

SkillLadder: Multi-User Skill Evolution with Compliance-Oriented Synchronization and Quality-Gated Synthesis

Nan Chen^{1*}, Yi Wang², Bin Benjamin Zhu^{3†}, Haoyi Xiong^{2‡}

¹Shanghai Jiao Tong University

²Microsoft Corporation

³Beijing Institute of Mathematical Sciences and Applications

arcs-ur@sjtu.edu.cn, yiwang4@microsoft.com

binzhu@ieee.org, haoyi.xiong.fr@ieee.org

Abstract

LLM agents accumulate valuable procedural knowledge during task execution, but these traces are session-bound, model-specific, and often entangled with sensitive content, making them unsafe to pool directly across users. Prior approaches to skill discovery often sacrifice cross-user generalization, iterative quality, or compliance-oriented sharing. We propose SKILLLADDER, a multi-user framework that distills local execution traces into a shared skill library without funneling raw data to a central server. The system evolves user-contributed traces into a multi-level hierarchy (L0–L3) using five quality gates that prevent regression below the strongest parent skill under a judge-defined score. To ensure safe sharing, SKILLLADDER distributes only level- ≥ 1 skills that have undergone entity scrubbing and lineage sanitization. On τ^2 -bench, where each isolated task execution simulates one contributing user—a deliberately conservative proxy for many small local contributors, not a model of longitudinal human users—evolved skills improve a fixed Claude Sonnet 4.6 agent in three of four domains and generalize to simulated users running different agent models. Compliance audits verify that entity retention reaches zero by L3 and source-attribution accuracy is roughly halved, substantially reducing measured sharing risks.

1 Introduction

A rapidly growing body of work argues that practical agent progress increasingly comes from externalizing procedural knowledge — skills, protocols, and reusable memory — rather than enlarging model weights (Zhou et al., 2026a), and dedicated benchmarks confirm that well-crafted agent skills consistently improve professional task

performance (Li et al., 2026b; Wang et al., 2026b; Zhang et al., 2026a). This has motivated a rich design space along three axes. *Representation* ranges from hierarchical or multi-level skill documents (Wang et al., 2026a; Ni et al., 2026; Zhou et al., 2026b), to executable Skill-MDP units (Mi et al., 2026), to compact non-documentation forms (Wang et al., 2026c). *Discovery* is driven by failure- or resource-driven mining (Alzubi et al., 2026; Shen et al., 2026) and by domain-grounded enterprise pipelines (Liu et al., 2026b,a; Tao et al., 2026; Hu et al., 2026; Ren et al., 2026; Lei et al., 2026). *Refinement* is realized through surrogate co-evolutionary verifiers (Zhang et al., 2026b), prioritized-and-clustered evolution (Xiong et al., 2026), and training-free GRPO-style optimizers (Tian et al., 2026); a complementary line couples skill evolution with policy training rather than keeping the agent frozen (Xia et al., 2026; Shi et al., 2026; Li et al., 2026c; Ouyang et al., 2026).

Existing systems make substantial progress in skill generation, refinement, retrieval, and collective evolution (Zhang et al., 2026b; Wang et al., 2026a; Ma et al., 2026; Xia et al., 2026; Ouyang et al., 2026). These approaches, however, almost universally assume a single agent, single user, or single trusted trace pool. In practice, LLM agents are deployed across many users and organizations, each accumulating procedural knowledge that is also entangled with credentials, customer attributes, internal URLs, and proprietary workflows. The natural next step—*multi-user skill evolution*—is to distill these lessons into a shared, improving library without funneling raw traces to a central server (Ma et al., 2026), but doing so raises the governance bar. A multi-user skill-evolution framework is therefore simultaneously a learning system and a governance system: it applies **data-minimization** principles by removing or transforming direct identifiers before any cross-user sharing, acting as an empirical **data-loss-prevention (DLP)** layer that blocks acci-

* Work done during an internship at Microsoft.

† Work completed as a FTE at Microsoft.

‡ Corresponding author.

dental propagation of sensitive artifacts across user, organizational, or tenant boundaries. Appendix C compares these objectives with representative skill-evolution frameworks.

Technical challenges of multi-user skill evolution. Building such a framework raises three coupled challenges, and each rules out a naive composition of existing techniques.

- **C1: Cross-user, cross-task generalization.** Skills mined from one user’s traces must abstract beyond a single session while preserving executable steps, so that they help a different user on a related but non-identical task. Per-user local mining (Alzubi et al., 2026; Shen et al., 2026) and single-source representations (Wang et al., 2026a; Ni et al., 2026) sidestep sharing but forfeit cross-source generalization, yielding fragmented libraries with no cumulative gain across users.
- **C2: Non-degradation under iterative, multi-source evolution.** As skills from different users are merged and re-abstracted, synthesis can hallucinate steps, drop preconditions, or over-generalize. Existing refiners verify or rank one skill at a time (Zhang et al., 2026b; Xiong et al., 2026; Tian et al., 2026), but naively chaining such per-skill refiners does not enforce that a merged, higher-level skill be no worse than its sources, and quality silently degrades as evolution deepens.
- **C3: Compliance-oriented sharing across user boundaries.** Traces may expose personal data, credentials, file paths, internal endpoints, or proprietary workflows. Approaches that ship raw traces or run centralized evolution and retrieval over them (Ma et al., 2026; Li et al., 2026a; Su et al., 2026) expose far more data than is needed for skill reuse, violating data-minimization and DLP goals; central-propose / distributed-verify variants reduce but do not eliminate central access to sensitive content. Sharing must therefore implement quantifiable data minimization, de-identification / pseudonymization hooks, and DLP-style release checks at every cross-user boundary, rather than overclaiming anonymization or cryptographic privacy.

Our approach. We present SKILLLADDER, a *framework* for multi-user skill evolution instanti-

ated as a four-phase cycle. **(i) Collect:** each contributor executes tasks while raw traces remain in its environment. **(ii) Extract:** a contributor-side trusted processor distills and scrubs trace-local L0 skills before they enter a trusted evolution coordinator. **(iii) Evolve:** the coordinator clusters skills across contributors and proposes higher-level children; five quality gates compare each child with its strongest parent, and rejected abstractions leave their parents intact, jointly addressing C1 (cross-source abstraction) and C2 (no silent regression). **(iv) Distribute:** only level- ≥ 1 skills cross the sharing boundary; entity scrubbers replace covered identifiers with typed placeholders, and lineage sanitization strips source-task and source-user metadata; once updated, the latest version is synchronized to every user’s local library so all participants share the same evolving state, jointly addressing C3. The distribute phase is deliberately empirical: rather than asserting cryptographic guarantees, SkillLadder is designed for practical compliance deployments and exposes pluggable slots for stronger sanitization. We measure residual release risk with entity retention (ER) and source-attribution accuracy (SA); these metrics quantify what survives the release pipeline, and leave anonymization or non-membership modules pluggable.

Our contributions are:

1. We formulate the problem of multi-user skill evolution as a joint utility-and-release problem, characterizing the compliance objectives and privacy concerns that distinguish it from single-user skill discovery.
2. We propose an integrated solution comprising compliance-oriented synchronization and quality-gated synthesis, which together address cross-user generalization, non-degradation under iterative evolution, and safe cross-boundary sharing.
3. We conduct experiments on τ^2 -bench that treat each isolated task execution as a simulated contributing user and explicitly bind evidence to challenges: C1 via cross-simulated-user reward (Table 4), C2 via the leave-one-gate-out ablation (Table 14), and C3 via ER, SA, and source-set mixing μ_ℓ (Tables 1–2), with AWM (Wang et al., 2025) as the external memory baseline.

The code, configurations, prompts, and exact task splits are available at <https://github.com/Arcs-ur/SkillLadder>.

2 Method

SkillLadder is organized around Collect, Extract, Evolve, and Distribute. Section 2.1 formalizes the setting and trusted computing base; §2.2 describes contributor-local collection and extraction; §2.3 defines hierarchical evolution and lineage; §2.4 specifies peer distribution and retrieval; §2.5 defines candidate acceptance; and §2.6 states the trust and release boundaries.

2.1 Overview

Given K contributors $\{U_1, \dots, U_K\}$, each running an LLM agent on tasks from a shared domain, contributor k accumulates a local trace pool $\mathcal{D}_k$. Raw traces remain in the contributor environment. A contributor-side extractor converts them into data-minimized L0 artifacts, which may enter a trusted evolution coordinator; the coordinator receives neither raw traces nor permission to release L0 to peers. The current trusted computing base comprises the contributor-side extractor, the coordinator, and the model services they invoke. From the internal artifacts, SkillLadder builds a release-filtered library $\mathcal{L}$ that recipient agents can consult at task time. The library is optimized to maximize expected return on a task distribution $\mathcal{T}$ (ξ is one task from $\mathcal{T}$),

$$\max_{\mathcal{L}} \mathbb{E}_{\xi \sim \mathcal{T}} [R(\xi, \pi, \Omega_\xi)], \quad \Omega_\xi = \text{RETRIEVE}(\xi, \mathcal{L}), \quad (1)$$

and is produced by aggregating per-user level-0 skills and applying R rounds of an evolution operator $\mathcal{E}$:

$$\mathcal{L} = \mathcal{E}^{(R)}(\mathcal{S}^{(0)}), \quad \mathcal{S}^{(0)} = \bigcup_k \text{EXTRACT}(\mathcal{D}_k). \quad (2)$$

Indices. The *evolution level* $\ell \in \{0, 1, \dots\}$ records abstraction depth; the *round* r indexes one clustering–synthesis pass; and the *cycle* c denotes one full **Collect–Extract–Evolve–Distribute** invocation.

2.2 Collect and Extract

Collect accumulates each contributor’s task descriptions and ReAct-format traces inside its local environment. **Extract** uses a contributor-approved model service to turn success traces into effective

procedures and failure traces into defensive checks and fallbacks. Before an L0 artifact leaves the contributor environment for trusted evolution, covered paths, URLs, keys, identifiers, and named entities are replaced with typed placeholders. The resulting $\mathcal{S}_k^{(0)}$ remains single-source and internal: it can be inspected by the coordinator for clustering and gating but is never distributed to peer users.

Skill representation. Every skill, whether level-0 or evolved, is stored as the same composite record of natural-language guidance, optional executable scripts, and runtime metadata:

$$s = (\text{name}, \text{desc}, \text{trigger}, \text{proc}, \text{example}, \text{scripts}, \text{cat}, \ell, \pi, \eta). \quad (3)$$

The first five fields encode what an agent needs to recognize and apply the skill; *scripts* stores optional executable code with placeholders in place of concrete paths and keys; and the metadata block records the semantic category *cat*, the evolution level ℓ , parent IDs π , and a thresholded empirical success rate $\eta \in [0, 1]$:

$$\eta(s) \leftarrow \frac{\#\{t \in I_s : r_t > 0.5\}}{|I_s|}, \quad (4)$$

where I_s is the set of sessions in which s has been used. A second runtime signal, the EMA $\bar{r}(s)$ of continuous task reward, is maintained in parallel and combined with η for per-task retrieval scoring (Appendix H).

2.3 Evolve: Hierarchy and Lineage

Levels L0–L3 form a lineage-preserving abstraction DAG rather than independent skill categories. Each accepted level- $(\ell-1)$ artifact can parent a level- ℓ candidate; the child stores its parent IDs internally, and its strongest parent supplies the non-degradation reference. A rejected candidate does not replace its parents. Moving from L0 to L1 changes release eligibility, while L1–L3 record increasing abstraction depth and source mixing. Appendix M traces one saved L0→L3 lineage.

Each evolution round r then lifts the skill set by one level via clustering and synthesis:

$$\mathcal{S}^{(r)} = \mathcal{E}(\mathcal{S}^{(r-1)}) = \bigcup_{j=1}^J \text{QG}(\text{SYNTH}(\mathcal{C}_j)), \quad (5)$$

where $\{\mathcal{C}_1, \dots, \mathcal{C}_J\}$ partitions $\mathcal{S}^{(r-1)}$ and QG denotes the quality-gate stack of §2.5. Algorithm 1 in Appendix A gives one full cycle.

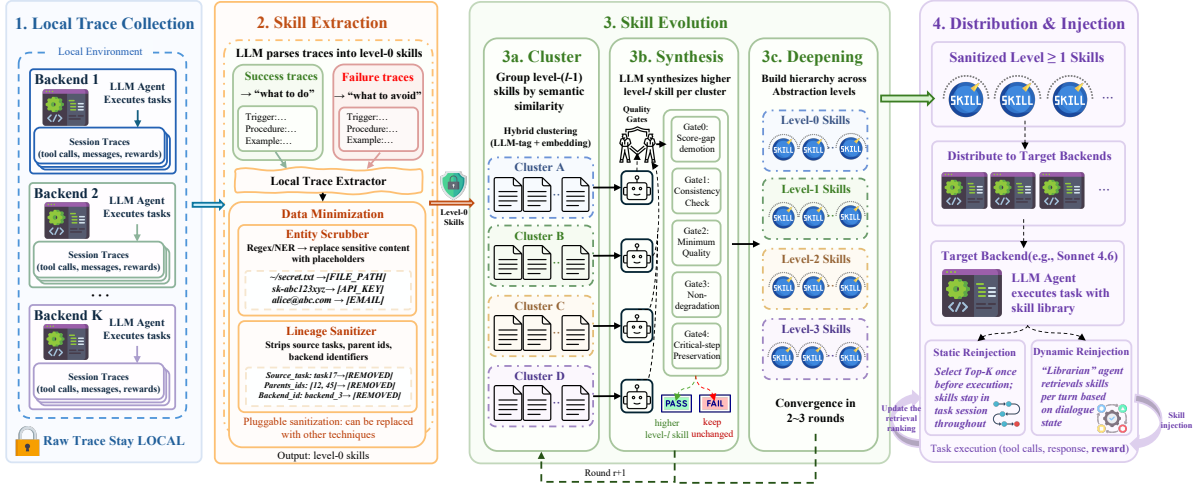


Figure 1: **SkillLadder architecture.** SkillLadder’s four-phase workflow and trust boundaries. Contributors retain raw traces locally and send only scrubbed L0 artifacts to trusted evolution. Clustered parents produce higher-level candidates along an explicit L0-to-L3 lineage; candidates that fail the strongest-parent quality gates are retried or discarded while their parents remain available. Only gated, re-scrubbed, and lineage-sanitized L1-L3 skills cross the peer-user release boundary.

Clustering and synthesis. Clustering is a swappable component; the default *hybrid* strategy uses one LLM call to discover $C \in [3, 15]$ categories, assigns skills via MiniLM-L6-v2 (Wang et al., 2021) embeddings, and splits oversized categories by agglomerative clustering ($\delta=0.5$, $C_{\max}=8$); alternatives are benchmarked in Appendix F. For every cluster C_j with $|C_j| \geq \tau_{\min}=2$, the LLM synthesizes a higher-level skill preserving the cluster’s procedural backbone; candidates are screened by the quality-gate stack of §2.5, and failed or undersized clusters pass through unchanged. Later batches of contributor can reopen evolution.

2.4 Distribute: Release and Retrieval

After evolution, only release-eligible skills are synchronized to recipient libraries. Three mechanisms reduce what crosses the peer-user boundary:

- **Level-gated release.** L0 artifacts remain internal to trusted evolution and are never peer-distributed. Only level- ≥ 1 skills can cross the release boundary after generalization and source mixing (Proposition 1, Appendix D).
- **Entity scrubbing (pluggable).** A release-side scrubber—regex patterns for file paths, URLs, IPs, API keys, emails, and usernames; a lightweight NER model for person, organization, and location names; and a deployment blocklist—again replaces matched values with typed placeholders such as [FILE_PATH].

Stricter detectors or human release review can replace or augment this empirical DLP layer.

- **Lineage sanitization (pluggable).** Skill records carry `source_task`, `parent_ids`, and source-user identifiers internally; these fields are stripped at the boundary so receivers see only the sanitized skill body and level ℓ , removing the metadata path to attribution and fingerprinting.

Per-task retrieval. Sanitized skills are written into each user’s library in agent-native format. SkillLadder retrieves a compact top- k (default 10) subset per task by static scoring or a dynamic librarian; see Appendix H for retrieval details and Appendix J, Table 17 for the prompt/latency cost trade-off.

2.5 Quality-gated Synthesis

Every synthesized candidate is screened by a stack of five gates (Table 7); failure triggers up to two retries before falling back to the original cluster members. Gates 2 and 3 share a generalizability score,

$$\text{score}(s; \mathcal{T}) = \frac{|\{\xi \in \mathcal{T} : \text{app}(s, \xi) \geq 3\}|}{|\mathcal{T}|} \times \overline{\text{rel}}_{\text{app} \geq 3}, \quad (6)$$

where per-task applicability $\text{app}(s, \xi)$ and relevance $\text{rel}(s, \xi)$ are judged by an LLM on a 0–10 scale (exact aggregation protocol in Appendix G).

Gate 3 enforces non-degradation:

$$\text{score}(s^{(r)}; \mathcal{T}) \geq \max_{s \in \mathcal{C}_j} \text{score}(s; \mathcal{T}), \quad (7)$$

so an evolved skill is accepted only if it scores at least as well as the strongest cluster member; otherwise originals carry forward. Audits are in Appendix G.

2.6 Trust and Release Boundaries

The current trusted computing base contains the contributor-side extractor, the trusted evolution coordinator, and their approved model services. The evaluated adversary is instead an honest-but-curious receiving user who can inspect every distributed skill and run automatic analyses but cannot see raw traces, internal L0 artifacts, private lineage, or sanitizer configuration. This boundary yields three visibility classes: raw traces remain contributor-local; scrubbed L0 artifacts are coordinator-visible but not release-eligible; and only gated, sanitized L1–L3 skills are peer-visible. Active poisoning, coordinator compromise, collusion, and high-effort inversion remain out of scope and require controls such as audited provenance, isolated execution, or differential privacy.

Table 8 in Appendix B maps six receiver capabilities to mechanisms and evidence. T1 (content leakage) is quantified by entity retention (ER)—the fraction of covered entities surviving scrubbing. T2 (source attribution) is quantified by source-attribution accuracy (SA), the accuracy of an LLM adversary linking each skill to candidate source pools; T6 (style fingerprinting) is an SA proxy. T3–T5 receive structural or pluggable mitigations via source-set mixing μ_ℓ , abstraction, and pluggable controls.

Deterministic scrubbing for covered patterns is stated formally as Property 1 in Appendix D. Intuitively, the release boundary applies a hard pre-distribution filter that replaces matched identifiers with typed placeholders. This is a coverage-limited design invariant rather than an end-to-end privacy guarantee: the substantive question is detector coverage, which is measured empirically by $\text{ER}(\ell)$ in §3.3.

3 Experiments

We evaluate SkillLadder on τ^2 -bench across four customer-service domains, reporting both skill-performance improvements and data-protection metrics. Section 3.1 describes datasets and metrics;

§3.2 explains the multi-user evaluation protocol; §3.3 reports overall results addressing C1 (cross-simulated-user generalization) and C3 (release-risk reduction); §3.4 presents ablation studies that address C2 (non-degradation) through gate and level ablations; and §3.5 analyzes case studies. Hyperparameter settings are listed in Appendix K, Table 18.

3.1 Datasets and Metrics

We evaluate on τ^2 -bench (Barres et al., 2025), which provides domain-specific user simulations, APIs, databases, and reward checks across four customer-service splits: airline (50 tasks), retail (114), banking-knowledge (97), and telecom (200). The main experiments use Claude Sonnet 4.6 for both agent and user simulator. Under the honest-but-curious threat model of §2.6, ER directly measures T1 (whether credentials, file paths, URLs, phone numbers, and other detected entities survive into shared skills); SA measures T2 (source attribution) directly and proxies T6 (style fingerprinting) by testing whether an LLM adversary can link a shared skill to one of $m=5$ candidate source pools; and μ_ℓ is structural evidence for source aggregation, relevant to T3 though not a membership-inference guarantee.

3.2 Multi-User Evaluation Setup

Each task in a domain serves as one independent *simulated contributing user*: it is executed by an isolated agent instance, contributes only its own trace-derived L0 skills, and later receives the synchronized library for evaluation. This one-task-per-user abstraction is intentional rather than a claim about longitudinal human behavior: τ^2 -bench already provides independent personas, API states, and reward checks, and isolating each execution stress-tests whether SkillLadder can aggregate many small, privacy-bounded contributions without relying on repeated interactions from the same source. To probe generalization in a multi-user setting where users may run agents with diverse models, we evaluate Claude Sonnet 4.6 (the contributor model, measuring same-simulated-user reuse), Qwen3-32B, and Qwen3-8B (Yang et al., 2025), with Claude Sonnet 4.6 retained as the user simulator throughout. We compare AWM (Wang et al., 2025) as the external memory baseline and use L0-only, L1+-only, full-static, and full-dynamic variants as policy-feasible internal baselines. We omit head-to-head comparisons with systems such as

SkillClaw (Ma et al., 2026) and Skill-Pro (Mi et al., 2026) for two reasons: (a) they expose centralized traces or source metadata that SkillLadder deliberately withholds, so a reward comparison would conflate utility with a different governance regime; and (b) they iteratively refine skills on a single task rather than aggregating across many independent users, so they do not target the cross-task, cross-user generalization regime we evaluate.

3.3 Overall Results and Comparison

Results are reported in two parts: skill-performance improvements from library injection (Table 4) and empirical release-risk measurements across evolution levels (Tables 1–2).

Skill Performance

Table 4 reports fixed-agent results under no skills, static reinjection, and dynamic per-turn retrieval across all four domains and three simulated users.

SkillLadder wins on three of four domains and dominates raw-trace pooling on all twelve (domain, model) cells. Evolved injection lifts reward in three domains (largest +0.177 telecom dynamic; retail is the sole regression, analyzed in §3.5). Routing raw trace segments through the same scoring function with the level-bonus and runtime-quality terms zeroed (B1, App. H) does not match SkillLadder on any cell: Δ ranges from -0.190 (airline / Qwen3-32B) to $+0.030$ (telecom / Qwen3-8B), with 11/12 cells at or below baseline, while B1 inherits the privacy upper bound of Tables 1–2 (ER=1.0, SA=1.0). The (utility, ER, SA) frontier is therefore dominated by SkillLadder Full static/dynamic across all measured conditions.

The synchronized library generalizes across diverse simulated users, but the best level tracks the user’s capability. On a weaker simulated user (Qwen3-8B), full-evolved injection lifts reward by $+0.244 / +0.108 / +0.253 / +0.211$ on airline/banking/retail/telecom while L0-only is markedly weaker (airline -0.043 ; 0.09 – 0.22 below the full library elsewhere). On a stronger user (Qwen3-32B) the pattern partially reverses—L0-only matches or exceeds Full-static on every domain—but per-turn dynamic retrieval still recovers the evolved-skill advantage (banking $+0.085$, retail $+0.133$). We dissect this level shift in §3.4.

Dynamic retrieval delivers our largest multi-user generalization gain on retail. In the “Full dynamic” row, dynamic retrieval recovers on retail what static reinjection loses ($+0.287$ on the

Table 1: Entity retention rate (ER) by domain and evolution level. Raw traces (B1 baseline) expose all detected entities (ER=1.0). Evolution progressively removes covered entities from shared skills.

Domain	Trace entities	L0 ER	L1 ER	L2 ER	L3 ER
Airline	34	0.000	0.000	0.000	0.000
Banking	261	0.081	0.015	0.012	0.000
Retail	587	0.019	0.009	0.000	0.000
Telecom 200	9	0.333	0.333	0.333	0.000
B1: Raw traces		ER = 1.0 (all entities exposed)			

Table 2: Privacy metrics by evolution level (aggregated). μ_ℓ : source mixing degree (min / mean). ER: entity retention rate. SA: LLM-adversary source attribution accuracy ($m=5$, random baseline = 0.2).

Level ℓ	μ_ℓ (min / mean)	ER(ℓ)	SA(ℓ)
B1 (raw traces)	1 / 1.0	1.000	1.00
0 (extracted)	1 / 1.0	0.033	0.70
1 (1st evolution)	2 / 4.1	0.008	0.57
2 (2nd evolution)	4 / 6.2	0.004	0.55
3 (3rd evolution)	8 / 12.4	0.000	0.46

Qwen3-8B simulated user, the largest single improvement among diverse simulated users in the table; $+0.133$ on Qwen3-32B), with a similar but smaller effect on banking. Telecom dynamic stays within noise of static ($|\Delta| \leq 0.007$); on airline, dynamic slightly underperforms static on Sonnet and Qwen3-32B, suggesting per-turn retrieval can over-prune when static reinjection is already strong. Dynamic injection is therefore most valuable where static reinjection underperforms, incurring roughly $2\times$ wall-clock overhead and two additional LLM calls per turn (App. J).

Data Protection Metrics

Detector-covered entity retention drops monotonically with evolution level, reaching zero by L3 across all domains. Table 1 reports detector-based ER by domain and level; Table 2 adds source-attribution accuracy (SA) and source-set mixing μ_ℓ .

L0 extraction plus default scrubbing already removes 67–100% of detected entities; residual banking entities are date-formatted path fragments, and telecom retains example phone numbers as domain-relevant operational data until L3. Because detector-based ER is bounded by detector coverage, we also conduct the human audit below.

Human adjudication confirms low released-skill ER while exposing detector errors. We audit 199 stratified trace/skill documents and all 650 detector candidates. Human review confirms 646 candidates as true positives and 4 as false positives;

Table 3: Skill library statistics after $R=3$ evolution rounds (aggregated across airline, banking, retail, and telecom domains).

	Level-0	Level-1	Level-2	Level-3
# Skills	885	107	48	9
Avg. tokens	466	1,491	2,521	2,949
Avg. score	0.00	1.98	2.48	2.60
# With scripts	282	85	39	7
# Categories	43	32	20	8

complete-document, candidate-free review finds 2 additional missed spans. Against this human gold, released L1–L3 skills retain 5 of 648 source-entity opportunities (0.77%), and L3 retention is 0.0 in all four domains. This finite audit supports empirical data minimization but cannot certify absence of leakage; Appendix E gives the protocol and result table.

Source attribution decreases but remains above chance and protocol-sensitive. Under the original five-way protocol, L3 SA is $13/28 = 0.464$ against a 0.20 random baseline, a 0.236 absolute (33.7% relative) reduction from L0 rather than a halving; its approximate Wilson 95% interval is $[0.30, 0.64]$. A separate DeepSeek-v3.2 audit covers all 29 L3 skills with repeated same-domain, shared-tool, same-task-type, embedding-nearest hard negatives. Five-way SA is 0.286 (skill-bootstrap 95% CI $[0.210, 0.369]$; chance-normalized advantage 0.108), and paired ten-way SA is 0.167 ($[0.112, 0.228]$; advantage 0.075). Expanding the lineup reduces absolute SA by 0.119 (paired CI $[-0.179, -0.062]$). Because the judge family and distractor protocol both change, this comparison establishes sensitivity, not a single-factor causal effect. Together with μ_ℓ , the audits support measured risk reduction while leaving residual linkability explicit.

3.4 Ablation Studies

Learned skill library. Table 3 summarizes the library after $R=3$ evolution rounds: evolution compresses 885 L0 skills into 164 evolved skills while the average generalizability score grows from 0 (L0) to 2.60 (L3). Token count grows with level because synthesis merges multiple parent procedures; Appendix M shows an artifact-backed telecom lineage.

Injection audit. The Mix column of Table 4 audits which skill levels enter the prompt under static vs. dynamic retrieval (mean skills

per dynamic consult 2.3/2.2/2.1/1.3 on airline/banking/retail/telecom). **Both policies favor evolved skills (typically 68–87% L2+).** Banking under dynamic retrieval is the sole outlier at $\sim 38\%$ L2+ (avg $\ell \approx 1.3$), suggesting state-conditioned lookup sometimes prefers narrower L1 operational procedures over evolved abstractions on that domain.

Skill-level effects within and across simulated users. L0 skills suffice for same-simulated-user reuse, while L1+ carries most multi-user generalization gain on Qwen3-8B. On Sonnet, L0-only already improves airline/banking/telecom ($+0.060/ +0.052/ +0.125$). On Qwen3-8B, L0-only is much less reliable and L1+ recovers nearly all full-library gain; on Qwen3-32B, L0-only often matches or exceeds Full static, while dynamic retrieval recovers evolved-skill gains on banking and retail. The useful level therefore tracks the source–target capability gap, so the ladder is a tunable interface rather than a monotone “higher is better” axis. Detailed per-domain deltas, release-policy caveats, and same-row sample-size notes for Table 4 are in App. L; the retail mechanism analysis remains in §3.5.

Quality-gate. Across 830 candidate clusters from all four domains, acceptance drops from 82% at R1 to 43% at R3 while accepted skills beat their strongest parent on average ($5.93 \rightarrow 7.58$ at R2; $5.88 \rightarrow 7.16$ at R3). However, the median margin is only 0.14 on a 0–10 scale and only 21% of accepted skills beat the strongest parent by ≥ 1.0 , indicating that the non-degradation gate enforces “no regression” rather than strict improvement. On airline R1 (51 clusters), leave-one-gate-out ablations confirm that relaxing any single gate lowers mean judge score: -0.34 from non-degradation (G3), -0.38 from critical-step preservation (G4), and -0.11 from consistency (G1). Crucially, this non-degradation pattern and the role of Gate 3 reproduce outside the Claude family. Independent rescoring of 122 saved L2/L3 child–parent pairs rates the child no worse in 121/122 cases with DeepSeek-v3.2 and 117/122 with GLM-5.2 (both rate all 29 L3 children as non-degrading). In a stronger replication replacing Claude with DeepSeek-v3.2 for both R2/R3 synthesis and gate scoring (320 candidate clusters, 1,280 condition evaluations), the full gate accepts 175/320 candidates (100% non-degrading), whereas removing Gate 3 accepts 235/320 candidates but reduces non-

Table 4: Joint skill-level ablation. Sonnet, Qwen3-32B, and Qwen3-8B share the same evolved library (§3). Each cell reports reward $R_{\pm SE}^{\Delta}$, where Δ is relative to the per-agent baseline. “Mix” reports injected skill-slot percentages in L0/L1/L2/L3 order; the retriever is identical across consumers, so mix differences are condition-driven. **B1 (raw-trace concat)** reuses the SkillLadder static retriever with raw trace segments as the corpus (no abstraction, no scrub) and is the privacy upper bound corresponding to the B1 row of Tables 1–2 (ER=1.0, SA=1.0). Detailed interpretation, release-policy caveats, and paired-contrast sample sizes are in App. L.

Condition	Sonnet	Qwen3-32B	Qwen3-8B	Mix
<i>Airline</i>				
○ Baseline	0.760 $\pm$ 0.012	0.380 $\pm$ 0.020	0.277 $\pm$ 0.020	—
◇ AWM	0.761 $\pm$ 0.001	0.432 $\pm$ 0.052	0.302 $\pm$ 0.025	100/0/0/0
□ B1 raw-trace	0.730 $\pm$ 0.030	0.190 $\pm$ 0.190	0.212 $\pm$ 0.065	raw segs
△ L0-only	0.820 $\pm$ 0.014	0.502 $\pm$ 0.025	0.234 $\pm$ 0.014	100/0/0/0
▽ L1+ only	0.839 $\pm$ 0.011	0.501 $\pm$ 0.121	0.520 $\pm$ 0.018	0/32/42/26
● Full static	0.840 $\pm$ 0.015	0.498 $\pm$ 0.118	0.521 $\pm$ 0.244	0/32/42/26
★ Full dynamic	0.780 $\pm$ 0.012	0.456 $\pm$ 0.076	0.494 $\pm$ 0.217	2/25/40/33
<i>Banking</i>				
○ Baseline	0.125 $\pm$ 0.017	0.071 $\pm$ 0.016	0.035 $\pm$ 0.016	—
◇ AWM	0.130 $\pm$ 0.005	0.080 $\pm$ 0.009	0.064 $\pm$ 0.013	100/0/0/0
□ B1 raw-trace	0.105 $\pm$ 0.015	0.056 $\pm$ 0.014	0.035 $\pm$ 0.000	raw segs
△ L0-only	0.177 $\pm$ 0.012	0.108 $\pm$ 0.037	0.053 $\pm$ 0.013	100/0/0/0
▽ L1+ only	0.168 $\pm$ 0.020	0.091 $\pm$ 0.016	0.140 $\pm$ 0.104	0/13/44/43
● Full static	0.168 $\pm$ 0.043	0.091 $\pm$ 0.019	0.143 $\pm$ 0.011	0/13/44/43
★ Full dynamic	0.208 $\pm$ 0.012	0.156 $\pm$ 0.017	0.164 $\pm$ 0.017	28/33/18/21

Condition	Sonnet	Qwen3-32B	Qwen3-8B	Mix
<i>Retail</i>				
○ Baseline	0.811 $\pm$ 0.010	0.568 $\pm$ 0.010	0.247 $\pm$ 0.013	—
◇ AWM	0.790 $\pm$ 0.021	0.598 $\pm$ 0.030	0.297 $\pm$ 0.050	100/0/0/0
□ B1 raw-trace	0.761 $\pm$ 0.050	0.512 $\pm$ 0.012	0.203 $\pm$ 0.044	raw segs
△ L0-only	0.773 $\pm$ 0.038	0.658 $\pm$ 0.090	0.283 $\pm$ 0.016	100/0/0/0
▽ L1+ only	0.721 $\pm$ 0.013	0.648 $\pm$ 0.011	0.500 $\pm$ 0.009	0/22/47/32
● Full static	0.719 $\pm$ 0.090	0.654 $\pm$ 0.012	0.500 $\pm$ 0.253	3/21/46/31
★ Full dynamic	0.782 $\pm$ 0.017	0.701 $\pm$ 0.133	0.534 $\pm$ 0.287	1/13/67/18
<i>Telecom 200</i>				
○ Baseline	0.425 $\pm$ 0.012	0.363 $\pm$ 0.012	0.046 $\pm$ 0.016	—
◇ AWM	0.429 $\pm$ 0.004	0.386 $\pm$ 0.023	0.147 $\pm$ 0.101	100/0/0/0
□ B1 raw-trace	0.416 $\pm$ 0.012	0.356 $\pm$ 0.014	0.076 $\pm$ 0.015	raw segs
△ L0-only	0.416 $\pm$ 0.009	0.356 $\pm$ 0.007	0.076 $\pm$ 0.030	raw segs
▽ L1+ only	0.550 $\pm$ 0.125	0.455 $\pm$ 0.013	0.147 $\pm$ 0.101	100/0/0/0
● Full static	0.599 $\pm$ 0.017	0.446 $\pm$ 0.083	0.257 $\pm$ 0.211	0/13/54/33
★ Full dynamic	0.600 $\pm$ 0.023	0.447 $\pm$ 0.012	0.257 $\pm$ 0.014	0/13/54/33
	0.602 $\pm$ 0.021	0.450 $\pm$ 0.017	0.264 $\pm$ 0.218	0/25/54/21

degradation to 70.2%. Thus, the evidence confirms that the quality gate reliably enforces a conservative no-regression threshold across tested model families. Appendix G details the Claude and non-Claude audits.

3.5 Case Studies

Retail failure analysis. Retail’s reward drop comes from agent-level interference, not infrastructure noise. Table 5 decomposes the failure modes that actually enter the reward computation (infrastructure errors are excluded from reward by τ^2 -bench convention). **Static reinjection roughly doubles the behavioral-failure rate** (7.8% $\rightarrow$ 12.4%), indicating that over-general skills can distract the agent on exact-argument API calls, and also raises the max-step rate (11.1% $\rightarrow$ 15.7%). **Dynamic retrieval removes the behavioral excess** (6.9%, on par with the 7.8% baseline) but still incurs slightly more max-step failures (14.9%), so its residual reward gap is driven by longer trajectories rather than wrong actions. Appendix N traces two of the six baseline-pass / static-fail tasks to the specific L1 skills that caused them; the corrective directions there are consistent with those two cases rather than established over the full set.

Pluggable scrubbing strategies. No single scrubbing strategy dominates across domains, validating the pluggable architecture. Table 6 compares no scrubbing, regex, and Presidio (Data Privacy Stack and Microsoft, 2026) NER at the

Table 5: Retail 114 failure breakdown, restricted to the failure modes that contribute to reward. Cells report rates as % of evaluated tasks. Behavioral = DB/action mismatch; Max-step = agent exhausted the step budget.

Method	Behavioral (%)	Max-step (%)
Baseline	7.8	11.1
Static reinjection	12.4	15.7
Dynamic librarian	6.9	14.9

Table 6: Entity retention rate (ER) at L0 under different scrubbing strategies. Best per domain in **bold**. No single strategy dominates, validating the pluggable design.

Strategy	Airline	Banking	Retail	Telecom 200
No scrub	0.000	0.081	0.019	0.333
Regex	0.000	0.000	0.015	0.222
Presidio NER	0.000	0.073	0.000	0.000

same sanitization slot, reporting L0 ER where scrubbing matters most. **Regex and Presidio are complementary, not substitutable:** regex removes banking’s date-path fragments but leaves some phone-number fragments, while Presidio catches retail and telecom phone numbers but leaves banking date paths. Sanitization should therefore be a configurable DLP component rather than a fixed release mechanism.

4 Discussion and Conclusion

We introduced SkillLadder, a four-phase framework for accumulating procedural knowledge across contributors without peer-distributing raw

traces or single-source L0 artifacts. Its key idea is to couple a lineage-preserving abstraction ladder with strongest-parent quality gates and an explicit release boundary: trusted evolution can use data-minimized internal artifacts, while recipients receive only gated and sanitized L1–L3 skills. On τ^2 -bench, evolved skills improve three of four same-model domains and transfer to Qwen3-8B and Qwen3-32B recipients; the useful abstraction level depends on recipient capability, and dynamic retrieval is most valuable where static reinjection interferes with exact tool arguments. The release audits calibrate rather than eliminate risk. Human-adjudicated ER is 5/648 for released L1–L3 skills and zero at L3 in all four domains, while original and hard-negative SA remain above chance. DeepSeek and GLM rescoring plus the full DeepSeek synthesis/gating rerun show that the no-regression trend is not confined to Claude under the tested protocol. The resulting contribution is a governed transfer layer with measurable utility and residual risk, not an anonymization mechanism. Longitudinal deployments, policy-matched comparisons, prompt perturbation studies, and user-aware contribution controls remain the most direct extensions.

5 Limitations

The evidence is limited to customer-service/API tasks in τ^2 -bench; code generation, web navigation, and open-ended long-horizon settings remain untested. One isolated task execution represents one simulated contributor, which prevents self-trace reuse but does not model recurring users, correlated sessions, unequal contribution volume, or organization-specific drift. Longitudinal deployments should therefore use user-aware clustering, contribution caps, or unique-user thresholds rather than treating source sessions as distinct people.

The non-Claude audits reuse a fixed disclosed rubric and prompt templates; they test portability across the evaluated model families, not invariance to prompt paraphrases, random seeds, unrelated datasets, or broad hyperparameter perturbations. Reward cells use mean $\pm$ SE without paired-bootstrap or multiple-comparison correction, and close cells should not be read as a strict ordering. The static scorer’s unablated level bonus can affect which abstraction depth is retrieved, while dynamic-librarian latency is measured only on airline-50. External comparisons are limited

to AWM and policy-feasible internal baselines because centralized trace-sharing systems instantiate a different release regime. Finally, source mixing and symmetric synthesis do not prevent cross-cluster attribution, and neither a finite human audit nor deterministic scrubbing proves absence of leakage.

Ethics Statement

SkillLadder is designed to reduce privacy risk when procedural agent knowledge is shared without sharing raw traces. The framework targets the failure mode where naive trace reuse can leak credentials, user records, file paths, or proprietary workflows: only level- ≥ 1 skills cross a sharing boundary, and entity scrubbing plus lineage sanitization are applied before distribution. The default scrubbers cover regex-detectable identifiers and Presidio NER spans; the architecture exposes pluggable slots so deployments can add stronger PII detectors or differential privacy when their threat model demands it. Residual exposure is measured (Table 1) rather than assumed absent.

We position these mechanisms as practical data minimization and data loss prevention for agentic workflows, not as an anonymization claim or a formal privacy guarantee. Entity scrubbing, lineage sanitization, and source-set mixing are mechanisms that support these goals by reducing exposed identifiers, trace-specific details, and source linkability. SkillLadder distills operational procedures while measuring how much trace-specific content remains (ER) and how much source linkage persists (SA). Whether this is sufficient in a deployment depends on the chosen scrubbers, organizational release policy, and the applicable governance requirements.

All experiments use τ^2 -bench, a publicly available simulated benchmark with synthetic user personas, databases, and APIs; no human subjects, real customer data, or private user records are involved. With publication, we will release the implementation code (including parsers and audit scripts), configurations, prompts, and exact task IDs/splits. We will not release raw traces, generated skill artifacts, or experimental result files. The closed-source Claude Sonnet 4.6 is accessed via the standard provider API under its commercial terms; the open-source Qwen3-8B and Qwen3-32B targets run locally as isolated simulated user/agent configurations.

A quality-gated skill layer that improves a frozen agent can also be misused to make a poorly aligned agent more capable. The same retrieval interface that injects helpful procedures could in principle inject undesirable ones if the upstream library were poisoned. We mitigate this within scope by requiring that synthesized skills pass non-degradation gates against an LLM judge, but we do not claim defenses against an adversarial skill provider, and we recommend that deployments treat the skill library as an audited artifact rather than as user-modifiable content.

References

- Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. 2026. [Evoskill: Automated skill discovery for multi-agent systems](#). *Preprint*, arXiv:2603.02766.
- Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. 2025. [\$\tau^2\$ -bench: Evaluating conversational agents in a dual-control environment](#). *Preprint*, arXiv:2506.07982.
- Data Privacy Stack and Microsoft. 2026. [Presidio: Data protection and de-identification sdk](#).
- Jinchao Hu, Meizhi Zhong, Kehai Chen, and Min Zhang. 2026. [Searchskill: Teaching llms to use search tools with evolving skill banks](#). *Preprint*, arXiv:2605.09038.
- Da Lei, Feng Xiao, Lu Li, and Yuzhan Liu. 2026. [Signalclaw: Llm-guided evolutionary synthesis of interpretable traffic signal control skills](#). *arXiv preprint arXiv:2604.05535*.
- Fangzhou Li, Pagkratos Tagkopoulos, and Ilias Tagkopoulos. 2026a. [Skillflow: Scalable and efficient agent skill retrieval system](#).
- Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, Chujun Tao, Binxu Li, Xuandong Zhao, Hejia Geng, Xiaojun Wu, Junwei Zhou, Xiaokun Chen, Hanwen Xing, Yubo Li, and 59 others. 2026b. [Skillsbench: Benchmarking how well agent skills work across diverse tasks](#). *Preprint*, arXiv:2602.12670.
- Yu Li, Rui Miao, Zhengling Qi, and Tian Lan. 2026c. [Arise: Agent reasoning with intrinsic skill evolution in hierarchical reinforcement learning](#). *Preprint*, arXiv:2603.16060.
- Bochao Liu, Zhipeng Qian, Yang Zhao, Xin Jiang, Zihan Liang, Yufei Ma, Jun Zhuang, Ben Chen, Shuo Yang, Hong Wan, Yaoliang Wu, Chenyi Lei, and Xiaoyang Liang. 2026a. [Bian que: An agentic framework with flexible skill arrangement for online system operations](#).
- Xingyan Liu, Xiyue Luo, Linyu Li, Gang Huang, Jianfeng Liu, and Hongli Qiao. 2026b. [Skillforge: Forging domain-specific, self-evolving agent skills in cloud technical support](#).
- Ziyu Ma, Shidong Yang, Yuxiang Ji, Xucong Wang, Yong Wang, Yiming Hu, Tongwen Huang, and Xiangxiang Chu. 2026. [Skillclaw: Let skills evolve collectively with agentic evolver](#). *arXiv preprint arXiv:2604.08377*.
- Qirui Mi, Zhijian Ma, Mengyue Yang, Haoxuan Li, Yisen Wang, Haifeng Zhang, and Jun Wang. 2026. [Skill-pro: Learning reusable skills from experience via non-parametric ppo for llm agents](#). *Preprint*, arXiv:2602.01869.
- Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjuan Jiang. 2026. [Trace2skill: Distill trajectory-local lessons into transferable agent skills](#). *Preprint*, arXiv:2603.25158.
- Siru Ouyang, Jun Yan, Yanfei Chen, Rujun Han, Zifeng Wang, Bhavana Dalvi Mishra, Rui Meng, Chun-Liang Li, Yizhu Jiao, Kaiwen Zha, Maohao Shen, Vishy Tirumalashetty, George Lee, Jiawei Han, Tomas Pfister, and Chen-Yu Lee. 2026. [Skillos: Learning skill curation for self-evolving agents](#). *Preprint*, arXiv:2605.06614.
- Tianfei Ren, Zhipeng Yan, Yiming Zhao, Zhen Fang, Yu Zeng, Guohui Zhang, Hang Xu, Xiaoxiao Ma, Shiting Huang, Ke Xu, Wenxuan Huang, Lionel Z. Wang, Lin Chen, Zehui Chen, Jie Huang, and Feng Zhao. 2026. [Scope: Structured decomposition and conditional skill orchestration for complex image generation](#). *Preprint*, arXiv:2605.08043.
- Shuaike Shen, Wenduo Cheng, Mingqian Ma, Alistair Turcan, Martin Jinye Zhang, and Jian Ma. 2026. [Skillfoundry: Building self-evolving agent skill libraries from heterogeneous scientific resources](#). *Preprint*, arXiv:2604.03964.
- Yaorui Shi, Yuxin Chen, Zhengxi Lu, Yuchun Miao, Shugui Liu, Qi GU, Xunliang Cai, Xiang Wang, and An Zhang. 2026. [Skill1: Unified evolution of skill-augmented agents via reinforcement learning](#). *Preprint*, arXiv:2605.06130.
- Weihang Su, Jianming Long, Qingyao Ai, Yichen Tang, Changyue Wang, Yiteng Tu, and Yiqun Liu. 2026. [Skill retrieval augmentation for agentic ai](#). *Preprint*, arXiv:2604.24594.
- Zhen Tao, Riwei Lai, Chenyun Yu, Weixin Chen, Li Chen, Beibei Kong, Lei Cheng, Chengxiang Zhuo, Zang Li, and Qingqiang Sun. 2026. [Sager: Self-evolving user policy skills for recommendation agent](#). *Preprint*, arXiv:2604.14972.
- Yu Tian, Jiawei Chen, Lifan Zheng, Mingxiang Tao, Xinyi Zeng, Zhaoxia Yin, Hang Su, and Xian Sun. 2026. [Skills-coach: A self-evolving skill optimizer via training-free grpo](#). *Preprint*, arXiv:2604.27488.

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

Jianing Wang, Linsen Guo, Zhengyu Chen, Qi Guo, Hongyu Zang, Wenjie Shi, Haoxiang Ma, Xiangyu Xi, Xiaoyu Li, Wei Wang, and Xunliang Cai. 2026b. [Heavyskill: Heavy thinking as the inner skill in agentic harness](#). *Preprint*, arXiv:2605.02396.

Junjie Wang, Yiming Ren, and Haoyang Zhang. 2026c. [From procedural skills to strategy genes: Towards experience-driven test-time evolution](#). *Preprint*, arXiv:2604.15097.

Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2021. [Minilmv2: Multi-head self-attention relation distillation for compressing pre-trained transformers](#). *Preprint*, arXiv:2012.15828.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2025. [Agent workflow memory](#). In *Forty-second International Conference on Machine Learning*.

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. 2026. [Skillrl: Evolving agents via recursive skill-augmented reinforcement learning](#). *Preprint*, arXiv:2602.08234.

Feng Xiong, Zengbin Wang, Yong Wang, Xuecai Hu, Jinghan He, Liang Lin, Yuan Liu, and Xiangxiang Chu. 2026. [Ace-skill: Bootstrapping multimodal agents with prioritized and clustered evolution](#).

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. [Qwen3 technical report](#). *Preprint*, arXiv:2505.09388.

Aimin Zhang, Jiajing Guo, Fuwei Jia, Chen Lv, Boyu Wang, and Fangzheng Li. 2026a. [Evoagent: An evolvable agent framework with skill learning and multi-agent delegation](#). *Preprint*, arXiv:2604.20133.

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. 2026b. [Coevoskills: Self-evolving agent skills via co-evolutionary verification](#). *Preprint*, arXiv:2604.01687.

Chenyu Zhou, Huacan Chai, Wenteng Chen, Zihan Guo, Rong Shan, Yuanyi Song, Tianyi Xu, Yingxuan Yang, Aofan Yu, Weiming Zhang, Congming Zheng, Jiachen Zhu, Zeyu Zheng, Zhuosheng Zhang, Xingyu Lou, Changwang Zhang, Zhihui Fu, Jun

Table 7: Five-gate stack applied to each synthesized skill. Gates are applied in order; up to two retries are allowed before fallback to the cluster members.

#	Gate	Description
0	Score-gap demotion	Members scoring $< \alpha$ (0.3) times the best member are marked [CONTEXT-ONLY].
1	Consistency	Verify that the child preserves the strategy, generalizes it, and remains actionable.
2	Minimum quality	Reject candidates below $\theta_{\min}=1.0$ under the cross-task score.
3	Non-degradation	Require the child score to match or exceed its strongest parent.
4	Critical-step preservation	Reject candidates that drop concrete procedural steps.

Table 8: Threat-to-evidence map. Claims concern empirical data minimization and release-risk reduction, not anonymity or a formal privacy guarantee.

#	Threat	Mechanism	Evidence / claim status
T1	Content leakage	Entity scrubbing	Directly evaluated by ER
T2	Source attribution	Level gate + lineage sanitization	Directly evaluated by SA
T3	Membership inference	Source-set mixing	Structural reduction; no guarantee
T4	Skill inversion	Abstraction	Residual risk; not directly evaluated
T5	Cross-skill correlation	Pluggable release controls	Residual risk
T6	Style fingerprinting	Lineage sanitization + mixing	SA proxy

Wang, Weiwen Liu, and 2 others. 2026a. [Externalization in llm agents: A unified review of memory, skills, protocols and harness engineering](#). *Preprint*, arXiv:2604.08224.

Huichi Zhou, Siyuan Guo, Anjie Liu, Zhongwei Yu, Ziqin Gong, Bowen Zhao, Zhixun Chen, Menglong Zhang, Yihang Chen, Jinsong Li, Runyu Yang, Qiangbin Liu, Xinlei Yu, Jianmin Zhou, Na Wang, Chunyang Sun, and Jun Wang. 2026b. [Memento-skills: Let agents design agents](#). *Preprint*, arXiv:2603.18743.

A Algorithm Pseudocode

Algorithm 1 gives one full Collect–Extract–Evolve–Distribute cycle (§2.3).

B Method Reference Tables

Tables 7 and 8 summarize the gate stack and the threat-to-evidence mapping used by the four-phase workflow.

C Comparison with Skill-Evolution Frameworks

SkillLadder is not the first system for generating or evolving reusable agent skills. Table 9 compares representative systems by their stated primary mechanisms and evaluation axes. The distinguishing scope of SkillLadder is the joint treatment of hierarchical cross-contributor evolution, raw-trace

Algorithm 1 SkillLadder: One Evolution Cycle

Require: Users $\{U_1, \dots, U_K\}$ (each with its own agent) with new sessions; rounds R ; min cluster size $\tau_{\min}$; score threshold $\theta_{\min}$; score-gap ratio α ; max retries M
Ensure: Evolved skill library $\mathcal{L}$

```
1: // Phase 1: Collect
2: for each user  $k = 1, \dots, K$  do
3:    $\mathcal{D}_k \leftarrow$  new session traces from  $U_k$ 
4: end for
5: // Phase 2: Extract inside the contributor-controlled boundary
6: for each user  $k = 1, \dots, K$  do
7:    $\mathcal{S}_k^{(0)} \leftarrow \bigcup_{d \in \mathcal{D}_k} \text{EXTRACT}(d, \text{analyst}(d))$  ▷ dual-mode level-0
8:    $\mathcal{S}_k^{(0)} \leftarrow \text{ENTITYSCRUB}(\mathcal{S}_k^{(0)})$  ▷ data-minimize before trusted evolution
9: end for
10:  $\mathcal{S}^{(0)} \leftarrow \text{CHOOSE SOURCES}(\{\mathcal{S}_k^{(0)}\}_k)$ ;  $\mathcal{T} \leftarrow$  task descriptions from chosen source sessions
11: // Phase 3: Evolve at the trusted coordinator
12: for round  $r = 1, \dots, R$  do
13:    $\{\mathcal{C}_1, \dots, \mathcal{C}_J\} \leftarrow \text{HYBRIDCLUSTER}(\mathcal{S}^{(r-1)})$  ▷ Step 1: cluster
14:    $\mathcal{S}^{(r)} \leftarrow \emptyset$ 
15:   for each cluster  $\mathcal{C}_j$  with  $|\mathcal{C}_j| \geq \tau_{\min}$  do
16:      $s^* \leftarrow \arg \max_{s \in \mathcal{C}_j} \text{SCORE}(s, \mathcal{T})$  ▷ best parent
17:     Demote members with score  $< \alpha \cdot \text{SCORE}(s^*, \mathcal{T})$  to context-only ▷ Gate 0
18:      $s' \leftarrow \text{LLM-SYNTHESIZE}(\mathcal{C}_j)$  ▷ Step 2: merge & generalize
19:     if  $\neg \text{CONSISTENCYCHECK}(s', \mathcal{C}_j)$  or  $\text{SCORE}(s', \mathcal{T}) < \theta_{\min}$  then
20:       Retry up to  $M$  times; if all fail:  $\mathcal{S}^{(r)} \leftarrow \mathcal{S}^{(r)} \cup \mathcal{C}_j$ ; continue
21:     end if
22:     if  $\text{SCORE}(s', \mathcal{T}) < \text{SCORE}(s^*, \mathcal{T})$  then
23:        $\mathcal{S}^{(r)} \leftarrow \mathcal{S}^{(r)} \cup \mathcal{C}_j$  ▷ non-degradation: keep originals
24:     else if  $\neg \text{STEPPRESERVED}(s', s^*)$  then
25:       Retry up to  $M$  times; if all fail:  $\mathcal{S}^{(r)} \leftarrow \mathcal{S}^{(r)} \cup \mathcal{C}_j$  ▷ Gate 4
26:     else
27:        $s'.\ell \leftarrow r$ ;  $\mathcal{S}^{(r)} \leftarrow \mathcal{S}^{(r)} \cup \{s'\}$ 
28:     end if
29:   end for
30:    $\mathcal{S}^{(r)} \leftarrow \mathcal{S}^{(r)} \cup \{s \in \mathcal{S}^{(r-1)} : s \text{ in singleton cluster}\}$ 
31:   if no new evolved skills produced then
32:     break ▷ safe-abstraction stopping point
33:   end if
34: end for
35: // Phase 4: Distribute across the peer-user release boundary
36: for each skill  $s \in \bigcup_r \mathcal{S}^{(r)}$  with  $s.\ell \geq 1$  do
37:    $s \leftarrow \text{ENTITYSCRUB}(s)$ ;  $s \leftarrow \text{SANITIZELINEAGE}(s)$ 
38: end for
39: Distribute scrubbed level- $\geq 1$  skills to the target user(s)
40: return  $\mathcal{L} \leftarrow \bigcup_r \mathcal{S}^{(r)}$ 
```

locality, explicit lineage and fallback, peer-release controls, and leakage measurement; this scope is complementary to utility-oriented generation, refinement, retrieval, and policy learning.

D Privacy Analysis Details

Consistent with §2.6, we consider an honest-but-curious adversary who observes the shared skill library $\mathcal{L}$ after scrubbing and lineage sanitization and attempts either to recover sensitive entities from source traces or to attribute a shared skill to one of several candidate source pools. The adversary can inspect skill text and run auxiliary LLM analyses, but does not control upstream skill creation or see raw traces. Active poisoning, collusion with hold-

ers of raw logs, and deliberate multi-step inversion are outside this default model.

Source-attribution audit protocol. The SA metric instantiates the attribution part of this adversary as a five-way identification game. Each candidate source pool is represented by one source-task trace in the audit. For each evolution level, we sample up to 30 skills with non-empty lineage using a fixed random seed. For a sampled skill, we recover its ancestral source-task set from parent links, choose one true source trace from that set, and sample four distractor traces from outside the ancestral set. The candidate order is randomized and trace identifiers are hidden; each trace is truncated to 2000 characters to keep prompt length bounded.

Table 9: Representative skill-evolution frameworks. “Not central” means that the cited work does not foreground cross-user release leakage as a primary method and evaluation axis; it does not imply absence of general safety practices.

Framework	Evolution mechanism	Hierarchy / lineage	Verification signal	Cross-user release boundary
SkillLadder	Cross-contributor clustering and L0–L3 synthesis	Explicit parent DAG and fallback	Child vs. strongest-parent gates; execution reward	Raw traces local; L0 internal; sanitized L1–L3 peer release; ER/SA audits
SkillClaw (Ma et al., 2026)	Collective refinement and extension from interactions	Shared repository; lineage not primary	Recurring interaction and failure patterns	Cross-user repository; release leakage not a primary evaluation axis
SkillX (Wang et al., 2026a)	Automated construction and iterative refinement	Strategic, functional, and atomic levels	Skill validation and transfer tests	Not central
SkillForge (Liu et al., 2026b)	Failure diagnosis and domain-skill rewriting	Not a cross-user abstraction ladder	Failure and response-consistency checks	Not central
SkillIOS / SkillRL (Ouyang et al., 2026; Xia et al., 2026)	Continual or RL-based skill-library evolution	Higher-level or hierarchical skill banks	Task reward and retrieval feedback	Not central
CoEvoSkills (Zhang et al., 2026b)	Iterative multi-file skill refinement	Not a cross-user abstraction ladder	Co-evolving surrogate verifier	Not central

The LLM adversary receives the sanitized skill text and the five candidate traces, then returns a single 1-indexed choice and confidence. SA is the fraction of trials where the chosen candidate is the true source; the random baseline is therefore $1/5 = 0.2$. Invalid JSON outputs are counted as incorrect with zero confidence. This protocol evaluates residual linkability under a bounded, release-side adversary; it does not model collusion with raw-log holders or high-effort reconstruction from external side information.

Source-set mixing. The level- ≥ 1 sharing boundary provides a structural risk-reduction property through source mixing. This is not classical k -anonymity over known quasi-identifiers; instead, it lower-bounds the number of source sessions aggregated into each shared skill and upper-bounds any single source’s contribution weight.

Proposition 1 (Source-set k -mixing). *Let $\tau_{\min} \geq 2$ be the minimum cluster size and let μ_ℓ denote the number of distinct source sessions whose content is aggregated into a shared level- ℓ skill. Define the contribution weight of source session i in a level- ℓ skill as the fraction $1/\mu_\ell$ of its aggregated source-set membership, i.e., a uniform-counting share over the aggregated sessions; this is a combinatorial share, not a bound on textual information copied by the synthesizer. Then every shared skill ($\ell \geq 1$) satisfies $\mu_\ell \geq \tau_{\min}^\ell$, and the contribution weight of any single source session is at most $\tau_{\min}^{-\ell}$.*

Proof. By induction. At level 0, $\mu_0 = 1$. At round

r , each synthesized skill is produced from a cluster with at least $\tau_{\min}$ members, each carrying mixing degree at least μ_{r-1} . Thus $\mu_r \geq \tau_{\min} \mu_{r-1} = \tau_{\min}^r$. The per-source weight bound follows because no source can contribute more than one unit of the aggregated source set, giving weight at most $1/\mu_\ell \leq \tau_{\min}^{-\ell}$. $\square$

Deterministic scrubbing for covered patterns.

Let $\mathcal{C}$ denote the scrubber-covered pattern set: regex families for file paths, URLs, IP addresses, API keys, emails, and usernames; the NER tag set; and deployment blocklists. For a source trace t , let $\text{Ent}_{\mathcal{C}}(t)$ be substrings matched by $\mathcal{C}$, and let $\text{Tok}(s)$ be the literal token sequence of a shared skill s .

Property 1 (Deterministic scrubbing for covered patterns). *For every source trace t and every shared skill s that crosses the sharing boundary,*

$$\text{Ent}_{\mathcal{C}}(t) \cap \text{Tok}(s) = \emptyset \quad (8)$$

after the configured scrubber is applied.

This is a coverage-limited design invariant rather than an end-to-end privacy guarantee: the scrubber is a hard deterministic filter applied immediately before distribution, so any token matching $\mathcal{C}$ is replaced by a typed placeholder before the skill can be shared. The substantive question is the *coverage* of $\mathcal{C}$, which is empirical and measured by $\text{ER}(\ell)$ in §3.3.

Symmetric synthesis. A raw LLM can have position bias, so SkillLadder randomly permutes cluster members before synthesis. Let $\text{Synth} : \mathcal{T}^m \rightarrow$

$\Delta(S)$ be the randomized synthesis map. We call synthesis symmetric if, for every permutation $\sigma \in S_m$, $\text{Synth}(t_{\sigma(1)}, \dots, t_{\sigma(m)}) \stackrel{d}{=} \text{Synth}(t_1, \dots, t_m)$. Uniform random permutation enforces this property in distribution.

Theorem 1 (Intra-cluster source indistinguishability). *Assume symmetric synthesis. For any adversary $\mathcal{A} : \mathcal{S} \rightarrow \{i, j\}$ trying to decide whether a level- ℓ skill originated with member i or j of the same ancestral cluster,*

$$\Pr[\mathcal{A}(s) = i] = \Pr[\mathcal{A}(s) = j]. \quad (9)$$

Proof. Swapping members i and j is a permutation of the input cluster. Symmetric synthesis gives the same output distribution after the swap, while the labels i and j exchange roles. Therefore any adversary has equal probability of choosing either member. $\square$

Residual-risk metrics. The formal properties above are limited to scrubber-matched entities and within-cluster source symmetry. Pattern-class-external entities, stylistic fingerprints, and cross-cluster attribution remain residual risks and are measured by entity retention rate $\text{ER}(\ell)$ and source attribution accuracy $\text{SA}(\ell)$ in §3.3. Deployments requiring worst-case differential privacy can add a DP post-processing layer after deterministic scrubbing, at the cost of possible skill-text degradation.

E Human Entity and Hard-Negative Attribution Audits

Human entity audit. The audit stratifies 199 full trace/skill documents across domains and evolution levels. Annotators review all 650 detector candidates and then inspect complete documents without candidate cues to find omissions. Table 10 distinguishes detector accuracy from retention in released skills. False positives are operationally mitigated by typed replacement rather than deleting entire procedures, but false negatives remain bounded by detector coverage and motivate the complete-document pass.

Hard-negative attribution audit. The original SA protocol samples unrelated distractors and can conflate intended topical fidelity with source-specific linkability. The sensitivity audit therefore evaluates all 29 L3 skills with up to five paired lineups per skill and same-domain, shared-tool, same-task-type, embedding-nearest distractors. Table 11 reports skill-bootstrap intervals. The

Table 10: Human-adjudicated entity audit. Released artifacts comprise L1–L3 skills; no raw traces or L0 artifacts are peer-distributed.

Audit quantity	Result
Stratified documents	199
Detector candidates	650
Human-confirmed candidates	646
False-positive candidates	4
Candidate-free missed spans	2
Human-gold source-entity opportunities	648
Retained in released L1–L3	5 (0.77%)
L3 retention, each of four domains	0.0

Table 11: L3 source-attribution sensitivity. Normalized advantage is $(\text{SA} - 1/m)/(1 - 1/m)$ for m candidates.

Protocol	Skills	Chance	SA (95% CI)	Norm. adv.
Original five-way	28	0.20	0.464 [0.30, 0.64]	—
Hard-negative five-way	29	0.20	0.286 [0.210, 0.369]	0.108
Hard-negative ten-way	29	0.10	0.167 [0.112, 0.228]	0.075

paired five-to-ten-way difference is -0.119 (95% CI $[-0.179, -0.062]$). Because the audit changes both judge family and distractor construction relative to the original protocol, the magnitude difference should not be interpreted as a controlled judge effect.

F Clustering Strategy Comparison

Clustering in Algorithm 1 is a swappable component: any partition of $\mathcal{S}^{(r-1)}$ is a valid input to the synthesis step. SkillLadder implements five interchangeable clustering strategies; this appendix benchmarks all of them on the airline level-0 pool ($N=265$). Let $C \in [3, 15]$ denote the number of categories discovered (when applicable).

Strategies. (i) *LLM one-shot*: a single LLM prompt returns one cluster label per skill. (ii) *Embedding-based*: MiniLM-L6-v2 sentence embeddings followed by agglomerative clustering at cosine threshold δ , with $C_{\max}=8$; we sweep $\delta \in \{0.4, 0.6, 0.8\}$ (smaller δ is stricter). (iii) *Open tag-based*: an LLM discovers categories and assigns each skill in one call; oversized categories are split by sequential chunking at $C_{\max}=8$. (iv) *Hybrid* (default): an LLM discovers C open categories using short per-skill summaries; skill-to-category assignment is delegated to local MiniLM embedding similarity; categories with $|C| > C_{\max}$ are sub-divided by agglomerative clustering at $\delta=0.5$. (v) *Reverse-hybrid*: the order of stages (ii) and the LLM call is swapped—embedding clustering produces seeds, then the LLM merges seeds

into named categories.

Cost scaling. LLM one-shot and tag-based clustering must emit one assignment label per skill, so their output grows as $O(N)$. Hybrid removes this output dependency: only the category-discovery output remains, bounded by $O(C)$, while input scales as $O(N)$ over short skill summaries. At $N=265$, this yields roughly a $4\times$ estimated clustering-token reduction relative to LLM-only clustering. Beyond a single context window, LLM-only strategies also require chunk-and-merge logic that hybrid avoids by local embedding assignment. Embedding-only clustering has no LLM calls but produces no named categories, which affects downstream auditing and human inspection rather than evolution correctness.

Results. Table 12 reports per-strategy structural metrics and the mean judge-rated generalizability score (Equation 6) on the airline level-0 pool. All strategies produce viable evolved skills, with acceptance rates of 0.86–1.00 and mean scores in the 3.3–6.7 band; the choice of clustering strategy primarily affects the *number* and *size* of clusters reaching synthesis rather than whether synthesis succeeds. Within the embedding sweep, looser thresholds ($\delta=0.6, 0.8$) merge more aggressively and lift mean score from 3.30 to 4.00, but all three embedding rows remain below the LLM-aware tag and hybrid clusterers, consistent with the fact that pure structural similarity offers no semantic guidance to the synthesizer. We adopt hybrid as the default on cost-scaling grounds (above); deployments without lifelong scale or with strict reproducibility requirements may prefer embedding-only (no LLM cost, deterministic given a fixed model) or tag-based (highest mean score on this pool, but $O(N)$ output cost and shares an inductive bias with the synthesis judge, which limits direct quality comparisons).

G Quality-Gate Diagnostics

This appendix provides the per-round process audit and the leave-one-gate-out ablation summarized in §3.4.

How app and rel are computed for Eq. 6. For every candidate skill s and evaluation task $\xi \in \mathcal{T}$, the same judge used by Gates 2–3 returns two scores on a 0–10 scale: applicability $\text{app}(s, \xi)$ (can this skill be applied to this task) and relevance $\text{rel}(s, \xi)$ (how useful/appropriate it is if ap-

Strategy	#Cl	#Multi	Avg sz	Acc	Score
LLM one-shot	80	50	3.33	0.86	3.46
Embedding ($\delta=0.4$)	97	60	2.73	0.92	3.30
Embedding ($\delta=0.6$)	57	51	4.65	0.98	3.92
Embedding ($\delta=0.8$)	52	51	5.10	0.98	4.00
Tag-based	34	33	7.79	1.00	6.67
Hybrid (default)	81	54	3.27	0.94	3.87
Reverse-Hybrid	79	65	3.35	0.94	3.49

Table 12: Clustering ablation on airline level-0 pool ($N=265$). #Cl: total clusters; #Multi: clusters with $|\mathcal{C}| \geq 2$ (those entering synthesis); Avg sz: average cluster size; Acc: synthesis acceptance rate after the quality gates of Table 7; Score: mean generalizability score (Eq. 6) over accepted evolved skills. Embedding rows sweep the agglomerative-clustering distance threshold δ ; larger δ merges more aggressively, yielding fewer, larger clusters.

Table 13: Evolution process audit over saved artifacts from airline, banking, retail, and telecom. A candidate is a cluster with $|\mathcal{C}_j| \geq \tau_{\min}$. Preserved candidates are candidate clusters that did not yield a newly synthesized accepted skill, so their original members were carried forward. Gate-level retry causes were not logged in these artifacts.

Round	Clust.	Cand.	Accepted	Pres. cand.	Acc. rate	$\bar{s}_{p,\max} \rightarrow \bar{s}_e$	Gap dem.
R1	658	510	418	92	82.0%	0.00→6.78	0/0
R2	340	196	114	82	58.2%	5.93→7.58	73/145
R3	238	124	53	71	42.7%	5.88→7.16	39/93
All	1,236	830	585	245	70.5%	1.69→6.97	112/238

plied). We then compute Eq. 6 in three deterministic steps: (i) define the applicable subset $\mathcal{T}_{\text{app}}(s) = \{\xi \in \mathcal{T} : \text{app}(s, \xi) \geq 3\}$; (ii) compute $\text{rel}_{\text{app} \geq 3} = \frac{1}{|\mathcal{T}_{\text{app}}(s)|} \sum_{\xi \in \mathcal{T}_{\text{app}}(s)} \text{rel}(s, \xi)$; (iii) set $\text{score}(s; \mathcal{T}) = \frac{|\mathcal{T}_{\text{app}}(s)|}{|\mathcal{T}|} \cdot \text{rel}_{\text{app} \geq 3}$. If $|\mathcal{T}_{\text{app}}(s)| = 0$, we define $\text{score}(s; \mathcal{T}) = 0$. This makes the score a coverage term (fraction of tasks where $\text{app} \geq 3$) times a quality term (mean relevance on that covered subset).

Process audit across rounds. Table 13 audits how the five synthesis quality gates behave across rounds using saved evolution artifacts from all four domains: how many candidate clusters reach synthesis, what fraction the gates accept, and how the accepted skills score relative to their strongest parent.

Across 830 candidates, SkillLadder accepts 585 syntheses and preserves originals for 245; acceptance falls from 82% (R1) to 43% (R3) as clusters become more abstract, yet accepted skills still beat the strongest parent on average in R2 (5.93 → 7.58) and R3 (5.88 → 7.16). Score-gap demotion flags 112 later-round clusters (238 par-

Table 14: Leave-one-gate-out ablation on airline R1 (51 clusters; Claude Sonnet 4.6 synthesizer). Turning a gate off only *relaxes* that gate; the others still apply. R1 parents are L0 skills with $s_{p,\max}=0$, so $\bar{s}_e = \text{margin}$ by construction; the $\geq 0.5/\geq 1.0$ margin floors equal the score floors enforced by Gate 2. Bold marks the full configuration.

Condition	Acc. rate	Mean $\bar{s}_e$	Median margin	Δ score
full (all gates on)	0.88	4.77	4.3	—
no_consistency (G1 off)	0.88	4.66	4.2	−0.11
no_nondegrad (G3 off)	0.94	4.43	3.9	−0.34
no_critical (G4 off)	0.92	4.39	3.7	−0.38

ent skills) as context-only. This is a process audit, not a gate-by-gate ablation; exact retry causes require instrumented logs. Restricting to the 122 R2/R3 acceptances whose strongest-parent score is locally resolvable from the saved artifacts, the non-degradation gate is strictly enforced (100% have margin $\bar{s}_e - \bar{s}_{p,\max} \geq 0$), and 35.2% / 21.3% of acceptances beat the strongest parent by $\geq 0.5 / \geq 1.0$ respectively (median margin 0.14, $p_{90}=1.73$).

Leave-one-gate-out ablation. To isolate the contribution of individual gates, we re-run the round-1 synthesis pass on the airline cycle (51 candidate clusters, $\tau_{\min}=2$) with each discriminative gate turned off in turn, holding all other components (clusterer, LLM, retry budget) fixed. Table 14 reports acceptance rate and mean judge score under four conditions: *full* (all gates on); *no_consistency* (G1 off); *no_nondegrad* (G3 off); and *no_critical* (G4 off).

Every discriminative gate improves the judge-rated accepted set: relaxing any single gate raises acceptance but lowers the mean judge score. Turning G3 (non-degradation) off lifts acceptance from 88% to 94% but drops mean score by 0.34; turning G4 (critical-step preservation) off lifts acceptance to 92% and yields the largest score reduction (−0.38). G1 (consistency check) has the smallest effect (−0.11) but remains measurable: a few cluster-level inconsistencies still slip through when it is removed. We ablate the three discriminative gates only: Gate 0 is a member-level demotion rather than an accept/reject decision, and removing Gate 2 ($\theta_{\min}=1.0$) reduces to a trivial “accept-all” setting that we report separately as the baseline floor. Two clusters were rejected in all four conditions, indicating that the score floor (Gate 2) is the binding constraint when no synthesizer pass produces a usable abstraction.

Table 15: Independent child–parent rescoring. “No worse” counts margins ≥ 0 .

Auditor	Levels	Pairs	No worse	Mean margin
DeepSeek-v3.2	L2/L3	122	121 (99.2%)	+0.380
GLM-5.2	L2/L3	122	117 (95.9%)	+0.699
DeepSeek-v3.2	L1/L2	511	506 (99.0%)	+0.714
GLM-5.2	L1/L2	511	507 (99.2%)	+0.717

Table 16: DeepSeek-v3.2 R2/R3 synthesis and gate replication over 320 candidate clusters.

Setting	Accepted	Non-degrading	Margin (mean / median)
Full gate	175/320 (54.7%)	100.0%	+2.589 / +1.350
No consistency	181/320 (56.6%)	100.0%	+2.686 / +1.280
No non-degradation	235/320 (73.4%)	70.2%	+1.411 / +0.310
No critical-item	177/320 (55.3%)	100.0%	+2.595 / +1.280

Independent non-Claude rescoring. Table 15 reloads accepted artifacts and independently compares every evolved child with its strongest parent under the same task-conditioned rubric. DeepSeek-v3.2 and GLM-5.2 differ in absolute margins but reproduce the no-worse trend, including all 29 L3 pairs.

Full non-Claude synthesis and gate replication. DeepSeek-v3.2 replaces Claude for both synthesis and gate scoring on all 320 R2/R3 candidate clusters across four domains. Table 16 holds clustering, prompts, task-conditioned rubric, and retry budget fixed across 1,280 condition evaluations. Removing Gate 3 raises acceptance but admits degrading children, whereas removing consistency or critical-step checks leaves the accepted set non-degrading under the remaining score gate. This isolates the tested role of Gate 3 without claiming that every model family or prompt formulation will behave identically.

H Per-task Retrieval

Sanitized skills are written into each user’s library in agent-native format (e.g. structured SKILL.md files). Because injecting the full library is expensive—164 evolved skills total roughly 300K tokens—SkillLadder retrieves a compact subset per task. The default *static reinjection* policy scores each non-retired skill against the task description by token overlap, runtime quality, and a level bonus, and injects the top- k (default 10) into the system prompt for the entire episode:

$$\text{SCORE}_{\text{static}}(s, \xi) = |T(\xi) \cap T(s)| + q(s) + 0.1 \ell(s), \quad (10)$$

where $T(\cdot)$ is the lower-cased token set extracted by the regex `[a-z][a-z0-9_-]{3,}` from the task

description (for $T(\xi)$) and from the concatenated skill fields (for $T(s)$). The runtime quality estimate $q(s)$ combines the EMA of continuous reward,

$$\bar{r}_t(s) = (1 - \alpha) \bar{r}_{t-1}(s) + \alpha r_t, \quad \alpha = 0.35, \quad (11)$$

with the thresholded success rate $\eta(s)$ of Eq. 4:

$$q(s) = 0.7 \bar{r}(s) + 0.3 \eta(s), \quad (12)$$

seeded from the pre-deployment judge score so newly minted skills are not penalized for lack of evidence. The EMA $\bar{r}$ tracks continuous-reward trends so partial credit is retained, while η acts as a Bernoulli prior that dampens single-session noise. The dynamic librarian policy (Appendix I) delegates per-turn retrieval to a hidden subagent. A skill is *retired*—and excluded from both retrievers—once it has been used in at least $N_{\min}=5$ sessions and $q(s)$ falls below $q_{\text{ret}}=0.15$. SkillLadder also supports a *daemon mode*: a background loop monitors new sessions, triggers evolution, and re-synchronizes updated skills over time.

B1 (raw-trace concat) baseline. B1 in Table 4 is the privacy upper bound that mirrors B1 in Tables 1–2: it injects the same kind of past-session content that a naive trace-pooling system would expose, but goes through the SkillLadder static retriever rather than a different ranker so that any reward gap relative to SkillLadder reflects the abstraction/scrub pipeline, not retrieval choice. Concretely, for each evaluated task ξ :

- (i) **Corpus.** All historical trace files from the same domain, excluding ξ ’s own trace. Each trace is a JSON record of $\text{agent} \leftrightarrow \text{user} \leftrightarrow \text{tool}$ turns and is concatenated into a flat text string.
- (ii) **Segmentation.** Each trace string is chunked into $\sim 1,500$ -token segments (matching the average length of an evolved skill in Table 3) so that the segment-level corpus is comparable in granularity to the SkillLadder skill library. Trace metadata (task_id , user_id , timestamps) is retained verbatim—no scrubbing, no abstraction, no lineage sanitization.
- (iii) **Retriever.** The static scoring function of Eq. 10 is reused with $\ell(s)=0$ (raw segments have no level) and $q(s)=0$ (no runtime quality estimate is available for raw segments), reducing to pure token-overlap ranking. The top- $k=10$ segments are concatenated and prepended to the system prompt, identical to the Full-static injection slot.

- (iv) **Privacy accounting.** Because (ii) preserves trace metadata, the released artifact contains every covered entity in the source traces by construction, so B1 inherits the ER=1.0, SA=1.0 row of Tables 1–2; reward is the only free variable.

What B1 is and is not. B1 is a controlled *utility* measurement for "what if we shipped raw traces through the same retrieval slot." It is not a re-implementation of any published trace-pooling system, and it does not attempt the centralized indexing, summarization, or per-skill verifiers that papers such as SkillClaw or AWM perform on top of raw logs. It exists so that the (reward, ER, SA) trade-off in Table 2 can be read against a real reward number rather than an imagined upper bound.

I Dynamic Librarian Implementation

The dynamic-librarian injection mode (§2.4) is implemented as a hidden two-stage subagent that runs before every agent response. Algorithm 2 summarizes one consultation; the underlying prompts are reproduced verbatim below.

Stage 1: query decision prompt. The query LLM is called at temperature 0 with the system prompt below and a user prompt of the form “Task context: $\{\xi\}$ Recent conversation/tool context: $\{h\}$ Should the agent consult the SkillLadder skill library before the next response?”

You are the hidden SkillLadder librarian controller for a customer-service agent. Decide whether the next agent response needs a skill-library lookup.

Return valid JSON only, with this schema:

```
{
  "need_skills": true or false,
  "query": "short search query if
  need_skills is true, otherwise empty
  string",
  "reason": "brief reason"
}
```

Set $\text{need_skills}=\text{true}$ for complex or high-risk situations such as policy eligibility, cancellation, refund, compensation, upgrade, reservation changes, irreversible write actions, user claims that need verification, multi-reservation retrieval, cost calculation, or uncertainty about the next step. Set $\text{need_skills}=\text{false}$ for simple greetings, acknowledgements, or purely clerical follow-up questions.

If the JSON call fails, the librarian falls back to a small keyword set (cancel, refund, policy,

Algorithm 2 Dynamic librarian consultation

Require: task description ξ , recent dialogue h , skill library $\mathcal{L}$, max rounds R_{lib} , candidate count k_c , top- k to return k_s
1: $d \leftarrow \text{QUERYDECISION}(\xi, h)$ ▷ JSON: {need_skills, query, reason}
2: **if** $\neg d.\text{need_skills}$ **or** $d.\text{query} = \emptyset$ **then**
3: **return** $\emptyset$
4: **end if**
5: $q \leftarrow d.\text{query}$
6: **for** $r = 1, \dots, R_{\text{lib}}$ **do**
7: $C \leftarrow \text{TOKENOVERLAPTOPK}(q, \mathcal{L}, k_c)$ ▷ Eq. 10
8: $j \leftarrow \text{JUDGECANDIDATES}(q, C, h)$ ▷ JSON: {sufficient, selected_skill_ids, refined_query, reason}
9: $S \leftarrow \{c \in C : c.\text{id} \in j.\text{selected_skill_ids}\}$
10: **if** $j.\text{sufficient}$ **or** $j.\text{refined_query} \in \{\emptyset, q\}$ **then**
11: **break**
12: **end if**
13: $q \leftarrow j.\text{refined_query}$
14: **end for**
15: **return** $S[:k_s]$ injected as a transient `<skillladder_retrieved_skills>` system message

reservation, eligib, ...) over the recent context.

Stage 2: judge prompt. Given the issued query and k_c token-overlap candidates retrieved with the same scorer used for static reinjection (Eq. 10), a second LLM call chooses which subset to return and whether to retry with a refined query (up to $R_{\text{lib}}=2$ rounds).

You are the hidden SkillLadder librarian quality judge. You receive the main agent’s skill query and retrieved candidate skills. Decide whether the candidates satisfy the query, then select only the useful skills to return to the main agent.

Return valid JSON only, with this schema:
{
 “sufficient”: true or false,
 “selected_skill_ids”: [“skill id”,
 “...”],
 “selected_skill_names”: [“skill name”,
 “...”],
 “refined_query”: “better query if
 sufficient is false, otherwise empty
 string”,
 “reason”: “brief reason”
}

Select at most the few skills that are directly useful. Reject generic, duplicated, or off-topic candidates. If the candidates do not cover the need, set sufficient=false and provide a refined query.

Injection. The selected skills are injected as a transient system message wrapped in a `<skillladder_retrieved_skills>` block; the agent itself only sees the librarian’s output, never the controller/judge prompts. The agent’s system prompt instructs it to treat retrieved skills as optional procedural hints, to defer to domain policy on conflict, and not to mention the internal librarian to the user.

J Cost and Efficiency Audit

Evolution-stage cost. For an incremental window with M new trace/L0 artifacts, at most C candidate clusters per round, and R abstraction rounds, the dominant evolution-stage LLM-call count scales as $O(M + RC)$; token volume additionally depends on artifact length and the gate-evaluation set size T . A DeepSeek-v3.2 audit of 10 round-stratified airline R2/R3 candidates, each scored against 10 task descriptions, uses 30,236 total tokens per candidate on average (median 28,897; interquartile range 23,513–40,170), including retries. The score gate accounts for 52.0% of these tokens and synthesis for 34.3%, identifying smaller/local judges, cascaded gates, and smaller T as direct scaling levers. Entity scrubbing and lineage sanitization are local text/metadata operations and add no per-release LLM call; the human audit is an offline evaluation, not part of the deployment path.

Online retrieval cost. We report per-task token usage and wall-clock from the airline-50 Sonnet runs (same agent/user models, identical infrastructure, sequential execution). “First prompt” is the prompt-token count of the agent’s first turn, which absorbs the entire injected skill block under static policies. “Avg in / out” are summed over all agent turns in a task.

The two injection policies trade off prompt mass against wall-clock. Static reinjection roughly triples the first-prompt size (5.6K $\rightarrow$ 14.2K tokens) and more than doubles total prompt usage (67.5K $\rightarrow$ 156.4K) because the full top-10 skill block enters the system prompt once and is re-resent every turn. L0-only static is much cheaper on the prompt side (+41% total in) because L0 skills

Table 17: Per-task cost on airline-50 (Sonnet agent + Sonnet user simulator). Tokens are summed over agent turns; wall-clock is per task, single-concurrency.

Condition	Turns	First-prompt	Avg in	Avg out	Wall (s)
Baseline (no skills)	8.9	5,578	67,503	1,406	190.8
L0-only static	9.3	7,864	95,053	1,547	195.3
Full static	9.3	14,240	156,353	1,724	196.7
Dynamic librarian	9.0	8,566	89,411	1,507	392.2

Table 18: Default hyperparameters used in all main-paper experiments. Symbols match the ones introduced in §2.

Component	Symbol / Name	Value
<i>Skill extraction (§2.2)</i>		
Extractor temperature	—	0.4
Extraction mode	—	dual (success / failure)
<i>Clustering (§2.3)</i>		
Embedding cosine threshold	—	0.60
Discover-cluster temperature	—	0.30
Min cluster size	$\tau_{\min}$	2
<i>Synthesis & quality gates (§2.3)</i>		
Score gate (judge)	$\theta_{\min}$	1.0
Score-gap demotion ratio	α_{gap}	0.30
Max evolve retries	$\bar{M}$	2
Evolve temperature	—	0.50
Consistency / score / step-preservation temp	—	0.20
Evolution rounds	R	3
<i>Runtime quality (§2.2, Eq. 11, Eq. 12)</i>		
EMA coefficient	α	0.35
EMA / η mix in $q(s)$	—	$0.7F + 0.3\eta$
Positive-session threshold	—	$r_t > 0.5$
Retire quality threshold	—	0.15
Min injections before retire	—	5
<i>Static retrieval (§2.4)</i>		
Top- k injected	k	10
Level bonus weight	—	0.1ℓ
Min token overlap	—	1
<i>Dynamic librarian (App. 1)</i>		
Candidates per round	k_c	8
Returned per turn	k_s	3
Max refinement rounds	R_{lib}	2
Controller / judge temperature	—	0.0
<i>LLMs</i>		
Synthesizer / judge	—	Claude Sonnet 4.6
Target agent (same-model)	—	Claude Sonnet 4.6
Target agent (simulated user)	—	Qwen3-8B, Qwen3-32B
User simulator	—	Claude Sonnet 4.6

are shorter than evolved skills. Dynamic librarian keeps the first-prompt close to baseline (+54%) by retrieving only when needed, but roughly doubles wall-clock (191s $\rightarrow$ 392s) because every agent turn first triggers two hidden LLM calls (query decision + judge). Completion-token counts and turn counts stay within 10% of baseline in all conditions, so the injected skills do not noticeably change response length or interaction depth. None of these overheads include the one-off extraction and evolution cost, which is amortized across all subsequent task evaluations.

K Hyperparameter Settings

Table 18 lists every hyperparameter used in the main experiments. The same defaults are applied across all four domains (airline, banking, retail, telecom); no per-domain tuning was performed.

Tuned vs. fixed. Only the two parameters that combine the EMA reward and the thresholded success rate in $q(s)$ were tuned: the mixing weight $\{0.5, 0.7, 0.9\}$ and the EMA coefficient $\alpha \in \{0.10, 0.35, 0.50\}$, swept on a held-out subset of 10 airline sessions, and selected to minimise the rank-correlation gap between $q(s)$ and the realised end-of-episode reward. All other entries in Table 18 were fixed by design intent ($\tau_{\min}=2$ is the smallest non-trivial cluster; $R=3$ matches the depth at which the L3 skill count plateaus in our pilots; the score-gap ratio 0.3 and retire threshold 0.15 are conservative defaults that demote outliers without aggressive pruning) or carried over from preliminary pilots, and were not searched.

L Extended Table-6 Interpretation

This appendix consolidates details omitted from §3.4 for readability and should be read with Table 4.

Per-domain skill-level effects. For same-simulated-user reuse (Sonnet), L0-only gains are +0.060/ + 0.052/ − 0.038/ + 0.125 on airline/banking/retail/telecom, while L1+ gains are +0.079/ + 0.043/ − 0.090/ + 0.174. For Qwen3-8B, L1+ gains are +0.243/ + 0.104/ + 0.253/ + 0.211 and closely track Full-static +0.244/ + 0.108/ + 0.253/ + 0.211, whereas L0-only is weaker (−0.043/ + 0.018/ + 0.036/ + 0.101). For Qwen3-32B, L0-only matches or exceeds Full-static on all four domains (+0.122/ + 0.037/ + 0.090/ + 0.092 vs. +0.118/ + 0.020/ + 0.086/ + 0.084), while Full-dynamic recovers gains on banking and retail (+0.085, +0.133).

Release-policy interpretation of Full rows. Under the deployment policy, only scrubbed level- ≥ 1 skills are distributed; this policy-faithful effect is reported in the “L1+ only” row. “Full static” and “Full dynamic” retrieve over the entire synchronized library for within-paper comparability and should be read as upper bounds on the deployment setting when nonzero L0 appears in Mix. On Sonnet, L1+ is within 0.002 of Full-static on all four domains, so the same-simulated-user conclusion is unchanged by the release policy.

Sample sizes for paired contrasts. All same-row paired contrasts in Table 4 use $n=50/97/114/200$ tasks on airline/banking/retail/telecom, respectively.

M Skill Evolution Case Study

To illustrate how SkillLadder progressively abstracts task-specific knowledge into reusable skills, we trace a strict $L0 \rightarrow L1 \rightarrow L2 \rightarrow L3$ lineage from the saved telecom artifacts ($\tau^2_{\text{telecom_remaining200_max50/skills_from_train}}$). In the displayed path, each child is synthesized only from parents at the immediately preceding level.

Level 0: Task-specific extraction. Two level-0 skills are extracted from individual mobile-data troubleshooting traces:

- `exhaust-all-identified-issues-before-retesting` (id 32df93d1): “Prevents repeated failed speed tests and user frustration by identifying ALL potential issues upfront and fixing multiple problems before running a verification test.”
- `address-all-identified-issues-before-retesting` (id 58e161bb): “Prevents repeated failed speed tests by ensuring ALL identified issues are resolved before asking the user to retest, rather than fixing one issue at a time and retesting after each.”

Both L0 skills are local lessons from concrete telecom traces: diagnose multiple blockers such as airplane mode, VPN, roaming, data limits, or network mode, fix them as a batch, then test.

Level 1: Cross-trace rule. Round 1 synthesizes those two L0 skills into `batch-fix-all-issues-before-verification` (id 3452289c, `parent_ids=[32df93d1,58e161bb]`):

“Prevents repeated failed verification attempts and user frustration by identifying ALL potential issues upfront through comprehensive diagnostics, resolving them in batch before running any verification test, rather than fixing one issue at a time with a test after each.”

The L1 skill abstracts away the specific mobile-data cases while preserving the reusable rule: run all diagnostics, compile all anomalies, resolve every known issue, and verify only after the batch is complete.

Level 2: Multi-issue troubleshooting procedure. Round 2 synthesizes that L1 skill with other level-1 variants into `batch-all-fixes-before-verification` (id 99891bfa, including parent id 3452289c):

“When troubleshooting a problem with multiple concurrent root causes, identify ALL issues through comprehensive upfront diagnostics, apply all fixes in a prioritized sequence, and only then run a single verification test.”

The L2 procedure adds ordering and scope: categorize complete blockers, partial blockers, and optimization concerns; fix account/billing blockers before device or application settings; and re-diagnose comprehensively if the final verification fails.

Level 3: General layered-system workflow. Round 3 synthesizes two level-2 batch-fix skills into `batch-all-fixes-before-verification` (id 0dfcecbf, `parent_ids=[99891bfa,3d92a855]`):

“Prevents repeated failed verification tests, wasted effort, and user/operator frustration by comprehensively diagnosing ALL issues upfront across every relevant layer, resolving every one of them in dependency order, confirming clean state, and only then running a single validation test.”

Although this L3 skill shares its surface name with its L2 parent (99891bfa), the two records are distinct artifacts—different ids, different parent sets, and different scope language—because the synthesizer reused the parent’s short slug when the abstraction step only widened the procedure’s applicable domain rather than rewriting its surface form. The L3 skill generalizes beyond mobile data to layered systems more broadly: device/client settings, network state, account or billing status, configuration, resource usage, permissions, active services, logs, and external dependencies. It expresses the final abstraction as a reusable `diagnose-all  $\rightarrow$  fix-all  $\rightarrow$  confirm-clean  $\rightarrow$  verify-once` workflow.

Abstraction pattern. This artifact-backed chain illustrates the general pattern: level-0 skills capture trace-specific procedures, level-1 skills capture recurring procedures across similar traces, level-2 skills turn the rule into an ordered troubleshooting procedure, and level-3 skills generalize it into a domain-agnostic workflow. The quality gates ensure that each level preserves concrete procedural steps from its parents while removing task-specific bindings.

N Retail Failure Case Studies

To make the aggregate retail numbers (§3.4, Table 5) concrete, we trace two of the six tasks that pass under the no-skill baseline but fail under static

Table 19: Concrete lineage from trace-specific telecom lessons to a domain-general layered-system workflow. Parent IDs are internal provenance and are removed before peer release.

Level	Artifact ID	Parent relation	Semantic change	Visibility
L0	32df93d1, 58e161bb	Two separate trace-derived artifacts	Identify all mobile-data blockers, batch fixes, then retest	Trusted evolution only
↓ L1	3452289c	Parents: both L0 artifacts	Abstract two traces into a recurring batch-fix-before-verification rule	Peer-release eligible after gates/scrubbing
↓ L2	99891bfa	Includes L1 3452289c and peer L1 variants	Add dependency ordering and multi-issue troubleshooting scope	Peer-release eligible after gates/scrubbing
↓ L3	0dfcecbf	Parents: L2 99891bfa, 3d92a855	Generalize to diagnose-all → fix-all → confirm-clean → verify-once across layered systems	Peer-release eligible after gates/scrubbing

reinjection, to identify the specific injected skill responsible.

Case 1 (task 66): over-rationalized cancellation reason. The expected gold action is a single call,

```
cancel_pending_order(
  order_id="#W3361211",
  reason="no longer needed").
```

Under no-skill baseline the agent emits this verbatim and the task passes. Under static reinjection, the agent calls the same API but writes `reason="ordered by mistake"`, so the resulting database state diverges from the gold and the task fails. The top-10 injected block for this task contains an L1 skill `pivot-order-action-by-status-and-intent` (id 627e6e0c) whose procedure instructs the agent to “acknowledge the change of preference... identify the best alternative action that achieves the customer’s underlying goal.” The user utterance for task 66 contains light hedging that this skill treats as an “intent shift,” prompting the agent to substitute what it judges a more plausible reason. The API and order id are correct; only the discrete reason enum is wrong.

Case 2 (task 71): wrong-API substitution from a sequencing heuristic. The expected gold action is

```
modify_pending_order_items(
  item_ids=["2492465580"],
  new_item_ids=["5917587651"],
  order_id="#W5270061",
  payment_method_id="paypal_7729105").
```

The baseline executes this. Under static reinjection the agent instead first calls `modify_pending_order_address(...)` on the same order, never reaches the items modification, and fails on both the action-matching and DB-state components.

The injected block contains the L1 skill `sequential-order-modification-address-before-items` (id 186e5204), explicitly recommending an address-first sequencing whenever a pending order is being modified. Task 71 never asks for an address change, so the skill triggers an unnecessary and counter-productive call.

What these cases imply for static reinjection.

Both cases share the same failure mode: a generic, well-formed L1 skill injects a strategy that is locally plausible but task-irrelevant, and the agent treats it as instruction rather than guidance. They align with the aggregate Table 5 pattern: of the behavioral failures under static reinjection (12.4% of 114 tasks), six are on tasks that the baseline passes, and the two examined here trace to a single injected skill. These cases suggest two corrective directions for retail: (i) the static scorer should down-weight skills whose pre-conditions are not satisfied by the current task description; and (ii) per-turn retrieval avoids most of this interference—dynamic librarian’s behavioral-failure rate (6.9%) is on par with the baseline (7.8%). This motivates dynamic mode as the retail policy in our final configuration.