia et al., 2026b) compiles flat skill sets into typed directed acyclic graphs, observing that providing agents with more skills does not monotonically improve performance. Weight-based methods such as SAGE (Wang et al., 2025) and FireAct (Chen et al., 2023) internalise skills into model parameters via RL or trajectory fine-tuning, avoiding context-window limitations but requiring retraining for each base model. Our contribution connects conditional library repair and deployment selection through task-indexed execution measurements. For measurement, we draw on component attribution (Lundberg & Lee, 2017) and randomized ablation (Covert et al., 2021), applied to natural-language skill instructions. Bernoulli sampling targets a specified co-occurrence distribution; it does not use Shapley coalition weights. Full-library leave-one-out evaluates each skill at one fixed background and requires $(N + 1)M$ rollouts when each configuration is evaluated once per task. Shared masks estimate all skills with KM executions. This budgeted measurement feeds both library edits and deployment selection.

5 Conclusion

We presented ASSAY, which connects skill-library repair and task-specific use through shared execution measurements. Preserving a task-indexed profile makes the same evidence available for proposing conditional instructions and selecting eligible skills at deployment. Across models and environments, the complete pipeline improves task completion. Matched-removal controls demonstrate the selection advantage of task-conditioned scores in the controlled setting; this advantage persists across resampled development sets. Task completion ranges from 76.2% to 78.0% across the tested attribution budgets. Skill generation produces candidate knowledge; empirical curation determines how that knowledge enters the agent's working context.

AI USE STATEMENT

LLMs are central to this work: as the agents under evaluation (Section 3), as the skill generators during upstream curation and the rewriters in the split step (Section 2), and as simulated users in τ-bench (Section 3). The split step reintroduces LLM judgment into curation, explicitly bounded and validated through the development gate. In preparing this manuscript, AI tools assisted with literature search, text revision and language polishing, and figure and LaTeX formatting.

REPRODUCIBILITY STATEMENT

Both benchmarks are publicly available. Appendix E provides pipeline hyperparameters, model API identifiers, benchmark versions, random seeds, and library sizes. The attribution protocol and thresholds are specified in Section 2 and Appendix E; operational templates are provided in Appendix B. Algorithm 1 describes inference, and Appendices F and I provide additional ablation and statistical details.

REFERENCES

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*, 2023.

Ian Covert, Scott Lundberg, and Su-In Lee. Explaining by removing: A unified framework for model explanation. *Journal of Machine Learning Research*, 22(209):1–90, 2021.

Shivanshu Gupta, Sameer Singh, Ashish Sabharwal, Tushar Khot, and Ben Bogin. Leveraging in-context learning for language model agents. *arXiv preprint arXiv:2506.13109*, 2025.

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.

Yuan Liang, Ruobin Zhong, Haoming Xu, Chen Jiang, Yi Zhong, Runnan Fang, Jia-Chen Gu, Shumin Deng, Yunzhi Yao, Mengru Wang, et al. Skillnet: Create, evaluate, and connect AI skills. *arXiv preprint arXiv:2603.04448*, 2026.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

Ziyu Ma, Shidong Yang, Yuxiang Ji, Xucong Wang, Yong Wang, Yiming Hu, Tongwen Huang, and Xiangxiang Chu. Skillclaw: Let skills evolve collectively with agentic evolver. *arXiv preprint arXiv:2604.08377*, 2026.

Sami Marreed, Alon Oved, Avi Yaeli, Segev Shlomov, Ido Levy, Offer Akrabi, Aviad Sela, Asaf Adi, and Nir Mashkif. Towards enterprise-ready computer using generalist agent. *arXiv preprint arXiv:2503.01861*, 2025.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Weihang Su, Jianming Long, Qingyao Ai, Yichen Tang, Changyue Wang, Yiteng Tu, and Yiqun Liu. Skill retrieval augmentation for agentic AI. *arXiv preprint arXiv:2604.24594*, 2026.

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

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.

Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. Reinforcement learning for self-improving agent with skill library. *arXiv preprint arXiv:2512.17102*, 2025.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, and Botian Shi. EvolveR: Self-evolving LLM agents through an experience-driven lifecycle. *arXiv preprint arXiv:2510.16079*, 2025.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026a.

Tianle Xia, Lingxiang Hu, Yiding Sun, Ming Xu, Lan Xu, Siying Wang, Wei Xu, and Jie Jiang. GraSP: Graph-structured skill compositions for LLM agents. *arXiv preprint arXiv:2604.17870*, 2026b.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Yi Yu, Liuyi Yao, Yuexiang Xie, Qingquan Tan, Jiaqi Feng, Yaliang Li, and Libing Wu. Agentic memory: Learning unified long-term and short-term memory management for large language model agents. *arXiv preprint arXiv:2601.01885*, 2026.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, Xue Liu, Xiaoxiao Li, and Philip S. Yu. Coevoskills: Self-evolving agent skills via co-evolutionary verification. *arXiv preprint arXiv:2604.01687*, 2026.

Jianguo Zhang, Tian Lan, Rithesh Murthy, Zhiwei Liu, Weiran Yao, Ming Zhu, Juntao Tan, Thai Hoang, Zuxin Liu, Liangwei Yang, et al. Agentohana: Design unified data and training pipeline for effective agent learning. *arXiv preprint arXiv:2402.15506*, 2024.

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. *arXiv preprint arXiv:2510.04618*, 2025.

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 Extended Analysis and Discussion

A.1 The Landscape of Skill Effects

The task-indexed matrix supports inspection at two resolutions. Figure 4 shows selected score profiles and their row summaries; Appendix D connects two profiles to the corresponding skill instructions and task contexts. The following analysis describes how selection varies across test tasks.

A.2 Masking Behaviour

We now examine how per-task masking operates in practice. On GPT-5.1 / *test_challenge*, masking remains active on 329 of 417 tasks, suppressing a mean of 5.5 skills (5.3% of the library; standard deviation 2.8) within this non-fallback subset. The remaining 88 tasks (21.1%) use the full library through fallback and are excluded from these suppression statistics. Only 2 skills are suppressed on more than 80% of non-fallback tasks. The selected subsets vary with the task, following negative predicted effects transferred from nearby development tasks.

Figure 1 illustrates task-specific selection directly from the upstream library. Task `6474048_2` (the coffee grinder from §1) fails both without skills (incorrect dimension filtering) and with the full library (off-topic Spotify and side-effect-checking rules distract the agent). Per-task masking drops exactly these two irrelevant skills, and the agent succeeds, illustrating how measured effects guide a task-specific change in skill exposure.

A.3 Limitations

The evaluation uses a fixed skill library and attribution measured separately for each model–benchmark configuration. Extending ASSAY to continuously evolving libraries would require refreshing this evidence as skills change. With a finite attribution budget, task-level estimates carry sampling uncertainty, and transfer to new tasks depends on coverage by the development instructions. The measured effects are marginal over sampled skill subsets; effects within a particular deployed combination may differ.

Gains vary across model–environment combinations. All seven models improve on both AppWorld splits, while five improve on τ-bench retail and two retain their unaugmented scores. Robustness experiments on GPT-5.1 / AppWorld *test_normal* evaluate changes to attribution masks, development tasks, and mask budgets.

B Operational Templates

The five AppWorld templates and five τ-bench retail templates below are appended to every skill library as a protected bedrock layer. They are exempt from downstream modification and deployment-time masking; they participate in randomized attribution trials. The templates encode benchmark-specific operational instructions for data collection, validation, authentication, entity resolution, sequencing, and confirmation. In code-generation agents (AppWorld), templates take the form of executable code scaffolds; in conversational agents (τ-bench), they take the form of procedural checklists derived from the benchmark's policy document.

AppWorld Templates. `[tpl-00001] PAGINATION TEMPLATE.` Use for any API that returns a list. Always paginate with a `while True` loop over `page_index`; never use `range(N)` or fetch only one page. Accumulate results into `all_items` and break when the page is empty.

`[tpl-00002] DATA VALIDATION TEMPLATE.` Before acting on any data, print and inspect intermediate results (item count, sample keys, first item). Only proceed after confirming the data looks correct.

`[tpl-00003] SAFE COMPLETE_TASK TEMPLATE.` Before submitting an answer, print it alongside the number of source data points from which it was derived. Only then call `complete_task`.

`[tpl-00004] LOGIN + ACCESS TOKEN TEMPLATE.` Standard login flow: retrieve account passwords via `supervisor.show_account_passwords`, match by app name, call the app's `login` endpoint, and store the access token.

`[tpl-00005] CROSS-APP IDENTITY RESOLUTION TEMPLATE.` When resolving a person across apps, always start from the phone contacts directory (paginated). Match by email or phone number, never by name alone.

τ-bench Retail Templates. `[tpl-tb-001] ITEM MODIFICATION CHECKLIST.` Before calling `modify_pending_order_items` or `exchange_delivered_order_items`: (1) call `get_product_details` for each item, (2) present options and confirm exact choice, (3) ask if these are all the items, (4) collect all changes into a single list, (5) make one tool call. These tools can only be called once.

`[tpl-tb-002] EXCHANGE ITEM RESOLUTION.` Six-step procedure: get current item IDs, fetch product details for available variants, match customer requirements, present ambiguous options, compare prices if requested, confirm exact new item ID.

`[tpl-tb-003] MULTI-ORDER SEQUENCING.` For requests involving multiple orders: get all order IDs, check each order's status, process sequentially, confirm each action individually, track completed orders.

`[tpl-tb-004] STATUS-GATED ACTION.` Before any order action, verify status via `get_order_details`. Cancel/modify require "pending"; return/exchange require "delivered". If status does not match, inform the customer of available actions.

`[tpl-tb-005] CONFIRMATION PROTOCOL.` Before any destructive action, present a summary of: order ID, specific items and changes, payment method for refund, expected outcome. Wait for explicit customer confirmation before executing.

C Causal Attribution Matrix

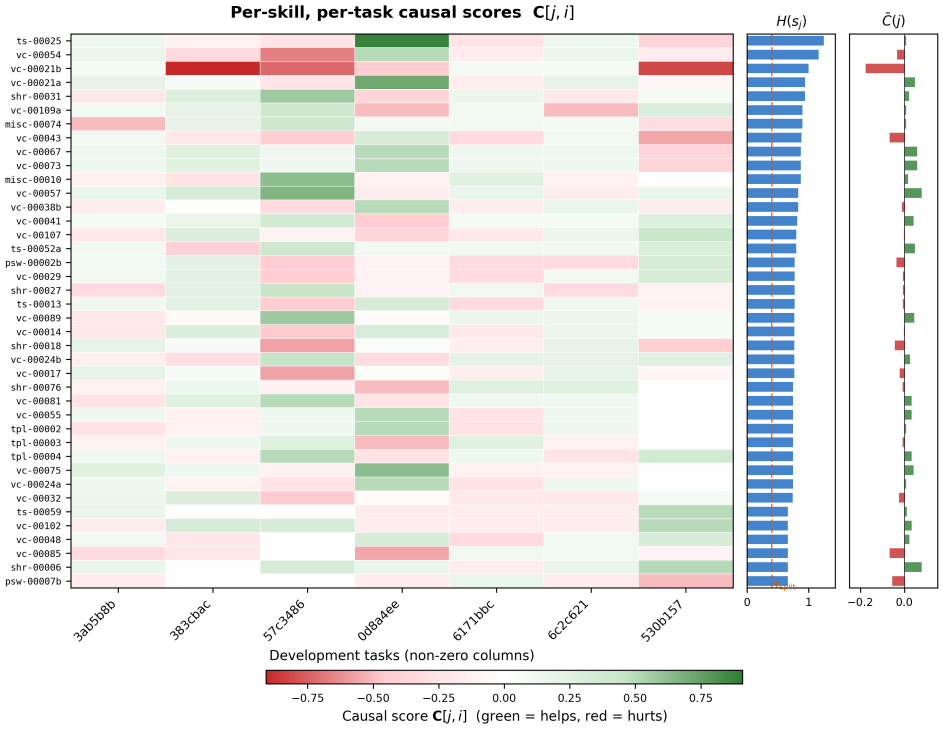

Figure 4: **Causal attribution matrix for GPT-5.1 on AppWorld** (top 40 skills by heterogeneity, 7 selected development tasks). Each cell shows the estimated causal score $\mathbf{C}[j, i]$: green indicates a positive estimate and red a negative estimate. *Left*: the matrix itself, sorted by heterogeneity $H(s_j)$ descending. The selected rows illustrate contrasting task-level estimates. *Centre*: the heterogeneity score $H(s_j)$; the dashed orange line marks the split threshold τ_{split}=0.40. *Right*: the global causal score $\bar{C}(j)$. A near-zero global score can conceal the variation in these task profiles.

Figure 4 visualises the causal attribution matrix for GPT-5.1 on AppWorld, restricted to the 40 skills with the largest estimated row ranges and 7 selected development tasks. Each cell encodes the difference-in-means causal score $\mathbf{C}[j, i]$ (Eq. 1): green indicates a positive estimate and red a negative estimate. The display selects high-range rows to show their estimated task profiles. It illustrates the information used for conditional rewriting and task-specific prediction (§A.1); finite-budget uncertainty is analyzed in Appendix I. The right-hand panels display the heterogeneity score $H(s_j)$ and the global causal score $\bar{C}(j)$; a near-zero mean can conceal substantial variation among the displayed task estimates.

D TASK-DEPENDENT SKILL PROFILES

The following GPT-5.1 / AppWorld profiles describe two skills in the upstream library.

Example 1: `vc-00054`. "Validate that note names map unambiguously to contacts before creating Venmo requests." Global causal score: $\bar{C} = -0.03$ (near zero). Displayed task-level estimates range from $+0.50$ in shared-expense reconciliation to -0.67 in single-app execution. These contexts illustrate the distinction between resolving name ambiguities and introducing additional contact cross-referencing. The near-zero mean compresses these contrasting task-level estimates.

Example 2: `vc-00021a`. "For repeated side-effecting API calls, capture and check each response." Global causal score: $\bar{C} = +0.05$ (near zero). Displayed task-level estimates range from $+0.71$ in side-effecting API use to -0.23 in read-only execution. Response checking can expose silent failures in the former context while adding steps in the latter. Both examples have a near-zero global estimate and a large estimated task range. Their profiles motivate retaining task-level information rather than relying on the row mean alone.

E REPRODUCIBILITY DETAILS

This section records the pipeline hyperparameters, model identifiers, and software versions. Table 3 lists every hyperparameter used across the pipeline; all values are fixed across every (model, benchmark) configuration with no per-cell tuning. Table 4 lists the recorded API identifiers, including hosted aliases.

Table 3: **Hyperparameters.** All values are fixed across every (model, benchmark) cell.

Stage	Parameter	Value
Upstream curation	Difficulty-estimation rollouts	2
	Task ordering	Hardest-first
	Development tasks M	15
	Random masks K	12
	Keep probability f	0.4
Offline restructuring	Split threshold τ_{split}	0.40
	Max split candidates	15
	Retire threshold τ_{retire}	0.10
	Merge similarity τ_{merge}	0.85
	Nearest neighbours k	8
	Softmax inverse temperature τ	5
	Mask threshold τ_{mask}	-0.10
	Min retained skills	30
Shared	Embedding model	Qwen3-Emb-0.6B

Model identifiers. All agent models are accessed via API; Table 4 lists the recorded identifiers, including hosted aliases. Both our DeepSeek ASSAY runs and ACE reproduction use DeepSeek-V3.2. The DeepSeek ReAct reference is reported separately; its exact snapshot is not specified by the available record.

Table 4: **Model API identifiers.**

Model	API identifier	Provider
GPT-5.1	`gpt-5.1-2025-11-13`	Azure OpenAI
GPT-5.4	`gpt-5.4-2026-03-05`	Azure OpenAI
GPT-4.1	`gpt-4.1-2025-04-14`	Azure OpenAI
GPT-4o	`gpt-4o-2024-11-20`†	Azure OpenAI
DeepSeek-V3.2	`deepseek-chat` (V3.2)	DeepSeek direct
Sonnet 4.5	`claude-sonnet-4-5-20250929`	Anthropic direct
Gemini 2.5 Pro	`gemini-2.5-pro`	Google AI Studio
Embedding	Qwen3-Embedding-0.6B	HuggingFace (local)
τ-bench user sim	`gpt-4o-2024-11-20`	Azure OpenAI

†Our agent and the Gupta et al. (2025) baseline use this version. The ReAct baseline (48.8%) is from the AppWorld leaderboard entry using `gpt-4o-2024-05-13`.

Benchmark versions. AppWorld: `appworld==0.1.4.dev0` (pip, aligned with the ACE codebase). τ-bench: commit `4754e6b` of `sierra-research/tau-bench`.

Decoding and randomization. All agent calls use temperature 0 and seed 100 where supported. Attribution masks are generated with Python's `random.Random(seed)` where seed = 42 for all (model, benchmark) cells except GPT-5.1 on AppWorld, which uses seed = 45. The independent GPT-5.1 / AppWorld replication uses mask seed 7. Attribution configuration is otherwise uniform: K=12 masks, M=15 development tasks, keep probability f=0.4 (Bernoulli).

Library sizes. After upstream curation (before offline restructuring), the skill library contains 103 skills for GPT-5.1, 87 for GPT-4.1, 126 for GPT-4o, 80 for DeepSeek-V3.2, 95 for GPT-5.4, 71 for Claude Sonnet 4.5, and 88 for Gemini 2.5 Pro on AppWorld. On τ-bench, library sizes are 76, 72, 61, 161, 81, 94, and 68 respectively. DeepSeek-V3.2's large τ-bench library (161 skills) reflects more aggressive skill generation during curation: DeepSeek-V3.2 proposed 115 ADD actions over 500 training tasks, compared to 15–30 for the other models under the same curation prompt. The attribution rollout count is fixed across library sizes; prompt length and execution cost can still vary. Variation arises because each model's curation run produces a different set of skills from the same training tasks. **Offline and deployment cost.** The 180 attribution rollouts cover the randomized masking experiment. Other offline work includes difficulty estimation (two executions per training task), upstream curation, rewrite generation, and development-gate evaluations. Deployment adds instruction embedding, neighbor lookup, and weighted score computation, without another test-task reward query.

F Sequential Ablation

Table 6 specifies the skill context in each configuration. Row (*b*) uses templates alone; row (*c*) adds both the learned ACE library and offline restructuring. Rows (*c*) and (*d*) share the same restructured library and differ only in deployment masking. The increments use the displayed percentages.

The complementary comparison in Table 5 evaluates ACE's learned library with the same five templates against full ASSAY.

Table 5: **Controls on GPT-5.1 / AppWorld *test_normal*.** Template content is matched between ACE + templates and ASSAY; fresh attribution reruns all 180 development executions and the downstream pipeline.

Configuration	TGC (%)	Successes / tasks
ACE	67.3	—
ACE + same five templates	70.8	—
ASSAY, original masks	77.4	130/168
ASSAY, fresh masks (seed 7)	76.8	129/168

Table 6: **Ablation configurations** (GPT-5.1, AppWorld *test_normal*, 168 tasks). All rows use ReAct. Rows (*b*)–(*c*) add the learned library and restructuring; rows (*c*)–(*d*) vary only per-task masking. Δ uses displayed percentages.

Configuration	Learned library	Templates	Offline restruct.	Task masking	TGC	Δ
(*a*) ReAct	None	No	No	No	61.9	—
(*b*) + templates	None	Yes	No	No	67.9	+6.0
(*c*) Restructured	ACE	Yes	Yes	No	69.6	+1.7
(*d*) Full ASSAY	Same as (*c*)	Yes	Yes	Yes	**77.4**	+7.8

F.1 MATCHED-REMOVAL SELECTION CONTROLS

The controls use GPT-5.1 / AppWorld *test_normal* and the same restructured library as full ASSAY. For each task, the removal count is matched to ASSAY. Random masking samples that many eligible skills uniformly; global-score masking removes that many eligible skills with the lowest row means $\bar{C}(j)$. Thus, global-score ranking is task-independent while the removal count remains task-matched. Table 7 reports all three random-seed outcomes and the two attribution-based selectors. Differences use displayed percentages.

Table 7: **Selection controls with matched per-task removal counts.** GPT-5.1 / AppWorld *test_normal*, 168 tasks. All configurations use the same restructured library. Random masking reports the mean across three seeds; individual results are shown.

Selection rule	TGC (%)	Individual runs / successes
Random masking	70.0	69.0 / 70.2 / 70.8
Global-score masking	73.2	123/168
ASSAY: task-conditioned masking	**77.4**	130/168

G OFFLINE RESTRUCTURING: SPLIT EXAMPLE

To illustrate what offline restructuring produces in practice, we show one of the 13 splits performed on the GPT-5.1 AppWorld library (bifurcation score $H = 1.56$).

Before (single skill). `[psw-00007]` "Many APIs return items in pages. Make sure to run through all the pages by looping over `page_index`."

This rule is essential for tasks that require scanning many items (e.g., browsing products, enumerating emails), but harmful for targeted purchases where the agent already knows which item it wants: exhaustive pagination wastes the step budget.

After (two conditional variants). `[psw-00007a]` *[IF task involves browsing or comparing multiple product options or scanning through multiple emails/messages]:* "Many APIs return items in pages. Make sure to run through all the pages by looping over `page_index`."

`[psw-00007b]` *[IF task is a targeted purchase of a specific known item or constrained to prior sellers]:* "Do NOT run through all pages or loop over `page_index`; instead, focus only on the specific item or on items from previously used sellers without exhaustive pagination."

The rewritten pair passes the development gate (no regression on any of the M=15 attribution tasks). Both the protected pagination template and the accepted conditional variants remain in the prompt, where the agent interprets their natural-language instructions.

G.1 Score Assignment for Split Variants

The randomized attribution experiment is run once per pipeline run. An accepted split produces a helping variant s_j^+ and a correcting variant s_j^-, with assigned rows

$$\mathbf{C}[j^+, i] = \max\{\mathbf{C}[j, i], 0\}, \qquad \mathbf{C}[j^-, i] = \max\{-\mathbf{C}[j, i], 0\}.$$

The helping variant inherits positive parent entries, and the correcting variant inherits the magnitudes of negative entries, with zeros elsewhere. The superscripts identify these parent-score subsets.

Accepted variants carry these assigned rows through restructuring and are protected from retirement and per-task masking; rejected splits retain the original skill and row. Application conditions are expressed in the instruction text. Merge follows the highest-scoring-member rule in §2.2.

Downstream row means retain the denominator M, including zero entries. The score array remains aligned with the surviving skills; the masking threshold is applied only to unprotected entries.

H Evaluation Metrics

AppWorld Task Goal Completion (TGC) measures the fraction of tasks passing all evaluation tests. Scenario Goal Completion (SGC) measures success on all task variants of a scenario, rather than partial completion of sub-goals (Trivedi et al., 2024). The main comparisons use TGC.

I Attribution Diagnostics and Replication

We evaluate the measurement-to-decision pipeline at three levels: sensitivity of task completion to fresh attribution, reproducibility of aggregated scores and selections, and uncertainty in individual estimates. The empirical diagnostics use GPT-5.1 / AppWorld *test_normal*. The default attribution protocol (K=12, M=15, f=0.4) applies to all seven models; the budget sweep below varies K.

I.1 Fresh-Mask Replication

The seed-7 experiment repeats all 12 masks, 180 development executions, and the downstream pipeline on GPT-5.1 / AppWorld. Original and fresh runs solve 130/168 and 129/168 test tasks. The paired counts are 123 successes in both runs, seven original-only successes, six fresh-only successes, and 32 failures in both: 13 outcomes change, with a net difference of one success. The exact two-sided McNemar test detects no difference ($p = 1.00$). This experiment evaluates sensitivity to mask regeneration with the model and development set held fixed.

Aligning the same upstream skills by skill ID across the two attributions, correlations are 0.34 for individual cells, 0.68 for global scores, and 0.61 for neighbor-smoothed predictions. The respective drop rates are 5.3% and 5.6%; overall keep/drop agreement is 91.3% and includes the predominant keep/keep pairs. Fallback activates on 21.4% and 19.6% of *test_normal* tasks. These statistics describe score and selection reproducibility within the same evaluation setting.

I.2 Development-Set Sensitivity

We independently sample three development sets, D1–D3, from the same pool of 57 development tasks using Python's `random.Random` with seeds 1, 2, and 3. Each set contains 15 distinct tasks; overlap between sets is allowed. All runs start from the same upstream library of 103 skills and repeat attribution, split/retire/merge, and per-task masking. The mask seed remains 45 and all hyperparameters are fixed. D0 is the original seed-45 run, reused as the reference rather than rerun. Each setting's random and global-score controls use its restructured library and match ASSAY's per-task removal counts as in Appendix F.1.

Table 8: **Development-set sensitivity** on GPT-5.1 / AppWorld *test_normal* (168 tasks). Random masking reports success counts for three seeds and their mean TGC; global-score and ASSAY entries report TGC and success counts. D0 achieves 77.4% (130/168).

Dev set	ASSAY	Global-score	Random successes	Random mean
D1	**76.2%** (128/168)	72.0% (121/168)	117 / 119 / 116	69.8%
D2	**75.6%** (127/168)	72.6% (122/168)	114 / 117 / 116	68.8%
D3	**78.0%** (131/168)	73.8% (124/168)	119 / 120 / 117	70.6%

ASSAY averages 76.6% across D1–D3 and solves 7, 5, and 7 more tasks than global-score masking under D1, D2, and D3 (Table 8). Against D0, the paired success counts (resampled-only, D0-only) are (5,7), (4,7), and (7,6), respectively.

Selection agreement. Comparisons align skill IDs in the intersection of the two restructured libraries; split variants are matched by parent skill and trigger condition. Keep/drop agreement uses final selections after fallback over all test tasks and these aligned skills. Removed-set Jaccard is computed per task on the aligned skills and averaged over tasks with at least one nonempty removal set. Tasks with two empty removal sets are excluded only from the Jaccard average. Table 9 reports both selection metrics and the number excluded. The task-completion gains persist across development sets even as selections among shared skills vary.

Table 9: **Selection agreement with D0 on shared skills.** Excluded tasks have two empty removal sets; the Jaccard mean uses the remaining tasks. Keep/drop agreement includes all 168 tasks.

Comparison	Keep/drop	Mean Jaccard	Tasks used	Tasks excluded
D1 vs D0	90.2%	0.44	136/168	32 (19.0%)
D2 vs D0	88.9%	0.40	132/168	36 (21.4%)
D3 vs D0	90.8%	0.47	139/168	29 (17.3%)

I.3 Attribution-Budget Sensitivity

With D0 and all other hyperparameters fixed, we independently generate mask sets of sizes $K = 24$ and $K = 48$ and repeat the full downstream pipeline. These sets are not nested extensions of the original $K = 12$ masks. Increasing attribution cost from 180 to 360 and 720 rollouts yields 130, 131, and 128 test successes, respectively (Table 10).

Table 10: **Attribution-budget sensitivity** on GPT-5.1 / AppWorld *test_normal*. Rollout counts cover attribution; keep/drop agreement is relative to $K = 12$ and uses the shared-skill alignment in Appendix I.2.

Masks K	Attribution rollouts	TGC (successes/tasks)	Keep/drop agreement
12	180	77.4% (130/168)	—
24	360	78.0% (131/168)	92.8%
48	720	76.2% (128/168)	94.6%

Between $K = 12$ and $K = 48$, correlations are 0.86 for global skill scores and 0.90 for neighbor-weighted predictions, computed on the same upstream skill IDs before restructuring. Across these budgets, task completion ranges from 128 to 131 successes, with no monotonic increase as K grows. These results support $K = 12$ as a practical operating point in this configuration.

I.4 Single-Cell Power and Range Diagnostics

Table 11 reports effect-detection power and null range exceedance under the simulation. At $K = 12$, a permutation test at $\alpha = 0.05$ detects a true ± 0.30 effect with 38.5% power; larger budgets increase

detection power. Null profiles also frequently exceed $H \geq 0.40$, so the range threshold serves as a permissive screen for rewrite candidates. Rewrite acceptance is determined by development execution, and deployment selection combines task estimates through Eq. 6.

Table 11: **Per-cell effect-detection power and null range diagnostics.** Power is computed for a true ± 0.30 effect via 1000-trial permutation simulation. Null exceedance is the fraction of simulated null profiles satisfying $H \geq 0.40$.

Masks K	**Per-cell** σ	**Power** ($\pm$**0.30**)	**Null exceedance rate**
12 (ours)	0.290	38.5%	$\sim$100%
20	0.225	59.5%	—
30	0.183	80.0%	—
50	0.142	98.0%	—

I.5 Bootstrap Score Summaries

To characterize uncertainty in the scores underlying masking decisions, we resample the $K{=}12$ attribution masks with replacement 1000 times and recompute the predicted scores. Table 12 summarises the results. For 97.2% of the 287 selected (skill, task) pairs on GPT-5.1 / AppWorld *test_normal*, the median predicted score across bootstrap resamples remains below the masking threshold -0.10. For fixed neighbor weights, prediction variance is $\mathbf{w}^\top \Sigma_j \mathbf{w}$, where Σ_j is the covariance of the task estimates in score row j. Shared masks and nonuniform weights affect how averaging changes uncertainty. At the stricter 95% CI level, 4.2% of the selected predictions have intervals wholly below the threshold. These criteria summarize score uncertainty conditional on the observed masks; neither percentage measures decision accuracy. Fresh-mask replication evaluates the resulting selections and task completion independently of this resampling.

Table 12: **Bootstrap summaries** (1000 resamples). For masking, the resampling criterion is a median below -0.10. The interval criterion requires the full 95% interval to fall below the masking threshold.

Decision type	**Total**	**Resampling criterion** (see caption)	**Interval criterion** (95% CI)
Mask ($\hat{C} < -0.10$)	287 cells	97.2%	4.2%

I.6 Reverse-Masking Protocol

The control uses GPT-5.1 / AppWorld *test_normal*, starting from the restructured library before per-task masking (Table 6, row *c*). Keeping the same protection set as standard masking, we suppress the five most positively scored unprotected skills. TGC falls from 69.6% to 66.7% (-2.9 points), as reported in the main ablation.

I.7 Development Set Coverage

Figure 5 shows the cumulative distribution of mean cosine similarity between each test task and its $k{=}8$ nearest development tasks. On *test_normal*, 78.6% of tasks have similarity ≥ 0.50; on *test_challenge*, 30.9% do. These distributions characterize the semantic neighborhoods used by the predictor on the two test splits.

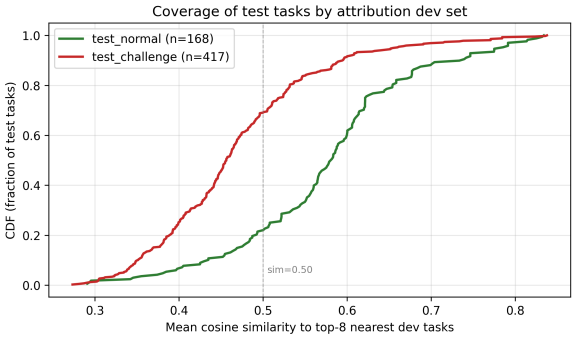

Figure 5: **Coverage of test tasks by the attribution development set.** CDF of mean cosine similarity to the top-8 nearest development tasks. The dashed line marks similarity = 0.50.