Skill²: Fully Agentic Co-Evolution of Policy and Skill Library

Anonymous authors
Paper under double-blind review

Abstract

Skills have become the native carrier of procedural knowledge in deployed agents. However, most skill learning is misaligned with deployment in three respects, the artifact it produces, the pathway through which skills are invoked, and the tasks on which it is validated. The few *fully agentic* efforts evolve only one side, training the policy against a fixed skill library or evolving the skill library under a frozen model. Yet a skill matures on neither side alone. We present Skill², a fully agentic co-evolution framework in which a policy and its skill library advance together by alternating two separately trained roles around one skill library. The actor learns whether, when, and which skill to invoke through the native harness with Fork-GRPO, which forks a control branch at each skill decision so that credit lands on the tokens that made it. The extractor learns to write and maintain skills with counterfactual-utility policy optimization, which credits edits with the paired gains the current actor obtains from the revised skill library and from invoking the skills they shaped. Extensive experiments on seven real long-horizon agentic benchmarks demonstrate that Skill² is ❶ *effective*, surpassing baselines with relative gains of up to 46.7% on terminal tasks and 59.8% on web tasks, ❷ *scale-consistent*, with its average gain growing from Qwen3.5-4B to Qwen3.5-9B, and ❸ *transferable*, lifting four larger frozen models from three families by up to 13.0% through the evolved skill library alone. Our code is available here.

1 Introduction

Large language model (LLM) agents are moving beyond conversational assistants toward systems that autonomously complete long-horizon tasks in real environments, driving rapid progress from software engineering to scientific discovery (Wang et al., 2024b; Xi et al., 2025). Reliable work, however, hinges on procedural knowledge of how to get things done in a particular context, and an agent that retains and reuses it need not start from scratch each time (Zhang et al.). The unit that carries this knowledge is the *skill*, a reusable procedural artifact that packages workflow knowledge with the resources it requires, to be discovered, loaded, and followed by the agent when a task calls for it (Zhou et al., 2026). Unlike a tool, which provides a callable capability, a skill specifies when and how such capabilities should be used (Jiang et al., 2026b). Skills have become a standard layer of modern agent systems (Agent Skills), and *skill learning* has emerged as an active direction that lets agents acquire skills from their own experience, consolidate them into a reusable *skill library*, and learn to use them in tasks (Xia et al., 2026a; Wang et al., 2026; Zhang et al., 2026b). Together, these efforts point toward an agent that improves itself through its skills.

In deployment, a skill is defined by its form, its invocation pathway, and the tasks that validate it. Most existing research is misaligned with deployment on all three (Figure 1, top), leaving its skill learning short of *fully agentic*. ❶ **Artifact Misalignment.** The artifact is mostly text injected into the context (Xia et al., 2026a; Yang et al., 2026a), rather than the directory artifact, a `SKILL.md` file with scripts and resources, that a deployed agent loads (Zhang et al.). ❷ **Invocation Misalignment.** Invocation is decided by a hand-built workflow outside the action loop (He et al., 2026a; Ouyang et al., 2026; Xia et al., 2026b), rather than by the agent itself while acting inside a skill-native harness that exposes only skill names and descriptions (Wang et al., 2025a). ❸ **Validation Misalignment.** Validation takes place in text simulators such as ALFWorld and WebShop (Shi et al., 2026; Zhang et al., 2026d), rather than on the real long-horizon agentic tasks whose multi-step, tool-intensive

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

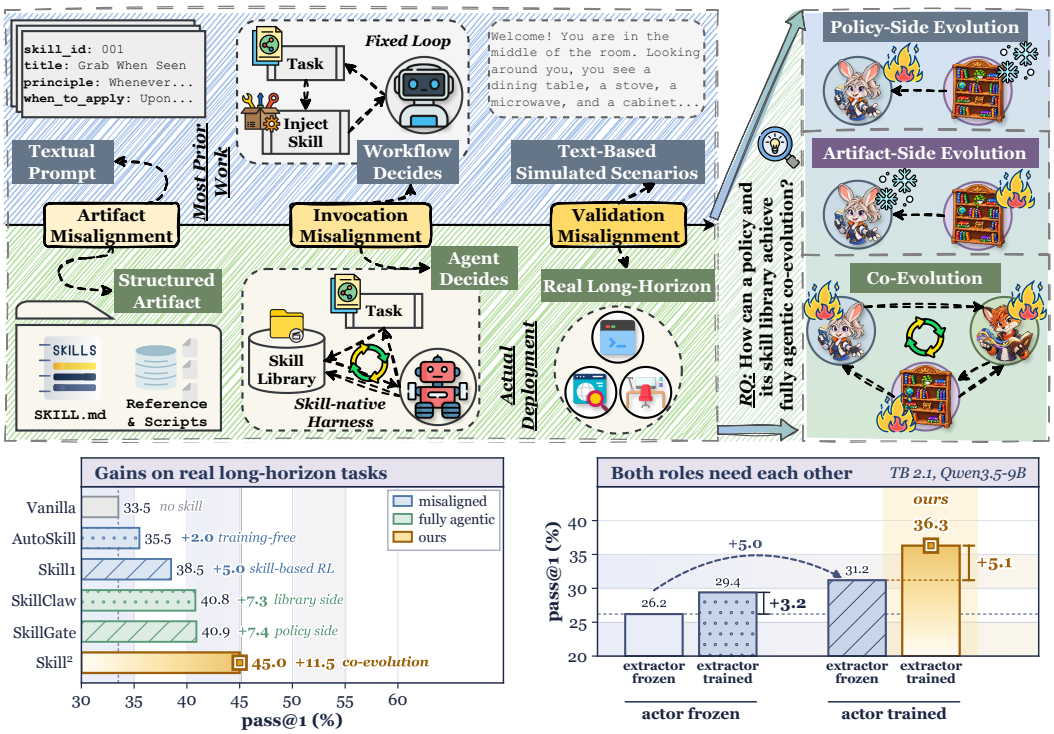


Figure 1: **Motivation of Skill².** (***Top***) Three misalignments of existing skill learning with deployment. (***Bottom left***) Gains of representative methods on seven real long-horizon benchmarks with Qwen3.5-9B. (***Bottom right***) Each role gains more when the other is also trained.

operations most need distilling into reusable procedures (Li et al., 2026b; Merrill et al., 2026). Given these misalignments, two questions arise. ♣ *Does existing research lack fully agentic skill learning altogether?* Not entirely. A few recent efforts do bring skill learning back to a fully agentic setting, either training the policy to use existing skills inside a native harness (Tan et al., 2026; Li et al., 2026a) or evolving the skill library with a frozen model under the same harness (Zhang et al., 2026a; Ma et al., 2026). ♠ *Do these works let the policy and the library advance together?* Unfortunately not. In both cases, evolution remains confined to one side (Table 3), and the cost shows on real long-horizon tasks, where co-evolution gains 11.5 points while misaligned skill methods gain at most 6.1 and one-sided fully agentic ones at most 7.4 (Table 1 and Figure 1, bottom left).

This gap is expected, since skills do not mature on one side alone. In cognitive science, skill acquisition is a dual process in which competence first forms implicitly through practice, and explicit knowledge extracted from that practice in turn speeds it up, the two developing best in interaction (Berry & Broadbent, 1984; Sun et al., 2005). An agent mirrors this split, with the policy holding implicit competence and the skill library the explicit knowledge distilled from it. Because a library changes only through the model that writes it, letting the library side learn means training that writer alongside the policy. A controlled comparison bears this out (Figure 1, bottom right). On Terminal-Bench 2.1, training the writer adds 3.2 points when the policy is frozen but 5.1 when the policy is also trained, and training the policy adds 5.0 with a frozen writer but 6.9 with a trained one. Training both is not straightforward, however, as the only reward is the task outcome, which does not reveal which skill decision or edit made the difference. This leads to our research question:

RQ: *How can a policy and its skill library achieve fully agentic co-evolution?*

To this end, we propose Skill², which achieves this by alternating two separately trained roles around one skill library. The **actor** is the policy that accomplishes tasks with skills. Trained with Fork-GRPO inside a skill-native harness on real long-horizon agentic tasks, it decides by itself whether, when, and which skill to invoke, with skills disclosed and loaded by the harness as in deployment. At every skill decision, Fork-GRPO forks a control branch that proceeds without the skill, so that the decision is compared against its alternative and credited on the tokens that made it.

The **extractor** is the policy that maintains the skill library, turning the actor's practice into new skills and keeping existing ones in step with a changing actor. Trained with counterfactual-utility policy optimization, it reads the actor's latest rollouts and edits the skill library into one candidate, which is rewarded by the paired gain the current actor obtains from it over the original on held-out probe tasks and credited at two levels, macro over the whole revision and micro over each skill. The **skill library**, the shared repository of skill artifacts, is the medium through which the two roles shape each other. As the two roles train in alternation, the skill library is written from the current actor's practice and valued against its use, while the actor trains on the version just released, so neither side adapts to a stale version of the other. Our contributions are summarized as follows:

- **Dilemma Identification.** We identify three misalignments between existing skill learning and real deployment, in artifact, invocation, and validation. The few fully agentic works evolve only one side, the policy or the skill library, leaving their co-evolution unaddressed.
- **Practical Solution.** We propose `Skill`^{2}, in which a policy and its skill library co-evolve in alternation in a fully agentic setting. The actor learns whether, when, and which skill to invoke through Fork-GRPO, which credits each skill decision against a control branch. The extractor learns to write and maintain skills through counterfactual-utility policy optimization, which credits edits with paired gains to the current actor at two levels.
- **Empirical Validation.** Extensive experiments on seven real long-horizon agentic benchmarks demonstrate that `Skill`^{2} is ❶ ***effective***, surpassing baselines with relative gains of up to 46.7% on terminal tasks and 59.8% on web tasks, ❷ ***scale-consistent***, with its average gain growing from Qwen3.5-4B to Qwen3.5-9B, and ❸ ***transferable***, lifting four larger frozen models from three families by up to 13.0% through its evolved skill library alone.

2 METHOD

As shown in Figure 2, `Skill`^{2} couples two separately parameterized models through one skill library inside a skill-native harness. The **actor** π_θ accomplishes tasks with skills, invoking them at its own discretion, and the **extractor** ρ_ϕ distills the actor's latest trajectories into the **skill library**, updating existing skills as the actor changes. We first introduce the notation and formalize the problem (§2.1), then present how the actor learns to use skills (§2.2) and how the extractor learns to write them (§2.3), and finally how the two roles alternate around the shared skill library (§2.4).

2.1 PRELIMINARY

Notation. We formalize an agent in a skill-native harness. A skill library is a set of structured artifacts $\mathcal{B} = \{\sigma_j\}_{j=1}^{|\mathcal{B}|}$, each skill $\sigma_j = (n_j, d_j, b_j)$ comprising a name, a description, and a body that may carry scripts and resources (Zhang et al.). Following the Agent Skills specification (Agent Skills), skills are disclosed progressively. The agent first sees only the metadata $c(\mathcal{B}) = \{(n_j, d_j)\}_j$, the names and descriptions, and a body enters the context only once invoked. For a task x, the harness places $c(\mathcal{B})$ in the context. At turn t the agent π_θ with parameters θ generates a token span $y_t \sim \pi_\theta(\cdot \mid h_t)$ from the history $h_t = (x, c(\mathcal{B}), a_{<t}, o_{<t})$, parsed by the harness into an action $a_t \in \mathcal{A} = \mathcal{A}_{\text{env}} \cup \{\text{invoke}(n) : n \in \text{names}(\mathcal{B})\}$, either an environment action from the harness's tool set $\mathcal{A}_{\text{env}}$ or a skill invocation by name. Throughout, $\text{invoke}(n)$ abstracts how the harness implements skill invocation. The observation o_t is the tool's return for an environment action and the skill body b_n for $\text{invoke}(n)$. The trajectory is $\tau = (a_1, o_1, \ldots, a_T, o_T)$.

Problem Formalization. `Skill`^{2} seeks a policy and a skill library that are jointly optimal:

$$\max_{\theta,\ \mathcal{B}} \ \mathbb{E}_{x\sim\mathcal{D}}\, \mathbb{E}_{\tau\sim\pi_\theta(\cdot|x,\mathcal{B})}\big[R(\tau)\big], \tag{1}$$

where $\mathcal{D}$ is the task distribution, $R(\tau)$ the task reward of a trajectory, and $\mathcal{B}$ ranges over skill libraries reachable from an initial $\mathcal{B}_0$ by editing. Existing fully agentic skill learning is a special case of Eq. 1, optimizing θ against a fixed $\mathcal{B}$ (Tan et al., 2026; Li et al., 2026a) or rewriting $\mathcal{B}$ with an untrained model while θ stays fixed (Zhang et al., 2026a; Ma et al., 2026). Since $\mathcal{B}$ changes only through edits, we optimize it through its editor. The extractor is a separately parameterized policy ρ_ϕ whose actions are edits of the skill library and whose reward is the gain those edits bring to the current actor. We then solve Eq. 1 in alternation. With $\mathcal{B}_k$ fixed, the actor updates θ by policy gradient (§2.2), and with θ_{k+1} fixed, the extractor ρ_ϕ rewrites $\mathcal{B}_k$ into $\mathcal{B}_{k+1}$ (§2.3).

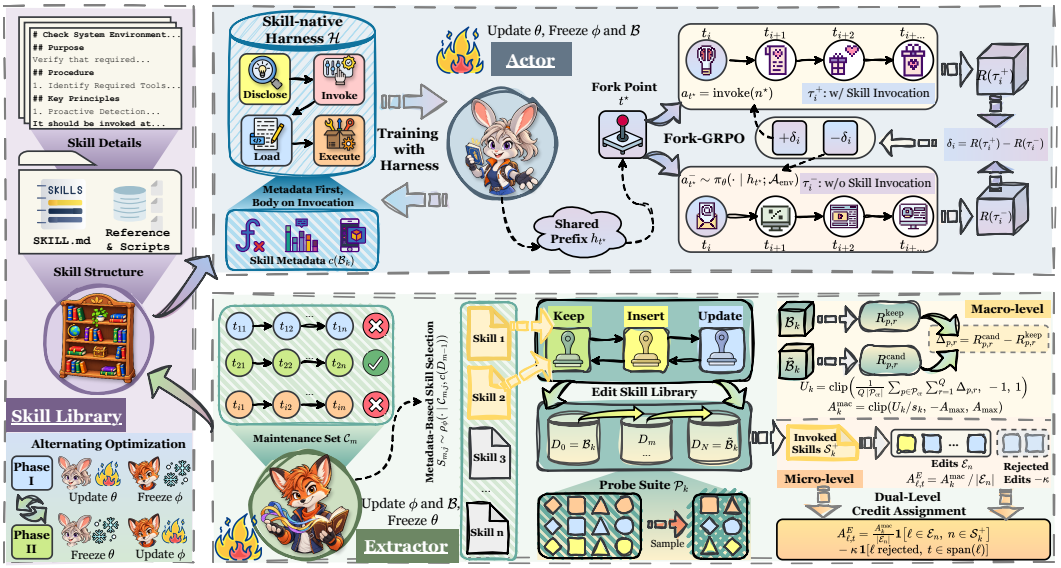


Figure 2: Overview of `Skill`^{2}.

2.2 Actor

The actor is the policy that carries out tasks with the aid of skills. Beyond solving the task, it learns whether, when, and which skill to invoke, a decision that the environment reward does not score directly. With a single outcome reward over thousands of tokens, the actor cannot tell whether a success came from the skill it invoked or would have happened anyway. We propose Fork-GRPO, which builds on GRPO (Shao et al., 2024) and branched agentic rollouts (Dong et al., 2025) and forks a control branch at each skill decision, so that the decision is compared against its alternative and credited on the tokens that made it. Throughout, $(\mathcal{B}_k, \phi_k)$ are held fixed.

Training with a Skill-Native Harness. The actor is trained inside a skill-native harness $\mathcal{H}$, where skills are disclosed, invoked, and loaded and actions are executed by the harness itself. An episode opens with the context $h_1 = (x,\, c(\mathcal{B}_k))$, in which the current skill library is *disclosed* only through its metadata. At turn t the actor generates $y_t \sim \pi_\theta(\cdot \mid h_t)$, and the harness parses it into an action a_t. If $a_t = \mathrm{invoke}(n)$, the actor *invokes* skill n, and the harness *loads* it and returns its body as the observation, $o_t = b_n$. If a_t is an environment action, the harness *executes* it and returns the outcome as the observation $o_t = \mathrm{Exec}_{\mathcal{H}}(a_t)$. The history advances to $h_{t+1} = h_t \oplus (a_t, o_t)$, and the actor proceeds with environment actions, following the loaded instructions and drawing on the bundled scripts and resources, deciding alone when to invoke which skill and when to proceed without one. Training through the deploy-time harness in this way aligns the interface the actor learns on with the one it meets in deployment (He et al., 2026b; Xu et al., 2026; Kim et al., 2026).

Decision-Point Forking. For each task x we sample a group $\mathcal{G}(x)$ of G rollouts under the current skill library, and every skill decision in the group is a fork point. Whenever a rollout τ^+ invokes a skill, $a_{t^\star} = \mathrm{invoke}(n^\star)$, we fork from the same state $h_{t^\star}$ a rollout τ^- that proceeds without the skill, taking an environment action, $a^-_{t^\star} \sim \pi_\theta(\cdot \mid h_{t^\star}; \mathcal{A}_{\mathrm{env}})$, and thereafter follows the same policy to the end of the episode. Forks fill the group alongside the rollouts they stem from, so its size stays G. Since τ^+ and τ^- share every token before $t^\star$ and follow the same policy after it, they differ only in the decision at $t^\star$. Their reward difference therefore estimates the effect of this one decision, both its timing and its choice of skill.

Credit Assignment. The paired comparison should reach only the tokens that made the decision and leave the credit of all other tokens intact. Following GRPO (Shao et al., 2024), each rollout in the group, forks included, first receives an advantage by normalizing its reward within the group, $\hat{A}_i = \frac{R_i - \mathrm{mean}_j R_j}{\mathrm{std}_j R_j + \epsilon}$, where ϵ is a numerical constant. A prefix shared by a rollout and its fork, however, should not answer for their diverging outcomes, so a token in the prefix shared by a lineage

$L \subseteq \mathcal{G}(x)$ receives the lineage mean while a token in a private suffix keeps its own advantage,

$$A_{i,t} = \begin{cases} \frac{1}{|L|} \sum_{j \in L} \hat{A}_j, & t \text{ in the prefix shared by } L, \\ \hat{A}_i, & \text{otherwise.} \end{cases} \tag{2}$$

Finally, for each pair we form the paired difference $\delta = R(\tau^+) - R(\tau^-)$ and write it onto the decision tokens $\mathrm{dec}(\tau)$ of turn $t^\star$ alone, that is, the tokens of the action taken at the fork point, including the skill name of an invocation. Writing $\mathcal{G}^+(x)$ for the rollouts that invoked a skill at their fork point, $\mathcal{G}^-(x)$ for their control branches, and δ_i for the paired difference of the pair of τ_i,

$$A_{i,t} = \begin{cases} +\delta_i, & t \in \mathrm{dec}(\tau_i),\ \tau_i \in \mathcal{G}^+(x), \\ -\delta_i, & t \in \mathrm{dec}(\tau_i),\ \tau_i \in \mathcal{G}^-(x), \\ \text{Eq. 2}, & \text{otherwise.} \end{cases} \tag{3}$$

Written onto the decision tokens in place of the group-relative value, δ_i leaves every other token trained by Eq. 2 and turns the gradient at the fork point into a paired comparison of the two decisions. A positive δ_i raises the probability of invoking $n^\star$ at $h_{t^\star}$ and lowers that of proceeding without it, and vice versa, so the actor learns whether, when, and which skill to invoke.

Training Objective. Let y_i denote the tokens generated by the actor in τ_i. With the token-level advantages of Eq. 3, the actor is updated with the clipped, KL-regularized objective of GRPO,

$$\mathcal{J}_A(\theta) = \mathbb{E}_{x, \mathcal{G}(x)} \Big[\frac{1}{G} \sum_{i=1}^{G} \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \mathcal{L}_{i,t}^{\mathrm{clip}} \Big] - \beta\, \mathbb{D}_{\mathrm{KL}} \big[\pi_\theta \,\|\, \pi_{\mathrm{ref}} \big], \tag{4}$$

$$\mathcal{L}_{i,t}^{\mathrm{clip}} = \min \big(r_{i,t} A_{i,t},\ \mathrm{clip}(r_{i,t}, 1 - \varepsilon_{\mathrm{low}}, 1 + \varepsilon_{\mathrm{high}})\, A_{i,t} \big), \tag{5}$$

where $r_{i,t}$ is the probability ratio of token t between the current and the sampling policy, π_{ref} the reference policy, and β the KL coefficient. The loss covers actor-generated tokens only, so skill metadata and skill bodies are treated as observations rather than as targets.

2.3 Extractor

The extractor is the policy that maintains the skill library. It turns the actor's practice into new skills and keeps existing ones in step with a changing actor, learning which edits help the actor in its current state. The value of an edit is not visible in the text of the edit itself. It shows only in how the actor behaves once the edited skill library is in place, together with the other edits of the round. We propose counterfactual-utility policy optimization, which rewards edits by the paired gain the current actor obtains from the revised skill library and assigns credit at two levels, macro over the whole revision and micro over each skill. Throughout, θ_{k+1} is held fixed.

Maintenance Evidence. The skill library is a set of skill artifacts that only the extractor revises, round by round. The extractor reads the actor's latest rollouts, organized into maintenance sets $\mathcal{C}_1, \ldots, \mathcal{C}_N$ that are processed in turn. Each set holds J complete same-task groups $\mathcal{C}_{m,j} = \big(x, \{(\tau_i, R_i, e_i)\}_{i=1}^{G}\big)$ with their rewards and bounded evidence e_i, such as tool interactions, outcome diagnostics, and how the episode ended. Successes and failures enter alike, and seeing all rollouts of one task side by side lets the extractor tell practices that hold across rollouts from those that are accidental. Each group is accompanied by rollouts of other tasks of the same type, so that what the extractor distills is shared across instances rather than tied to a single one. Like the actor, the extractor first sees the skill library through its metadata, which keeps its context short and lets it decide which skill to refine from the same names and descriptions that the actor chooses from.

Editing Operations. Each maintenance set yields one wave of J edits, one per task group, all made against the working draft D_{m-1} with $D_0 = \mathcal{B}_k$. For each group $\mathcal{C}_{m,j}$, the extractor first selects, from the metadata of the draft, at most M existing skills whose bodies it will read in full,

$$S_{m,j} \sim \rho_\phi\big(\cdot \mid \mathcal{C}_{m,j},\ c(D_{m-1})\big), \qquad S_{m,j} \subseteq \mathrm{names}(D_{m-1}),\ |S_{m,j}| \le M. \tag{6}$$

The selection is part of the editing decision, not a retrieval step, since only a selected skill may be updated, a new skill is inserted only when none of them covers its theme, and leaving the draft as it is

comes last. Given their bodies, the extractor writes one operation $o \in \mathcal{O}$, either KEEP, INSERT(σ), or UPDATE(n, σ'), and the J operations of a wave, sampled in parallel, are applied in order,

$$o_{m,j} \sim \rho_\phi\big(\cdot \mid \mathcal{C}_{m,j},\ c(D_{m-1}),\ \{b_n : n \in S_{m,j}\}\big), \qquad D_m = \mathrm{Apply}\big(D_{m-1},\ o_{m,1}, \ldots, o_{m,J}\big), \tag{7}$$

where Apply applies the operations in turn and resolves conflicts among them, such as two insertions under one name, and b_n is read from D_{m-1}. INSERT adds a complete skill $\sigma = (n, d, b)$, UPDATE replaces the fields of skill n by those given in σ', and KEEP leaves the draft as it is. The skill library has a fixed capacity, and an insertion at capacity retires the skill that the actor invoked least recently. Once every set is processed, the draft D_N becomes the single candidate $\tilde{\mathcal{B}}_k$, so that one candidate is evaluated per round and each wave edits the draft left by the waves before it.

Counterfactual Utility. The candidate is valued on a probe suite $\mathcal{P}_k$ of held-out tasks that is drawn anew in each round from a larger probe pool $\mathcal{P}$, disjoint from training and from every evaluation benchmark, by stratified sampling over task family and difficulty, so that the extractor is not rewarded on one fixed set of tasks. For each probe task $p \in \mathcal{P}_k$ and each of Q repetitions r, the current actor runs once under the base skill library and once under the candidate, the two runs differing only in the skill library. With $R^{\mathrm{keep}}_{p,r}$ and $R^{\mathrm{cand}}_{p,r}$ the rewards of the two runs, the paired difference is $\Delta_{p,r} = R^{\mathrm{cand}}_{p,r} - R^{\mathrm{keep}}_{p,r}$. The counterfactual utility of the candidate for the current actor is the clipped mean paired difference over the suite,

$$U_k = \mathrm{clip}\Big(\frac{1}{Q\,|\mathcal{P}_k|} \sum_{p \in \mathcal{P}_k} \sum_{r=1}^{Q} \Delta_{p,r},\ -1,\ 1\Big), \tag{8}$$

and no probe run enters any loss.

Dual-Level Credit Assignment. One candidate is probed per round, so U_k values the revision as a whole and says nothing about a skill it left unchanged, however often the actor invokes it. Credit is therefore assigned at two levels, each with its own intervention. ❶ *Macro credit* values the revision as the normalized counterfactual utility $A^{\mathrm{mac}}_k = \mathrm{clip}(U_k / s_k,\ -A_{\max},\ A_{\max})$, with s_k a running scale, shared equally by the edits $\mathcal{R}_k$ that reached the candidate, the very edits the pairing intervened on, including one whose skill is never invoked but shifts the metadata or retires another skill. ❷ *Micro credit* values each skill through the control branches of Fork-GRPO. In the actor phase of the same round, run under $\mathcal{B}_k$ before the candidate is written, every invocation of a skill n is paired with a branch that proceeds without it, and the mean $g_{k,n}$ of their paired differences δ_i is the gain of invoking the version of n in $\mathcal{B}_k$ over not invoking it. Every accepted edit waits in a credit buffer, keyed by the skill it shaped, until that skill is certified by pairs that disagree, $g_{k,n} \neq 0$, $n \in \mathcal{S}^+_k$, whereupon its waiting edits $\mathcal{E}_n$ leave the buffer as one unit sharing $A^{\mathrm{mic}}_{k,n}$, the gain normalized and bounded like A^{mac}_k. The edits of round k thus enter the buffer after the pairs of round k and can be certified from round $k+1$ on, an edit is credited, or blamed, through a counterfactual on the skill it wrote rather than on itself, and one never certified waits until it expires. An edit that fails a structural check, such as a malformed operation or a body whose scripts do not hold together, does not reach the skill library and takes a small penalty $-\kappa$ on the span responsible, $\mathrm{span}(\ell)$, with no bonus for passing. The two levels combine into the token-level advantage of the extractor, $A^E_{\ell,t} = A^{\mathrm{mac}}_k / |\mathcal{R}_k|$ for $\ell \in \mathcal{R}_k$, $A^E_{\ell,t} = A^{\mathrm{mic}}_{k,n} / |\mathcal{E}_n|$ for $\ell \in \mathcal{E}_n$ with $n \in \mathcal{S}^+_k$, $A^E_{\ell,t} = -\kappa$ for $t \in \mathrm{span}(\ell)$ of a rejected edit, and $A^E_{\ell,t} = 0$ otherwise. Macro credit thus ties the revision to the actor's outcome, including effects that pass through no invocation, and micro credit ties each skill to its own invocations. Both rest on paired outcomes and deterministic checks rather than on a judge model, so a skill earns credit only where invoking it pays off for the current actor.

Training Objective. Since credit can arrive rounds after an edit was sampled, an update in round k draws on the revision $\mathcal{R}_k$, the units released from the buffer in that round, and the round's rejected edits, collected as $\mathcal{U}_k$. An edit ℓ may thus stem from an earlier extractor ρ_{ϕ_ℓ} than the one being updated. Following decoupled policy optimization (Hilton et al., 2022; Fu et al., 2026), we separate the extractor that sampled an edit from the one before the update, $\rho_{\phi_{\mathrm{old}}}$, and correct the lag with a truncated importance weight (Espeholt et al., 2018), $w_{\ell,t} = \min\big(\rho_{\phi_{\mathrm{old}}}(y_{\ell,t}) / \rho_{\phi_\ell}(y_{\ell,t}),\ c\big)$, held fixed in the gradient, and edits older than a set number of updates are dropped. With y_ℓ the tokens of edit ℓ, the extractor is updated with the clipped, KL-regularized objective

$$\mathcal{J}_E(\phi) = \mathbb{E}\Big[\frac{1}{|\mathcal{U}_k|} \sum_{u \in \mathcal{U}_k} \sum_{\ell \in u} \frac{1}{|y_\ell|} \sum_{t=1}^{|y_\ell|} w_{\ell,t}\, \mathcal{L}^{\mathrm{clip}}_{\ell,t}\Big] - \beta_E\, \mathbb{D}_{\mathrm{KL}}\big[\rho_\phi \,\|\, \rho_{\mathrm{ref}}\big], \tag{9}$$

$$\mathcal{L}^{\text{clip}}_{\ell,t} = \min\left(r^E_{\ell,t}A^E_{\ell,t},\ \text{clip}(r^E_{\ell,t}, 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}})\, A^E_{\ell,t}\right), \tag{10}$$

where $r^E_{\ell,t}$ is the probability ratio of token t between the current extractor and $\rho_{\phi_{\text{old}}}$, ρ_{ref} the reference policy, and β_E the KL coefficient. Each unit counts once regardless of the number of edits it holds.

2.4 Alternating Co-Evolution

Unified View. The actor and the extractor are trained in alternation, meeting only through the skill library. In outer iteration k, the actor takes S_A steps of Fork-GRPO on the current version $\mathcal{B}_k$, and its rollouts on that version form the extractor's maintenance sets. The extractor then rewrites $\mathcal{B}_k$ into one candidate, values it by paired probes with the newly trained θ_{k+1}, updates on the resulting dual-level credit, and releases the candidate as $\mathcal{B}_{k+1}$ for the next iteration:

$$(\theta_k,\ \phi_k,\ \mathcal{B}_k) \xrightarrow[\mathcal{B}_k,\,\phi_k \text{ fixed}]{\S 2.2} (\theta_{k+1},\ \phi_k,\ \mathcal{B}_k) \xrightarrow[\theta_{k+1} \text{ fixed}]{\S 2.3} (\theta_{k+1},\ \phi_{k+1},\ \mathcal{B}_{k+1}). \tag{11}$$

The skill library is thus written from the current actor's practice and valued against the current actor's use, while the actor trains on the skill library just released, so neither side adapts to a stale version of the other. Algorithm 1 in Appendix B summarizes the procedure.

3 Experiments

In this section, we conduct extensive experiments to answer the following research questions ($\mathcal{RQ}$). ($\mathcal{RQ}$**1**) Is `Skill`^{2} effective on real long-horizon agentic tasks? ($\mathcal{RQ}$**2**) Do the gains of `Skill`^{2} hold across model scales? ($\mathcal{RQ}$**3**) Is the skill library evolved by `Skill`^{2} transferable to frozen models other than its actor? ($\mathcal{RQ}$**4**) What does the co-evolution of `Skill`^{2} look like over training?

3.1 Experimental Setup

Evaluation and Benchmarks. Our evaluation covers seven real long-horizon agentic benchmarks spanning two domains (Appendix D), ❶ *terminal* tasks on Terminal-Bench 2.1 (TB 2.1) (Merrill et al., 2026), OpenThoughts-TBLite (TBLite) (OpenThoughts-Agent team, 2026), TerminalWorld-Verified (TW-Verified) (Chu et al., 2026), and InterCode-Bash (IC-Bash) (Yang et al., 2023), and ❷ *web navigation* tasks on WebArena-Verified (WA-V) (hattami et al., 2025), MiniWoB++ (Liu et al., 2018), and WorkArena-L1 (WoA-L1) (Drouin et al., 2024). Each task runs in its live environment and is graded by the official verifier. We report pass@1 (%) averaged over 10 independent runs. Evaluation tasks are held out from training, from the probe pool, and from library updates.

Baselines. We compare `Skill`^{2} against nine baselines in five groups. ❶ **Vanilla** is the untrained backbone acting in the same harness without a skill library. ❷ **RL methods** train the policy on outcome rewards with no skill library, including GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025). ❸ **Training-free skill evolution** keeps the model frozen and relies on LLM-based reflection over its trajectories to induce and refine skills, including AutoSkill (Yang et al., 2026c) and SkillOpt (Yang et al., 2026b). ❹ **Skill-based RL** trains the policy together with a skill library distilled from its rollouts, including Skill1 (Shi et al., 2026) and SkillRL (Xia et al., 2026a). ❺ **Fully agentic methods** evolve one side under a deploy-time harness, including SkillClaw (Ma et al., 2026), which evolves the skill library with a prompted evolver for a frozen model, and SkillGate (Li et al., 2026a), which trains the policy to select skills from a fixed skill library, both run at the two scales of `Skill`^{2}.

Implementation Details. We instantiate `Skill`^{2} on Qwen3.5-4B and Qwen3.5-9B (Qwen Team, 2026). At each scale the actor and the extractor start from the same backbone as two separate sets of parameters and are trained with RL directly, so that the writer is no stronger than the actor it writes for. OpenHands (Wang et al., 2025a) serves as the skill-native harness $\mathcal{H}$ of all methods, while mini-SWE-agent (Yang et al., 2024b) and BrowserGym (Chezelles et al., 2025) execute the environment actions on terminal and web tasks, respectively. The actor trains for one epoch, with eight rollouts per task and at most 48 turns per episode, on the 4,567 verifiable tasks of SETA (Shen et al., 2026b) and on 4,000 tasks of WebShop-HTML, which runs WebShop (Yao et al., 2022) as real web pages in a browser rather than as the text simulator used in prior work. Probe suites for the counterfactual utility are drawn from separate pools of 1,536 terminal and 768 web tasks held out from training. Further settings and hyperparameters of both domains are given in Appendix E.

3.2 Main Results ($\mathcal{RQ}$1 & $\mathcal{RQ}$2)

Table 1 compares `Skill`^{2} with all baselines. Obs. ❶–❷ answer $\mathcal{RQ}1$ and Obs. ❸ answers $\mathcal{RQ}2$.

Table 1: Performance comparison on the terminal and web navigation benchmarks with Qwen3.5-4B and Qwen3.5-9B. Each cell reports pass@1 (%) averaged over 10 runs and its change relative to the vanilla backbone of the same scale, with gains marked by ↑ and drops by ↓. Avg. is the mean over the seven benchmarks. The best results are in **bold** and the second best are underlined.

Method	Terminal				Web Navigation			Avg.
	TB 2.1	TBLite	TW-Verified	IC-Bash	WA-V	MiniWoB++	WoA-L1	
Qwen3.5-4B								
Vanilla	22.3	31.9	18.3	33.2	11.2	48.6	29.7	27.9
GRPO	25.9↑3.6	38.7↑6.8	21.4↑3.1	37.0↑3.8	14.7↑3.5	56.8↑8.2	37.3↑7.6	33.1↑5.2
DAPO	25.9↑3.6	39.1↑7.2	22.2↑3.9	37.2↑4.0	15.5↑4.3	58.0↑9.4	37.9↑8.2	33.7↑5.8
AutoSkill	22.1↓0.2	34.2↑2.3	18.1↓0.2	34.9↑1.7	12.4↑1.2	52.5↑3.9	33.6↑3.9	29.7↑1.8
SkillOpt	22.7↑0.4	35.6↑3.7	18.3±0.0	34.4↑1.2	13.3↑2.1	53.3↑4.7	35.2↑5.5	30.4↑2.5
Skill1	23.4↑1.1	35.2↑3.3	19.3↑1.0	36.5↑3.3	14.6↑3.4	55.0↑6.4	36.1↑6.4	31.4↑3.5
SkillRL	24.4↑2.1	36.7↑4.8	20.4↑2.1	36.2↑3.0	14.9↑3.7	56.1↑7.5	36.4↑6.7	32.2↑4.3
SkillClaw	26.6↑4.3	40.7↑8.8	23.6↑5.3	36.1↑2.9	16.6↑5.4	54.6↑6.0	39.4↑9.7	33.9↑6.0
SkillGate	27.1↑4.8	41.3↑9.4	24.0↑5.7	35.9↑2.7	16.8↑5.6	54.5↑5.9	40.0↑10.3	34.2↑6.3
`Skill`^{2}	**29.2**↑6.9	**44.1**↑12.2	**26.1**↑7.8	**41.3**↑8.1	**17.9**↑6.7	**62.2**↑13.6	**41.5**↑11.8	**37.5**↑9.6
Qwen3.5-9B								
Vanilla	25.3	41.2	24.0	41.5	15.4	55.2	31.8	33.5
GRPO	29.1↑3.8	46.7↑5.5	29.3↑5.3	45.9↑4.4	20.9↑5.5	63.4↑8.2	41.8↑10.0	39.6↑6.1
DAPO	31.5↑6.2	47.2↑6.0	30.1↑6.1	45.7↑4.2	22.3↑6.9	64.1↑8.9	43.3↑11.5	40.6↑7.1
AutoSkill	25.9↑0.6	44.0↑2.8	24.6↑0.6	41.9↑0.4	16.9↑1.5	59.1↑3.9	35.8↑4.0	35.5↑2.0
SkillOpt	26.4↑1.1	43.6↑2.4	25.8↑1.8	42.7↑1.2	18.0↑2.6	59.8↑4.6	37.6↑5.8	36.3↑2.8
Skill1	28.4↑3.1	45.9↑4.7	26.5↑2.5	43.6↑2.1	21.0↑5.6	61.6↑6.4	42.4↑10.6	38.5↑5.0
SkillRL	29.7↑4.4	48.1↑6.9	27.8↑3.8	44.2↑2.7	21.5↑6.1	62.8↑7.6	43.0↑11.2	39.6↑6.1
SkillClaw	31.2↑5.9	47.8↑6.6	32.5↑8.5	43.4↑1.9	23.5↑8.1	61.4↑6.2	45.8↑14.0	40.8↑7.3
SkillGate	31.5↑6.2	48.1↑6.9	32.8↑8.8	43.2↑1.7	23.7↑8.3	61.0↑5.8	46.1↑14.3	40.9↑7.4
`Skill`^{2}	**36.3**↑11.0	**53.3**↑12.1	**35.2**↑11.2	**49.6**↑8.1	**24.2**↑8.8	**68.5**↑13.3	**47.9**↑16.1	**45.0**↑11.5

Obs. ❶ `Skill`^{2} achieves the best result in all fourteen comparisons across backbones and benchmarks. With Qwen3.5-4B, `Skill`^{2} reaches 29.2%, 44.1%, 26.1%, and 41.3% on Terminal-Bench 2.1, OpenThoughts-TBLite, TerminalWorld-Verified, and InterCode-Bash, improving over vanilla by 6.9, 12.2, 7.8, and 8.1 points. With Qwen3.5-9B, the improvements are 11.0, 12.1, 11.2, and 8.1 points. On average, `Skill`^{2} lifts pass@1 from 27.9% to 37.5% and from 33.5% to 45.0% at the two scales, whereas the strongest baseline, SkillGate, reaches 34.2% and 40.9%. Against the strongest baseline on each benchmark, the margin ranges from 0.5 points on WebArena-Verified to 5.2 points on OpenThoughts-TBLite, with a relative gain of up to 15.2% on Terminal-Bench 2.1.

Obs. ❷ Fully agentic co-evolution outperforms one-sided and misaligned alternatives. RL methods lead the misaligned baselines, with DAPO adding 5.8 and 7.1 points over vanilla, yet what they learn stays inside the parameters. Training-free skill evolution adds only 1.8 to 2.8 points, and the best skill-based RL method trails the best RL method in thirteen of fourteen comparisons, consistent with skills formed as plain text and developed on text simulators. The two fully agentic baselines gain the most, 6.0 to 7.4 points, as alignment with deployment pays off, yet each leaves one side fixed, SkillClaw the policy and SkillGate the library. `Skill`^{2} exceeds SkillGate by 3.3 and 4.1 points, as its actor invokes skills by itself and each side trains against the other's latest state.

Obs. ❸ The gains of `Skill`^{2} hold across model scales and lift a compact model past its larger vanilla counterpart. The average gain over vanilla rises from 9.6 points at the 4B scale to 11.5 points at the 9B scale, and `Skill`^{2} stays ahead of all baselines on all seven benchmarks at both scales. On Terminal-Bench 2.1, the gain over vanilla grows from 6.9 to 11.0 points. Moreover, `Skill`^{2} at the 4B scale surpasses the vanilla Qwen3.5-9B on six of the seven benchmarks (37.5% vs. 33.5% on average), and both training-free baselines at the 9B scale on at least five. Its average gain at 4B (9.6 points) also exceeds what moving the vanilla backbone from 4B to 9B brings (5.6 points). Co-evolving a policy with its skill library can thus outweigh doubling the model size.

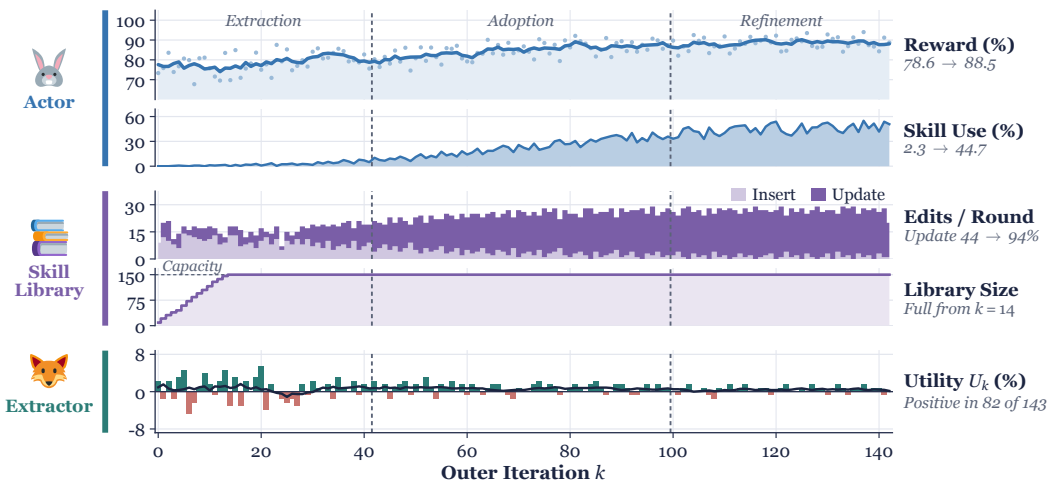


Figure 4: Co-evolution dynamics of Skill² with Qwen3.5-9B on terminal tasks.

3.3 Transferability ($\mathcal{RQ}3$)

We give four larger frozen models the skill library evolved with Qwen3.5-9B (Figure 3).

Obs. ❹ The evolved skill library transfers to larger frozen models across families and architectures. The four models invoke the skills through the same harness without training, and each gains on all seven benchmarks, by 3.0 to 4.6 points against 0.7 to 1.3 from Skill1's library (Appendix F.1). On average, the gain is largest for the two Qwen3.5 models, from the actor's family, and smallest for Muse Glimmer 30B, the strongest of the four (Table 6), consistent with skills supplying procedures that a model lacks rather than instructions it would already follow on its own.

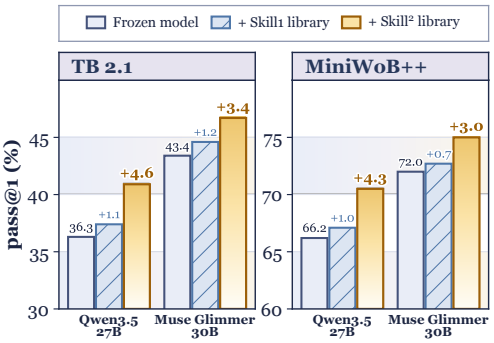


Figure 3: **Transfer to frozen models.** Pass@1 of two of the four frozen models alone and with the 9B skill library of Skill1 or Skill².

3.4 Framework Analysis ($\mathcal{RQ}4$)

We examine how the three parts of Skill² move over training and how one skill evolves, with mechanism ablations and a sensitivity analysis in Appendices F.2 and F.3.

Obs. ❺ Co-evolution proceeds from extraction to adoption to refinement. Figure 4 splits the run into three phases. ① *Extraction precedes adoption.* The extractor fills all 150 slots of the library by round 14, yet skill use averages only 2.3% over the first 42 rounds, so a library is not used just because it exists. ② *Adoption drives both roles.* As skill use passes 10% at round 42, the reward rises with it, and the extractor turns from inserting to revising skills from round 39 on, 25 rounds after the library filled, consistent with micro credit arriving only through invocations. Its revisions pay most here, positive in 64% of rounds. ③ *The library settles as the actor does.* Utility is non-negative in 84% of the last 43 rounds yet shrinks toward zero, so revision becomes fine adjustment.

Obs. ❻ Revision turns a one-off check into a reusable tool. Over seven updates (Appendix G), the extractor makes the delimiter, keys, and record contract of a CSV check into options of one tool and fixes two regressions along the way. It also rewrites the description twice, as the actor decides from metadata alone whether to call the skill. Baseline skills stay a fixed script or plain text.

4 Conclusion

We presented Skill², a fully agentic framework in which a policy and its skill library co-evolve through Fork-GRPO and counterfactual-utility policy optimization. Across seven real long-horizon benchmarks, Skill² outperforms all baselines at two scales, and its skill library transfers to frozen models. These results suggest that skills mature as a shared artifact between practice and the knowledge written from it, a step toward agents that grow their own capabilities in use.

AI USE STATEMENT

Large language models are the object of study in this work. Within it, the actor and the extractor are trained from Qwen3.5 models, four open-weight models are evaluated with the evolved skill library (§3.3), and GLM-5.2 labels the tasks of the probe pools and their families (Appendix E.1). Beyond these uses, LLMs were used only to polish the writing.

ETHICS STATEMENT

Our agents execute shell commands and operate web browsers only in isolated containers, locally hosted websites, and dedicated test instances, never on public websites or real user systems. All benchmarks are public and involve no human subjects or personal data. The evolved skills are machine-generated procedures that may contain unsafe commands, so they are stored as human-readable files with a logged history of every edit, and we recommend human review before an evolved skill library is used outside a sandbox.

REPRODUCIBILITY STATEMENT

The method is specified in §2, with its procedure in Algorithm 1 and its notation in Appendix A. The benchmarks are described in Appendix D, and the environments, training data, probe pools, and all hyperparameters of both domains in Appendix E, together with the execution pipeline. The prompts of the extractor are reproduced in Appendix I, and all results are pass@1 averaged over 10 runs. The code is available in the anonymous repository linked in the abstract.

REFERENCES

Agent Skills. Agent skills: Specification and documentation for agent skills. URL https://github.com/agentskills/agentskills.

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. Gepa: Reflective prompt evolution can outperform reinforcement learning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 8479–8565, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/0e9e708b6f48e14fd0ac29e167413f76-Paper-Conference.pdf.

Huan ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, Hongru Wang, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Qihan Ren, Cheng Qian, Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence, 2026. URL https://arxiv.org/abs/2507.21046.

Dianne C. Berry and Donald E. Broadbent. On the relationship between task performance and associated verbalizable knowledge. *The Quarterly Journal of Experimental Psychology Section A*, 36(2):209–231, 1984. doi: 10.1080/14640748408402156. URL https://doi.org/10.1080/14640748408402156.

Léo Boisvert, Megh Thakkar, Maxime Gasse, Massimo Caccia, Thibault Le Sellier De Chezelles, Quentin Cappart, Nicolas Chapados, Alexandre Lacoste, and Alexandre Drouin. Workarena++: Towards compositional planning and reasoning-based common knowledge work tasks. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 5996–6051. Curran Associates, Inc., 2024. doi: 10.52202/079017-0195. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/0b82662b6c32e887bb252a74d8cb2d5e-Paper-Datasets_and_Benchmarks_Track.pdf.

Zhengyu Chen, Teng Xiao, Huaisheng Zhu, Yige Yuan, Luan Zhang, and Jingang Wang. Co-harness: Co-evolving harnesses and model weights for llm agents, 2026. URL https://arxiv.org/abs/2607.22688.

Thibault Le Sellier De Chezelles, Maxime Gasse, Alexandre Drouin, Massimo Caccia, Léo Boisvert, Megh Thakkar, Tom Marty, Rim Assouel, Sahar Omidi Shayegan, Lawrence Keunho Jang, Xing Han Lù, Ori Yoran, Dehan Kong, Frank F. Xu, Siva Reddy, Quentin Cappart, Graham Neubig, Ruslan Salakhutdinov, Nicolas Chapados, and Alexandre Lacoste. The browsergym ecosystem for web agent research, 2025. URL https://arxiv.org/abs/2412.05467.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. In Inês Lynce, Nello Murano, Mauro Vallati, Serena Villata, Federico Chesani, Michela Milano, Andrea Omicini, and Mehdi Dastani (eds.), *ECAI 2025 - 28th European Conference on Artificial Intelligence, 25-30 October 2025, Bologna, Italy - Including 14th Conference on Prestigious Applications of Intelligent Systems (PAIS 2025)*, volume 413 of *Frontiers in Artificial Intelligence and Applications*, pp. 2993–3000. IOS Press, 2025. doi: 10.3233/FAIA251160. URL https://doi.org/10.3233/FAIA251160.

Zhaoyang Chu, Jiarui Hu, Xingyu Jiang, Pengyu Zou, Han Li, Chao Peng, Peter O'Hearn, Earl T. Barr, Mark Harman, Federica Sarro, and He Ye. Terminalworld: Benchmarking agents on real-world terminal tasks, 2026. URL https://arxiv.org/abs/2605.22535.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic reinforced policy optimization, 2025. URL https://arxiv.org/abs/2507.19849.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How capable are web agents at solving common knowledge work tasks? In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11642–11662. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/drouin24a.html.

Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1406–1415. PMLR, 2018. URL http://proceedings.mlr.press/v80/espeholt18a.html.

Jinyuan Fang, Yanwen Peng, Xi Zhang, Yingxu Wang, Xinhao Yi, Guibin Zhang, Yi Xu, Bin Wu, Siwei Liu, Zihao Li, Zhaochun Ren, Nikos Aletras, Xi Wang, Han Zhou, and Zaiqiao Meng. A comprehensive survey of self-evolving ai agents: A new paradigm bridging foundation models and lifelong agentic systems, 2025. URL https://arxiv.org/abs/2508.07407.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, Tongkai Yang, Binhang Yuan, and Yi Wu. Areal: A large-scale asynchronous reinforcement learning system for language reasoning, 2026. URL https://arxiv.org/abs/2505.24298.

Jinyi Han, Yuanjian Xu, Ying Liao, Xinyi Wang, Zishang Jiang, Zixiang Di, Fanyang Lu, Zhichao Hu, and Yanghua Xiao. Skill-use: Can llms actually use skills in agentic harnesses?, 2026a. URL https://arxiv.org/abs/2608.04828.

Tingxu Han, Yi Zhang, Wei Song, Chunrong Fang, Zhenyu Chen, Youcheng Sun, and Lijie Hu. Swe-skills-bench: Do agent skills actually help in real-world software engineering?, 2026b. URL https://arxiv.org/abs/2603.15401.

Amine El hattami, Megh Thakkar, Nicolas Chapados, and Christopher Pal. Webarena verified: Reliable evaluation for web agents. In *Workshop on Scaling Environments for Agents*, 2025. URL https://openreview.net/forum?id=94tlGxmqkN.

Zelin He, Haotian Lin, Boran Han, Wei Zhu, Haoyang Fang, Yuyang Wang, Xuan Zhu, Runze Li, and Matthew Reimherr. Reskill: Reconciling skill creation with policy optimization in agentic RL. *CoRR*, abs/2606.01619, 2026a. doi: 10.48550/ARXIV.2606.01619. URL https://doi.org/10.48550/arXiv.2606.01619.

Zhiyuan He, Siwei Zhang, Zhiwen Zhou, Yuqing Yang, Yu Kang, Yuge Zhang, Luna K. Qiu, Tin Yan Tsui, Jiahang Xu, and Chong Luo. Agent lightning v1.0: Towards harnessed agentic rl, 2026b. URL https://arxiv.org/abs/2608.17528.

Zhongyu He, Yuanfan Li, Fei Huang, Tianyu Chen, Siyuan Chen, Xingyang Li, Meng Hsuan Yu, Xiangrong Liu, Leyi Wei, Lu Pan, Ke Zeng, and Xunliang Cai. Siri: Self-internalizing reinforcement learning with intrinsic skills for llm agent training, 2026c. URL https://arxiv.org/abs/2606.02355.

Jacob Hilton, Karl Cobbe, and John Schulman. Batch size-invariance for policy optimization. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/6ceb6c2150bbf46fd75528a6cd6be793-Abstract-Conference.html.

Pengcheng Jiang, Jiacheng Lin, Zhiyi Shi, Zifeng Wang, Luxi He, Yichen Wu, Ming Zhong, Peiyang Song, Qizheng Zhang, Heng Wang, Xueqiang Xu, Hanwen Xu, Pengrui Han, Dylan Zhang, Jiashuo Sun, Chaoqi Yang, Kun Qian, Tian Wang, Changran Hu, Manling Li, Quanzheng Li, Hao Peng, Sheng Wang, Jingbo Shang, Chao Zhang, Jiaxuan You, Liyuan Liu, Pan Lu, Yu Zhang, Heng Ji, Yejin Choi, Dawn Song, Jimeng Sun, and Jiawei Han. Adaptation of agentic ai: A survey of post-training, memory, and skills, 2026a. URL https://arxiv.org/abs/2512.16301.

Yanna Jiang, Delong Li, Haiyu Deng, Baihe Ma, Xu Wang, Qin Wang, and Guangsheng Yu. Sok: Agentic skills - beyond tool use in LLM agents. *CoRR*, abs/2602.20867, 2026b. doi: 10.48550/ARXIV.2602.20867. URL https://doi.org/10.48550/arXiv.2602.20867.

Zhiyuan Jiang, Fangrui Huang, Hanwen Xing, Xander Wu, Yipeng Gao, Rui Cao, Mengdi Wang, Shilong Liu, and Yijiang Li. Demystifying agent skills: Why they work-until they don't, 2026c. URL https://arxiv.org/abs/2608.14036.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. Dspy: Compiling declarative language model calls into self-improving pipelines, 2023. URL https://arxiv.org/abs/2310.03714.

Kyungmin Kim, Youngbin Choi, Seoyeon Lee, Suhyeon Jun, Dongwoo Kim, and Sangdon Park. The interplay of harness design and post-training in llm agents, 2026. URL https://arxiv.org/abs/2606.25447.

Qingyao Li, Wenxiang Jiao, Shuai Shao, Kangning Zhang, Yuan Lu, Yi Guo, Weiwen Liu, Weinan Zhang, and Yong Yu. Skillgate: Training in-policy skill selection in long-horizon agents, 2026a. URL https://arxiv.org/abs/2608.18852.

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, Qunhong Zeng, Di Wang, Yuanli Wang, Roey Ben Chaim, Penghao Jiang, Haotian Shen, Luyang Kong, Xinyi Liu, Runhui Wang, Xuanqing Liu, Jiachen Li, Xin Lan, Yueqian Lin, Wengao Ye, Junwei He, Songlin Li, Yue Zhang, Yipeng Gao, Yijiang Li, Ze Ma, Liqiang Jing, Tianyu Wang, Kaixin Li, Yiqi Xue, Haoran Lyu, Yizhuo He, Yuchen Tian, Shutong Wu, Bowei Wang, Yixuan Gao, Bo Chen, Litong Liu, Sikai Cheng, Jiajun Bao, Shuaicheng Tong, Shuwen Xu, Terry Yue Zhuo, Tinghan

Ye, Qi Qi, Miao Li, Longtai Liao, Zelin Tan, Chang Shi, Xilin Tang, Srinath Tankasala, Boqin Yuan, Yaoyao Qian, Jianhong Tu, Chenguang Wang, Yizhou Sun, Wei Wang, Aaron Taylor, Ziyue Yang, Changkun Guan, Zhikang Dong, Xinyu Zhang, Steven Dillmann, Han chung Lee, and Dawn Song. Skillsbench: Benchmarking how well agent skills work across diverse tasks, 2026b. URL https://arxiv.org/abs/2602.12670.

Xi Victoria Lin, Chenglong Wang, Luke Zettlemoyer, and Michael D. Ernst. NL2Bash: A corpus and semantic parser for natural language interface to the linux operating system. In Nicoletta Calzolari, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga (eds.), *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL https://aclanthology.org/L18-1491/.

Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. Reinforcement learning on web interfaces using workflow-guided exploration. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=ryTp3f-0-.

Hongyi Liu, Haoyan Yang, Tao Jiang, Bo Tang, Feiyu Xiong, Yuyu Luo, and Zhiyu Li. Skillsvote: Lifecycle governance of agent skills from collection, recommendation to evolution, 2026. URL https://arxiv.org/abs/2605.18401.

Zhengxi Lu, Zhiyuan Yao, Jinyang Wu, Chengcheng Han, Qi Gu, Xunliang Cai, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. Skill0: In-context agentic reinforcement learning for skill internalization, 2026. URL https://arxiv.org/abs/2604.02268.

Ziyu Ma, Shidong Yang, Yuxiang Ji, Xucong Wang, Yong Wang, Yiming Hu, Tongwen Huang, and Xiangxiang Chu. Skillclaw: Let skills evolve collectively with agentic evolver, 2026. URL https://arxiv.org/abs/2604.08377.

Fanzhe Meng, Guoxin Chen, Jiale Zhao, Shuang Sun, Zhiyu Lin, Wayne Xin Zhao, Ruihua Song, Ji-Rong Wen, and Kai Jia. Calibforge: Adversarial solver calibration for scaling learnable terminal tasks, 2026. URL https://arxiv.org/abs/2608.06352.

Mike A. Merrill, Alexander Glenn Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, Estefany Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Jenia Jitsev, Di Lu, Orfeas Menis-Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Kumar Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Jesse Hu, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *CoRR*, abs/2601.11868, 2026. doi: 10.48550/ARXIV.2601.11868. URL https://doi.org/10.48550/arXiv.2601.11868.

Dilxat Muhtar, Jiashun Liu, Wei Gao, Weixun Wang, Shaopan Xiong, Ju Huang, Siran Yang, Wenbo Su, Jiamang Wang, Ling Pan, and Bo Zheng. Complementary rl: Towards efficient experience-driven agent learning, 2026. URL https://arxiv.org/abs/2603.17621.

Bespoke Labs OpenThoughts-Agent team, Snorkel AI. OpenThoughts-TBLite: A High-Signal Benchmark for Iterating on Terminal Agents. https://www.openthoughts.ai/blog/openthoughts-tblite, February 2026.

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. Skillos: Learning skill curation for self-evolving agents. *CoRR*, abs/2605.06614, 2026. doi: 10.48550/ARXIV.2605.06614. URL https://doi.org/10.48550/arXiv.2605.06614.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Junhao Shen, Zhanqiu Zhang, Yiwen Guo, and Hong Cheng. Progressive agent skill generation via reinforcement learning, 2026a. URL https://arxiv.org/abs/2608.01678.

Qijia Shen, Zhiqi Huang, Vamsidhar Kamanuru, Aznaur Aliev, Jay Rainton, Ahmed Awelkair, Zhichen Zeng, Jiajun Li, Shi Dong, Yueming Yuan, Boyuan Ma, Qizheng Zhang, Jiwei Fu, Yuzhen Mao, Wendong Fan, Ping Nie, Philip Torr, Bernard Ghanem, Changran Hu, Jonathan Lingjie Li, Urmish Thakker, and Guohao Li. Seta: Scaling environments for terminal agents, 2026b. URL https://arxiv.org/abs/2607.10891.

Yaorui Shi, Yuxin Chen, Zhengxi Lu, Yuchun Miao, Shugui Liu, Qi Gu, Xunliang Cai, Xiang Wang, and An Zhang. Skill1: Unified evolution of skill-augmented agents via reinforcement learning. *CoRR*, abs/2605.06130, 2026. doi: 10.48550/ARXIV.2605.06130. URL https://doi.org/10.48550/arXiv.2605.06130.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020. URL https://arxiv.org/abs/1909.08053.

Ron Sun, Paul Slusarz, and Chris Terry. The interaction of the explicit and the implicit in skill learning: A dual-process approach. *Psychological Review*, 112:159–192, 01 2005. doi: 10.1037/0033-295X.112.1.159.

Mirac Suzgun, Mert Yüksekgönül, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. In Vera Demberg, Kentaro Inui, and Lluís Marquez (eds.), *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2026 - Volume 1: Long Papers, Rabat, Morocco, March 24-29, 2026*, pp. 7080–7106. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.EACL-LONG.333. URL https://doi.org/10.18653/v1/2026.eacl-long.333.

Zelin Tan, Yiqun Zhang, Hao Li, Zhiyao Cui, Hejia Geng, Shao Zhang, Hangfan Zhang, Yang Chen, Xiaosong Wang, Lilong Wang, Zhenfei Yin, Shuyue Hu, Chen Zhang, and Lei Bai. SKT: skill-use training at scale via verified synthetic data generation. *CoRR*, abs/2608.02287, 2026. doi: 10.48550/ARXIV.2608.02287. URL https://doi.org/10.48550/arXiv.2608.02287.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Trans. Mach. Learn. Res.*, 2024, 2024a. URL https://openreview.net/forum?id=ehfRiF0R3a.

Jiongxiao Wang, Qiaojing Yan, Yawei Wang, Yijun Tian, Soumya Smruti Mishra, Zhichao Xu, Megha Gandhi, Panpan Xu, and Lin Lee Cheong. Reinforcement learning for self-improving agent with skill library. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 1529–1550. Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.ACL-LONG.69. URL https://doi.org/10.18653/v1/2026.acl-long.69.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. *Frontiers Comput. Sci.*, 18(6): 186345, 2024b. doi: 10.1007/S11704-024-40231-1. URL https://doi.org/10.1007/s11704-024-40231-1.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, and et al. Openhands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025a. URL https://openreview.net/forum?id=OJd3ayDDoF.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025b. URL https://proceedings.mlr.press/v267/wang25bx.html.

Tianxin Wei, Zhan Shi, Minhua Lin, Bing He, Zewen Liu, Yisi Sang, Yuanchen Bei, Xuying Ning, Jiaru Zou, Ting-Wei Li, Xiao Lin, Yanjun Zhao, Chi Wang, Benoit Dumoulin, Dakuo Wang, Jingrui He, and Hanqing Lu. Evo-harness: Context-to-harness skill compilation for self-evolving agents, 2026. URL https://arxiv.org/abs/2608.15071.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, and Botian Shi. Evolver: Self-evolving llm agents through an experience-driven lifecycle, 2026. URL https://arxiv.org/abs/2510.16079.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, Qi Zhang, and Tao Gui. The rise and potential of large language model based agents: a survey. *Sci. China Inf. Sci.*, 68(2), 2025. doi: 10.1007/S11432-024-4222-0. URL https://doi.org/10.1007/s11432-024-4222-0.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *CoRR*, abs/2602.08234, 2026a. doi: 10.48550/ARXIV.2602.08234. URL https://doi.org/10.48550/arXiv.2602.08234.

Peng Xia, Jianwen Chen, Xinyu Yang, Haoqin Tu, Jiaqi Liu, Kaiwen Xiong, Siwei Han, Shi Qiu, Haonian Ji, Yuyin Zhou, Zeyu Zheng, Cihang Xie, and Huaxiu Yao. Metaclaw: Just talk – an agent that meta-learns and evolves in the wild, 2026b. URL https://arxiv.org/abs/2603.17187.

Binfeng Xu, Hao Zhang, Shaokun Zhang, Songyang Han, Mingjie Liu, Jian Hu, Shizhe Diao, Zhenghui Jin, Yunheng Zou, Michael Demoret, Jan Kautz, and Yi Dong. Polar: Agentic rl on any harness at scale, 2026. URL https://arxiv.org/abs/2605.24220.

Renjun Xu and Yang Yan. Agent skills for large language models: Architecture, acquisition, security, and the path forward, 2026. URL https://arxiv.org/abs/2602.12430.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024a. URL https://openreview.net/forum?id=Bb4VGOWELI.

John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. Intercode: Standardizing and benchmarking interactive coding with execution feedback. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/4b175d846fb008d540d233c188379ff9-Abstract-Datasets_and_Benchmarks.html.

John Yang, Carlos Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 50528–50652. Curran Associates, Inc., 2024b. doi: 10.52202/079017-1601. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/5a7c947568c1b1328ccc5230172e1e7c-Paper-Conference.pdf.

Min Yang, Jinghua Piao, Xu Xia, Xiaochong Lan, Jiaju Chen, Yongshun Gong, and Yong Li. Skillmaster: Toward autonomous skill mastery in LLM agents. *CoRR*, abs/2605.08693, 2026a. doi: 10.48550/ARXIV.2605.08693. URL https://doi.org/10.48550/arXiv.2605.08693.

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

Yutao Yang, Junsong Li, Qianjun Pan, Bihao Zhan, Yuxuan Cai, Lin Du, Jie Zhou, Kai Chen, Qin Chen, Xin Li, Bo Zhang, and Liang He. Autoskill: Experience-driven lifelong learning via skill self-evolution, 2026c. URL https://arxiv.org/abs/2603.01145.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

Haoran Ye, Xuning He, Vincent Arak, Haonan Dong, and Guojie Song. Meta context engineering via agentic skill evolution. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=P1jHroBS5E.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: an open-source LLM reinforcement learning system at scale. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5,*

2025, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/a4277440d50f1f15d2cb4c14f7e0c0d2-Abstract-Conference.html.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic "differentiation" via text, 2024. URL https://arxiv.org/abs/2406.07496.

Barry Zhang, Keith Lazuka, and Mahesh Murag. Equipping agents for the real world with Agent Skills. URL https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills.

Genrui Zhang, Erle Zhu, Jinfeng Zhou, Caiyan Jia, and Hongning Wang. Skillevolver: Skill learning as a meta-skill. *CoRR*, abs/2605.10500, 2026a. doi: 10.48550/ARXIV.2605.10500. URL https://doi.org/10.48550/arXiv.2605.10500.

Hanrong Zhang, Shicheng Fan, Henry Peng Zou, Yankai Chen, Zhenting Wang, Jiayu Zhou, Chengze Li, Wei-Chieh Huang, Yifei Yao, Kening Zheng, Xue Liu, Xiaoxiao Li, and Philip S. Yu. Coevoskills: Self-evolving agent skills via co-evolutionary verification. *CoRR*, abs/2604.01687, 2026b. doi: 10.48550/ARXIV.2604.01687. URL https://doi.org/10.48550/arXiv.2604.01687.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Y Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 86069–86100, 2026c. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/8a94ff6f922d995d7d3f4ebf4143e442-Paper-Conference.pdf.

Zhiwei Zhang, Yudi Lin, Nikki Lijing Kuang, Linlin Wu, Xiaomin Li, Songtao Liu, and Fenglong Ma. Co-evolving skill generation and policy optimization. *CoRR*, abs/2606.08755, 2026d. doi: 10.48550/ARXIV.2606.08755. URL https://doi.org/10.48550/arXiv.2606.08755.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: LLM agents are experiential learners. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (eds.), *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2024, February 20-27, 2024, Vancouver, Canada*, pp. 19632–19642. AAAI Press, 2024. doi: 10.1609/AAAI.V38I17.29936. URL https://doi.org/10.1609/aaai.v38i17.29936.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark W. Barrett, and Ying Sheng. Sglang: Efficient execution of structured language model programs. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/724be4472168f31ba1c9ac630f15dec8-Abstract-Conference.html.

Shuyan Zhou. Webarena-infinity: Generating browser environments with verifiable tasks at scale. *shuyanzhou.com*, March 2026. URL https://webarena.dev/webarena-infinity/.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

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

2026. doi: 10.48550/ARXIV.2605.07358. URL https://doi.org/10.48550/arXiv.2605.07358.

Kaijie Zhu, Yuzhou Nie, Yijiang Li, Yiming Huang, Jialian Wu, Jiang Liu, Ximeng Sun, Zhenfei Yin, Lun Wang, Zicheng Liu, Emad Barsoum, William Yang Wang, and Wenbo Guo. Termigen: High-fidelity environment and robust trajectory synthesis for terminal agents, 2026. URL https://arxiv.org/abs/2602.07274.

Zilin Zhu, Chengxing Xie, Xin Lv, and slime Contributors. slime: An llm post-training framework for rl scaling. https://github.com/THUDM/slime, 2025. GitHub repository. Corresponding author: Xin Lv.

A NOTATIONS

Table 2 lists the notation of the paper, grouped by the part of `Skill`^{2} it belongs to.

Table 2: Notations used throughout the paper.

Notation	Definition
	Preliminary
$\mathcal{B} = \{\sigma_j\}_{j=1}^{\lvert\mathcal{B}\rvert}$	the skill library, a set of skill artifacts
$\sigma_j = (n_j, d_j, b_j)$	a skill, with its name, description, and body
$c(\mathcal{B}) = \{(n_j, d_j)\}_j$	the metadata of the skill library, disclosed before any invocation
$\mathcal{H}$	the skill-native harness
π_θ, ρ_ϕ	the actor with parameters θ and the extractor with parameters ϕ
$x \sim \mathcal{D}$	a task drawn from the task distribution
h_t, y_t	the history at turn t and the token span generated from it
$a_t \in \mathcal{A} = \mathcal{A}_{\text{env}} \cup \{\text{invoke}(n)\}$	the action at turn t, an environment action or a skill invocation
o_t	the observation at turn t, a tool return or a skill body
τ, $R(\tau)$	a trajectory and its task reward
	Actor
$\mathcal{G}(x)$, G	the group of rollouts for task x and its size
$t^\star$, $n^\star$	the turn of a skill decision and the skill invoked
τ^+, τ^-	an invoking rollout and its control rollout without the skill
$\mathcal{G}^+(x)$, $\mathcal{G}^-(x)$	the invoking and the control members of the group
$\delta_i = R(\tau_i^+) - R(\tau_i^-)$	the paired difference of a fork pair
$\hat{A}_i$	the group-relative advantage of rollout τ_i
L	a lineage, rollouts that share a prefix
$\text{dec}(\tau)$	the decision tokens of a rollout at its fork turn
$A_{i,t}$	the token-level advantage of the actor
$\mathcal{J}_A(\theta)$, $r_{i,t}$, β	the actor objective, its probability ratio, and its KL coefficient
ε_{low}, $\varepsilon_{\text{high}}$	the ratio clipping range, shared by the two objectives
	Extractor
$\mathcal{C}_m$, $\mathcal{C}_{m,j}$, N	a maintenance set, its j-th task group, and the number of sets per round
e_i	the bounded evidence attached to rollout τ_i
D_m	the working draft after set m, with $D_0 = \mathcal{B}_k$
$S_{m,j}$, M	the skills read in full for group j of set m, and their maximum number
$o_{m,j} \in \mathcal{O}$, J	the operation for group j of set m, and the number of groups per set
Apply	the in-order application of a set's operations to the draft
$\mathcal{B}_k$, $\tilde{\mathcal{B}}_k$	the skill library of iteration k and its candidate revision
$\mathcal{P}$, $\mathcal{P}_k$	the probe pool, and the probe suite drawn from it in round k
Q	the number of repetitions per probe task
$R_{p,r}^{\text{keep}}$, $R_{p,r}^{\text{cand}}$, $\Delta_{p,r}$	the paired rewards under the two libraries, and their difference
U_k	the counterfactual utility of the candidate
$g_{k,n}$, $\mathcal{S}_k^+$	the invocation gain of skill n under $\mathcal{B}_k$, and the skills it certifies
A_k^{mac}, $A_{k,n}^{\text{mic}}$, s_k, $A_{\max}$	the macro and micro credit, their running scale, and their bound
$\mathcal{R}_k$, $\mathcal{E}_n$	the edits of round k that reached the candidate, and the buffered edits that shaped the version of skill n in $\mathcal{B}_k$
κ, $\text{span}(\ell)$	the structural penalty and the span of edit ℓ it applies to
$A_{\ell,t}^E$	the token-level advantage of the extractor
$\mathcal{U}_k$	the credit units of round k, the revision, the released units, and the rejected edits
ρ_{ϕ_ℓ}, $\rho_{\phi_{\text{old}}}$	the extractor that sampled edit ℓ and the extractor before the update
$w_{\ell,t}$, c	the truncated importance weight and its cap
$\mathcal{J}_E(\phi)$, $r_{\ell,t}^E$, β_E	the extractor objective, its probability ratio, and its KL coefficient
	Co-Evolution
k, K	the outer iteration and the number of iterations
S_A	the number of actor steps per iteration
$(\theta_k, \phi_k, \mathcal{B}_k)$	the state at the start of iteration k

B ALGORITHM OF Skill²

Algorithm 1 gives the full procedure of Skill², alternating the actor phase (§2.2), the extractor phase (§2.3), and the release of the skill library (§2.4).

Algorithm 1 Skill²

Input: actor π_{θ_0}, extractor ρ_{ϕ_0}, skill library $\mathcal{B}_0$, probe pool $\mathcal{P}$, iterations K, actor steps S_A

for $k = 0, 1, \ldots, K-1$ **do**

Actor phase (§2.2) — $\mathcal{B}_k$, ϕ_k fixed

for $s = 1, \ldots, S_A$ **do**

$x \sim \mathcal{D}$; $\mathcal{G}(x) \leftarrow G$ rollouts of π_θ under $c(\mathcal{B}_k)$

fork τ_i^- at every skill decision; $\delta_i \leftarrow R(\tau_i^+) - R(\tau_i^-)$ ▷ decision-point forking

$A_{i,t} \leftarrow$ Eqs. 2, 3; $\theta \leftarrow$ one step on Eq. 4 ▷ credit assignment

$\theta_{k+1} \leftarrow \theta$; $g_{k,n} \leftarrow$ mean δ_i over pairs invoking n ▷ invocation gain under $\mathcal{B}_k$

Extractor phase (§2.3) — θ_{k+1} fixed

$\mathcal{C}_1, \ldots, \mathcal{C}_N \leftarrow$ maintenance sets of J task groups each; $D_0 \leftarrow \mathcal{B}_k$

for $m = 1, \ldots, N$ **do** ▷ one wave of edits per set

for $j = 1, \ldots, J$ **in parallel do** ▷ Eqs. 6, 7

$S_{m,j} \sim \rho_\phi(\cdot \mid \mathcal{C}_{m,j},\ c(D_{m-1}))$; $o_{m,j} \sim \rho_\phi(\cdot \mid \mathcal{C}_{m,j},\ c(D_{m-1}),\ \{b_n\}_{n \in S_{m,j}})$

$D_m \leftarrow \mathrm{Apply}(D_{m-1}, o_{m,1}, \ldots, o_{m,J})$ ▷ in order, resolving conflicts

$\tilde{\mathcal{B}}_k \leftarrow D_N$ ▷ one candidate per round

$\mathcal{P}_k \leftarrow$ stratified draw from $\mathcal{P}$ with seed and round k ▷ drawn anew every round

run $\pi_{\theta_{k+1}}$ on $\mathcal{P}_k$ under $\mathcal{B}_k$ and $\tilde{\mathcal{B}}_k$; $U_k \leftarrow$ Eq. 8 ▷ counterfactual utility

$A^E_{\ell,t} \leftarrow A^{\mathrm{mac}}_k$ on $\mathcal{R}_k$, $A^{\mathrm{mic}}_{k,n}$ on $\mathcal{E}_n$, $n \in \mathcal{S}^+_k$ ▷ dual-level credit assignment

$\phi \leftarrow$ one step on Eq. 9; $\phi_{k+1} \leftarrow \phi$

Skill library release

$\mathcal{B}_{k+1} \leftarrow \tilde{\mathcal{B}}_k$ ▷ seen by the actor from iteration $k+1$

Output: actor π_{θ_K}, skill library $\mathcal{B}_K$

C RELATED WORK

Self-Evolving Agents. Self-evolving agents improve by accumulating and reusing their own experience (ang Gao et al., 2026; Fang et al., 2025; Jiang et al., 2026a). By the carrier of experience, prior work falls into five families. ❶ **Prompt optimization** compresses experience into instructions and demonstrations that are rewritten by the LLM itself or by evolutionary search (Yang et al., 2024a; Khattab et al., 2023; Yuksekgonul et al., 2024; Agrawal et al., 2026). ❷ **Agent memory** stores trajectories, reflections, and insights in an external store and retrieves them back into the context (Shinn et al., 2023; Zhao et al., 2024; Wang et al., 2025b; Chhikara et al., 2025). ❸ **Context engineering** treats the context itself as an object that is revised continually (Suzgun et al., 2026; Zhang et al., 2026c; Ye et al., 2026). ❹ **Experience-driven RL** updates the policy and its experience store within one training loop (Muhtar et al., 2026; Wu et al., 2026; Chen et al., 2026), a two-sided stance that Skill² shares. ❺ **Skill learning** packages experience into reusable skills that an agent learns to use or to write (Xia et al., 2026a; Zhang et al., 2026a; Tan et al., 2026). Skill² belongs to this family.

Skill Learning for LLM Agents. Skills, procedural artifacts that package instructions, metadata, and optional scripts and resources, have become a standard capability layer of agent systems (Zhang et al.; Agent Skills) and the subject of dedicated surveys (Xu & Yan, 2026; Jiang et al., 2026b; Zhou et al., 2026). By the object being optimized, prior work falls into three lines. ❶ **Policy side** fixes the skill library and trains the model to select and follow given skills (Tan et al., 2026; Li et al., 2026a), or internalizes them so that no skill is needed at inference (Lu et al., 2026; He et al., 2026c). ❷ **Library side** freezes the acting model and lets a prompted model induce and revise skills from experience (Wang et al., 2024a; Zhang et al., 2026b;a; Ma et al., 2026; Yang et al., 2026b), or

Table 3: Comparison with representative skill-learning methods. The three *Fully agentic* columns follow the three misalignments of §1. **Artifact**, skills are directory artifacts with scripts and resources rather than text spliced into the prompt. **Invocation**, the agent itself decides whether, when, and which skill to invoke through the harness rather than a pipeline injecting skills. **Validation**, evaluation on real long-horizon agentic tasks rather than text simulators. The two *Co-evolution* columns ask who learns. **Actor**, whether the model that uses skills is trained. **Extractor**, whether the model that writes skills is trained, sharing weights with the actor (†) or as a separate model (‡). ✓, ●, and ✗ denote full, partial, and absent. 🔥 trainable, ❄ frozen.

Method	Fully agentic			Co-evolution	
	Artifact	Invocation	Validation	Actor	Extractor
Misaligned with deployment					
Voyager (Wang et al., 2024a)	●	✗	✓	❄	❄
Evo-Harness (Wei et al., 2026)	●	✗	✓	❄	❄
SkillOpt (Yang et al., 2026b)	●	●	✗	❄	❄
SkillOS (Ouyang et al., 2026)	●	✗	✗	❄	🔥‡
Skill-α (Shen et al., 2026a)	●	✗	●	❄	🔥‡
SkillRL (Xia et al., 2026a)	✗	✗	✗	🔥	❄‡
Skill1 (Shi et al., 2026)	✗	●	✗	🔥	🔥†
ReSkill (He et al., 2026a)	●	✗	✗	🔥	❄‡
SkillMaster (Yang et al., 2026a)	✗	✗	✗	🔥	🔥†
SAGE (Wang et al., 2026)	✗	●	✓	🔥	🔥†
MetaClaw (Xia et al., 2026b)	●	✗	●	🔥	❄‡
Fully agentic					
SKT (Tan et al., 2026)	✓	✓	✓	🔥	✗
SkillGate (Li et al., 2026a)	✓	✓	✓	🔥	✗
CoEvoSkills (Zhang et al., 2026b)	✓	✓	✓	❄	❄
SkillEvolver (Zhang et al., 2026a)	✓	✓	✓	❄	❄
SkillClaw (Ma et al., 2026)	✓	✓	✓	❄	❄
SkillsVote (Liu et al., 2026)	✓	✓	✓	❄	❄
`Skill`^{2} (Ours)	✓	✓	✓	🔥	🔥‡

trains a separate writer against a fixed executor (Ouyang et al., 2026; Shen et al., 2026a). ❸ **Co-evolution** lets the policy and the skill library advance together within one loop (Xia et al., 2026a; He et al., 2026a; Yang et al., 2026a; Zhang et al., 2026d; Xia et al., 2026b), the line we draw most on. Benchmark studies meanwhile find that the benefit of skills concentrates on real tasks graded by executable verification (Li et al., 2026b; Han et al., 2026b;a; Jiang et al., 2026c). Across the three lines, most works remain misaligned with deployment in artifact, invocation, or validation, while the few fully agentic ones evolve only one side and lack co-evolution. Table 3 places representative works on both axes, and among them `Skill`^{2} is the only one that co-evolves a trained actor and a separately trained extractor in the fully agentic setting.

D BENCHMARKS

All seven benchmarks of §3.1 execute the agent's actions in a live environment and grade success by a programmatic check of the resulting state or answer, without a judge model. Unless stated otherwise, we use the complete released task set with its official environment and checker.

Terminal-Bench 2.1 (TB 2.1). Terminal-Bench (Merrill et al., 2026) evaluates agents on hard, realistic tasks in command-line interfaces, spanning software engineering, system administration, security, and scientific computing. Each of its 89 tasks provides an instruction, a Docker image, an oracle solution, and tests that check the final state of the container rather than the agent's commands. Version 2.1 (Merrill et al., 2026) keeps the tasks of version 2.0 and repairs those affected by drifting dependencies, mismatched resources, or instructions that contradict their tests.

OpenThoughts-TBLite (TBLite). OpenThoughts-TBLite (OpenThoughts-Agent team, 2026) is a curated set of 100 terminal tasks in the format of Terminal-Bench, built as a faster proxy whose scores track those of Terminal-Bench 2. Its tasks span nine categories, led by data processing and security, and are balanced in difficulty, with 40 easy, 26 medium, 26 hard, and 8 extreme tasks.

TerminalWorld-Verified (TW-Verified). TerminalWorld (Chu et al., 2026) derives 1,530 terminal tasks in 18 categories from public asciinema recordings of real terminal sessions, through an agent pipeline that rebuilds each environment in Docker and generates its tests. Its Verified subset holds 200 tasks, chosen for diversity and long command sequences and reviewed by hand.

InterCode-Bash (IC-Bash). InterCode (Yang et al., 2023) casts interactive coding as an environment with execution feedback. Its Bash split has 200 tasks adapted from NL2Bash (Lin et al., 2018) and run in an Ubuntu shell over four groups of file systems, and a task counts as solved when both the command output and the file-system changes match the gold solution.

WebArena-Verified (WA-V). WebArena (Zhou et al., 2024) provides 812 tasks on self-hosted websites, namely an online store and its administration panel, a forum, a GitLab instance, and a map service. WebArena-Verified (hattami et al., 2025) audits these tasks, repairs their evaluators and instructions, and replaces substring matching and the judge model with deterministic checks. We evaluate on all 812 tasks with the evaluators of WebArena-Verified, while the actor trains on WebShop-HTML rather than on the sites of WebArena.

MiniWoB++. MiniWoB++ (Liu et al., 2018) is a suite of synthetic web pages, each posing a short interaction task such as clicking a button, filling a form, picking a date, or operating an email widget. We use the 125 tasks integrated in BrowserGym (Chezelles et al., 2025), which renders each page in a real browser and counts an episode as solved when its programmatic reward is positive.

WorkArena-L1 (WoA-L1). WorkArena (Drouin et al., 2024) evaluates agents on common knowledge-work tasks on the ServiceNow enterprise platform, comprising 33 tasks with 19,912 unique instances over lists, forms, the knowledge base, the service catalog, menus, and dashboards. These tasks form level L1 of WorkArena++ (Boisvert et al., 2024), and each is validated by a task-specific function that inspects the state of the instance, such as the records the agent created.

E Experimental Details

E.1 Settings

Table 4 lists the settings that differ between the two domains and Table 5 the hyperparameters shared by both. The paragraphs below give what the tables leave out.

Environments. Terminal tasks run in containers managed by Harbor (Merrill et al., 2026), and the actor acts through Bash commands issued by mini-SWE-agent (Yang et al., 2024b). Each command starts a fresh shell in the working directory, while files and services persist across commands, and its output is cut to its first and last 4,000 characters. Web tasks run in Chromium driven through BrowserGym (Chezelles et al., 2025) and Playwright. The observation holds the task, the URL, the open tabs, the accessibility tree of the page with element identifiers, the previous actions, and the last error, without screenshots, and each turn issues one function call, either a browser operation on an element identifier or `invoke_skill`. In both domains the harness discloses the names and descriptions of the skill library, returns the body of an invoked skill as the observation, and keeps all other bodies out of reach of the environment tools. An episode ends when the task is submitted, after 48 turns, or when the time limit of the task, 900 seconds by default, runs out.

Data. Terminal training visits the 4,567 tasks of SETA (Shen et al., 2026b) once, in 142 batches of 32 and a final batch of 23, with the score of the task's verifier in $[0, 1]$ as reward. Web training visits 4,000 tasks of WebShop-HTML once, with the binary success of the purchase as reward. WebShop (Yao et al., 2022) provides both a text simulator, the setting of prior skill-learning work (Shi et al., 2026; Zhang et al., 2026d), and a web server that renders its full product catalog

as HTML pages. WebShop-HTML is the latter, opened in Chromium through BrowserGym, so the actor searches, compares products, sets options, and buys by acting on page elements, as on the other web benchmarks. Its tasks follow the frozen order of the official training range from goal index 1,500, skipping duplicate and invalid goals, and keep the original goals, attributes, options, and scoring. Each episode has its own session, while the catalog and search index are shared read-only. The probe pools are disjoint from training and from all seven benchmarks, deduplicated by intent, inputs, target state, and verifier. The terminal pool holds 768 tasks each from CalibForge (Meng et al., 2026) and TermiGen (Zhu et al., 2026), and the web pool holds 512 tasks from the development range of WebShop and 256 from WebArena-Infinity (Zhou, 2026). GLM-5.2 labels every pooled task at temperature 0 with a family group, a difficulty, and a quality verdict, keeping accepted tasks labelled with confidence at least 0.7. Each round draws one twelfth of every stratum of family and difficulty without replacement, over ten terminal and six web family groups.

Rollouts and Maintenance. The actor samples four natural rollouts per task, forks a control branch at every skill decision in them, and fills the rest of the group of eight with further natural rollouts. A branch starts from a clone of the container, or from a fresh copy of the site on which the shared prefix is replayed and checked, and a pair whose states do not match keeps only group-relative credit. Each task group of a batch forms one maintenance case, joined by at most two tasks of the same fine-grained family from the current batch and the last 12 rounds, and cases are edited in parallel sets of eight. At capacity, an insertion retires the least recently invoked skill.

Evaluation. Results use the final actor and skill library, frozen, with one rollout per task in each of ten runs, and pass@1 averages the runs. MiniWoB++ and WorkArena-L1 average over task types with ten paired seeds per type. Malformed actions and exhausted budgets count as failures. Trained baselines start from the same backbones and use RL on the same task pools and budgets, and retrieval-based baselines keep their own access to skills within the same budget.

E.2 Efficient Execution

Model computation and environment execution run as two separately scheduled pools, the former on 48 H100 GPUs split as in Table 5, training with Slime (Zhu et al., 2025) on Megatron (Shoeybi et al., 2020) and generating with SGLang (Zheng et al., 2024), and the latter on three CPU servers with 192 terminal or 48 web environment slots, joined by an asynchronous request layer. A rollout that waits for a tool thus leaves the GPUs to other rollouts, and an environment that waits for the model serves other episodes. Four overlaps shorten the critical path of a round. ❶ A control branch generates its first action while its environment copy is prepared from the cut point. ❷ The environments of the next batch and of the round's probes are prepared while the current rollouts and edits run. ❸ The eight cases of a maintenance set are edited in parallel against one draft, while evidence is organized in the background. ❹ Models stay resident across rounds, with weights synchronized at update

Table 4: Settings that differ between the two domains, with shared ones in Table 5.

Setting	Terminal	Web Navigation
Data		
Training tasks, visited once	SETA, 4,567	WebShop-HTML, 4,000
Tasks per batch	32	16
Outer iterations (K)	143	250
Probe pool	1,536	768
Probe tasks per round	128	64
Interaction		
Environment	mini-SWE-agent	BrowserGym
Observation limit	8,000 characters	16,384 tokens
Actor new tokens per turn	8,192	2,048
Time limit per command or action	60 s	30 s
Skill Library		
Library capacity	150	64
Maintenance sets per round (N)	4	2

Table 5: Hyperparameters shared by the two domains. The KL reference is the backbone.

Hyperparameter	Actor	Extractor
Generation		
Group size (G), edits per set (J)	8	8
Temperature, top-p	1.0, 1.0	0.8, 1.0
Context length (tokens)	139,264	262,144
New tokens per generation	Table 4	32,768 (512 to select)
Turns per episode	48	—
Skills read in full per case (M)	—	4
Limit of a description, a body	—	512 chars, 128 KiB
Optimization		
Optimizer	Adam, β=(0.9, 0.98), ϵ=10^{-8}	
Learning rate, schedule	5×10^{-7}, constant without warmup	
Weight decay, precision	0.1, BF16	
Ratio clipping (ε_{low}, $\varepsilon_{\text{high}}$)	(0.2, 0.28)	
KL coefficient	0.01	0.02
Credit Assignment		
Utility clip, advantage bound $A_{\max}$	—	$[-1, 1]$, 5
Structural penalty κ	—	0.1
Importance weight cap c	—	2
Credit wait W, policy age bound	—	16 rounds, 32 updates
Edits per update, probe repetitions Q	—	≤ 64, 1
Optimizer steps per round	1	≤ 1
Compute		
H100 GPUs to train, to generate	16, 20	8, 4

boundaries, and finished episodes are verified and released at once, so that a freed slot is refilled immediately. These overlaps keep the order of co-evolution, since the probes compare libraries under the updated actor and the next round starts only once the library is released.

F Additional Results

This section complements §3.3 and §3.4 with the full transfer results, an ablation of the mechanisms of both roles, and a sensitivity analysis of four hyperparameters that shape credit and updates.

F.1 Transfer to Frozen Models

Table 6 presents the full results behind Figure 3 on all seven benchmarks. The skill libraries evolved with Qwen3.5-9B are given to four frozen models larger than the actor, namely Qwen3.5-27B and the mixture-of-experts Qwen3.5-35B-A3B from the actor's family, and Gemma 4 31B-IT and Muse Glimmer 30B from other families, and each model is compared without a library, with the library of Skill1, and with that of `Skill`^{2}. As shown in Table 6, the library of `Skill`^{2} raises all four models on all seven benchmarks, by 3.0 to 4.6 points, whereas that of Skill1 adds 0.7 to 1.3. The gain is smallest for Muse Glimmer 30B, which leads the four without a library (53.2%), at 3.3 points on average against 3.9 for the two Qwen3.5 models. Notably, the mixture-of-experts model gains as much as the dense one of its family, which indicates that the transfer does not hinge on the architecture of the receiving model. The reference block further separates the library of Skill1 from its training. Loaded into the untrained Qwen3.5-9B, it adds only 1.0 point on average, against 5.0 for Skill1 trained with it, so most of the gain of Skill1 stays in the parameters it trains, whereas the library of `Skill`^{2} carries its gain over to models it never co-evolved with.

F.2 Ablation Study

To attribute the gains of `Skill`^{2} to its mechanisms, we remove one mechanism at a time with Qwen3.5-9B on all seven benchmarks, as summarized in Table 7. Each variant keeps both roles

Table 6: Transfer of the skill libraries evolved with Qwen3.5-9B to frozen models. Each cell reports pass@1 (%) averaged over 10 runs and its change relative to the same model without a library, where + Skill1 and + `Skill`^{2} load the library of that method, with gains marked by ↑. The last block lists Qwen3.5-9B for reference, where Skill1 and `Skill`^{2} are the trained systems of Table 1.

Model	Terminal				Web Navigation			Avg.
	TB 2.1	TBLite	TW-Verified	IC-Bash	WA-V	MiniWoB++	WoA-L1	
Qwen3.5-27B								
No library	36.3	56.3	33.9	53.5	28.3	66.2	53.3	46.8
+ Skill1	37.4↑1.1	57.6↑1.3	34.9↑1.0	54.2↑0.7	29.2↑0.9	67.1↑1.0	54.5↑1.2	47.8↑1.0
+ `Skill`^{2}	**40.9**↑4.6	**60.5**↑4.2	**37.6**↑3.7	**56.6**↑3.1	**31.9**↑3.7	**70.5**↑4.3	**57.3**↑3.9	**50.7**↑3.9
Qwen3.5-35B-A3B								
No library	35.7	54.8	33.4	51.9	27.6	65.8	51.8	45.9
+ Skill1	36.9↑1.1	56.0↑1.2	34.5↑1.1	52.8↑1.0	28.6↑1.0	67.0↑1.1	53.0↑1.2	47.0↑1.1
+ `Skill`^{2}	**40.0**↑4.3	**59.1**↑4.3	**37.1**↑3.7	**55.1**↑3.2	**31.2**↑3.6	**69.9**↑4.1	**55.8**↑3.9	**49.7**↑3.9
Gemma 4 31B-IT								
No library	35.1	55.1	34.0	52.1	31.0	65.4	50.3	46.1
+ Skill1	36.1↑1.0	56.4↑1.3	35.1↑1.1	53.1↑1.0	32.0↑1.0	66.7↑1.3	51.5↑1.2	47.3↑1.1
+ `Skill`^{2}	**39.0**↑3.9	**59.0**↑3.9	**37.2**↑3.2	**55.2**↑3.1	**34.4**↑3.4	**69.3**↑3.8	**54.2**↑3.9	**49.8**↑3.6
Muse Glimmer 30B								
No library	43.4	61.7	39.8	58.7	38.3	72.0	58.8	53.2
+ Skill1	44.6↑1.2	62.8↑1.1	40.6↑0.9	59.8↑1.1	39.4↑1.1	72.7↑0.7	60.0↑1.2	54.3↑1.0
+ `Skill`^{2}	**46.7**↑3.4	**65.4**↑3.7	**42.9**↑3.2	**61.7**↑3.0	**41.7**↑3.4	**75.0**↑3.0	**62.4**↑3.6	**56.6**↑3.3
Qwen3.5-9B (reference)								
No library	25.3	41.2	24.0	41.5	15.4	55.2	31.8	33.5
+ Skill1	26.3↑1.0	42.2↑1.0	25.1↑1.1	42.5↑1.0	16.3↑0.9	56.3↑1.1	33.0↑1.2	34.5↑1.0
Skill1	28.4↑3.1	45.9↑4.7	26.5↑2.5	43.6↑2.1	21.0↑5.6	61.6↑6.4	42.4↑10.6	38.5↑5.0
`Skill`^{2}	**36.3**↑11.0	**53.3**↑12.1	**35.2**↑11.2	**49.6**↑8.1	**24.2**↑8.8	**68.5**↑13.3	**47.9**↑16.1	**45.0**↑11.5

trained and the skill library evolving. On the side of the actor, *w/o Fork-GRPO* samples eight natural rollouts per task and trains them with group-relative advantages, and *w/o Decision Credit* keeps the forks and Eq. 2 but drops the paired advantage of Eq. 3. On the side of the extractor, *w/o Counterfactual Utility* values a candidate by its own mean probe reward instead of the paired difference of Eq. 8, *w/o Micro Credit* passes macro credit to all buffered edits rather than to the skills certified by their invocation gain, *w/o Delayed Credit* settles each edit in its own round, and *w/o Family-Aware Evidence* draws the supporting rollouts of a maintenance set from other attempts at the same task rather than other tasks of the same type.

The results reveal three observations. First, both parts of Fork-GRPO contribute. Removing it lowers the average from 45.0% to 43.1%, and keeping the forks without the decision credit recovers part of the loss (44.0%) and stays above *w/o Fork-GRPO* on all seven benchmarks, which indicates that the paired rollouts help by themselves while crediting the decision tokens adds a further 1.0 point. Second, the extractor depends most on what its credit measures and where the credit goes. Valuing a candidate by its own success rate costs 2.5 points and broadcasting credit to uncertified edits costs 3.0, the two largest drops, while crediting only the edits of the current round costs 1.9. The two kinds of paired evidence are thus useful training signals, though they do not identify the contribution of any single edit, since the edits of one revision or of one skill share their credit. Third, evidence from other tasks of the same type helps distillation, as replacing it with other attempts at the same task costs 1.5 points. Collectively, the full model beats all six variants on all seven benchmarks, which, with the role-level comparison of Figure 1 (bottom right), ties the gains of `Skill`^{2} to both roles and to the mechanisms that train them rather than to individual edits.

F.3 Sensitivity Analysis

We further examine four hyperparameters that shape the credit and the policy updates, namely the importance cap c of the extractor objective (Eq. 9), the clip range $(\varepsilon_{\text{low}}, \varepsilon_{\text{high}})$ shared by both roles

Table 7: Ablation of the mechanisms of Skill^2 with Qwen3.5-9B. Each cell reports pass@1 (%) averaged over 10 runs and its change relative to the full model, with drops marked by ↓. Each variant removes one mechanism and keeps both roles trained (Appendix F.2).

Variant	Terminal				Web Navigation			Avg.
	TB 2.1	TBLite	TW-Ver.	IC-Bash	WA-V	MiniWoB++	WoA-L1	
Skill^2 (full)	**36.3**	**53.3**	**35.2**	**49.6**	**24.2**	**68.5**	**47.9**	**45.0**
Actor								
w/o Fork-GRPO	34.4↓1.9	51.4↓1.9	33.2↓2.0	48.5↓1.1	22.4↓1.7	66.8↓1.7	45.2↓2.7	43.1↓1.9
w/o Decision Credit	35.1↓1.2	52.2↓1.1	34.2↓1.0	48.7↓0.9	23.5↓0.7	67.4↓1.1	47.0↓0.9	44.0↓1.0
Extractor								
w/o Counterfactual Utility	34.0↓2.2	50.8↓2.5	32.7↓2.5	47.5↓2.1	22.3↓1.8	65.7↓2.8	44.2↓3.6	42.5↓2.5
w/o Micro Credit	33.9↓2.4	50.0↓3.3	32.4↓2.8	47.1↓2.5	21.6↓2.5	65.2↓3.3	43.9↓3.9	42.0↓3.0
w/o Delayed Credit	34.5↓1.8	51.5↓1.8	33.4↓1.8	48.2↓1.4	22.9↓1.2	66.3↓2.2	44.8↓3.0	43.1↓1.9
w/o Family-Aware Evidence	34.8↓1.5	52.1↓1.2	33.8↓1.4	48.2↓1.4	23.3↓0.9	66.7↓1.8	45.8↓2.1	43.5↓1.5

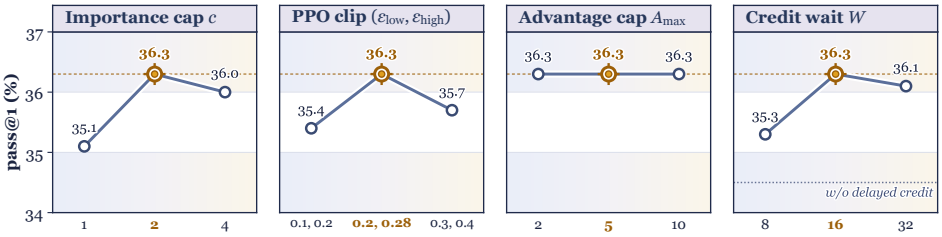
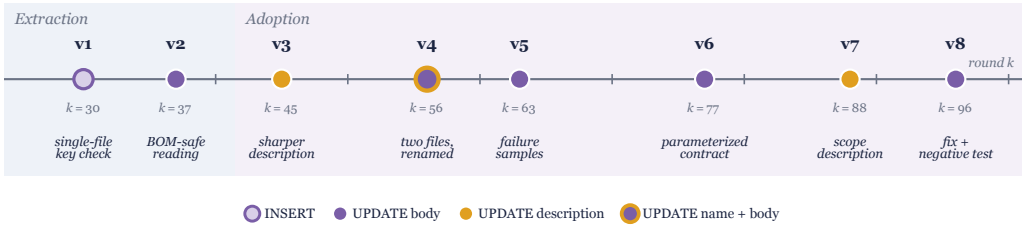


Figure 5: **Sensitivity to four hyperparameters.** Pass@1 on Terminal-Bench 2.1 with Qwen3.5-9B when one hyperparameter takes three values around its default (gold) and the others stay fixed. The dashed line marks the default, and the dotted line in the last panel, *w/o Delayed Credit* (Table 7).

(Eqs. 5 and 10), the bound $A_{\max}$ of the macro credit, and the wait limit W on the rounds an edit waits for certification before it expires. Each takes three values around its default while the others stay fixed, with Qwen3.5-9B on Terminal-Bench 2.1. As shown in Figure 5, the defaults reach the best result in each panel (36.3%), and the eight other configurations stay within 1.2 points of it. A tight importance cap (c=1) costs the most (35.1%), as it truncates more of the weights of edits sampled by earlier extractors, while c=4 stays close to the default (36.0%). Both a narrower and a wider clip range lower the result (35.4% and 35.7%), and $A_{\max}$ leaves it unchanged from 2 to 10, consistent with normalized utilities that rarely reach the bound. A short wait (W=8) costs 1.0 point yet still exceeds *w/o Delayed Credit* on the same benchmark (34.5%), so credit arriving a few rounds late is worth waiting for, while W=32 adds nothing further (36.1%). Collectively, Skill^2 is robust around its defaults, which all experiments use.

G Case Study

We study one theme, checking the records and keys of CSV exports, a recurring step of the terminal tasks. Figure 6 traces how one skill of Skill^2 evolves over eight versions. Its interface grows from a check of one key in one file at round 30 to a comparison of two files at round 56, when the extractor also renames it, and to a parameterized contract auditor at round 77, while two edits of its description keep the metadata in step with its scope. The rewrite at round 77 drops the negative test and lets an empty comparison path pass, and the last edit at round 96 repairs both, so revision corrects the skill as well as extending it. Figure 7 shows the final skill as the actor loads it. Figure 8 sets it beside the skills that SkillGate, Skill1, and SkillRL hold on the same theme and follows one task through each. The script of SkillGate has to be edited before it can compare a CSV with a TSV on a composite key, and the guidance of Skill1 and SkillRL has to be turned into a checker by the actor, whereas the tool of Skill^2 handles the task in one call, without any edit to its code.



How the actor calls the skill

```
v1  python3 "$work/check_csv.py" "$CSV_FILE" "$KEY_FIELD"      one file, one key
v4  python3 /the/printed/path/compare_keys.py left.csv right.tsv \
        --key id --right-delimiter tab                          two files, a delimiter each
v8  python3 "$work/csv_contract.py" "$work/left.csv" \
        --columns '["id","note"]' --rows 2 --key id --unique \
        --against "$work/right.tsv" --against-delimiter tab    a contract on columns, rows, and keys
```

v1 k=30 · INSERT

one file, one key

```
+ reader = csv.reader(stream, strict=True)
+ if not value.strip() or value in seen:
+     raise SystemExit(f"record {count}: empty or
    duplicate key")
```

v2 k=37 · body

a BOM no longer hides the key

```
- with open(sys.argv[1], encoding="utf-8",
    newline="") as stream:
+ with open(sys.argv[1], encoding="utf-8-sig",
    newline="") as stream:
- Use UTF-8 without a BOM here.
+ UTF-8 may have a BOM.
```

v3 k=45 · description

the metadata names the job

```
- Use when one CSV needs a unique key; provides
    logical-record validation
+ Use when quoted CSV records need key validation
    before a join; provides key-record checks;
    choose when line counts are misleading
```

v4 k=56 · name + body

two files, renamed

```
- name: check-csv-key-records
+ name: build-delimited-contract-auditor
+ p.add_argument("right")
+ p.add_argument("--right-delimiter", choices=[",",
    ";", "tab"])
```

v5 k=63 · body

failures point to records

```
+ print("left-only sample:", sorted(missing)[:3])
+ print("right-only sample:", sorted(extra)[:3])
+ Read the small key sample to locate the missing
    record.
```

v6 k=77 · body

a parameterized contract, and two regressions

```
+ parser.add_argument("--key", action="append",
    default=[])
+ parser.add_argument("--unique",
    action="store_true")
+ parser.add_argument("--columns", ...), ("--rows",
    ...), ("--against", ...)
+ MAX_BYTES = 32 * 1024 * 1024
- Remove a key from the copy; exit 1 is expected.
! if args.against:   an empty path skips the
    comparison
```

v7 k=88 · description

the metadata names the scope

```
- Use when quoted CSV records need key validation
    before a join
+ Use when delimited reports need reliable schema
    or key checks; provides explicit structural and
    key-coverage evidence
```

v8 k=96 · body

both regressions repaired

```
+ if args.against == "":
+     parser.error("--against requires a nonempty
    file path")
+ if python3 "$work/csv_contract.py"
    "$work/left.csv" --key id \
+     --against "$work/missing.csv"; then
+     echo "ERROR: missing key was not detected"
    >&2
+ test "$rc" -eq 1
```

Figure 6: **Evolution of one skill of `Skill`^{2}.** Top, its eight versions on the round axis of Figure 4, coloured by the edit that produced them. Middle, how the actor calls the skill at three versions. Bottom, what each edit changed, verbatim, with v6 dropping the negative test and letting an empty `--against` skip the comparison, and v8 repairing both.

build-delimited-contract-auditor/SKILL.md v8, after round 96

METADATA • disclosed to the actor before any invocation

name build-delimited-contract-auditor

description use when delimited reports need reliable schema or key checks; provides explicit structural and key-coverage evidence; choose when quoted fields or reordered records make line-based checks misleading; avoid when the input uses an undocumented custom dialect.

BODY • loaded when the actor invokes the skill

```
## Preconditions
Python 3.9+. Use a private writable directory. Inputs are stable,
    regular UTF-8 CSV/TSV files; a BOM is OK. Know the delimiter
    first. Read logical records, not lines. This version allows 32 MiB
    per file, 100,000 data records and 1,048,576 characters per field.
    For larger inputs, change the budget deliberately before running.
    ❶
## Procedure
Create the checker below. Save the printed path. A later Bash call
    may start a fresh shell, so set work again there. The checker only
    reads the input files. ❷
~~~bash
set -eu
work="$(mktemp -d)"
cat > "$work/csv_contract.py" <<'PY'
.
.
.
def audit(path, delimiter, keys, unique, columns=None,
    expected_rows=None):
    .
    .
    .
        reader = csv.reader(stream, delimiter=delimiter, strict=True)
            ❸
    .
    .
    .
    parser.add_argument("--key", action="append", default=[]) ❹
    parser.add_argument("--unique", action="store_true")
    parser.add_argument("--columns", help="JSON array; checked on the
        first file")
    parser.add_argument("--rows", type=int, help="data records in the
        first file")
    parser.add_argument("--against", help="require the same key set")
    .
    .
    .
    if args.against == "":
        parser.error("--against requires a nonempty file path")
    .
    .
    .
PY
~~~
## Invocation
Give the delimiter for each input. Repeat --key for a composite key.
    [...] Keep IDs as strings: 001 is not 1. [...]
python3 "$work/csv_contract.py" "$work/left.csv" \
    --columns '["id","note"]' --rows 2 --key id --unique \
    --against "$work/right.tsv" --against-delimiter tab
## Verification
Exit 0: requested checks passed. Exit 1: a contract mismatch. Exit 2:
    bad options, unreadable/malformed input, or a limit was hit. Also
    run a missing-key case. It should exit 1, not just print a
    message. ❺
if python3 "$work/csv_contract.py" "$work/left.csv" --key id \
    --against "$work/missing.csv"; then
    echo "ERROR: missing key was not detected" >&2
    .
    .
    .
    test "$rc" -eq 1
## Recovery
For exit 2, check the path, delimiter and limits first. For exit 1,
    read the small sample of issues and fix the producer, then rerun.
    Leave the inputs in place; matching keys still says nothing about
    totals or business rules. ❻ [...]
```

❶ **Explicit budget.** The limits are stated, and a larger input calls for a new budget rather than a partial check.

❷ **Reusable tool.** The skill writes its checker once and tells the actor how to reach it from a later shell.

❸ **Record-level parsing.** A strict reader over logical records handles quoted fields and reordered records, the case its description names.

❹ **Parameterized contract.** Composite keys, uniqueness, column order, record count, and a second file are options of one call.

❺ **Tested failure.** Exit codes separate a mismatch from a bad input, and a missing-key case has to return 1.

❻ **Safe recovery.** Fixes go to the producer of the data, and a key match is not read as a match of totals.

Figure 7: **The final skill.** Version v8 of the skill in Figure 6, abridged, with [...] and ⋮ marking omitted text and code. The actor sees only the metadata until it invokes the skill, and the body is then loaded as the observation. The tags mark the four clauses of the description, and the markers ❶–❻ link parts of the body to the notes on the right.

SkillGate fixed `SKILL.md` with a script

```
---
name: check-report-key-coverage
description: Read when a CSV export must have unique
   IDs and the same ID coverage as another export.
   Handles quoted records and literal string keys;
   does not verify numeric totals.
---
Use this for two header-bearing CSV files. The key
   column must be specified by the task. If the task
   asks for a composite key or a custom dialect, adapt
   the key extraction deliberately before using the
   result. [...]
: "${LEFT:?set first CSV}" "${RIGHT:?set second CSV}"
   "${KEY:?set key column}"
python3 - "$LEFT" "$RIGHT" "$KEY" <<'PY'
def keys(path, column):
    with open(path, encoding="utf-8-sig", newline="")
       as f:
        reader = csv.reader(f, strict=True)
        .
        .
        .
            if not value.strip() or value in result:
                raise ValueError(f"record {number}:
                   empty or duplicate key")
.
.
.
print("left_only", len(a-b), "right_only", len(b-a))
raise SystemExit(1 if a != b else 0)
PY
Exit 0 means unique keys with equal coverage. Exit 1
   means a coverage difference, and exit 2 means
   invalid input. [...]
```

Skill1 scenario and strategy

```
{
  "scenario_desc": "Two CSV exports contain quoted
     text fields and may use a different row order;
     verify whether the same record IDs survived
     processing.",
  "strategy": "Parse both files with Python csv rather
     than split or wc. Read the header and find the
     key column, checking duplicate headers and each
     record's width. Keep keys as strings so 001 is
     not changed to 1. Count duplicate and empty keys
     before comparing the key sets. Report keys
     missing on either side, not differences in line
     order. Test the check by removing one known key
     from a copy. Leave the original files unchanged;
     matching IDs does not prove matching numeric
     values."
}
```

SkillRL general and task-specific rules

```
"general_skills": [{ "title": "Parse before counting",
  "principle": "Use a format-aware parser before
     counting records or comparing fields. A CSV
     record may contain commas and newlines inside
     quotes.",
  "when_to_apply": "A terminal task checks or converts
     a delimited report." }],
"task_specific_skills": { "data-processing": [
  { "title": "Check key coverage",
    "principle": "Read the header, preserve identifier
       strings, and compare key sets independently of
       row order. Check duplicate keys separately when
       uniqueness is required.",
    "when_to_apply": "A join or export may have lost
       records." },
  { "title": "Verify the checker",
    "principle": "Use a reordered positive pair and a
       missing-key negative pair. A script that prints
       a difference but exits successfully can let a
       broken pipeline continue.",
    "when_to_apply": "A new validation command will
       decide whether a report is acceptable." } ] }
```

Skill² evolved `SKILL.md` with a tool (Figure 7)

```
---
name: "build-delimited-contract-auditor"
description: "Use when delimited reports need reliable
   schema or key checks; provides explicit structural
   and key-coverage evidence; choose when quoted
   fields or reordered records make line-based checks
   misleading; avoid when the input uses an
   undocumented custom dialect."
---
## Procedure  [creates a reusable tool, 114 lines]
usage printed by the tool:
csv_contract.py [-h] [--delimiter {,,;,|,tab}] [--key
   KEY]
                [--unique] [--columns COLUMNS] [--rows
                   ROWS]
                [--against AGAINST]
                [--against-delimiter {,,;,|,tab}]
                file
## Verification
Exit 0: requested checks passed. Exit 1: a contract
   mismatch. Exit 2: bad options, unreadable/malformed
   input, or a limit was hit. Also run a missing-key
   case. [...]
```

📋 **One task, four skills** Compare `a.csv` with `b.tsv` on the key (`id`, `region`), require unique keys, and fail on a missing key.

SkillGate	edit the script	It reads commas only, takes one key column, and always requires unique keys, as its own text warns ("adapt the key extraction deliberately").
Skill1	write a checker	The strategy says what to check, and the actor still has to write, run, and test the code in the episode.
SkillRL	write a checker	The rules say what to check, and the actor still has to write, run, and test the code in the episode.
Skill²	one call	`python3 "$work/csv_contract.py" a.csv --key id --key region --unique --against b.tsv --against-delimiter tab` exits 1 and lists the missing key, for instance `[["003", "north"]]`.

Figure 8: **Skills on the same theme.** Top, the skills that SkillGate, Skill1, SkillRL, and **Skill²** hold on checking CSV exports in the 9B setting, abridged, where the skill of SkillGate comes from its fixed library. Bottom, what the actor has to do with each of them on one task, where the call of **Skill²** was checked on a pair of toy files.

H LIMITATIONS

One Harness. All methods are trained and evaluated inside the skill interface of OpenHands. Heavier harnesses such as Codex CLI and Claude Code add long system prompts, many tools, and sub-agents, which multiply the context and the cost of each rollout, and running hundreds of forked rollouts and probe episodes per round inside them is beyond our compute. Their skills follow the same specification of names, descriptions, and bodies disclosed on demand (Agent Skills), so the evolved libraries can be loaded there, and training inside them is left to future work.

Credit Attribution. We do not claim to identify the contribution of each edit. The macro level intervenes on a whole revision and the micro level on an invocation, so an edit shares its credit with the other edits of its revision and of its skill, an ineffective edit can be credited alongside an effective one, and a historical edit that receives micro credit is one whose skill helped the actor when invoked, which is evidence of use rather than proof that the edit improved the skill. The ablations show that this credit trains the extractor better than broadcasting it or settling it within the round, while per-edit counterfactuals would need one probe pass per edit and are left to future work.

Model Scale. We train backbones of up to 9B parameters, since one run occupies 48 GPUs for its whole schedule and a larger backbone enlarges both the actor and the extractor. The transfer to frozen models of 27B to 35B parameters (§3.3) suggests that the evolved libraries stay useful at larger scales, while co-evolving them with such models remains untested.

I PROMPTS OF THE EXTRACTOR

Figures 9 and 10 give the system prompts of the two steps of an editing operation, selecting the skills to read in full (Eq. 6) and writing the operation (Eq. 7). The user message of each step carries the maintenance set as JSON, the task, the trajectories with their tool interactions and rewards, and the metadata of the working draft, together with the bodies of the selected skills in the second step. With family-aware evidence, the second sentence of the editing prompt is replaced by *"The input explicitly separates task instances grouped by an offline task-type annotation. Compare common mechanisms and varying inputs; relatedness is a hypothesis, not proof of one shared solution."*.

Skill selection prompt

```
Select the existing skills whose complete content is needed before editing this SkillBank.
Return exactly one JSON object with key selected_skill_names and zero to 4 unique names
copied from the supplied cards. Select same-theme or plausibly reusable skills; do not
invent names and do not emit prose. Each trajectory reward is in the inclusive [0,1]
range, where higher means better verified task completion. Prioritize Cards needed to
diagnose low and zero scores, while using high scores as positive controls for behavior to
preserve. Interpret each Card as trigger, reusable outcome, key discriminator, and
exclusion boundary. Match the trajectory's scenario family and observed state, not literal
path, number, filename, or command overlap. INSERT and UPDATE have equal priority. Select
an existing Card only when its capability and applicability genuinely match the evidence,
not merely its tool or topic. Return an empty selection when this capability is missing,
so the editor can INSERT it. Never force an unrelated UPDATE or duplicate a skill. A
genuinely matching Skill with a verbose or misleading description may need complete-detail
inspection for description-only UPDATE; do not require its Card to be perfect before
selecting it. Do not infer hidden capabilities. If skill_use_feedback names a Skill the
Actor actually read, inspect its full body to diagnose usefulness or a routing mismatch.
This does not authorize replacing that Skill with an unrelated capability; INSERT the
missing capability instead.
```

Figure 9: **System prompt for selecting skills to read.** The number of skills is the limit M of Eq. 6.

Editing prompt

```
Maintain useful terminal capabilities, not Skill counts. The supplied trajectories are
repeated attempts at ONE task instance, not independent examples of generality. Extract
ONE reusable operation or decision rule: what varies between instances, what must remain
true, and how the result is checked. A broad title over a task-specific script is not
generalization. A small complete capability beats an entire copied solution.

INSERT a missing capability; UPDATE only a genuinely matching one, with equal priority and
reward weight. Preserve supported scenarios and interfaces. KEEP when no justified change
exists. A description-only UPDATE is valid if the name and body remain correct; selecting
a Card does not oblige UPDATE.

For both INSERT and UPDATE, description only routes usage: Use when <scenario>; provides
<capability/result>; choose when <observable discriminator>; avoid when <real exclusion>.
Mention each fact once. No commands, procedure, tests, paths, example values or invented
exclusions; necessary technical discriminators may remain. The soft target is not a
minimum. Reconcile the final description with the actual body, including KEEP fields;
remove promises the implementation cannot deliver.

Return one <SKILL2_EDIT_V2> envelope without thinking or commentary. Follow metadata order
and end with </SKILL2_EDIT_V2> on its own line. CONTENT is raw Markdown: ## Preconditions,
## Procedure, optional ## Assets, ## Verification, and ## Recovery for mutation. Close
code fences, functions and heredocs.

In Preconditions, define caller inputs, output and supported boundary. In Procedure,
implement the reusable core with those SAME inputs, not a fixed project replacement.
Paths, users, host patterns, database/table names, keys, thresholds and expected counts
from the task are inputs, even when all attempts repeat them. Standard paths can identify
a format, but a mounted target root must not silently become the live host root. Preserve
unspecified data and valid existing behavior.

Give a complete command or useful self-contained Bash/Python asset. Define and import
everything used; keep creation, invocation and verification interfaces consistent. No
undefined helper, private test suite, bare placeholder or shell variables assumed to
survive another call. For a new tool, show Bash creating the complete code and invoking it
with caller inputs. For a useful small model, include fitting inputs, saving/loading,
inference and validation. Do not train one just to add complexity. Use the smallest useful
implementation.

For state changes, save the original BEFORE mutation. Stage and validate before
replacement; a direct heredoc overwrite is not atomic. Keep changes scoped to the
requested records or hosts. Restore the actual original on failure, not an uncreated
backup or a known-broken implementation. Respect offline/security constraints.

Verification must make failure observable. Check the promised semantic result AND what
must remain unchanged using the same inputs; a fixed fixture count, printed status or zero
exit after swallowing an exception is not proof. A checker may itself be the useful
capability. Do not claim execution or generality beyond the evidence. Every sentence and
asset must remove concrete work or errors; length, file count and complexity earn no
reward.
```

Figure 10: **System prompt for writing an operation.** The output is one <SKILL2_EDIT_V2> envelope with a KEEP, INSERT, or UPDATE operation, its skill body laid out as in the **Skill**^{2} card of Figure 8.