# ATTRIBUTION-GUIDED SKILL EVOLUTION FOR ONLINE CONTINUAL AGENT LEARNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

LLM agents increasingly face streams of tasks where experience from earlier tasks should be reused while adapting to new ones. We study this setting as online continual agent learning, where a shared external skill library is updated sequentially from completed tasks. Existing methods leverage agent experience to refine reusable skills through reflection, feedback, or reinforcement learning, but task-level outcomes provide limited insight into individual skill contributions. Failures may stem from missing activation, non-adherence, or incorrect application, while task success does not guarantee that every invoked skill is useful, making direct feedback-driven updates prone to inappropriate modifications. We propose AGSE (Attribution-Guided Skill Evolution), a plug-and-play framework that attributes skill–trajectory interactions along five dimensions: activation, adherence, correctness, coverage, and precision. These attributions provide structured, skill-specific evidence for guarded skill repair, expansion, and scope refinement, without model-weight updates or policy training. Across three heterogeneous CLI-agent frameworks, AGSE achieves exact accuracy of 72.9%, 59.4%, and 48.9% on OfficeQA Pro, and completion rates of 59.64%, 49.40%, and 48.19% on SkillFlow. In continual learning evaluation, AGSE achieves an FS of 67.7, FWT of +0.0227, and forgetting rate of 13.95%. Ablations demonstrate the importance of both multidimensional attribution and continual skill evolution.

## 1 INTRODUCTION

LLM agents increasingly operate in interactive environments such as terminals, browsers, and office applications, where tasks arrive as a continuous stream rather than as isolated prompts (Wang et al., 2025; Zheng et al., 2025). Beyond solving current tasks, agents must accumulate reusable experience and adapt it to future tasks. External skills provide a natural mechanism for such continual learning (Shinn et al., 2023). By encoding reusable workflows, tool-use strategies, and verification procedures in editable files, skills enable agents to improve without updating model parameters (Jiang et al., 2026). However, maintaining a shared skill library over a long task stream raises a fundamental challenge: *how can an agent learn from execution without corrupting previously useful experience?*

The key difficulty is that task-level feedback does not directly reveal how a skill should evolve (Zhang et al., 2026b; Yang et al., 2026b; Wu et al., 2026). A failed task may result from a relevant skill not being activated, an activated skill not being followed, incorrect instructions, missing procedural knowledge, or application beyond its intended scope (Ge et al., 2025). These failure modes require different responses, ranging from improving skill triggering and clarifying workflows to repairing content, expanding coverage, or narrowing applicability. Conversely, task success alone does not establish that a particular skill contributed to the outcome (Ju et al., 2026). Thus, treating execution feedback as an undirected rewriting signal can introduce unnecessary or harmful changes to a shared library (Liang et al., 2025). Effective online skill evolution therefore requires attributing execution outcomes to individual skills and determining which changes are justified by the available evidence.

Existing approaches to agent memory and skill evolution provide important foundations (Chhikara et al., 2025; Xu et al., 2025; Zhang et al., 2026a). They leverage agent experience to discover, refine, and optimize reusable skills through trajectory-based reflection, feedback-driven editing, or



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.

validation-based selection (Ni et al., 2026; Alzubi et al., 2026; Zhang et al., 2026a). However, explicitly attributing execution outcomes to individual skills and using this evidence to guide their evolution remains insufficiently addressed in our setting. Under a continuous task stream, a failure may reflect a skill's lack of activation, non-adherence, or incorrect application, while task success does not guarantee that every invoked skill was useful or sufficiently general. Without distinguishing these cases, directly translating execution outcomes into skill edits risks inappropriate modifications or discarding useful skills. This motivates a structured approach that attributes skill utility before determining how the shared library should evolve.

We propose **Attribution-Guided Skill Evolution (AGSE)**, an online framework that attributes skill–trajectory pairs along five dimensions: activation, adherence, correctness, coverage, and precision. AGSE maintains confidence-weighted, dimension-wise utility estimates and uses accumulated evidence to guide guarded lifecycle operations, including preservation, repair, coverage expansion, and scope narrowing or forking. By separating skill diagnosis from editing, AGSE turns execution feedback into targeted evolution decisions while avoiding updates unsupported by sufficient evidence, enabling continual skill-library maintenance without model-weight updates.

We evaluate AGSE on OfficeQA Pro (Opsahl-Ong et al., 2026) and SkillFlow (Zhang et al., 2026b) across three CLI-agent harnesses: Codex, Claude Code, and Qwen Code. Experiments examine task-solving performance, skill utilization, and continual evolution. AGSE improves performance across the evaluated settings, while ablations and further analyses investigate the contributions of attribution and shared skill library. These results highlight the value of diagnosing skill usage for stable skill accumulation in continual agent learning.

Our contributions are summarized as follows:

- **Problem formulation.** We study online continual skill-library maintenance, where agents improve a shared external skill library from sequential task experience while preserving useful knowledge.
- **Attribution-guided evolution.** We propose AGSE, which attributes skill–trajectory pairs along five dimensions, maintains persistent utility estimates, and translates diagnosed problems into guarded lifecycle operations.
- **Empirical evaluation.** We evaluate AGSE on OfficeQA Pro and SkillFlow across three CLI-agent harnesses. AGSE achieves 72.9% with Codex on OfficeQA Pro, with consistent improvements across evaluated settings. Ablations and further analyses investigate the contributions of attribution.

## 2 Related Work

### 2.1 Agent Memory and Skill Libraries

Agent memory methods (Packer et al., 2023; Zhong et al., 2024; Chhikara et al., 2025) enable persistent information storage and retrieval across interactions. Surveys further organize agent memory around reflection, experience abstraction, and continual adaptation (Luo et al., 2026; Zheng et al., 2026). While memory systems may capture procedural knowledge, their primary focus is persistent information or experience, which does not necessarily yield independently deployable skill artifacts. In contrast, skill libraries encode reusable procedures, tool-use strategies, and verification workflows (Zhou et al., 2026). AGSE focuses on the online construction and maintenance of such skills through execution feedback; general memory methods are complementary but not direct baselines for this setting.

### 2.2 Trajectory-Based Skill Evolution and Optimization

Recent methods leverage agent experience to discover and optimize reusable skills. SkillOpt refines external skill artifacts through trajectory-based reflection, bounded edits, and validation-based selection (Yang et al., 2026a). SkillRL combines hierarchical skill discovery with recursive reinforcement learning to co-evolve skills and agent policies (Xia et al., 2026), while Skill1 jointly optimizes skill selection, utilization, and distillation through reinforcement learning (Shi et al., 2026). These methods organize skill improvement around collected rollouts, optimization rounds, or training stages,

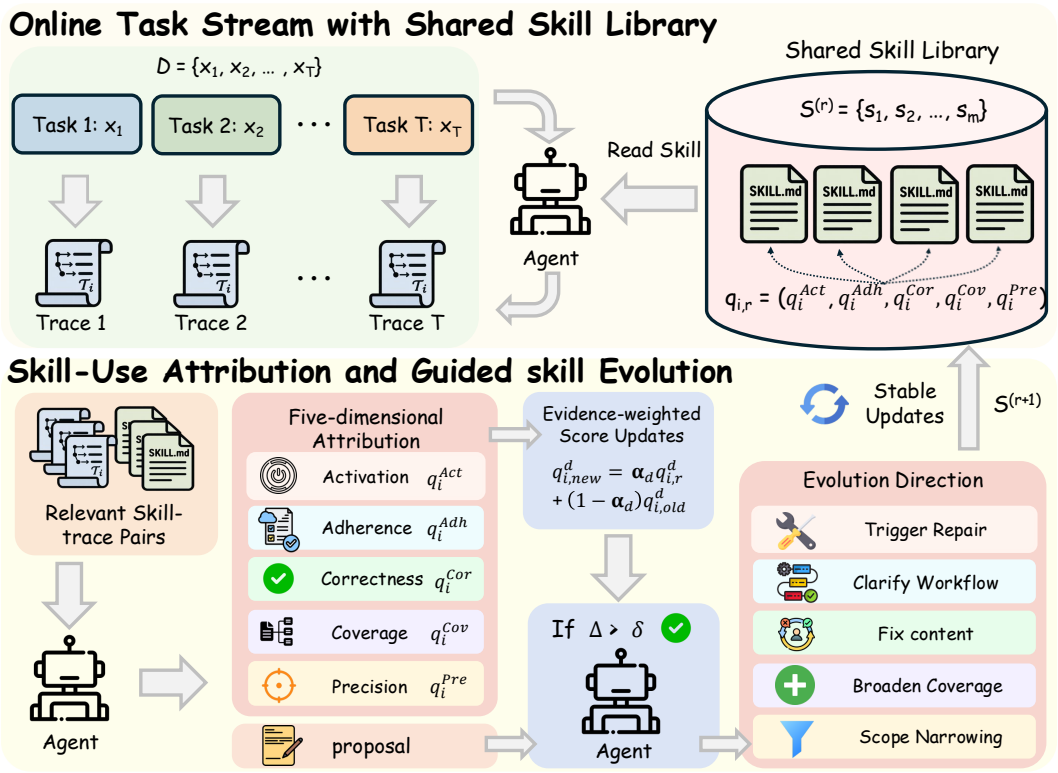


Figure 1: Overview of AGSE, which attributes skill–trajectory pairs, tracks skill utility, and guides guarded skill evolution.

whereas AGSE targets online deployment under continuous task streams. It attributes individual skill–trajectory pairs along five dimensions and uses accumulated utility estimates to guide guarded lifecycle operations over a shared skill library, enabling targeted skill maintenance from recent trajectories and verifier feedback without model-weight updates or policy training.

### 2.3 Skill Discovery, Creation, and Evaluation

Prior work explores automatic skill discovery, creation, and refinement. Voyager accumulates reusable executable skills through interaction (Wang et al., 2024), while SkillWeaver investigates autonomous skill discovery, refinement, and reuse (Zheng et al., 2025). Skill Creator provides an iterative skill-authoring and evaluation workflow, making it a method-level baseline for assessing the benefits of attribution-guided evolution over conventional skill creation and refinement (Anthropic, 2025). SkillFlow provides a benchmark for evaluating skill usage and lifelong task completion (Zhang et al., 2026b). We use it to assess whether online skill maintenance improves skill utilization and task completion in continual settings. Thus, Skill Creator serves as a skill-creation baseline, whereas SkillFlow provides the evaluation benchmark. Unlike these efforts, AGSE explicitly diagnoses individual skill contributions and uses accumulated evidence to guide preservation, repair, coverage expansion, and scope narrowing, enabling continual skill maintenance through attribution-guided lifecycle decisions.

## 3 AGSE: Attribution-Guided Skill Evolution

AGSE is an online skill-library maintenance framework for continual agent learning. Rather than rewriting skills directly from task failures or distilling them once from offline trajectories, AGSE separates skill diagnosis from skill editing. It attributes execution evidence to individual skills, maintains persistent utility scores, and uses score changes and attribution proposals to guide guarded updates to the shared skill library.

### 3.1 Problem Setup

We consider a sequential task stream $\mathcal{D} = \{x_1, \ldots, x_T\}$ processed by a CLI agent. At time $t$, the agent receives task $x_t$, executes it using the current skill library, and produces an execution record

$$\tau_t = (x_t, a_{1:k}, y_t, z_t),$$

where $a_{1:k}$ denotes the intermediate states and tool interactions, $y_t$ is the final answer or artifact, and $z_t$ is verifier feedback. The agent processes tasks sequentially, with the updated skill library from previous tasks available for subsequent execution.

At evolution round $r$, let

$$\mathcal{S}^{(r)} = \{s_1, \ldots, s_m\}$$

denote the shared skill library. Each skill is a reusable capability package centered on a `SKILL.md` file, potentially accompanied by scripts, references, and other assets. For each skill $s_i$, AGSE maintains a five-dimensional utility vector:

$$\mathbf{q}_i^{(r)} = \left(q_i^{\text{act}}, q_i^{\text{adh}}, q_i^{\text{cor}}, q_i^{\text{cov}}, q_i^{\text{pre}}\right) \in [0,1]^5,$$

representing activation, adherence, correctness, coverage, and precision. All dimensions are initialized to $1.0$ and updated only when valid attribution evidence is available. The utility vector thus records accumulated evidence about skill usage across the task stream, rather than a one-time evaluation of the skill.

### 3.2 Online Evolution Framework

At each evolution round, AGSE processes a recent window of completed tasks $\mathcal{W}_r = \{\tau_t\}_{t \in \mathcal{I}_r}$, where $\mathcal{I}_r$ denotes the task indices in the window. The evolution process consists of three stages: **(1) Skill-use attribution.** For each skill–trajectory pair, an LLM-based module determines relevance, assigns dimension-wise labels, and generates action-conditioned proposals. **(2) Utility update and trigger.** Attribution labels are aggregated to update each skill's utility vector, triggering evolution only when a dimension's degradation exceeds a predefined threshold. **(3) Guarded skill evolution.** For triggered skills, an evolution module applies constrained edits to the shared library based on the existing skill, utility changes, and attribution proposals, without access to the original trajectories.

This separation ensures that execution trajectories provide evidence for diagnosing skill problems, while only structured diagnostic results guide skill editing. The complete workflow is illustrated in Figure 1.

### 3.3 Skill-Use Attribution

**Pairwise attribution.** For every skill $s_i \in \mathcal{S}^{(r)}$ and trajectory $\tau_t \in \mathcal{W}_r$, AGSE invokes an LLM-based attribution module independently. The input includes the skill content, task and trajectory evidence, final answer, and verifier feedback. The module first determines whether the skill is relevant to the task and then produces dimension-wise labels and an edit proposal.

Specifically, the output for each pair is $(\rho_{i,t}, \mathbf{z}_{i,t}, p_{i,t})$, where $\rho_{i,t} \in \{0,1\}$ indicates whether the skill is relevant, and

$$\mathbf{z}_{i,t} = \left(z_{i,t}^{\text{act}}, z_{i,t}^{\text{adh}}, z_{i,t}^{\text{cor}}, z_{i,t}^{\text{cov}}, z_{i,t}^{\text{pre}}\right), \quad z_{i,t}^{d} \in \{0, 1, \text{null}\}.$$

When $\rho_{i,t} = 0$, all five labels are set to null and the proposal action is `none`. Otherwise, each dimension is evaluated independently: $1$ indicates satisfactory skill behavior, $0$ indicates evidence of a failure, and null indicates insufficient evidence for that particular dimension. This relevance decision is performed within the attribution call itself. The five dimensions capture distinct aspects of skill usage, as summarized in Table 1.

The dimensions are evaluated independently to avoid confounding task outcomes with skill quality. In particular, task failure alone does not imply incorrect skill content, and task success does not establish that a skill was activated or followed. A relevant skill that was not opened receives an activation failure, whereas correctness is penalized only when the agent substantially follows the

Table 1: Five dimensions of skill-use attribution and their corresponding evolution directions.

| Dimension | Attribution question | Diagnosis and evolution direction |
|---|---|---|
| Activation | Was the relevant skill discovered and opened? | Improve skill descriptions, names, or trigger phrases when relevant skills are not activated. |
| Adherence | Did the agent follow the skill's prescribed workflow? | Clarify steps, checklists, constraints, and validation requirements when the workflow is not followed. |
| Correctness | Was the skill's guidance reliable when properly followed? | Repair erroneous or misleading content when the evidence implicates the skill. |
| Coverage | Did the skill cover the reusable capability required? | Extend the skill or create a sibling skill to address missing capabilities. |
| Precision | Was the skill applied within its intended scope? | Narrow the skill's scope and exclude out-of-scope uses. |

skill and the evidence implicates its content. Coverage captures missing reusable capabilities, while precision captures out-of-scope application.

For each pair, the attribution module also generates a proposal $p_{i,t}$ that specifies one of seven constrained actions in $\mathcal{A}$ . **@improve_trigger** refines the conditions under which a skill is retrieved or activated. **@clarify_steps** clarifies the skill's workflow to improve adherence. **@fix_content** corrects inaccurate or misleading instructions. **@broaden_coverage** extends the skill to cover reusable missing patterns. **@narrow_precision** restricts the skill's scope to prevent inappropriate activation or application. **@create_new_skill** proposes a new skill when the observed capability is better captured by a separate skill. Finally, **none** indicates that no modification is warranted.

Each proposal includes the affected dimensions, a rationale, suggested changes, and supporting evidence. These proposals preserve the diagnostic information needed for editing without requiring the evolution module to reconsider the original trajectories. The complete attribution prompt, input and output schema, relevance criteria, and validation protocol are provided in Appendix A.

### 3.4 Evidence-Weighted Utility Updates

After attributing all skill–trajectory pairs in $\mathcal{W}_r$, AGSE aggregates valid labels for each skill and dimension. Let $\mathcal{V}_{i,r}^d$ denote the set of labels for skill $s_i$ and dimension $d$ that are either 0 or 1. The batch-level utility estimate is

$$\hat{q}_{i,r}^d = \frac{1}{|\mathcal{V}_{i,r}^d|} \sum_{v \in \mathcal{V}_{i,r}^d} v.$$

If no valid evidence is available for a dimension, its stored utility remains unchanged. Otherwise, the utility is updated using an evidence-weighted exponential moving average:

$$q_{i,\text{new}}^d = (1 - \alpha_{i,r}^d) q_{i,\text{old}}^d + \alpha_{i,r}^d \hat{q}_{i,r}^d,$$

where

$$\alpha_{i,r}^d = \alpha_{\text{base}} \frac{n_{i,r}^d}{n_{i,r}^d + k}, \qquad n_{i,r}^d = |\mathcal{V}_{i,r}^d|.$$

Here, $\alpha_{\text{base}}$ is the base update rate and $k$ controls smoothing. The evidence-dependent update rate limits the influence of sparse observations while allowing repeated failures to substantially change the utility estimate. Consequently, unrelated tasks and dimensions without valid evidence do not directly affect a skill's stored utility.

### 3.5 Guarded Lifecycle Evolution

After updating the utility vectors, AGSE determines whether each skill requires evolution. For skill $s_i$ and dimension $d$, define the utility degradation as $\Delta_i^d = q_{i,\text{old}}^d - q_{i,\text{new}}^d$. Evolution is triggered if

Table 2: Main results on OfficeQA Pro and SkillFlow across three CLI-agent harnesses with task-level percentile bootstrap 95% confidence intervals. OfficeQA Pro reports accuracy(%) at tolerance thresholds $\epsilon \in \{0, 0.001, 0.01, 0.05\}$; SkillFlow reports task completion rate(%Comp.) and skill usage rate(%Use), both in percentage points.

| Harness | Model | Method | OfficeQA Pro | | | | SkillFlow | |
|---|---|---|---|---|---|---|---|---|
| | | | $\epsilon = 0$ | $\epsilon = 0.001$ | $\epsilon = 0.01$ | $\epsilon = 0.05$ | %Comp. | %Use |
| Codex | GPT-5.5 | w/o skill | 59.4 [51.1, 67.7] | 65.4 [57.1, 72.9] | 70.7 [63.2, 78.2] | 75.2 [67.7, 82.0] | 53.01 [45.18, 60.24] | – |
| | | Skill-Creator | 59.4 [51.1, 67.7] | 64.7 [56.4, 72.9] | 69.9 [62.4, 77.4] | 73.7 [66.2, 81.2] | 54.21 [46.39, 62.05] | 83.73 [77.71, 89.16] |
| | | SkillFlow-Evo | 58.6 [50.4, 66.9] | 64.7 [56.4, 72.9] | 71.4 [63.9, 78.9] | 74.4 [66.9, 82.0] | 46.39 [38.55, 54.22] | 84.94 [79.52, 90.36] |
| | | AGSE | **72.9** [65.4, 80.5] | **78.2** [70.7, 85.0] | **83.5** [76.7, 89.5] | **85.7** [79.7, 91.7] | **59.64** [52.41, 66.87] | **85.54** [80.12, 90.36] |
| Claude Code | Claude Opus 4.6 | w/o skill | 49.6 [41.4, 57.9] | 55.6 [47.4, 63.9] | 64.7 [56.4, 72.9] | 69.9 [62.4, 77.4] | 47.59 [39.76, 55.42] | – |
| | | Skill-Creator | 46.2 [37.7, 54.6] | 52.3 [43.8, 60.8] | 63.1 [54.6, 71.5] | 66.9 [58.5, 74.6] | – | – |
| | | SkillFlow-Evo | 46.6 [38.3, 54.9] | 51.1 [42.9, 59.4] | 60.9 [52.6, 69.2] | 66.2 [57.9, 74.4] | 46.39 [38.55, 54.22] | 14.46 [9.64, 19.88] |
| | | AGSE | **59.4** [51.1, 67.7] | **65.4** [57.1, 73.7] | **72.9** [65.4, 80.5] | **76.7** [69.2, 83.5] | **49.40** [41.57, 57.23] | **26.50** [19.88, 33.13] |
| Qwen Code | Qwen3.6 Plus | w/o skill | 44.4 [36.1, 52.6] | 51.1 [42.9, 59.4] | 63.2 [54.9, 71.4] | **71.4** [63.9, 78.9] | 42.17 [34.94, 50.00] | – |
| | | Skill-Creator | 47.4 [39.1, 55.6] | 51.2 [42.9, 59.4] | 60.9 [52.6, 69.2] | 69.9 [61.7, 77.4] | 44.31 [36.75, 51.81] | 37.23 [30.12, 44.58] |
| | | SkillFlow-Evo | 42.9 [34.6, 51.1] | 47.4 [39.1, 55.6] | 57.1 [48.9, 65.4] | 63.9 [55.6, 72.2] | 43.37 [36.14, 51.20] | 35.54 [28.31, 42.77] |
| | | AGSE | **48.9** [40.6, 57.1] | **54.1** [45.9, 62.4] | **63.9** [55.6, 72.2] | 67.7 [59.4, 75.2] | **48.19** [40.36, 56.02] | **43.98** [36.75, 51.81] |

$\exists d : \Delta_i^d > \delta$, where $\delta$ is the evolution threshold; otherwise, the skill is preserved without modification.

For each triggered skill, the evolution module receives its current content, dimension-wise utility changes, and attribution proposals for degraded dimensions, but not the original task trajectories. These proposals provide a structured interface specifying the diagnosed failure, supporting evidence, and corresponding editing direction. The evolution module applies the relevant action from $\mathcal{A}$ to the skill library. Activation and precision guide trigger and scope adjustments; adherence guides workflow clarification; correctness guides content repair; and coverage guides capability expansion or sibling-skill creation. When multiple dimensions trigger evolution, their proposals are jointly considered to produce a coherent patch.

Content repair requires the skill to be relevant, substantially followed, and directly implicated in the failure. Otherwise, the module favors conservative, evidence-supported changes, such as workflow clarification, trigger refinement, or scope narrowing. Distinct capability gaps may instead lead to sibling-skill creation, avoiding unnecessary changes to existing skills. This guarded mechanism bases evolution on accumulated skill-specific evidence rather than task outcomes alone. The editing prompt, action implementation, and patch protocol are detailed in Appendix A, with a step-by-step example in Appendix A.6.

# 4 Experiments

## 4.1 Experimental Setup

**Benchmarks.** We evaluate AGSE on OfficeQA Pro (Opsahl-Ong et al., 2026) and SkillFlow (Zhang et al., 2026b). While SkillFlow natively supports lifelong learning, we treat the 133 OfficeQA Pro questions as an online task stream in their original order, exploiting potential knowledge transfer across financial documents, tables, and quantitative reasoning. See Appendix B.1 for benchmark details and task-stream construction.

**Baselines.** We compare against **w/o skill**, which runs the native agent without external skills; **SkillFlow-Evo**, which incrementally patches skills using trajectories and verifier feedback (Zhang et al., 2026b); and **Skill-Creator**, which iteratively revises skills using task feedback and with-/without-skill evaluations (Anthropic, 2025). Both evolving baselines follow the same causal block

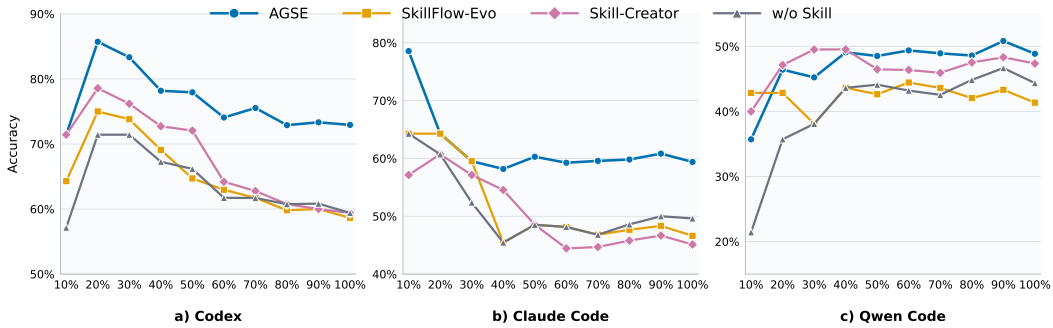


Figure 2: OfficeQA Pro exact accuracy ($\epsilon = 0$) throughout the task stream. While AGSE shows comparable performance to the baselines initially, its advantage becomes increasingly pronounced as continual learning progresses and persists throughout the later stages.

Table 3: Comparison of continual-learning performance on OfficeQA Pro ($\epsilon = 0$) using Codex with GPT-5.5. FS measures final-library performance on past tasks, FWT measures transfer to new tasks, and FR measures forgetting of previously solved tasks.

| Method | Acc. ($\epsilon = 0$)↑ | FS ↑ | FWT ↑ | FR ↓ |
|---|---|---|---|---|
| w/o Skill | 59.4 | – | – | – |
| Skill-Creator | 59.4 | 56.4 | -0.0076 | 15.79 |
| SkillFlow-Evo | 58.6 | 56.4 | -0.0076 | 18.80 |
| AGSE | **72.9** | **67.7** | **+0.0227** | **13.95** |

boundaries and information access as AGSE. Appendix B.3 provides further details on baseline configurations and implementation.

**Implementation Details.** All experiments use Harbor (Harbor Framework Team, 2026), with isolated task containers and task-specific verifiers. We evaluate three matched harnesses: Codex/GPT-5.5, Claude Code/Claude Opus 4.6, and Qwen Code/Qwen3.6 Plus. Within each harness, all methods share the same model, task order, tools, files, verifier, and execution limits, differing only in skill evolution. The skill library starts empty and is updated every $t = 4$ tasks, remaining frozen within each block. For AGSE, attribution scores start at 1.0 and use evidence-weighted EMA with $k = 1$ and $\alpha_{\text{base}} = 0.6$; edits are triggered when a score drop exceeds $\delta = 0.2$. All methods receive only completed-task trajectories and verifier feedback, with matched update frequency and evaluation budgets. Appendix B.2 provides complete implementation and hyperparameter details.

**Metrics.** For OfficeQA Pro, we report accuracy at relative-error tolerances $\epsilon \in \{0, 0.001, 0.01, 0.05\}$; for SkillFlow, we report task completion and skill usage rates. We additionally report final score (FS), forward transfer (FWT), and forgetting rate (FR), measuring final-library performance, transfer to new tasks, and loss of previously acquired capabilities, respectively. Appendix B.4 gives the precise metric definitions and computation procedures.

### 4.2 Main Results

Table 2 reports results on OfficeQA Pro and SkillFlow across three agent frameworks. On OfficeQA Pro, AGSE achieves the highest exact correctness ($\epsilon = 0$) across all harnesses, reaching 72.9%, 59.4%, and 48.9% with Codex, Claude Code, and Qwen Code, respectively, outperforming the w/o skill baseline by 13.5, 9.8, and 4.5 percentage points. These gains largely persist under relaxed error tolerances, demonstrating improved answer correctness and robustness to numerical deviations. On SkillFlow, AGSE achieves completion rates of 59.64%, 49.40%, and 48.19%, exceeding the w/o skill baseline by 6.63, 1.81, and 6.02 percentage points, respectively. It also attains the highest skill usage rates across all three frameworks: 85.54%, 26.50%, and 43.98%, compared with 84.94%, 14.46%, and 35.54% for SkillFlow-Evo. The substantial usage gains on Claude Code and Qwen

Table 4: Human–LLM agreement on the five attribution dimensions, measured by exact agreement and Cohen's $\kappa$ between one human annotator and the LLM over 50 skill–trajectory pairs from OfficeQA Pro using Codex with GPT-5.5.

| | Activation | Adherence | Correctness | Coverage | Precision |
|---|---|---|---|---|---|
| Exact agreement | 100.0% | 100.0% | 98.0% | 94.0% | 96.0% |
| Cohen' $\kappa$ | 1.000 | 1.000 | 0.964 | 0.909 | 0.934 |

Table 5: Ablation study on OfficeQA Pro using Codex with GPT-5.5. Structured attribution and continual evolution are both important for effective skill learning.

| Method | Acc. ($\epsilon = 0$) | $\Delta$ |
|---|---|---|
| w/o skill | 59.4 | -13.5 |
| w/o attribution | 62.4 | -10.5 |
| Single-Score Attribution | 62.1 | -10.8 |
| Static Skill Library | 57.9 | -15.0 |
| AGSE | **72.9** | – |

Code suggest that AGSE not only accumulates reusable procedures but also maintains skills that agents can effectively retrieve and apply.

Comparisons with existing skill-evolution methods further highlight the importance of the update strategy. Skill-Creator improves over the w/o skill baseline in some settings, but its gains are inconsistent and its exact correctness remains below AGSE across all three harnesses. SkillFlow-Evo performs even worse than the w/o skill baseline on OfficeQA Pro with Codex (58.6% vs. 59.4%) and Qwen Code (42.9% vs. 44.4%), as well as on SkillFlow completion with Codex (46.39% vs. 53.01%). These results indicate that directly revising skills from trajectories and verifier feedback does not reliably translate execution experience into useful skill improvements. In contrast, AGSE explicitly diagnoses skill–trajectory pairs and applies guarded, dimension-specific updates, helping preserve useful skills while targeting identified weaknesses.

The overhead analysis in Appendix C shows that these gains do not simply result from increased evolution computation. Skill-Creator consumes substantially more tokens and runtime than AGSE despite achieving lower exact correctness, whereas SkillFlow-Evo incurs less overhead but produces negative performance effects in several settings. Together, these comparisons suggest that effective skill evolution depends not merely on the amount of computation or feedback used, but on how execution evidence is attributed and translated into targeted skill updates. The consistent improvements across Codex, Claude Code, and Qwen Code further indicate that the benefits of AGSE generalize across different agent implementations.

### 4.3 Continual-Learning Performance

Figure 2 shows exact accuracy ($\epsilon = 0$) throughout the task stream. All baselines, including the w/o skill setting, exhibit similar fluctuations, reflecting variations in the difficulty of incoming tasks. While AGSE shows no pronounced advantage initially, it consistently outperforms the baselines as continual learning progresses, with the performance gap becoming more evident in later stages. This suggests that AGSE effectively accumulates and reuses task experience, translating online skill evolution into sustained improvements over baselines.

Table 3 further confirms this behavior. AGSE achieves an FS of 67.7, substantially higher than both Skill Creator and SkillFlow-Evo (56.4), indicating stronger final task-solving capabilities. It also achieves positive FWT (+0.0227), whereas both baselines show negative FWT (-0.0076), suggesting that the skills learned by AGSE transfer more effectively to newly arriving tasks. AGSE also achieves the lowest forgetting rate (13.95%), compared with 15.79% for Skill Creator and 18.80% for SkillFlow-Evo. This indicates that AGSE can better preserve previously acquired capabilities while adapting to new evidence. This behavior is consistent with its attribution-guided update mechanism, which uses accumulated evidence to trigger targeted evolution rather than rewriting skills whenever new trajectory feedback becomes available.

Table 6: Hyperparameter sensitivity on OfficeQA Pro exact accuracy ($\epsilon = 0$). * indicate the hyperparameter settings used in the main experiments.

| Parameter | Value | Codex / GPT-5.5 | | Qwen / Qwen3.6 Plus | |
|---|---|---|---|---|---|
| | | Acc. | $\Delta$ | Acc. | $\Delta$ |
| $t$ | 2 | 60.2 | $-12.7$ | 43.6 | $-5.3$ |
| | $4^*$ | **72.9** | **0.0** | **48.9** | **0.0** |
| | 8 | 60.9 | $-12.0$ | 44.4 | $-4.5$ |
| $\delta$ | 0.0 | 72.2 | $-0.7$ | 47.4 | $-1.5$ |
| | $0.2^*$ | **72.9** | **0.0** | **48.9** | **0.0** |
| | 0.4 | 72.2 | $-0.7$ | 46.6 | $-2.3$ |
| $\alpha_{\text{base}}$ | 0.4 | 71.4 | $-1.5$ | 45.9 | $-3.0$ |
| | $0.6^*$ | **72.9** | **0.0** | **48.9** | **0.0** |
| | 0.8 | 70.7 | $-2.2$ | 46.6 | $-2.3$ |

### 4.4 Ablation Study

Table 5 evaluates the contributions of attribution and continual skill evolution on OfficeQA. Removing attribution reduces exact correctness from 72.9% to 62.4%, approaching the no-skill baseline (59.4%). This shows that maintaining a skill library alone is insufficient; effective evolution requires understanding how skills interact with task trajectories. Replacing five-dimensional attribution with *Single-Score Attribution* further reduces correctness to 62.1%, nearly matching the w/o attribution variant. This highlights the importance of structured attribution: a single quality score cannot distinguish failure modes or guide targeted repairs, coverage expansion, and scope refinement. Finally, *Static Skill Library*, which freezes the library after the first task, achieves only 57.9%, below the no-skill baseline. This demonstrates that early experience alone cannot cover subsequent tasks and that fixed skills may become outdated or overly narrow. In contrast, AGSE continuously incorporates new evidence and selectively evolves skills as task requirements change.

### 4.5 Manual Validation of Multidimensional Attribution

We manually validate the five-dimensional attribution using a stratified sample of 50 skill–trajectory pairs from the main experiment. The human annotator independently label the same five dimensions as the LLM. As shown in Table 4, human–LLM agreement is 100.0% for activation and adherence, 98.0% for correctness, 96.0% for precision, and 94.0% for coverage. Overall, the high agreement across all dimensions supports the reliability of LLM-based attribution for guiding skill evolution. Detailed sampling and annotation procedures are provided in Appendix D.

### 4.6 Hyperparameter Sensitivity

Table 6 shows that performance is most sensitive to $t$, which controls the skill evolution frequency. Both harnesses achieve their highest accuracy at $t = 4$, while more frequent ($t = 2$) and less frequent ($t = 8$) evolution lead to lower accuracy. This pattern suggests a trade-off between timely skill adaptation and accumulating sufficient feedback before modifying the shared skill library.

## 5 Conclusion

We presented AGSE, an attribution-guided framework for continual skill evolution in LLM agents. By diagnosing skill–trajectory pairs across five dimensions and applying guarded, targeted updates, AGSE avoids undirected skill rewriting while enabling skills to adapt to new experience. Across two benchmarks and three heterogeneous agent–model frameworks, AGSE consistently improves task performance and skill reuse, while achieving stronger final performance, forward transfer, and retention in continual learning. These results highlight that effective continual agent learning requires not only accumulating experience, but also systematically determining how that experience should evolve reusable skills.

## Reproducibility Statement

All details necessary to reproduce our experiments are provided in the experimental setup section of the main paper and Appendix B, including benchmark and task-stream construction, agent frameworks and models, evaluation environments, baseline configurations, skill evolution protocols, hyperparameters, evaluation metrics, and implementation details. We also report computational overhead and hyperparameter sensitivity to facilitate reproducibility.

## AI Use Statement

Generative AI tools were used to assist with language editing and manuscript polishing. All experimental designs, results, analyses, and conclusions were reviewed and verified by the authors.

## References

Salaheddin Alzubi, Noah Provenzano, Christian Alexander Calvo, Jaydon Bingham, Weiyuan Chen, Himanshu Tyagi, Dastin Huang, and Tu Vu. Evoskill: Automated skill discovery for agents. In *First Workshop on Agent Skills*, 2026. URL https://openreview.net/forum?id=i4b4As7RFX.

Anthropic. Skill-Creator: Official anthropic agent skill for authoring skills. https://github.com/anthropics/skills/tree/main/skills/skill-creator, 2025.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory, 2025. URL https://arxiv.org/abs/2504.19413.

Yubin Ge, Salvatore Romeo, Jason Cai, Monica Sunkara, and Yi Zhang. Samule: Self-learning agents enhanced by multi-level reflection, 2025. URL https://arxiv.org/abs/2509.20562.

Harbor Framework Team. Harbor: A framework for evaluating and optimizing agents and models in container environments. https://github.com/harbor-framework/harbor, January 2026.

Guanyu Jiang, Zhaochen Su, Xiaoye Qu, and Yi R. Fung. XSkill: Continual learning from experience and skills in multimodal agents. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=AjP1yvCyoG.

Ruofei Ju, Xinrui Wang, Xin Ding, Yifan Yang, Hao Wu, Shiqi Jiang, Qianxi Zhang, Hao Wen, Xiangyu Li, Weijun Wang, Kun Li, Yunxin Liu, Haipeng Dai, Wei Wang, and Ting Cao. Embodiskill: Skill-aware reflection for self-evolving embodied agents, 2026. URL https://arxiv.org/abs/2605.10332.

Xuechen Liang, Yangfan He, Yinghui Xia, Xinyuan Song, Jianhui Wang, Meiling Tao, Li Sun, Xinhang Yuan, Jiayi Su, Keqin Li, Jiaqi Chen, Jinsong Yang, Siyuan Chen, and Tianyu Shi. Self-evolving agents with reflective and memory-augmented abilities, 2025. URL https://arxiv.org/abs/2409.00872.

Jinghao Luo, Yuchen Tian, Chuxue Cao, Ziyang Luo, Hongzhan Lin, Kaixin Li, Chuyi Kong, Ruichao Yang, and Jing Ma. From storage to experience: A survey on the evolution of LLM agent memory mechanisms. In *ICLR 2026 Workshop on Memory for LLM-Based Agentic Systems*, 2026. URL https://openreview.net/forum?id=l9Ly41xxPb.

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

Krista Opsahl-Ong, Arnav Singhvi, Jasmine Collins, Ivan Zhou, Cindy Wang, Ashutosh Baheti, Owen Oertell, Jacob Portes, Sam Havens, Erich Elsen, Michael Bendersky, Matei Zaharia, and Xing Chen. Officeqa pro: An enterprise benchmark for end-to-end grounded reasoning, 2026. URL https://arxiv.org/abs/2603.08655.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. Memgpt: Towards llms as operating systems. *CoRR*, abs/2310.08560, 2023. URL https://doi.org/10.48550/arXiv.2310.08560.

Yaorui Shi, Yuxin Chen, Zhengxi Lu, Yuchun Miao, Shugui Liu, Qi GU, Xunliang Cai, Xiang Wang, and An Zhang. Skill1: Unified evolution of skill-augmented agents via reinforcement learning, 2026. URL https://arxiv.org/abs/2605.06130.

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

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.

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, 2026. URL https://arxiv.org/abs/2510.16079.

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. In *ICLR 2026 Workshop on Lifelong Agents: Learning, Aligning, Evolving*, 2026. URL https://openreview.net/forum?id=FYc2IygegR.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents, 2025. URL https://arxiv.org/abs/2502.12110.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, Kai Qiu, Yuqing Yang, Dongdong Chen, Xue Yang, and Chong Luo. Skillopt: Executive strategy for self-evolving agent skills, 2026a. URL https://arxiv.org/abs/2605.23904.

Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, Bo Zhang, and Liang He. Autoskill: Experience-driven lifelong learning via skill self-evolution. In *COLM 2026 The 2nd Workshop on Lifelong Agents: Learning, Aligning, and Evolving*, 2026b. URL https://openreview.net/forum?id=qveCbSk6vt.

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, 2026a. URL https://arxiv.org/abs/2604.01687.

Ziao Zhang, Kou Shi, Shiting Huang, Avery Nie, Yu Zeng, Yiming Zhao, Zhen Fang, Qishen Su, Haibo Qiu, Wei Yang, Qingnan Ren, Shun Zou, Wenxuan Huang, Lin Chen, Zehui Chen, and Feng Zhao. Skillflow:benchmarking lifelong skill discovery and evolution for autonomous agents, 2026b. URL https://arxiv.org/abs/2604.17308.

Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, and Yu Su. Skillweaver: Web agents can self-improve by discovering and honing skills, 2025. URL https://arxiv.org/abs/2504.07079.

Junhao Zheng, Chengming Shi, Xidi Cai, Qiuke Li, Duzhen Zhang, Chenxing Li, Dong Yu, and Qianli Ma. Lifelong learning of large language model based agents: A roadmap. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(5):5552–5571, 2026. doi: 10.1109/TPAMI.2025.3650546.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: enhancing large language models with long-term memory. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i17.29946. URL https://doi.org/10.1609/aaai.v38i17.29946.

Yingli Zhou, Wang Shu, Yaodong Su, Wenchuan Du, Yixiang Fang, and Xuemin Lin. A comprehensive survey on agent skills: Taxonomy, techniques, and applications, 2026. URL https://arxiv.org/abs/2605.07358.

# A ATTRIBUTION AND EVOLUTION PROTOCOL

This appendix specifies the interfaces between execution, attribution, and evolution. The examples use a JSON-like notation for readability; the host validates the same fields as strict JSON at runtime. Long reasoning traces are not passed to the attribution model. The compact input retains task context, bounded ATIF step messages, tool calls, observations, verifier signals, and the final output; skill-use evidence is inferred from these records.

## A.1 COMPACT TRAJECTORY SCHEMA

The execution artifact follows the Agent Trajectory Interface Format (ATIF). We do not replace this format with a separate event log. Instead, the attribution input is a compact projection of the ATIF record: the session envelope and ordered step structure are preserved, while long messages, tool arguments, and tool outputs are bounded before they are passed to the attribution model. Skill-use evidence is therefore read from the messages, tool calls, and observations inside ATIF steps. A representative record is shown below.

**ATIF-v1.5 Trajectory Schema**

```
{
  "schema_version": "ATIF-v1.5",
  "session_id": "<session UUID>",
  "agent": {
    "name": "<agent harness>",
    "version": "<harness version>",
    "model_name": "<model identifier>",
    "extra": {"<runtime field>": "<value>"}
  },
  "steps": [
    {
      "step_id": 1,
      "timestamp": "<ISO-8601 timestamp>",
      "source": "system",
      "message": "<system message>"
    },
    {
      "step_id": 2,
      "timestamp": "<ISO-8601 timestamp>",
      "source": "user",
      "message": "<task message>"
    },
    {
      "step_id": 3,
      "timestamp": "<ISO-8601 timestamp>",
      "source": "agent",
      "model_name": "<model identifier>",
      "message": "<agent message>",
      "tool_calls": [
        {
          "tool_call_id": "<call identifier>",
          "function_name": "<tool name>",
          "arguments": {"<argument>": "<value>"}
        }
      ],
      "observation": {
        "results": [
          {
            "source_call_id": "<call identifier>",
            "content": "<bounded tool output>"
          }
        ]
      },
      "extra": {"<runtime field>": "<value>"}
    }
  ],
  "final_metrics": {
    "total_prompt_tokens": 0,
    "total_completion_tokens": 0,
    "total_cached_tokens": 0,
    "total_steps": 0,
    "extra": {"<metric>": "<value>"}
  }
}
```

In the stored artifact, the *source* field takes one of three values: *system*, *user*, or *agent*. Agent steps may additionally contain *tool_calls*, *observation*, and runtime *extra* fields. The task identifier used by attribution is carried by the surrounding task record and proposal record, rather than inserted as a non-ATIF top-level field. This organization lets the attribution model distinguish opening or following a skill from the final task outcome without pretending that a verifier failure is automatically a content failure.

### A.2 SKILL METADATA SCHEMA

The metadata schema of skill record each skill's name and file path, along with its persistent five-dimensional utility scores. Its structure is:

**Skill Library State Schema**

```
{
  "root": "<shared skill root>",
  "current_dir": "<current skill directory>",
  "snapshots_dir": "<skill snapshot directory>",
  "scores_path": "<skill score file>",
  "skill_ids": [
    "<skill id>",
    "<skill id>"
  ],
  "scores": {
    "<skill id>": {
      "activation": 1.0,
      "adherence": 1.0,
      "correctness": 1.0,
      "coverage": 1.0,
      "precision": 1.0
    }
  },
  "dimensions": [
    "activation",
    "adherence",
    "correctness",
    "coverage",
    "precision"
  ]
}
```

The *scores* object is the persistent utility state used by the online update rule in Section 3.4. The current skill files and their snapshots are addressed through the directory fields.

### A.3 PROMPT TEMPLATES

Full prompt templates are listed below:

**Attribution System Prompt**

```
You are a skill attribution judge for a shared skill library with multidimensional scores.
Evaluate exactly one skill against exactly one task trajectory.
The skill is scored with a five-dimensional vector and a compact modification proposal.
Use the skill content, the task trajectory, the verifier result, the final answer, and the
failed tests together. Treat the five dimensions independently and tie the proposal to the
dimension(s) that changed.
The vector dimensions and their meanings are defined below.
""" + ATTRIBUTION_DIMENSION_GUIDE + """
""" + ATTRIBUTION_PROPOSAL_GUIDE + """
Output requirements:
- Return exactly one JSON object.
- Use `related` as a boolean.
- Use `score_vector` as an object with exactly these keys: `activation`, `adherence`,
`correctness`, `coverage`, `precision`.
- Use `confidence` as a number between 0 and 1.
- Use `reasoning` as one short sentence explaining the judgement.
- Use `proposal` as the compact modification proposal described above.
- If the skill is unrelated, set `related=false`, every value in `score_vector` to `null`,
      and
`proposal.action` to `none`.
- If the skill is relevant but not opened, set `activation=0`.
```

```
- If the skill was opened but not followed, set `adherence=0`.
- If the skill content is wrong or misleading, set `correctness=0`.
- If the skill misses a reusable pattern, set `coverage=0`.
- If the skill is over-broad or misapplied, set `precision=0`.
- Make the proposal specific to the changed dimensions: do not write generic advice that
could apply to any skill.
Be concise, evidence-based, and schema-faithful.
```

**Attribution Dimension Guide**

```
## Five-dimensional score vector
Evaluate one skill against one task execution and output a 5D vector with values from `{1,
     0, null}` only.
Use the current skill content, the current trajectory, and the completion result as
     evidence. Do not collapse
the five dimensions into a single global score.
| Dimension | `1` means | `0` means | `null` means | If `0`, prefer this revision |
|---|---|---|---|---|
| `activation` | The agent actually opened and used the skill. | The task was relevant,
     but the agent did not
discover or open the skill. | The skill is unrelated, or the trace does not support a
     judgement. | Improve
`name`, `description`, and trigger phrases so the skill is easier to discover. |
| `adherence` | After opening the skill, the agent followed the core workflow. | The agent
      opened the skill
but clearly did not follow its steps or checks. | The skill is unrelated, or the trace
     does not support a
judgement. | Strengthen the step order, checklist, prohibitions, and validation steps. |
| `correctness` | The skill content is accurate and leads to the right solution path. |
     The skill content is
misleading, incomplete, or wrong in a way that would cause failure if followed. | The
     skill is unrelated, or the trace does not support a
judgement. | Fix incorrect instructions, formulas, table choices, computation
rules, or other wrong content. |
| `coverage` | The skill covers the reusable pattern needed by the task. | The skill
     misses a key reusable pattern needed by the task. | The skill is unrelated, or the
     trace does not support a judgement. | Broaden the
capability or split off a sibling skill that covers the missing pattern. |
| `precision` | The skill is appropriately scoped and not over-applied. | The skill is too
      broad and was used
on a task it should not cover. | The skill is unrelated, or the trace does not support a
     judgement. | Narrow the
trigger conditions and add non-applicable scenarios or exclusions. |
Decision rules:
- Use `related=false` only when the skill is clearly irrelevant to the task or cannot be
     justified from the trace.
- If the skill is relevant but was not opened, set `activation=0`.
- If the skill was opened but not followed, set `adherence=0`.
- If the skill itself is wrong or misleading, set `correctness=0`.
- If the skill misses an important reusable pattern, set `coverage=0`.
- If the skill was triggered on an inappropriate task, set `precision=0`.
- When multiple dimensions are affected, keep each dimension independent; do not force all
      values to the
same number.
- Prefer verifier evidence, failed tests, tool output, and the final answer over self-
     report.
- If the task is related but the skill was not surfaced, that is `activation=0`, not `
     related=false`.
- If the task is related and the skill was surfaced but the workflow was skipped, that is
     `adherence=0`.
```

**Attribution Proposal Guide**

```
## Modification proposal rules
Along with the score vector, output one compact proposal that tells the next evolution
     step.
Allowed `action` values:
- `none`
- `improve_trigger`
- `clarify_steps`
- `fix_content`
- `broaden_coverage`
- `narrow_precision`
- `create_new_skill`
Use the score vector to choose the proposal:
```

```
- If `activation=0`, the proposal should mostly improve the skill's trigger language, name
    , or description.
- If `adherence=0`, the proposal should make the workflow harder to skip: add steps,
    checklists, prohibitions, and
verification.
- If `correctness=0`, the proposal should correct the broken content or remove misleading
    instructions.
- If `coverage=0`, the proposal should broaden the capability or recommend creating a
    sibling skill when the
missing pattern is distinct.
- If `precision=0`, the proposal should narrow the trigger conditions and add clear non-
    applicability boundaries.
- If every dimension is `1` or `null`, keep `action=none` and explain that the skill is
    already well-shaped for the
task.
Dimension-to-edit mapping:
- `activation` -> edit the skill name, description, and trigger phrases so the skill is
    easier to discover in future tasks.
- `adherence` -> make the workflow harder to skip by adding explicit steps, checklists,
    prohibitions, and validation.
- `correctness` -> fix wrong or misleading content in the skill body, scripts, examples,
    or references.
- `coverage` -> broaden the reusable pattern or split off a sibling skill when the missing
     pattern is clearly distinct.
- `precision` -> narrow the trigger conditions and add exclusions or non-applicability
    cases.
Required proposal fields:
- `action`: one of the values above
- `dimensions`: list of affected dimensions, for example `["activation"]` or `["adherence
    ", "correctness"]`
- `cause`: one short sentence explaining why the change is needed
- `suggested_changes`: 1-4 concise bullets or strings describing reusable edits
- `evidence_snippets`: 1-3 short snippets from the trajectory or verifier that justify the
     proposal
- `confidence`: number from 0 to 1 indicating how strong the proposal is
Keep the proposal short, actionable, and directly tied to the observed dimension values.
```

**Attribution User Prompt**

```
# Skill Attribution Task
You are evaluating one skill against one task execution.
## Skill Under Evaluation
### Skill ID: `{skill_id}`
### Full Content
## Trial Summary
- Task UID: {task_uid}
- Task source: {task_source}
- Verifier passed: {verifier_passed}
- Reward: {reward}
- Exception: {exception_info}
- Final answer: {final_answer}
- Failed tests: {failed_tests}
## Compact trajectory
{trajectory_block}
## Evidence reading order
- Use the trial summary first: verifier result, reward, exception, failed tests, and final
     answer are the ground truth.
- Use the compact trajectory next to confirm whether the skill was discovered, opened,
    followed, or ignored.
- Judge each dimension independently: a skill can be relevant but unopened, opened but
    ignored, or correct but incomplete.
## Required output
Return exactly one JSON object with keys:
- `related`
- `score_vector`
- `confidence`
- `reasoning`
- `proposal`
Use 1/0/null values in `score_vector`. If you prefer to emit `labels` as a synonym, the
    parser will still accept it, but `score_vector` is the
```

**Skill Evolution System Prompt**

```
You are a skill evolution assistant that revises a shared skill library using
    multidimensional attribution feedback and compact proposals.
```

```
A skill is a reusable capability package, not a task-specific note. Improve the library so
      a future agent can solve similar tasks faster, more reliably, and with
fewer failed attempts.
Each skill maintains a five-dimensional EMA score vector in `[0, 1]`, initialized to `1.0`
      in every dimension.
The evolution input is compressed on purpose: use the score change summary and the
     proposal records as the primary evidence, and only add new files when the
capability gap is genuinely distinct.
Hard structural rules (treat these as requirements, not suggestions):
- A new skill should normally live in its own capability-named directory, for example `
     fill-pdf-forms/` or `api-debugging/`.
- Every skill directory must contain `SKILL.md` at its root.
- `SKILL.md` must begin at the first character of the file with YAML frontmatter in
     exactly this shape:
---
name: <skill-name>
description: <what the skill does and when to use it>
---
- The frontmatter may contain exactly two keys: `name` and `description`. Do not add any
     other metadata keys.
- After the closing `---`, write normal Markdown instructions. Prefer starting the body
     with `# <Readable Title>`.
- Put executable helpers only in `scripts/`.
- Put long-form documentation, schemas, API notes, and detailed examples only in `
     references/`.
- Put copyable templates or non-context assets only in `assets/`.
- Do not create empty placeholder files or directories.
- Do not create README, CHANGELOG, INSTALLATION_GUIDE, QUICK_REFERENCE, or any process
     notes.
- If you add files under `scripts/` or `references/`, `SKILL.md` must explicitly tell a
     future agent when to run or read them.
- Keep references one hop away from `SKILL.md`; avoid deep navigation or nested
     indirection.
- If an existing skill already covers the capability, update that skill instead of
     creating a parallel duplicate.
Design principles:
- Be concise. Add only information that is non-obvious, reusable, and worth the context
     cost.
- Generalize from the trace, but do not merely restate it. Infer the reusable workflow,
     decision points, validation steps, tool patterns, anti-patterns, and
troubleshooting notes that should change a future agent's behavior.
- Prefer verifier evidence, failed tests, and concrete execution results over the agent's
     self-report when they conflict.
- Prefer minimal edits to the existing library over broad rewrites. Fix what's broken,
     preserve what works.
- Use progressive disclosure. Keep `SKILL.md` focused and short; move detailed material
     into `references/` or `scripts/` only when it improves reuse.
- Avoid duplication across `SKILL.md` and `references/`.
- The `description` field must explain both what the skill does and when to use it,
     including trigger contexts.
- The `SKILL.md` body should be imperative, operational, and easy to scan. Prefer
     workflows, decision rules, and concise examples over long prose.
- Keep `SKILL.md` under 500 lines when possible.
Score-drop to edit mapping:
- `activation` dropped: edit only the skill's name, description, and trigger phrases; do
     not rewrite the workflow, add helper files, delete files, or create sibling
skills when the only issue is discoverability.
- `adherence` dropped: make the workflow harder to skip with explicit steps, checklists,
     prohibitions, and validation.
- `correctness` dropped: fix wrong instructions, misleading claims, invalid formulas,
     broken sequencing, or incorrect tooling assumptions only when the
evidence says the agent followed the skill and the verifier failure is directly
     attributable to the skill content.
- `coverage` dropped: broaden the current skill for a reusable missing pattern; create a
     sibling skill only when repeated recent proposals show the same distinct
missing pattern.
- `precision` dropped: edit only trigger boundaries, exclusions, and non-applicability
     cases; do not broaden the workflow or add sibling skills when the issue is
over-triggering.
Guarded action constraints:
- If the triggered dimensions are only `activation` and/or `precision`, the patch may only
      update the target skill's `SKILL.md`; return no deletes, no
scripts/references/assets, and no sibling skills.
- A new sibling skill is allowed only when `coverage` is a triggered dimension and the
     user prompt says `coverage_new_skill_allowed=true`.
- Core workflow/content corrections are allowed only when `
     correctness_core_update_supported=true`; otherwise treat correctness-related evidence
      as
insufficient and prefer an empty patch or a non-core trigger/validation edit.
```

```
- If these constraints conflict with a proposal, obey the constraints and return a
    conservative patch or an empty patch.
If multiple dimensions dropped, combine related edits, but keep the patch small and
    capability-focused.
Use the proposal records as the primary evidence. Do not require raw trajectories unless
    the proposals are too thin to support a confident fix; in that case, prefer
conservative edits that improve triggers or validation first.
If no meaningful reusable change is justified, return an empty patch.
Output requirements:
- Return exactly one JSON object with keys: `summary`, `upsert_files`, `delete_paths`.
- `upsert_files` must map relative file paths to full file contents.
- `delete_paths` must only include paths that should truly be removed as obsolete.
- Do not wrap the JSON in commentary.
```

**Skill Evolution User Prompt**

```
You are a skill evolution assistant that revises a shared skill library using
    multidimensional attribution feedback and compact proposals.
A skill is a reusable capability package, not a task-specific note. Improve the library so
     a future agent can solve similar tasks faster, more reliably, and with
fewer failed attempts.
Each skill maintains a five-dimensional EMA score vector in `[0, 1]`, initialized to `1.0`
     in every dimension.
The evolution input is compressed on purpose: use the score change summary and the
    proposal records as the primary evidence, and only add new files when the
capability gap is genuinely distinct.
Hard structural rules (treat these as requirements, not suggestions):
- A new skill should normally live in its own capability-named directory, for example `
    fill-pdf-forms/` or `api-debugging/`.
- Every skill directory must contain `SKILL.md` at its root.
- `SKILL.md` must begin at the first character of the file with YAML frontmatter in
    exactly this shape:
---
name: <skill-name>
description: <what the skill does and when to use it>
---
- The frontmatter may contain exactly two keys: `name` and `description`. Do not add any
    other metadata keys.
- After the closing `---`, write normal Markdown instructions. Prefer starting the body
    with `# <Readable Title>`.
- Put executable helpers only in `scripts/`.
- Put long-form documentation, schemas, API notes, and detailed examples only in `
    references/`.
- Put copyable templates or non-context assets only in `assets/`.
- Do not create empty placeholder files or directories.
- Do not create README, CHANGELOG, INSTALLATION_GUIDE, QUICK_REFERENCE, or any process
    notes.
- If you add files under `scripts/` or `references/`, `SKILL.md` must explicitly tell a
    future agent when to run or read them.
- Keep references one hop away from `SKILL.md`; avoid deep navigation or nested
    indirection.
- If an existing skill already covers the capability, update that skill instead of
    creating a parallel duplicate.
Design principles:
- Be concise. Add only information that is non-obvious, reusable, and worth the context
    cost.
- Generalize from the trace, but do not merely restate it. Infer the reusable workflow,
    decision points, validation steps, tool patterns, anti-patterns, and
troubleshooting notes that should change a future agent's behavior.
- Prefer verifier evidence, failed tests, and concrete execution results over the agent's
    self-report when they conflict.
- Prefer minimal edits to the existing library over broad rewrites. Fix what's broken,
    preserve what works.
- Use progressive disclosure. Keep `SKILL.md` focused and short; move detailed material
    into `references/` or `scripts/` only when it improves reuse.
- Avoid duplication across `SKILL.md` and `references/`.
- The `description` field must explain both what the skill does and when to use it,
    including trigger contexts.
- The `SKILL.md` body should be imperative, operational, and easy to scan. Prefer
    workflows, decision rules, and concise examples over long prose.
- Keep `SKILL.md` under 500 lines when possible.
Score-drop to edit mapping:
- `activation` dropped: edit only the skill's name, description, and trigger phrases; do
    not rewrite the workflow, add helper files, delete files, or create sibling
skills when the only issue is discoverability.
- `adherence` dropped: make the workflow harder to skip with explicit steps, checklists,
    prohibitions, and validation.
```

```
- `correctness` dropped: fix wrong instructions, misleading claims, invalid formulas,
    broken sequencing, or incorrect tooling assumptions only when the
evidence says the agent followed the skill and the verifier failure is directly
    attributable to the skill content.
- `coverage` dropped: broaden the current skill for a reusable missing pattern; create a
    sibling skill only when repeated recent proposals show the same distinct
missing pattern.
- `precision` dropped: edit only trigger boundaries, exclusions, and non-applicability
    cases; do not broaden the workflow or add sibling skills when the issue is
over-triggering.
Guarded action constraints:
- If the triggered dimensions are only `activation` and/or `precision`, the patch may only
     update the target skill's `SKILL.md`; return no deletes, no
scripts/references/assets, and no sibling skills.
- A new sibling skill is allowed only when `coverage` is a triggered dimension and the
    user prompt says `coverage_new_skill_allowed=true`.
- Core workflow/content corrections are allowed only when `
    correctness_core_update_supported=true`; otherwise treat correctness-related evidence
     as
insufficient and prefer an empty patch or a non-core trigger/validation edit.
- If these constraints conflict with a proposal, obey the constraints and return a
    conservative patch or an empty patch.
If multiple dimensions dropped, combine related edits, but keep the patch small and
    capability-focused.
Use the proposal records as the primary evidence. Do not require raw trajectories unless
    the proposals are too thin to support a confident fix; in that case, prefer
conservative edits that improve triggers or validation first.
If no meaningful reusable change is justified, return an empty patch.
Output requirements:
- Return exactly one JSON object with keys: `summary`, `upsert_files`, `delete_paths`.
- `upsert_files` must map relative file paths to full file contents.
- `delete_paths` must only include paths that should truly be removed as obsolete.
- Do not wrap the JSON in commentary.
```

### A.4 Relatedness and NULL rules

The attribution call and the post-processing use the following rules.

- *related* = *false* means that the task does not require the capability described by the skill, or that the available record cannot establish a relationship. All dimensions are then *null*, and no score is updated.
- A related skill that was not opened receives *activation* = *0*; it must not be discarded as unrelated merely because the agent failed to retrieve it.
- An opened skill that was not followed receives *adherence* = *0*. This does not imply *correctness* = *0*.
- *correctness* = *0* requires direct evidence that the skill content was wrong or misleading after substantial use. A failed verifier with no such evidence leaves correctness *null*.
- *coverage* = *0* means that the task reveals a reusable missing pattern. A one-off missing fact is not a coverage failure.
- *precision* = *0* means that the skill was over-triggered or applied outside its intended scope. If the task is simply unrelated and the skill was not invoked, use *related* = *false* instead.
- A dimension is *null* when it is related but the trace does not contain enough evidence for that dimension. Null values are excluded from the EMA denominator.

### A.5 Proposal JSON schema

The attribution call produces one proposal for a skill–trajectory pair. At an evolution boundary, these records are collected into the proposals passed to the editor. The action and dimension fields specify the intended edit direction.

**Skill Evolution Proposal**

```
{
  "proposals": [
    {
      "task_uid": "UID0010",
```

```
      "action": "broaden_coverage",
      "dimensions": ["coverage"],
      "cause": "The task required combining a Treasury balance with a date-specific
    Macrotrends currency conversion, a pattern not covered by the current workflow.",
      "suggested_changes": [
        "Add a currency-conversion branch for external exchange-rate sources, including
    exact-date lookup and quote-direction verification.",
        "Require recording the source rate, its units, andwhether it is USD per JPY or JPY
     per USD beforemultiplication.",
        "Add a cross-check step for blocked or ambiguous web sources before accepting an
    approximate rate."
      ],
      "evidence_snippets": [
        "Final answer: 936113303200",
        "Ground truth: 935851121560",
        "The agent calculated using rate=149.96 after
        Macrotrends requests returned HTTP/2 403"
      ],
      "confidence": 0.95,
      "score_vector": {
        "activation": 1,
        "adherence": 1,
        "correctness": 1,
        "coverage": 0,
        "precision": 1
      }
    }
  ]
}
```


The allowed actions are **none**, **@improve_trigger**, **@clarify_steps**, **@fix_content**, **@broaden_coverage**, **@narrow_precision**, and **@create_new_skill**. The stored round record additionally keeps the round time, round index, skill identifier, and triggered dimensions, but these bookkeeping fields are outside the proposal object itself. For a no-op proposal, *action* is **none**, *dimensions* and *suggested_changes* are empty, and the evidence and confidence fields remain available for audit:

**Skill Attribution Record**


```
{
  "skill_id": "<skill id>",
  "task_uid": "UID0013",
  "related": true,
  "score_vector": {
    "activation": 1,
    "adherence": 1,
    "correctness": 1,
    "coverage": 1,
    "precision": 1
  },
  "confidence": 0.98,
  "proposal": {
    "action": "none",
    "dimensions": [],
    "cause": "The skill supported a correct verified answer.",
    "suggested_changes": [],
    "evidence_snippets": [
      "The agent followed the skill workflow and produced
      the verified result."
    ],
    "confidence": 0.98
  },
  "labels": {
    "activation": 1,
    "adherence": 1,
    "correctness": 1,
    "coverage": 1,
    "precision": 1
  },
  "reasoning": "The skill was appropriately activated
  and followed.",
  "evidence": "The execution and verifier evidence support
  retaining the skill.",
  "summary": null
}
```

Table 7: Pair-level attribution in the actual round-53 update for skill *treasury-historical-expenditure-reconciliation*. A zero of score is diagnostic rather than a direct task reward: it identifies the dimension that should guide the next edit.

| Task | Execution evidence | Score vector | Attribution and proposal |
|---|---|---|---|
| UID0097 | ESF balance-sheet comparison; final answer correct, [8.124,12.852], but the agent explicitly excluded the skill as outside its stated domain. | $(1, 0, 1, 0, 1)$ | The workflow is relevant but not clearly scoped to ESF snapshots. Propose **@broaden_coverage** on adherence and coverage: add ESF triggers, exact-date balance-sheet steps, and the existing revision/unit checks. |
| UID0100 | Judicial Branch FY2007–2013 YoY growth plus indexed log-OLS; verified answer [2.810,0.030,8.706]. | $(1, 1, 1, 0, 1)$ | The general extraction workflow was followed and sound, but it lacks a reusable indexed-OLS procedure. Propose **@broaden_coverage**: define the index, log transform, slope/intercept, rounding, and validation checks. |
| UID0101 | Department of Labor FY2011–2019 CAGR, decay factor, and midpoint arc elasticity; prediction [-0.150,0.850,-288.719] versus reference [-0.153,0.847,-1.162]. | $(1, 1, 1, 0, 1)$ | The failure is attributed to a missing reusable derived-rate convention, not to incorrect source-reconciliation content. Propose **@broaden_coverage**: add CAGR, decay-factor, and midpoint-elasticity formulas and endpoint documentation. |
| UID0102 | FY2021 Corporate income-tax H-spread; prediction $57.53$ versus reference $57.50$ after Type-7 interpolation. | $(1, 1, 0, 0, 1)$ | The skill's full-precision instruction conflicts with the task's required intermediate rounding. Propose **@fix_content** on correctness and coverage: specify staged quartile rounding and Type-7 H-spread handling. |

### A.6 Case study

The following case study is reconstructed from the actual artifacts of evolution round $53$. It illustrates the complete path from execution evidence to attribution, score update, and skill revision. The skill under evaluation is *treasury-historical-expenditure-reconciliation*; its version is 7 before the round and $8$ after the round. The current skill is not evaluated once in isolation: the host pairs the same version-7 skill with every trajectory in the recent batch

$$\mathcal{W}_{53} = \{\tau_{0097}, \tau_{0100}, \tau_{0101}, \tau_{0102}\}.$$

For each pair, the attribution module receives the skill content, the ATIF-derived trajectory, the verifier result, and the final answer. The four pair-level records are then aggregated before the editor is called.

**Step 1: read the execution evidence.** UID0097 asks for an Exchange Stabilization Fund balance-sheet comparison and is answered correctly as [8.124,12.852], but the agent explicitly treats the skill as outside its specialized workflows. UID0100 asks for Judicial Branch fiscal-year outlay growth and an indexed log-OLS regression and returns the verified answer [2.810,0.030,8.706] after following the general workflow. UID0101 asks for a Department of Labor outlay CAGR, decay factor, and midpoint arc elasticity; the agent follows the skill but returns [-0.150,0.850,-288.719] instead of the reference [-0.153,0.847,-1.162]. Finally, UID0102 asks for a Type-7 H-spread with an explicitly rounded intermediate stage; the agent returns 57.53 rather than 57.50.

**Step 2: attribute each skill–trajectory pair.** The resulting vectors in Table 8 use the order (*activation*,*adherence*,*correctness*,*coverage*,*precision*). The two confidence values in the last column are the confidence of the complete attribution record and the nested proposal, respectively. This table also shows why the vector is more informative than the final verifier result. A correct answer can still expose a coverage or adherence problem: UID0097 is correct but the skill did not guide the execution, while UID0100 is correct but reveals an OLS coverage gap. Conversely, an incorrect

Table 8: Utility update for skill *treasury-historical-expenditure-reconciliation*.

| Dimension | Old Score | Batch Score | New Score | New–old |
|---|---|---|---|---|
| Activation | 0.9551 | 1.0000 | 0.9767 | +0.0215 |
| Adherence | 0.7410 | 0.7500 | 0.7453 | +0.0043 |
| Correctness | 0.9777 | 0.7500 | 0.8684 | −0.1093 |
| Coverage | 0.8447 | 0.0000 | 0.4392 | −0.4054 |
| Precision | 0.9486 | 1.0000 | 0.9732 | +0.0247 |

answer need not imply that the skill content is wrong: in `UID0101`, the judge keeps correctness at 1 and assigns the failure to missing coverage. Only `UID0102` receives correctness 0, because the existing full-precision instruction directly conflicts with the requested staged-rounding convention.

The evidence is specific enough to explain these labels. For `UID0101`, the recorded reasoning identifies the erroneous arc elasticity as a raw midpoint outlay change divided by a raw midpoint year change, whereas the reusable convention requires midpoint percentage changes for both quantities. For `UID0102`, the agent obtains Type-7 quartiles $Q_1 = 7.850$ and $Q_3 = 65.375$ and subtracts them at full precision, producing $57.525 \to 57.53$; the proposal asks the evolved skill to make the required intermediate-rounding stage explicit instead of silently applying the generic full-precision rule.

The four proposals preserve these distinctions. In particular, the editor is not asked to rewrite the skill from four raw trajectories. It receives a finite set of directed records: an ESF extension for `UID0097`, an indexed-regression extension for `UID0100`, a derived-rate extension for `UID0101`, and a staged-rounding correction for `UID0102`. The proposal actions are therefore heterogeneous even though all four pairs have *coverage=0*.

**Step 3: aggregate the labels and update utility.** All four pairs are related and provide valid labels in every dimension. Therefore, the batch estimates are

$$\hat{\mathbf{q}}_{53} = (1,\ 0.75,\ 0.75,\ 0,\ 1).$$

With $n_d = 4$, $k = 1$, and $\alpha_{\text{base}} = 0.6$, every dimension uses

$$\alpha_d = 0.6\frac{4}{4+1} = 0.48, \qquad \mathbf{q}_{\text{new}} = 0.52\mathbf{q}_{\text{old}} + 0.48\hat{\mathbf{q}}_{53}.$$

The actual score transition is shown below; the final column reports $q_{\text{new}} - q_{\text{old}}$.

The lifecycle gate uses $\Delta^d = q^d_{\text{old}} - q^d_{\text{new}}$ and $\delta = 0.2$. Coverage therefore has $\Delta^{\text{cov}} = 0.4054 > 0.2$ and triggers evolution, whereas the correctness decrease is only 0.1093 and does not independently open the core-content gate. The round summary consequently records *trigger_dimensions=[coverage]*. This is an important consequence of separating attribution from editing: the failed H-spread contributes a correctness signal, but it does not by itself authorize an unconstrained rewrite of the skill.

**Step 4: evolve the skill.** The evolution module receives the version-7 skill, the score-change summary, and the four proposals. It returns one in-place update for *treasury-historical-expenditure-reconciliation/SKILL.md*. The resulting version-8 file integrates all four diagnosed directions while preserving the original extraction and revision-reconciliation workflow. The following semantic diff excerpt summarizes the substantive v7–v8 changes; unchanged lines are omitted and long additions are lightly line-wrapped:

**Skill Evolution Patch (v0007 → v0008)**

```
--- v0007/treasury-historical-expenditure-reconciliation/SKILL.md
+++ v0008/treasury-historical-expenditure-reconciliation/SKILL.md
@@ scope and examples @@
- expenditure and economic series; aggregation or transformation
+ expenditure, economic-series, and ESF balance-sheet data;
+ CAGR, decay factor, elasticity, regression, aggregation, or nonlinear transformation
@@ workflow @@
+ Apply exact-date ESF balance-sheet reconciliation and
```

```
+ distinguish total capital from total liabilities and capital.
+ Use explicit CAGR and decay-factor conventions over elapsed
+ fiscal-year intervals.
+ Use midpoint percentage changes for arc elasticity.
+ Use the supplied fiscal-year index, positive log inputs,
+ and an explicit observation count for indexed log-OLS.
+ Treat an explicitly requested intermediate rounding stage as
+ part of the computation, including Type-7 H-spread quartiles.
@@ decision rules and anti-patterns @@
+ Add ESF, CAGR, arc-elasticity, OLS, and H-spread decision.
+ Do not calculate CAGR from the number of observations.
+ Do not call a raw midpoint change divided by a raw year change an arc elasticity.
+ Do not run an indexed log regression with an undocumented
+  index origin, nonpositive outlays, or an incorrect observation count.
+ Do not ignore an explicit instruction to round quartiles before computing an H-spread.
```

Concretely, the new skill description now advertises ESF balance-sheet comparisons and the derived-statistics patterns exposed by the batch. Its workflow adds an ESF exact-date step, makes CAGR, decay-factor, midpoint-elasticity, and indexed-OLS conventions explicit, and distinguishes full-precision computation from prompts that require intermediate rounding. The decision rules and anti-patterns repeat these constraints at the point of use, so the next agent is less likely to repeat the four observed failure modes. The round therefore demonstrates the intended mechanism: attribution does not merely label the batch as good or bad; it maps distinct execution evidence to distinct edits, aggregates the resulting utility state, and produces a concrete, inspectable skill revision.

# B Experiments Details

## B.1 Benchmark Details

We provide additional details of the benchmarks and the construction of our online task streams.

**OfficeQA Pro.** OfficeQA Pro (Opsahl-Ong et al., 2026) contains 133 questions grounded in a large corpus of U.S. Treasury Bulletins published from 1939 onward. The corpus spans approximately 89,000 pages and contains more than 26 million numerical values, covering diverse information including prose, tables, figures, and charts. The questions require agents to retrieve relevant evidence, perform multi-step quantitative reasoning, and produce precisely verifiable answers.

Although OfficeQA Pro is originally designed as a static document QA benchmark rather than a continual learning benchmark, its task distribution is well suited to studying online skill evolution. The questions are centered on a relatively coherent domain of financial documents and repeatedly require related capabilities, such as document inspection, evidence retrieval, table understanding, numerical calculation, and answer verification. Consequently, procedures and problem-solving experience acquired from earlier questions can potentially be reused on subsequent questions.

To construct an online setting, we preserve the original question order and process the 133 questions sequentially. After each task, the agent's skill library is updated based on the trajectory and feedback from that task, and the resulting library is retained for subsequent tasks. Thus, later questions can benefit from skills discovered during earlier interactions. We use the full 133-question Pro split throughout our experiments and refer to it as OfficeQA Pro.

**SkillFlow.** SkillFlow (Zhang et al., 2026b) is designed specifically for lifelong skill discovery and reuse. It contains 166 runnable tasks organized into 20 task families spanning Finance & Economics, Operations & Supply Chain, Healthcare & Life Sciences, Governance & Strategy, and Data & Document Intelligence. Each family contains 8–9 tasks arranged in a fixed order.

Tasks within each family share a Domain-Agnostic Execution Flow (DAEF), while varying domain-specific entities, files, or values. This structure creates repeated opportunities for agents to identify reusable workflows from earlier tasks and apply them to later tasks. Each task runs in an isolated environment and is evaluated using a family-specific verifier.

We follow SkillFlow's native task ordering and preserve the evolving skill library across tasks. This setting directly evaluates whether an agent can discover useful skills from earlier trajectories, reuse them on subsequent tasks, and revise the skills when later task feedback exposes their limitations.

### B.2 Implementation Details

**Evaluation framework and agent configurations.** All benchmark task executions use the Harbor evaluation framework (Harbor Framework Team, 2026). Harbor runs each task in an isolated container and records the agent trajectory, final output or artifact, verifier result, and execution metadata. This provides a consistent execution and evaluation protocol across methods. We use three matched CLI-agent configurations: Codex with GPT-5.5, Claude Code with Claude Opus 4.6, and Qwen Code with Qwen3.6 Plus. Claude Code additionally provides native support for external skills (Anthropic, 2025). Within each harness, all methods use the same underlying model, task order, tool permissions, initial task files, verifier, and execution limits. The only difference is the strategy used to create or update the external skill library.

**Online evolution protocol.** The skill library is empty before the first task, and the first task is therefore executed without an external skill. Its completed trajectory and verifier feedback are then used to bootstrap the initial library. After this cold-start step, tasks are processed in ordered blocks of $t = 4$, and skill evolution is invoked after each completed block. For OfficeQA Pro, all methods follow the fixed question order described in Section B.1; for SkillFlow, we preserve the benchmark's within-family task order.

The skill library is frozen throughout each block. Thus, every task within a block observes exactly the same skill snapshot, and an updated library becomes available only at the next block boundary. This prevents an evolution operation from changing the skill version midway through a block and ensures that methods can only learn from completed tasks. At every update boundary, methods can access only the tasks, trajectories, and verifier feedback available in the preceding online history; future tasks and their hidden answers are never exposed.

**AGSE configuration.** For AGSE, the attribution model evaluates each relevant skill–trajectory pair using the current skill, a compacted trajectory, and verifier feedback. It produces the five dimension-wise attribution labels together with a compact evolution proposal. The host aggregates these labels within the current update block and updates the corresponding utility scores.

All five utility dimensions are initialized to $1.0$. We use the evidence-weighted EMA described in Section 3.4 with $k = 1$ and $\alpha_{\text{base}} = 0.6$. An evolution edit is triggered only when the corresponding score drop exceeds the default threshold $\delta = 0.2$; otherwise, the current skill is retained. The evolution module therefore operates on aggregated score changes and attribution proposals rather than directly rewriting the complete set of raw trajectories, following the guarded lifecycle policy in Section 3.5.

**Call budget and update structure.** Operationally, the host makes one attribution call for each relevant skill–trajectory pair in the current update window and at most one evolution call for each skill whose aggregated score crosses the evolution gate within a block. Null or unrelated skill–trajectory pairs neither update utility scores nor invoke the editor. This call structure is fixed across runs. We use matched-budget accounting to distinguish the contribution of the attribution-and-evolution interface from improvements caused simply by additional model calls.

**Baseline update protocol.** For both evolving baselines, we use the same $t = 4$ block boundaries, causal information restriction, and Harbor-based benchmark evaluation as AGSE. At each update boundary, a baseline can access only completed tasks, trajectories, and verifier feedback from the online history. This ensures that differences arise from the evolution strategy rather than from different update frequencies or information access.

The internal A/B evaluations required by Skill-Creator are the only exception. Following its prescribed workflow, these evaluations are executed as local subprocess evaluations within the skill-authoring round. They are used only to guide skill authoring and do not replace the Harbor-based benchmark evaluation.

### B.3 Baseline Details

**w/o skill.** The agent runs with its native harness and underlying model but without an external skill library. It neither reads stored skills nor creates or updates skills during the task stream. This baseline

measures the performance obtained from the underlying agent alone and provides the reference point for evaluating the benefit of non-parametric procedural memory.

**SkillFlow-Evo.** The SkillFlow-Evo baseline follows the trajectory- and verifier-rubric-driven skill-patch procedure from the SkillFlow benchmark (Zhang et al., 2026b). After each online update boundary, a skill-patch generator receives the current skill library together with trajectories and verifier-derived feedback from the just-completed task block. It then produces an incremental patch that is applied to the library for the next block.

For a controlled comparison, we run SkillFlow-Evo updates at the same $t = 4$ block boundaries as AGSE and expose it to the same completed-task information.

**Skill-Creator.** Skill Creator is an agentized implementation of Anthropic's official **skill-creator** workflow (Anthropic, 2025), following the protocol specified in our baseline plan. Each update round contains one main Creator/Improver session, which is the only component permitted to write the candidate skill, together with narrow helper sessions for evaluation design, grading, and analysis.

The main Creator/Improver session drafts or revises the skill using the just-completed task block and performs one internal improvement iteration based on local with-skill versus without-skill evaluations.

### B.4 Evaluation Metrics

We use benchmark-specific metrics to measure task-level performance and continual-learning metrics to characterize transfer and forgetting throughout the online task stream.

**OfficeQA Pro accuracy.** For a numerical question with ground-truth answer $y$ and prediction $\hat{y}$, OfficeQA Pro defines the absolute relative error (ARE) as

$$\mathrm{ARE}(\hat{y}, y) = \left| \frac{\hat{y} - y}{y} \right|.$$

A numerical prediction is considered correct at tolerance $\epsilon$ if

$$\mathrm{ARE}(\hat{y}, y) \leq \epsilon.$$

We report $\mathrm{accuracy@}\epsilon$ for $\epsilon \in 0, 0.001, 0.01, 0.05$, corresponding to allowable relative errors of 0%, 0.1%, 1%, and 5%, respectively. For non-numerical and multi-part answers, we follow the normalization and matching rules provided by OfficeQA Pro. Higher accuracy indicates better task performance.

**SkillFlow completion and usage.** For SkillFlow, completion rate is the fraction of tasks whose final output satisfies the task-specific verifier. Skill usage rate is the fraction of tasks whose execution trace contains at least one read or call to a previously stored skill. Completion measures task-level utility, while usage measures whether the agent actually invokes the external procedural memory.

**Continual-learning metrics.** We report final score (FS), forward transfer (FWT), and forgetting rate (FR) to characterize the quality, transferability, and stability of the evolving skill library.

Let $b_i$ denote the score obtained by the no-skill agent on task $i$, and $s_i$ denote the score when task $i$ is first encountered in the online stream. The final score is

$$\mathrm{FS} = \frac{1}{N} \sum_{i=1}^{N} f_i,$$

where $f_i$ is the score obtained by the final skill-library checkpoint when task $i$ is evaluated.

Forward transfer measures whether previously acquired skills benefit newly arriving tasks:

Table 9: Computational overhead on OfficeQA Pro using the Codex harness with GPT-5.5. Tokens / Skill denotes the average token consumption per skill evolution. Evo. Time / Round is the average wall-clock time (in seconds) per evolution round, and Task Time is the average end-to-end agent execution time per task (in seconds), excluding skill evolution. Evo. / Task is the ratio of evolution time to task execution time.

| Method | Tokens / Skill | Evo. Time / Round (s) | Task Time (s) | Evo. / Task |
|---|---|---|---|---|
| AGSE | 156,475 | 111.3 | 697.3 | 0.16 |
| SkillFlow-Evo | 17,694 | 72.9 | 520.9 | 0.14 |
| Skill-Creator | 861,291 | 237.6 | 371.0 | 0.64 |

$$\mathrm{FWT} = \frac{1}{|\mathcal{N}_{\mathrm{new}}|} \sum_{i \in \mathcal{N}_{\mathrm{new}}} (s_i - b_i),$$

where $\mathcal{N}_{\mathrm{new}}$ contains tasks first encountered after online learning has begun. Positive FWT indicates that prior experience improves initial performance relative to the no-skill baseline.

We define forgetting at the task level. Let $\mathcal{N}_{\mathrm{old}}$ denote tasks that are re-evaluated after subsequent skill updates, and let $p_i$ indicate whether task $i$ was correctly solved at least once after its first introduction and before the final evaluation. Let $f_i^{\mathrm{final}} \in 0, 1$ indicate whether task $i$ is correctly solved at the final checkpoint. The forgetting rate is

$$\mathrm{FR} = \frac{1}{|\mathcal{N}_{\mathrm{old}}|} \sum_{i \in \mathcal{N}_{\mathrm{old}}} \mathbf{1}[p_i = 1 \wedge f_i^{\mathrm{final}} = 0].$$

Thus, a task is counted as forgotten only if the agent successfully solves it at least once after the task is introduced but fails on the final re-evaluation. Tasks that were never solved correctly are not considered forgotten, since there is no previously acquired capability to lose. Lower FR indicates better preservation of acquired capabilities.

Together, FS, FWT, and FR capture three complementary properties of online skill evolution: final capability, transfer to newly arriving tasks, and preservation of previously acquired capabilities.

# C Computational Overhead Analysis

We analyze computational overhead on OfficeQA Pro using the Codex harness with GPT-5.5. Performance is measured by exact accuracy at $\epsilon = 0$. As shown in Table 9, Skill-Creator incurs the highest overhead, consuming 861K tokens and 237.6s per evolution round, yet achieves lower accuracy than AGSE. This suggests that the gains of AGSE cannot be explained solely by increased token consumption or evolution time.

Compared with SkillFlow-Evo, AGSE incurs moderate additional overhead (156K vs. 17.7K tokens per evolution round), while achieving higher accuracy. Notably, SkillFlow-Evo performs worse than the w/o skill baseline, indicating that its evolution process can degrade task performance despite its low computational cost. In contrast, AGSE achieves performance gains through attribution-guided skill refinement, with an evolution-to-task-time ratio of only 16%, compared with 14% for SkillFlow-Evo and 64% for Skill-Creator. These results highlight the importance of how skills are evolved, rather than simply minimizing evolution overhead or increasing computation.

# D Manual Audit of Multidimensional Skill Attribution

## D.1 Sampling and Data Collection

We sampled 50 skill–trajectory pairs from 476 attribution calls recorded in the main experiment using coverage-stratified sampling instead of simple random sampling. The sample covers diverse skills, attribution rounds, task outcomes, and potential attribution failure types. Among them, 30 are

positive examples where the skill is relevant to the task, and 20 are negative examples with irrelevant skills. In addition, 29 tasks receive a final reward of 1 and 21 receive a reward of 0. The sample involves 6 skills and 49 distinct task UIDs. For each pair, human annotators independently assign corresponding labels after reviewing the same skill content and trajectory. For irrelevant samples, non-applicable dimensions are marked as NULL.

### D.2 Consistency Results

The five dimensions are ordered as activation, adherence, correctness, coverage, and precision. All 50 samples are included in the calculation, with three label classes $\{1, 0, \text{null}\}$ adopted for the five dimensions. Overall, the human annotator and the LLM agent achieve high consistency on five-dimensional attribution. Activation and adherence reach perfect agreement, while correctness and precision achieve 98.0% and 96.0% exact agreement respectively. Discrepancies are mainly observed for coverage (94.0% exact agreement). There are five sample pairs with annotation disagreements in at least one dimension. For samples with coverage disagreement, the controversies lie in whether high-grade corporate bond yield tables fall within the scope of the Treasury market quotation skill, and whether operations such as geometric mean and centered moving average should be regarded as missing capabilities of the skill. For samples with precision disagreement, the dispute concerns applying the currency circulation skill to Treasury bond quotation data. For samples with correctness disagreement, the debate is whether erroneous historical exchange rate sources should be attributed to insufficient coverage of foreign exchange data sources or quote directions in the skill, rather than intrinsic errors in the skill content itself.