EVOLVING-RL: END-TO-END OPTIMIZATION OF EXPERIENCE-DRIVEN SELF-EVOLVING CAPABILITY WITHIN AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Experience-driven self-evolving agents aim to overcome the static nature of large language models by distilling reusable experience from past interactions, thus enabling adaptation to novel tasks at deployment time. This process places substantial demands on the foundation model's capacities for abstraction, generalization, and in-context learning. However, existing methods either rely on external system-level designs or optimize only isolated components, without jointly improving the underlying model's capabilities to extract and utilize experience. To this end, we propose **Evolving-RL**, an algorithmic framework that jointly improves experience extraction and utilization. Specifically, we center the learning process on experience extraction and evaluation, using the two supervisory signals derived from evaluation to optimize the extractor and solver separately and thus enable their coordinated co-evolution. Experiments on ALFWorld, ScienceWorld, and Mind2Web demonstrate that Evolving-RL substantially improves experience extraction and reuse, enabling more effective experience-assisted out-of-distribution (OOD) adaptation. Under the same skill-conditioned protocol, it achieves 98.7% relative improvement over GRPO on ALFWorld unseen tasks and 35.8% in overall action accuracy on Mind2Web, together with a 6.3-point OOD score improvement on ScienceWorld. Furthermore, Evolving-RL also serves as an experience-augmented RL algorithm: the Evolving-RL-trained Qwen2.5-7B-Instruct model substantially outperforms its GRPO counterpart on ALFWorld and Mind2Web even without test-time skill injection, demonstrating benefits for both explicit skill reuse and the underlying policy. The code will be released after the review process.

1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable capabilities across a broad range of tasks, including complex reasoning (Huang & Chang, 2023; Qiao et al., 2023; Kojima et al., 2022; Wei et al., 2022) and autonomous agent decision-making (Wang et al., 2024a; Guo et al., 2024; Yao et al., 2023). However, once trained, LLMs are largely static: they lack the ability to continually adapt themselves to the complex out-of-distribution environments and tasks encountered during deployment. This fundamental limitation has motivated a growing body of research into test-time self-evolution (ang Gao et al., 2026; Xiang et al., 2026; Shinn et al., 2023; Zhao et al., 2023; Agrawal et al., 2026). As a prominent direction within this paradigm, experience-driven self-evolving agents accumulate reusable experiences from prior interactions and leverage them to solve related future tasks, thereby progressively enhancing their deployment-time capabilities.

Existing experience-driven self-evolution approaches primarily focus on system-level evolutionary design, with considerable research devoted to experience representation (Yang et al., 2024b; Zheng et al., 2025; Wang et al., 2024b; Ouyang et al., 2026), formulation, and management mechanisms (Cao et al., 2026; Tang et al., 2025; Yang et al., 2025). While these prompt-based methods have successfully demonstrated that injecting past experiences can significantly enhance downstream decision-making, their effectiveness is ultimately bounded by the underlying model's ability to extract and leverage these experiences (Shao et al., 2026)—a process that heavily relies on the model possessing robust in-context learning (Brown et al., 2020) and abstract reasoning capabilities.

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.

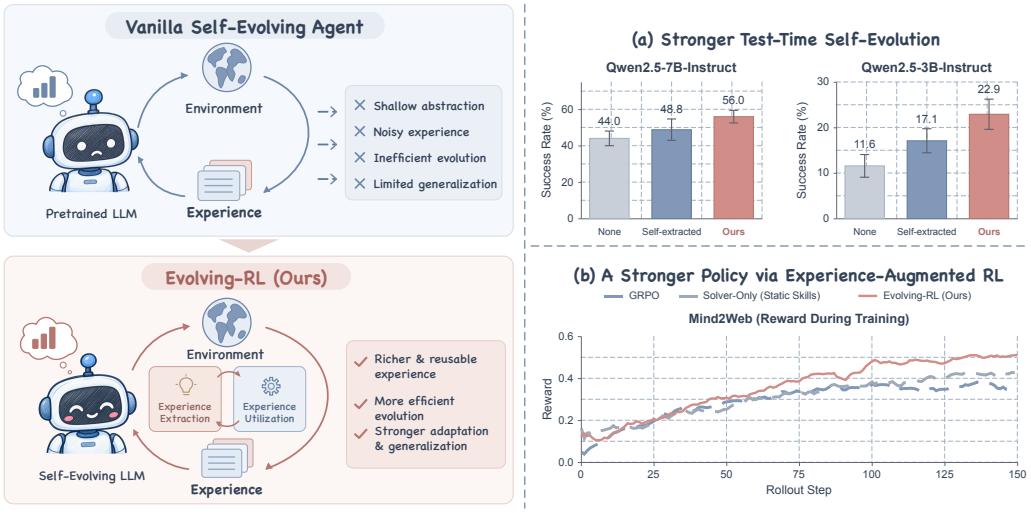


Figure 1: **Overview of Evolving-RL.** *Left:* Evolving-RL jointly optimizes experience extraction and utilization for more effective experience-driven self-evolution. *Right:* (a) Improved skill extraction and transfer across solver models. *None*, *Self-extracted*, and *Ours* denote no skills, skills extracted by the solver itself, and skills extracted by Evolving-RL, respectively. For each solver model, target tasks are fixed and both extractors receive identical source trajectories and feedback. (b) Online skill-conditioned training improves reward more rapidly over rollout steps than GRPO and solver-only training with static skills.

Several recent studies (Wu et al., 2025; Xia et al., 2026; Li et al., 2026) have explored reinforcement learning as a way to enhance the model's ability to utilize experience. However, these methods do not optimize self-evolution as a unified process. They improve only the utilization phase, while relying on stronger external models or hand-crafted filtering mechanisms to ensure the quality of extracted experience. Such a decoupled design is inherently limited, as extraction and utilization are not merely complementary capabilities, but mutually constitutive during learning. The quality of extracted experience directly influences the reinforcement learning dynamics for experience utilization. When the provided experience is noisy, conditioning on it introduces variance and ambiguity into the optimization process. Consequently, the policy often converges to a degenerate behavior: it learns to systematically ignore the provided experience, as bypassing unreliable guidance yields more stable returns than attempting to exploit it.

To bridge this gap, we propose **Evolving-RL**, a unified algorithmic framework that jointly optimizes both experience extraction and utilization within a shared policy. To make this joint optimization tractable and interpretable, we specifically instantiate experience as textual skills—compact procedural abstractions that capture actionable regularities regarding what to do, when to intervene, and how to recover from failures. Anchored by this representation, Evolving-RL employs an extractor-centric design: for each source interaction, the shared policy generates candidate textual skills. All candidates are evaluated on the same set of retrieved downstream tasks and receive rewards according to their average downstream performance. Concurrently, the trajectories collected during this evaluation are reused to refine the model's skill-utilization capability. This evaluation-centered design provides comparable transfer feedback for extractor optimization and skill-conditioned trajectories for solver optimization within the same training loop.

We evaluate on ALFWorld (Shridhar et al., 2021), Mind2Web (Deng et al., 2023), and ScienceWorld (Wang et al., 2022). Empirical results show that Evolving-RL substantially improves the model's ability to explicitly extract and reuse skills, leading to stronger experience-assisted OOD adaptation. Under the same skill-conditioned protocol, Evolving-RL boosts the success rate on ALFWorld unseen tasks from 44.6% for GRPO (Shao et al., 2024) to 88.6%, increases overall action accuracy on Mind2Web from 22.73% to 30.87%, and raises the ScienceWorld OOD score from 10.99 to 17.29. Moreover, with source trajectories and the solver model held fixed, skills extracted by Evolving-RL provide larger gains than self-extracted skills, demonstrating improved extraction quality and skill

transfer across solver models (Figure 1(a)). Iterative evaluation on ALFWorld further shows that the trained model effectively refines its skills through repeated interaction. Evolving-RL also strengthens the underlying policy. Without test-time skill injection, the Evolving-RL-trained Qwen2.5-7B-Instruct model achieves an 81.1% success rate on ALFWorld unseen tasks and a 28.05% overall action accuracy on Mind2Web, substantially outperforming the corresponding GRPO baselines of 33.7% and 22.83%, respectively. These results demonstrate its value both for experience-driven self-evolution and as an experience-augmented RL algorithm (Figure 1(b)).

2 Related Work

2.1 Experience-Driven Self-Evolving Agent

Experience-driven self-evolution enables agents to continuously accumulate and reuse knowledge, overcoming the static nature of their post-training capabilities (Zhao et al., 2023; ang Gao et al., 2026). Existing literature has advanced this paradigm across multiple dimensions. In terms of experience representation, prior studies have explored diverse granularities, ranging from raw interaction trajectories (Zhou et al., 2025; Yang et al., 2024b) and reusable workflows (Wang et al., 2024b) to executable skills (Zheng et al., 2025; Alzubi et al., 2026; Zhou et al., 2026) and strategic principles (Ouyang et al., 2026). Parallel efforts have also developed mechanisms for autonomous experience extraction (Alzubi et al., 2026; Ma et al., 2026; Shen et al., 2026) and efficient memory management (Tang et al., 2025; Cai et al., 2025; Yang et al., 2025). Despite these advancements, these systems predominantly rely on the base model's pre-existing ability to extract and utilize effective experience. Consequently, the benefits of experience-driven evolution remain fundamentally bounded by the underlying model's intrinsic capacities (Shao et al., 2026).

Recent methods such as EvolveR (Wu et al., 2025), SkillRL (Xia et al., 2026), and D2Skill (Tu et al., 2026) employ reinforcement learning to enhance the underlying model's capabilities. However, they primarily optimize experience utilization, while experience extraction remains dependent on hand-crafted filtering mechanisms or stronger external models. Concurrent with our work, Skill1 (Shi et al., 2026b) and Complementary RL (CRL) (Muhtar et al., 2026) independently explore joint experience–policy evolution: Skill1 trains a shared policy with distillation rewards derived from outcome variation, while CRL trains a separate extractor using the outcomes of experience-guided execution. Evolving-RL evaluates candidate skills on an identical set of retrieved downstream tasks and reuses the evaluation trajectories to jointly train extraction and utilization within a shared policy, directly aligning skill comparison with cross-task utility.

2.2 Experience-Augmented Reinforcement Learning

Enhancing experience utilization through reinforcement learning inherently constitutes an experience-augmented RL paradigm, in which experience collected from prior interactions is reused to guide subsequent policy optimization (Shi et al., 2026a; Xia et al., 2026; Li et al., 2026). Accordingly, most existing studies in this line of research primarily focus on how experience can improve reinforcement learning and policy optimization, while differing in the forms of experience they construct and incorporate. ERL (Shi et al., 2026a) and RetroAgent (Zhang et al., 2026) derive task-specific reflections from interaction trajectories and use them to augment the learning process. Other methods, such as EvolveR (Wu et al., 2025), SkillRL (Xia et al., 2026), and D2Skill (Tu et al., 2026), distill trajectories into reusable experience libraries that guide policy learning. Collectively, these studies demonstrate that incorporating experience into RL can improve training efficiency and policy performance, with concurrent work such as CRL (Muhtar et al., 2026) also allowing the experience itself to evolve together with the policy. However, their primary objective remains improving RL through experience augmentation, with less emphasis on whether the resulting model itself becomes better at extracting, transferring, and continually refining experience for self-evolution. Our work studies both aspects: the contribution of dynamically evolving skills to policy learning and the resulting model's ability to extract, transfer, and refine experience at test time.

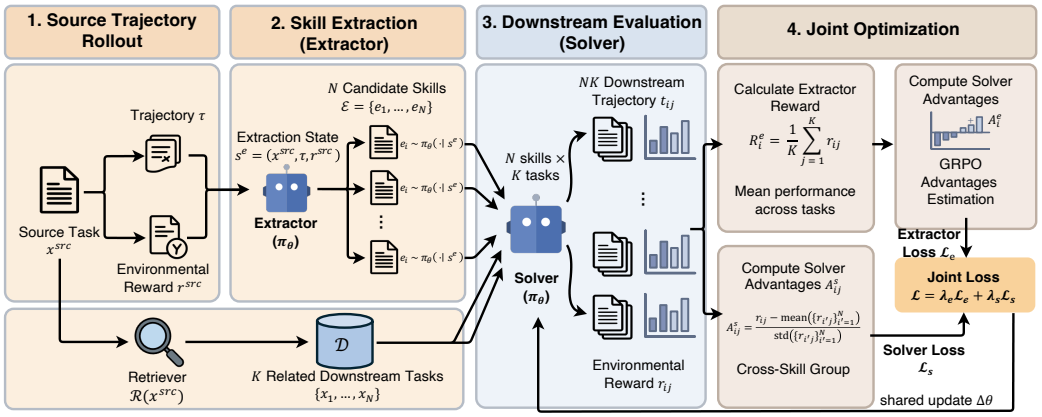


Figure 2: Overview of Evolving-RL. The framework begins with online skill extraction, followed by downstream evaluation of the extracted skills. This evaluation process produces training signals for both the extractor and the solver, enabling their joint optimization within a unified co-evolutionary framework.

3 Evolving-RL

3.1 Framework Overview

We propose **Evolving-RL**, a unified algorithmic framework that jointly improves the model's skill extraction and skill utilization capabilities within a single shared policy. As illustrated in Figure 2, our method generates skills online and evaluates them on downstream tasks. This process produces two coupled learning signals. Aggregated downstream rewards across multiple tasks supervise the extracted skills, while the skill-conditioned trajectories collected during downstream evaluation are reused to train skill utilization. The downstream evaluation rollouts therefore contribute to both objectives, allowing the two capabilities to co-evolve within a shared policy.

Roles and shared policy. Our method involves two key roles, the **extractor** and the **solver**, which share the same policy parameters. The **extractor** takes a completed source interaction as input and produces a compact textual *skill*—a procedural abstraction that encodes reusable decision rules, workflow steps, and error-handling strategies extracted from that experience. The **solver** takes a new task together with an injected skill and produces an action trajectory.

Rollout structure. Each training iteration begins by solving a source task $x^{\text{src}} \sim \mathcal{D}$ without any injected skill. The solver, parameterized by the shared policy π_θ, interacts with the environment conditioned on x^{src} and produces a trajectory:

$$\tau \sim \pi_\theta(\cdot \mid x^{\text{src}}).$$

The resulting trajectory τ and environment reward r^{src} form the extraction state $s^e = (x^{\text{src}}, \tau, r^{\text{src}})$, which conditions the extractor to produce N candidate skills. A retriever $\mathcal{R}$ then fetches K semantically related downstream tasks, and for every (skill, task) pair the solver rolls out a skill-conditioned trajectory. This structure—source solving → skill extraction → downstream evaluation—produces two coupled sample families within a single iteration: extractor samples and solver trajectories, both of which are consumed by the joint objective.

3.2 Extractor Training

Online skill generation. In Evolving-RL, the extractor is optimized in a fully online manner, such that each training iteration begins with the generation of new experience. Given the extraction state s^e, the extractor samples a set of N candidate skills:

$$\mathcal{E} = \{e_1, \dots, e_N\}, \qquad e_i \sim \pi_\theta(\cdot \mid s^e).$$

Each candidate e_i is instantiated as a skill, a structured textual abstraction that encodes both high-level procedural knowledge and concrete operational details distilled from the source interaction. The resulting N candidates constitute the comparison set for GRPO-based advantage estimation in extractor optimization.

Generalization-Oriented Evaluation. We evaluate the quality and generalization of a skill by its ability to improve solver performance on related tasks. To ensure fair evaluation, all candidate skills within the same comparison group must be assessed on an identical set of downstream tasks. Therefore, we do not use the skill content itself as the retrieval query, as this would cause different skills to be evaluated on different sets of downstream tasks. Instead, downstream tasks are retrieved based on the embedding of the task description.

Concretely, given a source task x^{src}, the retrieval function R returns a set of top-K semantically similar downstream tasks:

$$\mathcal{X}^{\mathrm{ret}} = R(x^{\mathrm{src}}) = \mathrm{TopK}_{x \in \mathcal{D}} \; s\left(\phi(x^{\mathrm{src}}), \phi(x)\right), \tag{1}$$

where $\phi(\cdot)$ denotes the embedding of the task description and $s(\cdot, \cdot)$ is instantiated as cosine similarity.

For each candidate skill e_i and each retrieved downstream task $x_j \in \mathcal{X}^{\mathrm{ret}}$, the solver generates a skill-conditioned trajectory $t_{ij} \sim \pi_\theta(\cdot \mid x_j, e_i)$ and receives environment reward r_{ij}. The reward assigned to skill e_i is defined as its mean downstream performance:

$$R_i^e = \frac{1}{K} \sum_{j=1}^{K} r_{ij}. \tag{2}$$

The resulting reward favors skills with higher average performance over the retrieved task set, aligning extractor optimization with empirical downstream utility.

Extractor Objective. The N candidate skills derived from the same source interaction constitute a single GRPO comparison group. We compute the group-normalized advantage as $A_i^e = \frac{R_i^e - \mathrm{mean}\left(\{R_{i'}^e\}_{i'=1}^N\right)}{\mathrm{std}\left(\{R_{i'}^e\}_{i'=1}^N\right)}$. The extractor loss is

$$\mathcal{L}_e(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \min\left(\rho_i^e A_i^e,\ \mathrm{clip}(\rho_i^e, 1-\epsilon, 1+\epsilon) A_i^e\right) + \beta_e \, D_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{ref}}) - \eta_e \, \mathcal{H}(\pi_\theta), \tag{3}$$

where $\rho_i^e = \frac{\pi_\theta(e_i \mid s^e)}{\pi_{\theta_{\mathrm{old}}}(e_i \mid s^e)}$ is the importance sampling ratio, D_{KL} denotes the KL regularization term, and $\mathcal{H}(\pi_\theta)$ denotes the entropy term. When $\eta_e < 0$, the final term penalizes rather than rewards entropy. We use this setting on ALFWorld, where unconstrained extractor entropy was associated with unstable training; Mind2Web and ScienceWorld use $\eta_e = 0$. Appendix A provides the corresponding empirical analysis.

3.3 Solver Training

Cross-Skill Group Advantage Estimation. During downstream evaluation, each retrieved task x_j yields a set of N trajectories $\{t_{ij}\}_{i=1}^N$, where each trajectory is conditioned on a distinct candidate skill. While these trajectories arise from heterogeneous prompts owing to the injected skills, they remain directly comparable, as the underlying task semantics, objective, and contextual structure are shared across all N rollouts for the same task x_j. We accordingly aggregate the N skill-conditioned trajectories of each task into a single GRPO comparison group and compute the solver-side advantage through within-group reward normalization:

$$A_{ij}^s = \frac{r_{ij} - \mathrm{mean}\left(\{r_{i'j}\}_{i'=1}^N\right)}{\mathrm{std}\left(\{r_{i'j}\}_{i'=1}^N\right)}. \tag{4}$$

Under this grouping mechanism, environment reward provides a task-matched relative signal. A trajectory conditioned on a more useful skill receives a positive advantage when it outperforms trajectories generated for the same task under other candidate skills; a less effective conditioning context receives a lower or negative relative advantage. This objective trains the solver to exploit useful guidance while reducing reliance on skills that produce lower returns within the comparison group.

Solver objective. We reuse all $N \times K$ skill-conditioned trajectories collected during downstream evaluation. Let A^s_{ij} be the heterogeneous-context advantage from Eq. equation 4. The solver loss is

$$\mathcal{L}_s(\theta) = -\frac{1}{NK}\sum_{j=1}^{K}\sum_{i=1}^{N}\min\big(\rho^s_{ij}A^s_{ij},\ \mathrm{clip}(\rho^s_{ij}, 1-\epsilon, 1+\epsilon)A^s_{ij}\big) + \beta_s\, D_{\mathrm{KL}}(\pi_\theta \| \pi_{\mathrm{ref}}), \quad (5)$$

where $\rho^s_{ij} = \frac{\pi_\theta(t_{ij}|x_j,e_i)}{\pi_{\theta_{\mathrm{old}}}(t_{ij}|x_j,e_i)}$, and $D_{\mathrm{KL}}(\pi_\theta\|\pi_{\mathrm{ref}})$ is the KL-divergence regularization term measuring the deviation of the current policy π_θ from the reference policy π_{ref}.

3.4 Joint Objective and Co-Evolution Dynamics

The final training objective is a weighted combination of the extractor and solver losses:

$$\mathcal{L} = \lambda_e\mathcal{L}_e + \lambda_s\mathcal{L}_s. \quad (6)$$

Because both losses operate on the same parameter vector θ, an update from either objective can affect the policy's behavior in both roles.

Concretely, the mechanism operates in two directions. On the extraction side, the extractor is incentivized to produce skills that are broadly useful— skills that help the solver on diverse retrieved tasks. On the utilization side, the solver is trained under a realistic distribution of skill quality: it encounters both informative and noisy skills during the same rollout, which mirrors the conditions it faces at test time and builds robustness accordingly.

4 Experiments

4.1 Experimental Setup

Our main experiments instantiate all methods with `Qwen2.5-7B-Instruct` (Yang et al., 2024a) and evaluate on ALFWorld (Shridhar et al., 2021), Mind2Web (Deng et al., 2023), and ScienceWorld (Wang et al., 2022). We compare with the prompt-based methods ExpeL (Zhao et al., 2023), Memento (Zhou et al., 2025), and ReasoningBank (Ouyang et al., 2026), and the RL baselines GRPO (Shao et al., 2024) and SkillRL (Xia et al., 2026). SkillRL requires substantial cold-start data, which are unavailable for our Mind2Web and ScienceWorld settings; we therefore evaluate it only on ALFWorld. Unless stated otherwise, the main-table results are averaged over five repeated evaluations of the same trained checkpoint. To assess robustness to training randomness, Appendix C additionally reports an ALFWorld study with five Evolving-RL models independently trained with different random seeds. Detailed evaluation protocols, baseline implementations, environment configurations, and training settings are provided in Appendix E.

Evaluation protocol. Unless otherwise specified, rows labeled *w/ skills* use one-shot test-time skill extraction. For each target, the method retrieves a source from a benchmark-specific source pool, samples its trajectory, and extracts one temporary skill. ALFWorld and ScienceWorld use original training instances; Mind2Web uses other official test-split tasks with the target excluded. No target is used as its own source, and skills are neither refined nor accumulated. Appendix E.1 provides the detailed evaluation protocol, including the source pools and information boundaries for each benchmark. On ALFWorld and ScienceWorld, evaluation on types unseen during policy training measures experience-assisted OOD adaptation using source experiences available at deployment. Figure 3(b) separately evaluates iterative skill refinement on ALFWorld Pick2 task.

ALFWorld. ALFWorld evaluates text-based household task completion. We partition its original training set by task type into *seen* and *unseen* subsets, restricting policy optimization and training-time skill evaluation to seen types. Original training instances from unseen types remain available for test-time skill extraction without parameter updates. We report success rate.

Mind2Web. Mind2Web evaluates web navigation under cross-task, cross-website, and cross-domain shifts. We train on its 1,009 official training tasks. Test-time extraction uses other official test tasks, excluding the target. The extractor receives reached observations, attempted actions,

Table 1: Performance on ALFWorld. Rows labeled "w/ skills" follow the one-shot test-time extraction protocol in Section 4.1. The best and second-best results in each column are highlighted in boldface and underlined, respectively. * denotes results replicated from Xia et al. (2026).

	Unseen tasks			Seen tasks					All
Method	Look	Pick2	Avg.	Pick	Heat	Cool	Clean	Avg.	Overall
Base Model	18.9	36.5	27.4	75.0	13.0	60.0	57.4	51.9	45.5
Base Model (w/ skills)	16.7	36.5	26.3	74.2	28.7	56.2	45.8	50.9	44.5
Prompt-based methods									
ReasoningBank	11.1	35.3	22.9	75.0	21.7	47.6	45.2	47.5	41.1
ExpeL*	67.0	6.0	37.4	21.0	52.0	71.0	55.0	49.5	46.3
Memento	0.0	41.2	20.0	79.2	39.1	76.2	64.5	64.6	53.0
RL-based methods									
GRPO	53.3	12.9	33.7	96.7	**100.0**	**100.0**	90.3	96.2	79.9
GRPO (w/ skills)	61.1	27.1	44.6	95.8	98.3	98.1	96.1	97.0	83.3
SkillRL	81.2	55.6	68.8	93.3	72.2	80.0	95.2	86.2	81.7
Ours (w/o skills)	**100.0**	61.2	81.1	94.2	99.1	**100.0**	96.8	97.4	93.1
Ours (w/ skills)	95.6	**81.2**	**88.6**	**97.5**	**100.0**	**100.0**	**97.4**	**98.6**	**96.0**

Table 2: Results on Mind2Web and ScienceWorld. Rows labeled "w/ skills" follow the one-shot protocol in Section 4.1. Mind2Web reports action accuracy over attempted actions, including the first incorrect action, and task success rate under cross-task (CT), cross-website (CW), cross-domain (CD), and overall (All) settings. ScienceWorld reports ID and OOD scores. Higher is better.

	Mind2Web								ScienceWorld	
	Action Accuracy				Success Rate				Score	
Method	CT	CW	CD	All	CT	CW	CD	All	ID	OOD
Base Model	12.57	7.95	7.91	8.79	0.24	0.00	0.44	0.34	-29.34	-30.95
Base Model (w/ skills)	14.92	15.01	12.68	13.41	0.40	0.00	0.50	0.42	-31.62	-36.19
ReasoningBank	11.84	17.76	12.09	12.79	0.00	0.56	0.66	0.52	-25.60	-18.73
Memento	15.27	15.46	13.13	13.84	0.40	0.56	0.88	0.75	-36.87	-26.13
GRPO	28.79	24.61	20.84	22.83	1.51	0.00	1.67	1.42	12.50	11.34
GRPO (w/ skills)	28.98	26.20	20.33	22.73	0.87	0.90	1.69	1.43	11.29	10.99
Ours (w/o skills)	35.32	**35.15**	24.67	28.05	1.59	**3.39**	1.21	1.57	19.09	9.39
Ours (w/ skills)	**41.99**	35.14	**26.97**	**30.87**	**2.38**	1.81	**1.91**	**1.99**	**21.93**	**17.29**

and terminal binary success, but no explicit per-action correctness labels, reference actions, gold trajectories, hidden annotations, or unreached HTML. Rollouts stop at the first incorrect action, implicitly revealing the correct prefix and failed terminal attempt. We report task success and action accuracy over all attempted actions, including the terminating error.

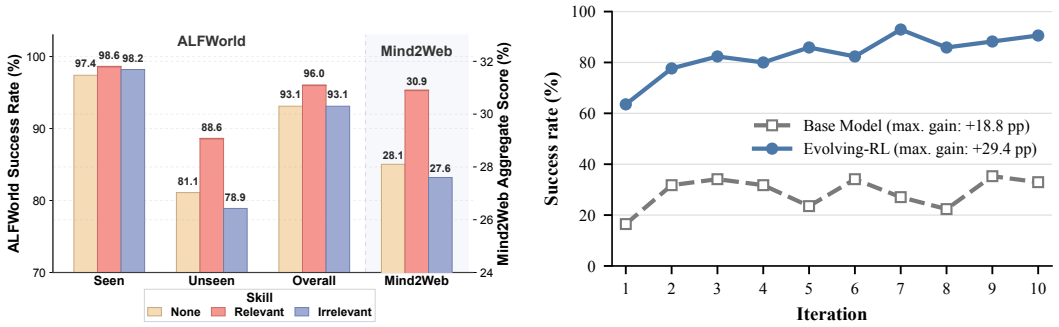
ScienceWorld. ScienceWorld evaluates interactive scientific tasks. We randomly split its 30 task types into 20 ID and 10 OOD types, restricting policy optimization and training-time skill evaluation to ID types. OOD types remain available for test-time skill extraction. All methods use the same GPT-4o-rewritten descriptions, which remove step-by-step guidance while preserving task objectives, entities, and constraints.

4.2 Main Results

Tables 1 and 2 report the main results on ALFWorld, Mind2Web, and ScienceWorld. Evolving-RL improves skill-conditioned performance across all three benchmarks. Without skill injection, our method also substantially outperforms GRPO on ALFWorld and Mind2Web, demonstrating improvement in the underlying policy.

Table 3: Objective ablation on ALFWorld. The co-evolution objective corresponds to the full Evolving-RL training.

Objective	Eval.	Seen	Unseen	Overall
Base Model	w/o skills	51.9	27.4	45.5
Base Model	w/ skills	50.9	26.3	44.5
Extractor-only	w/o skills	62.6	28.6	53.7
Extractor-only	w/ skills	73.7	27.6	61.7
Solver-only	w/o skills	98.2	70.3	90.9
Solver-only	w/ skills	97.8	69.7	90.4
Co-evolution	w/o skills	97.4	81.1	93.1
Co-evolution	w/ skills	**98.6**	**88.6**	**96.0**



(a) Sensitivity to skill relevance.

(b) Ten iterations of test-time skill evolution.

Figure 3: Test-time skill utility. (a) Relevant skills outperform no and irrelevant skills on ALFWorld and Mind2Web, while irrelevant skills remain near the no-skill condition. (b) On ALFWorld Pick2, both models start from identical trajectories and feedback and revise textual skills over ten iterations with parameters fixed. Gains are relative to the first iteration.

On ALFWorld, Evolving-RL reaches 96.0% overall success with skills and improves unseen-task success over GRPO (w/ skills) from 44.6% to 88.6%; without skill injection, it obtains 93.1% overall success. Relative to GRPO under the same skill-conditioned protocol, Mind2Web action accuracy improves by 44.9%, 34.1%, and 32.7% on the cross-task, cross-website, and cross-domain splits, respectively. Overall action accuracy increases from 22.73% to 30.87%, accompanied by an increase in task success rate from 1.43% to 1.99%, showing gains in both action prediction and complete task execution. On ScienceWorld, Evolving-RL (w/ skills) raises the OOD score from 10.99 for GRPO (w/ skills) to 17.29. Injecting skills also improves its own OOD score from 9.39 to 17.29, highlighting the contribution of explicit experience reuse to performance on task types unseen during training.

4.3 Ablations and Analysis

We further analyze Evolving-RL by investigating three key questions to better understand the sources of its gains and the properties of the learned self-evolution capabilities.

Q1: What drives the generalization gains of Evolving-RL? Table 3 compares the extractor- and solver-side objectives. *Solver-only* substantially improves the underlying policy on unseen tasks, but its self-extracted skills provide almost no additional gain, suggesting that stronger task-solving alone does not imply better experience reuse. *Extractor-only* improves skill-conditioned performance mainly on seen tasks. In contrast, co-evolution achieves strong no-skill performance and a further gain from skill injection, indicating that jointly optimizing extraction and utilization improves both the underlying policy and transferable experience use.

We further evaluate iterative test-time self-evolution in Figure 3(b). With parameters fixed and identical initial trajectories and feedback, Evolving-RL improves from 63.5% to 90.6% over ten iterations and peaks at 92.9%, while the Base Model remains below 36%. This suggests that part

Table 4: Results with `Qwen2.5-3B-Instruct`. CT, CW, and CD denote cross-task, cross-website, and cross-domain, respectively; best results are bold.

	ALFWorld			Mind2Web				ScienceWorld	
Method	Seen	Unseen	All	CT	CW	CD	All	ID	OOD
Base Model	12.22	9.72	11.57	10.93	6.84	8.00	8.59	-37.18	-36.43
GRPO	88.1	50.9	78.4	19.03	13.66	13.02	14.23	7.66	7.25
Evolving-RL (w/o skills)	86.9	40.0	74.6	16.41	12.38	13.72	14.05	7.05	8.51
Evolving-RL (w/ skills)	**92.9**	**60.0**	**84.3**	**25.53**	**16.19**	**17.73**	**18.99**	**10.12**	**11.44**

of the generalization gain comes from an improved ability to refine and reuse experience during deployment, rather than from a stronger static policy alone.

Q2: Are the gains merely due to the injected context format? We next examine whether the model has genuinely learned to interpret and use the semantic content of extracted skills, rather than simply overfitting to the presence of a skill-conditioned context. To test this, we compare three settings: relevant skills, skills extracted from the least-related retrieved tasks, and no skill injection. As shown in Figure 3(a), only relevant skills provide a clear improvement, whereas irrelevant skills remain close to the no-skill condition. This result suggests that the gains do not arise merely from exposing the model to an additional skill-formatted context, but depend on the semantic relevance of the injected experience. In other words, Evolving-RL learns to exploit useful skill content rather than relying on the presence of the skill prompt itself.

Q3: Do extracted skills transfer across solver models, and does Evolving-RL remain effective on other models? For each of the two solver models in Figure 1(a), we fix the source trajectories, outcome feedback, and target tasks, and compare their self-extracted skills with Evolving-RL skills. The latter provide larger gains. On `Qwen2.5-7B-Instruct`, overall success rises from 44.0% without skills and 48.8% with self-extracted skills to 56.0%. Because the solver, source evidence, and evaluation targets are all held fixed, the performance difference can be attributed to the extracted skills rather than to changes in the downstream policy or evaluation setting. This controlled comparison therefore demonstrates improved skill extraction quality as well as transfer across solver models; Appendix D provides a qualitative example.

We also apply the complete training pipeline to `Qwen2.5-3B-Instruct` to examine whether the observed benefits persist under a smaller backbone. Table 4 shows that skill-conditioned Evolving-RL outperforms GRPO across ALFWorld, all Mind2Web splits, and both ScienceWorld settings. Skill injection raises its overall ALFWorld success from 74.6% to 84.3% and Mind2Web action accuracy from 14.05% to 18.99%, confirming the benefit of explicit skill reuse on this model. Together, these results indicate that the gains of Evolving-RL are not specific to a single solver model, while the extracted skills remain useful when applied across different solver models.

In addition, we conducted further analyses of the training-compute budget and the sensitivity to the number of retrieved downstream tasks. Appendix B.1 reports a token-matched GRPO control and separately accounts for the extra source interaction and skill-generation stages. Appendix B.2 reports the sensitivity to K.

5 Conclusion

We present **Evolving-RL**, an end-to-end framework for optimizing experience-driven self-evolving capabilities by jointly training experience extraction and utilization within a shared policy. Centered on downstream transfer evaluation, Evolving-RL aligns extracted experience with its practical utility and reuses the resulting skill-conditioned trajectories to optimize the solver, forming a closed loop in which extraction and utilization mutually reinforce each other. Experiments across multiple agent benchmarks show that Evolving-RL improves both experience-assisted adaptation and the underlying policy, while further analyses confirm stronger experience extraction, transfer, and iterative refinement at test time. These results highlight the dual role of Evolving-RL as a framework for enhancing experience-driven self-evolution and as an experience-augmented reinforcement learning algorithm.

AI USE STATEMENT

We used generative AI tools to assist with method implementation, specifically in writing experimental code, to discuss experimental design and provide feedback on possible experimental choices, to assist with checking and discussing the derivation of the accuracy lower bound presented in the manuscript, and to polish the language and readability of the manuscript. AI-assisted discussions of experimental design and mathematical derivations were used only as sources of feedback and auxiliary reasoning support; all experimental choices, mathematical arguments, and conclusions were independently examined, verified, and ultimately determined by the authors. We also used GPT-4o to preprocess ScienceWorld task descriptions by removing step-by-step solution guidance while retaining the task goals, entities, and constraints, as described in Appendix E.1. All AI-assisted experimental code was manually reviewed by the authors, and all experiments, analyses, mathematical derivations, and reported results were conducted or independently verified by the authors. The authors take full responsibility for the final content of this work, including the methodology, experimental design, mathematical analysis, code, results, and claims.

ETHICS STATEMENT

This work studies experience-driven agent learning using ALFWorld, Mind2Web, and ScienceWorld. All experiments are conducted within these benchmark settings, and the results do not establish the safety or reliability of real-world deployment. Extracted skills may retain errors or biases from source interactions, which may propagate when the skills are reused across tasks.

REPRODUCIBILITY STATEMENT

Detailed evaluation protocols, information boundaries, baseline implementations, environment configurations, reward definitions, training hyperparameters, computational resources, and prompt templates are provided in Appendix E. Appendix C reports results across five independent training seeds on ALFWorld, and Appendix A presents the assumptions and analysis underlying skill-evaluation reliability. The code will be released after the review process.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. Gepa: Reflective prompt evolution can outperform reinforcement learning, 2026. URL https://arxiv.org/abs/2507.19457.

Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. Evoskill: Automated skill discovery for multi-agent systems, 2026. URL https://arxiv.org/abs/2603.02766.

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.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Zhicheng Cai, Xinyuan Guo, Yu Pei, Jiangtao Feng, Jinsong Su, Jiangjie Chen, Ya-Qin Zhang, Wei-Ying Ma, Mingxuan Wang, and Hao Zhou. FLEX: Continuous agent evolution via forward learning from experience. *arXiv preprint arXiv:2511.06449*, 2025. doi: 10.48550/arXiv.2511.06449.

Zouying Cao, Jiaji Deng, Li Yu, Weikang Zhou, Zhaoyang Liu, Bolin Ding, and Hai Zhao. Remember me, refine me: A dynamic procedural memory framework for experience-driven agent evolution, 2026. URL https://arxiv.org/abs/2512.10696.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *arXiv preprint arXiv:2306.06070*, 2023. doi: 10.48550/arXiv.2306.06070.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*, 2024. URL https://arxiv.org/abs/2402.01680.

Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 1049–1065, Toronto, Canada, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.67. URL https://aclanthology.org/2023.findings-acl.67/.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.09516.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pp. 22199–22213. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/8bb0d291acd4acf06ef112099c16f326-Paper-Conference.pdf.

Yu Li, Rui Miao, Zhengling Qi, and Tian Lan. Arise: Agent reasoning with intrinsic skill evolution in hierarchical reinforcement learning, 2026. URL https://arxiv.org/abs/2603.16060.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *International Conference on Learning Representations (ICLR)*, 2019.

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.

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

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=jL7fwchScm.

Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. Reasoning with language model prompting: A survey. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5368–5393, Toronto, Canada, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.294. URL https://aclanthology.org/2023.acl-long.294/.

Shuai Shao, Qihan Ren, Chen Qian, Boyi Wei, Dadi Guo, Jingyi Yang, Xinhao Song, Linfeng Zhang, Weinan Zhang, Dongrui Liu, and Jing Shao. Your agent may misevolve: Emergent risks in self-evolving llm agents, 2026. URL https://arxiv.org/abs/2509.26354.

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

Shuaike Shen, Wenduo Cheng, Mingqian Ma, Alistair Turcan, Martin Jinye Zhang, and Jian Ma. Skillfoundry: Building self-evolving agent skill libraries from heterogeneous scientific resources, 2026. URL https://arxiv.org/abs/2604.03964.

Taiwei Shi, Sihao Chen, Bowen Jiang, Linxin Song, Longqi Yang, and Jieyu Zhao. Experiential reinforcement learning, 2026a. URL https://arxiv.org/abs/2602.13949.

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, 2026b. URL https://arxiv.org/abs/2605.06130.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2021. doi: 10.48550/arXiv.2010.03768.

Xiangru Tang, Ge Zhang, Sirui Hong, Chenglin Wu, Hao Cheng, Jiaheng Liu, Wangchunshu Zhou, Xingyao Wang, He Zhu, Chi Wang, Peng Xia, Daniel Shao, Fang Wu, Xinming Wei, Tianhao Peng, Ziyang Zhou, Tingting Du, and Tianrui Qin. Agent kb: Leveraging cross-domain experience for agentic problem solving, 2025. URL https://arxiv.org/abs/2507.06229.

Songjun Tu, Chengdong Xu, Qichao Zhang, Yaocheng Zhang, Xiangyuan Lan, Linjing Li, and Dongbin Zhao. Dynamic dual-granularity skill bank for agentic rl, 2026. URL https://arxiv.org/abs/2603.28716.

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 of Computer Science*, 18(6):186345, 2024a. doi: 10.1007/s11704-024-40231-1. URL https://link.springer.com/article/10.1007/s11704-024-40231-1.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.emnlp-main.775. URL https://aclanthology.org/2022.emnlp-main.775/.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory, 2024b. URL https://arxiv.org/abs/2409.07429.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

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, 2025. URL https://arxiv.org/abs/2510.16079.

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

Zhishang Xiang, Chengyi Yang, Zerui Chen, Zhimin Wei, Yunbo Tang, Zongpei Teng, Zexi Peng, Zongxia Li, Chengsong Huang, Yicheng He, Chang Yang, Xinrun Wang, Xiao Huang, Qinggang Zhang, and Jinsong Su. A systematic survey of self-evolving agents: From model-centric to environment-driven co-evolution, February 2026. URL https://doi.org/10.36227/techrxiv.177203250.05832634/v2. Preprint.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024a. URL https://arxiv.org/abs/2412.15115.

Chen Yang, Quanquan Gu, Chenyang Zhao, and Dongruo Zhou. Cops: Empowering llm agents with provable cross-task experience sharing, 2024b. URL https://arxiv.org/abs/2410.16670.

Cheng Yang, Xuemeng Yang, Licheng Wen, Daocheng Fu, Jianbiao Mei, Rong Wu, Pinlong Cai, Yufan Shen, Nianchen Deng, Botian Shi, Yu Qiao, and Haifeng Li. Learning on the job: An experience-driven self-evolving agent for long-horizon tasks, 2025. URL https://arxiv.org/abs/2510.08002.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Xiaoying Zhang, Zichen Liu, Yipeng Zhang, Xia Hu, and Wenqi Shao. Retroagent: From solving to evolving via retrospective dual intrinsic feedback, 2026. URL https://arxiv.org/abs/2603.08561.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025. URL https://arxiv.org/abs/2506.05176.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: LLM agents are experiential learners. *arXiv preprint arXiv:2308.10144*, 2023. doi: 10.48550/arXiv.2308.10144.

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

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, and Jun Wang. Memento: Fine-tuning LLM agents without fine-tuning LLMs. *arXiv preprint arXiv:2508.16153*, 2025. doi: 10.48550/arXiv.2508.16153.

Huichi Zhou, Siyuan Guo, Anjie Liu, Zhongwei Yu, Ziqin Gong, Bowen Zhao, Zhixun Chen, Menglong Zhang, Yihang Chen, Jinsong Li, Runyu Yang, Qiangbin Liu, Xinlei Yu, Jianmin Zhou, Na Wang, Chunyang Sun, and Jun Wang. Memento-skills: Let agents design agents, 2026. URL https://arxiv.org/abs/2603.18743.

A Co-Evolution Stability

A.1 Reliability of skill evaluation

Because R_i^e is estimated from a finite number of downstream rollouts, we analyze how reliably it ranks two candidate skills e_a and e_b. We condition on a fixed solver policy, the two skills, and the same retrieved task set $\{x_i\}_{i=1}^K$. For each task, let

$$Y_{ai} = r(x_i, t_{ai}), \qquad t_{ai} \sim \pi_\theta(\cdot \mid x_i, e_a),$$

and define Y_{bi} analogously. Assume that these rollouts are sampled independently and that their rewards lie in $[m, M]$, with $M > m$. Write

$$u_{ai} = \mathbb{E}[Y_{ai}], \qquad u_{bi} = \mathbb{E}[Y_{bi}], \qquad R_a = \frac{1}{K}\sum_{i=1}^K u_{ai}, \qquad R_b = \frac{1}{K}\sum_{i=1}^K u_{bi}.$$

The empirical utilities are

$$\hat{R}_a = \frac{1}{K}\sum_{i=1}^K Y_{ai}, \qquad \hat{R}_b = \frac{1}{K}\sum_{i=1}^K Y_{bi}.$$

For a pair with $\Delta_{ab} = R_a - R_b > 0$, let $\hat{\Delta}_{ab} = \hat{R}_a - \hat{R}_b$. Then $\mathbb{E}[\hat{\Delta}_{ab}] = \Delta_{ab}$, and independent rollout sampling gives

$$\sigma_{ab}^2 = \mathrm{Var}(\hat{\Delta}_{ab}) = \frac{1}{K^2}\sum_{i=1}^K [v_{ai} + v_{bi}], \qquad v_{ai} = \mathrm{Var}(Y_{ai}), \quad v_{bi} = \mathrm{Var}(Y_{bi}). \tag{7}$$

Cantelli's one-sided variance inequality yields

$$\Pr(\hat{R}_a > \hat{R}_b) \geq \frac{\Delta_{ab}^2}{\sigma_{ab}^2 + \Delta_{ab}^2}. \tag{8}$$

To see this, set $Z = \Delta_{ab} - \hat{\Delta}_{ab}$, so that $\mathbb{E}[Z] = 0$ and $\mathrm{Var}(Z) = \sigma_{ab}^2$. For any $c \geq 0$, Markov's inequality implies

$$\Pr(Z \geq \Delta_{ab}) \leq \Pr\big((Z+c)^2 \geq (\Delta_{ab}+c)^2\big) \leq \frac{\sigma_{ab}^2 + c^2}{(\Delta_{ab}+c)^2}.$$

Choosing $c = \sigma_{ab}^2/\Delta_{ab}$ bounds the probability of an incorrect ordering or a tie by $\sigma_{ab}^2/(\sigma_{ab}^2 + \Delta_{ab}^2)$. Taking the complement gives Equation equation 8. If $\sigma_{ab}^2 = 0$, the empirical gap equals the positive true gap almost surely, and the bound equals one.

For bounded rewards, each variance is at most $(M-m)^2/4$, giving

$$\sigma_{ab}^2 \leq \frac{(M-m)^2}{2K}. \tag{9}$$

Combining the two inequalities gives the finite-sample bound

$$\Pr(\hat{R}_a > \hat{R}_b) \geq \frac{2K\Delta_{ab}^2}{(M-m)^2 + 2K\Delta_{ab}^2}. \tag{10}$$

This result holds for every $K \geq 1$ under the stated assumptions, including the $K = 4$ used in our experiments. For a given positive utility gap and reward range, the lower bound increases with K. It characterizes sampling uncertainty in ranking skills on the retrieved task set; the gap itself can change when the task set changes. Thus, finite-sample evaluation is particularly sensitive to skills with similar expected utilities.

Since K is a key design choice that directly controls both the amount of downstream evidence used to evaluate candidate skills and the associated evaluation cost, we additionally study the empirical sensitivity to K in Appendix B.2.

For ALFWorld's reward $r \in \{0, 10\}$, let p_{ai} denote the success probability of skill e_a on task x_i. Then $u_{ai} = 10p_{ai}$ and $v_{ai} = 100p_{ai}(1-p_{ai})$. Dividing rewards by ten gives the equivalent Bernoulli formulation without changing the skill ordering or the probability bounds.

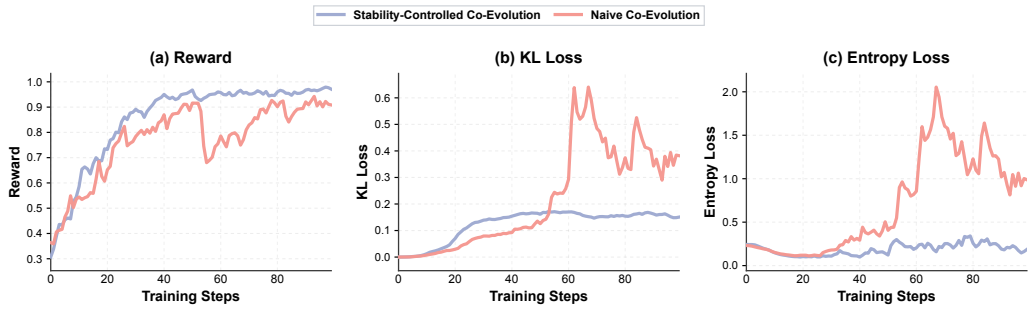


Figure 4: Training stability comparison between stability-controlled co-evolution and naive co-evolution. While naive co-evolution exhibits rapid growth in KL loss and entropy loss, stability-controlled co-evolution keeps both quantities bounded and yields smoother, more reliable reward improvement.

A.2 Training Stability

Equation 8 relates the lower bound on correct skill ordering to the utility gap Δ_{ab} and sampling variance σ^2_{ab}. For a fixed variance, a larger gap gives a stronger bound. During RL training, the extractor may occasionally generate rare expressions or anomalous characters. Such outputs do not always cause a substantial drop in downstream solver performance, so the utility gap between a malformed skill and a semantically similar, well-formed skill may remain small. Finite-sample feedback can therefore provide weak discrimination between these outputs.

When this margin is small, our empirical evaluation mechanism may fail to reliably distinguish between the two skills. Consequently, malformed skills can occasionally receive positive extractor-side advantages A^e_i. We hypothesize that repeatedly reinforcing low-probability tokens in such samples can increase extractor entropy and, because the two roles share parameters, can also affect solver behavior. The joint growth of the measured entropy and KL terms alongside degraded rewards in Figure 4 is consistent with this mechanism, although it does not establish it as the unique cause of instability.

To mitigate this issue, we introduce two explicit stability controls. First, we apply a rule-based filter that assigns zero reward to any candidate skill containing abnormal characters. Second, as defined in Equation 3, we include the entropy regularization term $-\eta_e \mathcal{H}(\pi_\theta)$ in the extractor objective to prevent the extraction policy from becoming excessively diffuse. Unlike the standard use of entropy regularization to encourage exploration, our goal here is to suppress uncontrolled entropy growth and constrain the extraction distribution. As shown in Figure 4, these two measures together substantially improve the stability of co-evolutionary training.

The first measure can be viewed as a lightweight modification of the reward function: in addition to transfer-based evaluation, we incorporate a small rule-based reward correction.

Discussion on Co-Evolution Instability. Related concurrent work (Muhtar et al., 2026) attributes similar instability to parameter conflicts between the extractor and solver and therefore separates the two roles across models. Our observations point to finite-sample evaluation noise and extractor entropy as an additional plausible mechanism. In our setting, rule-based gating and negative entropy regularization are sufficient to stabilize shared-policy training, showing that a decoupled architecture is not required for the configurations studied here.

B Additional Compute and Hyperparameter Analyses

B.1 Compute-Matched GRPO

The Evolving-RL training loop contains three generation stages: a source interaction, candidate skill generation, and skill-conditioned downstream evaluation. To test whether the main gains can be explained by a larger total generation budget, we extended GRPO training until its total generated-

Table 5: Per-update accounting of the additional stages in Evolving-RL. Skill generations are model-only generations and do not interact with the environment.

Component	Count	Relative quantity	Cost interpretation
GRPO solver rollouts	512	100.0%	Environment interaction
Evolving-RL downstream rollouts	512	100.0%	Environment interaction
Additional source rollouts	16	3.1% of GRPO	Environment interaction
Candidate-skill generations	128	25.0% of solver generations	Model only; no environment interaction

Table 6: Sensitivity to the number of retrieved downstream tasks on ALFWorld. Values are overall success rates (%).

K	w/ skills	w/o skills
2	91.42	86.94
4	96.00	93.10
8	95.90	91.79

token budget matched that of Evolving-RL, including the source-rollout and skill-generation stages. The token-matched GRPO run reached a held-out ALFWorld success rate of 80.6%. Since the original 75-step GRPO run had already largely converged, the additional updates did not yield a material improvement. This control suggests that the performance gap is not explained by giving GRPO more generated tokens alone.

We separately account for the extra stages under the default configuration $B = 16$, $N = 8$, and $K = 4$. Each Evolving-RL update uses $BNK = 512$ skill-conditioned downstream solver rollouts, matching the solver-rollout budget of GRPO. In addition, Evolving-RL performs $B = 16$ source rollouts, which interact with the environment and correspond to only $16/512 = 3.1\%$ of the matched GRPO environment-rollout budget. It also performs $BN = 128$ candidate-skill generations. These generations do not interact with the environment and are therefore not equivalent to full solver rollouts; their cost is lower and is better represented by their generated token count. The token-matched GRPO control above explicitly includes this model-generation budget. Thus, the extra environment interaction is small, while the skill-extraction stage adds a separate, lightweight model-only generation cost rather than 128 additional environment rollouts.

B.2 Sensitivity to the Number of Retrieved Tasks

We vary K while keeping the other training settings fixed. Table 6 summarizes the resulting overall success rates.

The $K = 2$ configuration exhibits earlier training collapse and lower final performance, consistent with the theoretical analysis that a smaller task set can lead to noisier skill evaluation and less reliable relative rewards. Increasing K from 4 to 8 results in very similar with-skill performance, indicating that $K = 4$ already provides sufficiently reliable skill evaluation in this setting. We therefore adopt $K = 4$ for the main experiments.

C ALFWorld Training-Seed Robustness

To quantify sensitivity to training randomness, we independently train Evolving-RL with five random seeds and evaluate the resulting checkpoints under the same ALFWorld protocol. We report the mean and sample standard deviation across the five independently trained models. The unseen and seen averages are weighted by their task counts (18, 17, 24, 23, 21, and 31; 134 tasks in total). Table 7 shows that the skill-conditioned variant retains a clear unseen-task advantage across training seeds, while both variants remain stable on seen tasks and overall performance. This experiment is distinct from the five repeated evaluations of a single checkpoint used in the main tables.

Table 7: ALFWorld robustness across five independently trained Evolving-RL models with different random seeds. Entries are mean ± sample standard deviation in success rate (%). Seen, unseen, and overall averages use the task counts described in Appendix C.

	Unseen tasks			All tasks
Method	Look	Pick2	Avg.	Overall
Evolving-RL (w/o skills)	93.33±10.92	63.61±11.12	78.89±2.67	93.00±1.33
Evolving-RL (w/ skills)	96.53±2.46	79.38±6.92	88.20±3.06	95.62±1.75

	Seen tasks				
Method	Pick	Heat	Cool	Clean	Avg.
Evolving-RL (w/o skills)	98.84±2.59	97.79±3.68	99.68±0.71	96.35±3.10	98.00±1.31
Evolving-RL (w/ skills)	98.39±2.46	97.10±2.71	98.73±2.84	98.62±1.48	98.24±1.45

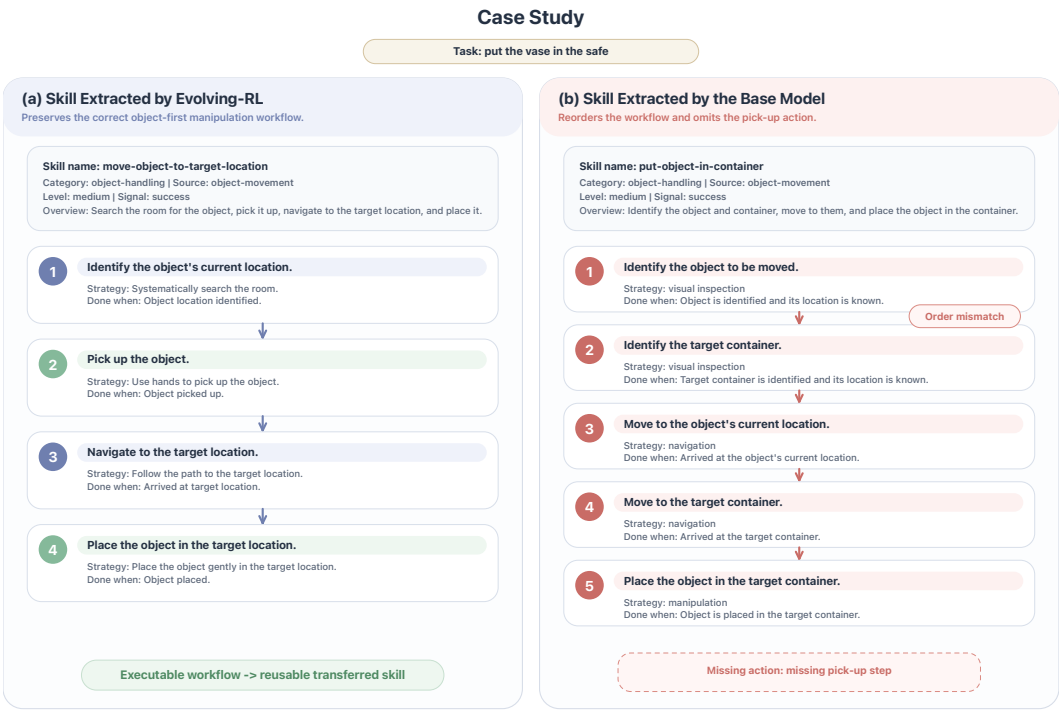


Figure 5: Case study. The skill extracted by the Evolving-RL-trained model is concise and provides a correct procedural guide, whereas the skill extracted by the untrained base model contains misordered steps and omits critical actions.

D Case Study

To better understand how Evolving-RL improves the quality of skill extraction, we compare the skills produced by different models from the same interaction trajectory. As shown in Figure 5, for a trajectory corresponding to the task *put the vase in the safe*, the Evolving-RL-trained model extracts a coherent and actionable skill that correctly captures the procedure of locating the target object, picking it up, finding the target container, and placing the object into it.

In contrast, the skill extracted by the Base Model contains clear procedural flaws. It introduces the step of confirming the container location before moving to the object, resulting in an incorrect ordering of subgoals. Moreover, after locating the object, it fails to include the crucial instruction to *pick up* the object. Injecting such a skill may therefore mislead the solver.

Table 8: Information access under the standard *w/ skills* protocol. One source trajectory and one skill are constructed per target task.

Benchmark	Policy-training exposure	Test-time extraction pool	Feedback visible during extraction	Persistent bank
ALFWorld	Seen task types	Original training instances, including the designated unseen types	Sampled interaction trace and terminal success/failure	No
Mind2Web	Official training split	Official test instances, excluding the target evaluation task itself	Interaction trace and terminal success/failure	No
ScienceWorld	20 ID task types	Training instances spanning all 30 task types	Sampled interaction trace and environment success/score	No

E Experimental and Implementation Details

This appendix provides the detailed experimental settings used throughout the paper, including evaluation protocols, information boundaries, baseline implementations, environment configurations, training settings, and prompt templates.

E.1 Evaluation Protocol and Information Boundaries

E.1.1 One-Shot Test-Time Skill Extraction

For a target task x, the standard *w/ skills* protocol retrieves the top-ranked semantically related source task x' from a benchmark-specific source pool. For ALFWorld and ScienceWorld, this pool consists of the benchmark-specific training instances described below; for Mind2Web, it consists of other official test-split tasks under leave-one-out exclusion. The evaluated policy first samples one source trajectory τ' without skill conditioning and observes the benchmark-specific feedback y'. The same policy then applies the extractor prompt once to construct a temporary skill $e = E(x', \tau', y')$, which conditions the rollout on x. No target instance is used as its own source. No extractor or policy update is performed, and e is neither refined nor accumulated across evaluation tasks. This protocol is distinct from the explicitly iterative experiment in Figure 3(b).

Controlled extractor comparison. Figure 1(a) compares skill extraction quality using identical source trajectories and environment feedback. For each solver model, we hold the model and target tasks fixed and provide the same source evidence to its own extractor and the Evolving-RL extractor. This design evaluates how effectively each extractor converts the same experience into skills that transfer to the solver model.

Mind2Web information boundary. Source interactions are collected from official test instances with the current target evaluation task excluded from the extraction pool. At each collection step, the agent receives the ordinary task instruction and the currently observable filtered HTML. The extractor is given the resulting trace of reached observations and attempted actions, together with a terminal binary task-success signal. It does not receive explicit per-action correctness labels, the reference action at a failed step, the remaining reference trajectory, hidden annotations, or unreached HTML states. Since the environment terminates the rollout after the first incorrect action, a failed trace contains the reached prefix and terminal attempted action, but no separately supplied correctness label for each action. The Base Model (w/ skills), GRPO (w/ skills), and Evolving-RL (w/ skills) use the same leave-one-out source pool and observable feedback, and consume only the top-ranked retrieved source to construct one temporary skill for each target task.

ScienceWorld split and preprocessing. The fixed 20/10 partition is defined over task types rather than individual variations. The 20 ID task types, used for both policy optimization and ID evaluation, are `boil`, `change-the-state-of-matter-of`, `chemistry-mix`, `chemistry-mix-paint-secondary-color`, `chemistry-mix-paint-tertiary-color`, `find-non-living-thing`, `find-plant`, `freeze`, `identify-life-stages-2`,

inclined-plane-friction-unnamed-surfaces, lifespan-longest-lived, lifespan-longest-lived-then-shortest-lived, measure-melting-point-known-substance, measure-melting-point-unknown-substance, mendelian-genetics-known-plant, mendelian-genetics-unknown-plant, power-component, test-conductivity, test-conductivity-of-unknown-substances, and use-thermometer. The 10 OOD task types, which are unseen during model training and used only for evaluation, are find-animal, find-living-thing, grow-fruit, grow-plant, identify-life-stages-1, inclined-plane-determine-angle, inclined-plane-friction-named-surfaces, lifespan-shortest-lived, melt, and power-component-renewable-vs-nonrenewable-energy.

For test-time skill extraction, the source pool contains training variations from all 30 task types; these variations construct temporary skills without contributing gradient updates. This protocol evaluates experience-driven generalization to task types unseen during model training.

ScienceWorld's original task descriptions often include explicit intermediate procedures. Because these cues overlap with the intended function of an extracted skill, we preprocess the descriptions with GPT-4o, asking it to remove step-by-step solution guidance while retaining the goal, referenced entities, initial conditions, thresholds, and other constraints required to determine success. All methods receive the same rewritten descriptions. During interaction, the environment is configured to use these de-guided descriptions from the dataset rather than replacing them with the original environment text.

E.1.2 Iterative Test-Time Skill Evolution

Figure 3(b) evaluates iterative test-time self-evolution on ALFWorld Pick2. Both models receive identical initial trajectories and outcome feedback. Model parameters remain fixed throughout the experiment. Each model extracts skills from this shared initial experience, then revises its skills using newly collected interaction feedback for use in the next iteration. We use the same Pick2 task set throughout: each iteration first evaluates the current skills, and the resulting failed trajectories are then used to revise skills for the following iteration. The reported curve therefore measures adaptation through repeated interaction with this task set. The main-table protocol separately evaluates one-shot skill transfer from source instances to distinct evaluation targets.

E.2 Baseline Implementations

ReasoningBank. We implement ReasoningBank as a trajectory-to-memory baseline. A rollout from each source instance is labeled by its environment outcome and summarized into at most three textual reasoning memories. At evaluation, task-relevant records are retrieved and their memory items are inserted into the solver prompt. We perform a single construction pass and do not run multi-round memory-bank evolution. ALFWorld and Mind2Web retrieve one record and use at most three memory items; ScienceWorld retrieves up to three records and caps the resulting context at five items.

Memento. Memento stores task–trajectory–outcome cases and compact plans in a non-parametric memory. We use sup-simcse-bert-base-uncased embeddings with cosine similarity and retrieve the top four cases for the current task. Retrieved positive cases provide procedural guidance, while negative cases provide failure-avoidance cues. Model parameters remain fixed during memory construction and evaluation.

GRPO. GRPO uses the same backbone, policy-training split, and optimizer as Evolving-RL. We match the number of policy update steps and solver samples per update, with each GRPO update using $B \times N \times K$ solver samples. Figure 1(b) reports reward progression over rollout steps. The row labeled GRPO is evaluated without skill conditioning; GRPO (w/ skills) applies the same one-shot extraction protocol as our method, without maintaining a SkillBank. SkillRL requires substantial cold-start data, which are unavailable for our Mind2Web and ScienceWorld settings, and is therefore evaluated only on ALFWorld. ExpeL is also reported only on ALFWorld; its asterisked values are reproduced from Xia et al. (2026) and are treated as contextual rather than compute-matched comparisons.

For the Base Model (w/ skills), GRPO (w/ skills), and Evolving-RL (w/ skills), each evaluation target uses the top-ranked retrieved source and one skill extracted from its sampled trajectory; the skill is not subsequently refined or updated. This gives the three rows the same test-time interaction and extraction budget. We use greedy solver decoding. Episodes are capped at 50 environment steps on ALFWorld and Mind2Web and at 30 steps on ScienceWorld.

E.3 Environment Configuration

ALFWorld. ALFWorld is a text-based interactive environment for embodied household task completion. At each step, the agent receives a textual observation describing the currently visible objects, together with a set of admissible actions. We use the *full interaction history* as the model context, including all previous observations and executed actions. Consequently, each optimization sample in ALFWorld corresponds to a *complete trajectory*. Following Jin et al. (2025), we apply a loss mask to environment observations within the trajectory, such that they are provided as conditioning context but do not contribute to the policy optimization loss.

The environment reward is defined at the trajectory level. If the task is successfully completed, the trajectory receives reward 10; otherwise, it receives reward 0. Formally, for a trajectory τ, the reward is

$$r(\tau) = \begin{cases} 10, & \text{if the task is completed successfully,} \\ 0, & \text{otherwise.} \end{cases}$$

Mind2Web. Mind2Web is a benchmark for grounded web navigation. In the official dataset, each task is paired with a reference action trajectory and the corresponding HTML state at each step. However, raw HTML is prohibitively long and noisy to be directly used as model input. We therefore preprocess each page by filtering the HTML and retaining only interactive elements, such as buttons, input fields, and hyperlinks; the resulting filtered text is used as the agent's observation.

Due to context-length constraints, we adopt a *turn-level* state representation. At each decision step, the agent is given the current filtered observation together with a textual history of previously executed actions, but not the full history of past observations. Therefore, each optimization sample in Mind2Web corresponds to a *single interaction step*, rather than an entire trajectory.

A key property of Mind2Web is that the dataset only provides the next state along the ground-truth trajectory. If the agent selects an incorrect action at any step, the corresponding next state is unavailable, and the trajectory is terminated immediately. Let a_i denote the number of attempted actions for task i, including the first incorrect action if one occurs, and let c_i denote the number of correct actions. We compute action accuracy as

$$\text{ActionAccuracy} = \frac{\sum_i c_i}{\sum_i a_i}.$$

Actions after termination are excluded, so each policy is evaluated on the action prefixes it reaches. We also report task success rate, defined as the proportion of tasks for which all required actions are completed correctly.

For solver optimization on Mind2Web, since correctness feedback is available at every step, we apply *turn-wise GRPO* to train the solver. Specifically, for the trajectory generated by skill e_i on retrieved task x_j, the reward at step t is defined as

$$r_{ij,t} = \begin{cases} 1, & \text{if the action predicted at step } t \text{ is correct,} \\ 0, & \text{otherwise.} \end{cases}$$

At each interaction step t, we form a comparison group from the rollouts that have reached that step. Since an incorrect action terminates a rollout, the group size may decrease at later steps. For groups containing multiple samples, rewards are normalized using the group mean and standard deviation; if all rewards are identical, the resulting advantages are zero. If only one rollout reaches a step, we avoid undefined group normalization by using an explicit fallback: a correct action inherits the preceding normalized reward when available and otherwise receives $+1$, while an incorrect action receives -1.

The extractor reward is computed using *trajectory-level* rewards. Specifically, for the downstream trajectory generated by skill e_i on retrieved task x_j, its trajectory-level reward is defined as the sum of step-level rewards:

Table 9: Main training hyperparameters across the three benchmarks.

Hyperparameter	**ALFWorld**	**Mind2Web**	**ScienceWorld**
Base model	Qwen2.5-7B-Instruct	Qwen2.5-7B-Instruct	Qwen2.5-7B-Instruct
Policy clip ϵ_{low}	0.1	0.1	0.1
Policy clip ϵ_{high}	0.15	0.15	0.15
Number of candidate skills N	8	8	8
Number of retrieved tasks K	4	4	4
Extractor reward weight λ_e	0.2	0.1	0.1
Solver reward weight λ_s	1.0	1.0	1.0
Rollout batch size B	16	16	16
Rollout temperature	1.0	1.0	1.0
Rollout top-p	0.9	0.85	0.9
Solver KL coefficient	0.01	0.01	0.01
Extractor KL coefficient	0.01	0.01	0.01
Solver entropy coefficient	0.0	0.0	0.0
Extractor entropy coefficient	-0.03	0.0	0.0
Training steps	75	150	300

$$r_{ij}^{\text{traj}} = \sum_{t=1}^{T_{ij}} r_{ij,t},$$

where $r_{ij,t}$ denotes the reward at step t, and T_{ij} is the trajectory length. The extractor reward for skill e_i is then computed as the average trajectory-level reward over the K retrieved downstream tasks:

$$R_i^e = \frac{1}{K} \sum_{j=1}^{K} r_{ij}^{\text{traj}}.$$

In this way, solver training exploits fine-grained step-level supervision, while extractor evaluation remains aligned with the overall quality of the procedural experience distilled from complete trajectories.

ScienceWorld. ScienceWorld exposes a textual observation, a set of possible actions and objects, and a cumulative environment score. We use the `easy` simplification setting and cap each episode at 30 interaction steps. Unlike ALFWorld, the solver receives a freshly constructed prompt at each step containing the de-guided task description, current observation, available actions, and prior action history. Training samples are formed at the step level, but reward assignment is trajectory-level: every sample from the same trajectory receives that trajectory's total score. We normalize across trajectories within a GRPO comparison group and broadcast the resulting advantage to all samples from the corresponding trajectory, without dividing it by the number of samples in that trajectory. At evaluation, we report the mean environment score on the fixed ID and OOD task-type splits.

E.4 Training Setup

The main experiments are initialized from `Qwen2.5-7B-Instruct` and trained with AdamW (Loshchilov & Hutter, 2019). Unless otherwise stated, we use a constant learning rate of 1×10^{-6}, weight decay 0.1, $\beta_1 = 0.9$, and $\beta_2 = 0.98$. Training is performed with GRPO-style policy optimization, using low-variance KL regularization and clipped policy updates. In each rollout, we sample 16 source tasks from the training set, i.e., the rollout batch size is 16. For similar task retrieval, task description embeddings are pre-computed offline using `Qwen3-Embedding-4B` (Zhang et al., 2025) and cached for use during training. All experiments are conducted on 8 NVIDIA H800 GPUs. Each Evolving-RL training run takes approximately 10 hours on ALFWorld and 17 hours on Mind2Web.

Table 9 summarizes the main task-specific and method-specific hyperparameters across all three benchmarks. We train for 75 steps on ALFWorld, 150 steps on Mind2Web, and 300 steps on ScienceWorld. For the GRPO baseline, we match the number of policy update steps and solver samples per update. Specifically, each GRPO update uses $B \times N \times K$ solver samples, matching the

solver-side rollout budget of Evolving-RL at each update. For ScienceWorld, we largely follow the ALFWorld configuration, with the task-specific differences reported in Table 9; in particular, we set the extractor reward weight to $\lambda_e = 0.1$ and both extractor and solver entropy coefficients to zero. We draw one rollout for each skill–task pair, allow at most 1,024 response tokens for skill extraction and for each solver step, use the `easy` simplification, and provide the de-guided dataset descriptions rather than the original instruction-rich environment text.

Among these hyperparameters, N determines the candidate comparison set for extractor-side GRPO, while K controls the number of downstream tasks used to evaluate each skill. The weights λ_e and λ_s balance the extraction and utilization objectives. We use $N = 8$ and $K = 4$ across all three benchmarks. For a source batch of size B, downstream evaluation requires BNK solver trajectories, each reused for solver optimization, in addition to B source trajectories and BN skill generations. Increasing K therefore broadens the task set used for skill comparison while increasing the downstream rollout count linearly for fixed B and N.

E.5 ALFWorld Prompt Templates

We provide the ALFWorld language interface because it exposes both the information supplied to the extractor and the recurrent interaction contract used by the solver. For readability, the listings consolidate repeated formatting instructions while preserving the runtime inputs, output fields, and decision rules; tokenizer-specific chat control tokens are omitted. Braced expressions denote values filled at runtime.

The extractor receives the source-task description, the complete sampled interaction trace, and the terminal environment-success signal. It is not given a corrected trajectory or any additional supervision.

ALFWorld Skill-Extraction Prompt

```
Task: {task_description}

Agent's Trajectory:
<<<BEGIN_TRAJECTORY>>>
{trajectory}
<<<END_TRAJECTORY>>>

True Environment Success: {success}

The case details above are the full evidence you should rely on. Read them
first, assess the trajectory quality, and extract a reusable skill. If the
sample failed, a brief failure analysis is especially helpful. Any analysis
must precede the final skill JSON.

The goal is not to retell the case, but to capture a reusable skill that could
help on similar tasks. A good skill should capture:
- what kind of task this is;
- when the skill should and should not be used;
- the stable workflow and its decision rules;
- common failure modes and how to handle them; and
- the expected final output.

Follow these rules:
1. Abstract from the example. Infer the broader task type, user intent,
   target outcome, and repeatable workflow. Do not copy the trajectory.
2. Make trigger conditions explicit: what the skill does, when it applies,
   how the need may be expressed, and when it should not be used.
3. Preserve workflow structure. Convert multi-step behavior into an ordered
   workflow and branching choices into decision rules.
4. Learn from success and failure. For a success, retain the key steps and
   checks; for a failure, identify the cause and a transferable correction.
5. Stay evidence-grounded. Do not invent unsupported tools, procedures, or
   domain knowledge. Produce a narrower skill when evidence is limited.
6. A short plain-text analysis may precede the final JSON, but the final skill
   must be a single JSON object at the end of the response.

Use the following JSON schema:
{
  "name": "string, kebab-case",
  "description": "what the skill does and when to use it",
  "metadata": {
```

```
    "source_task_type": "string",
    "success_signal": "success or failure",
    "abstraction_level": "low, medium, or high",
    "skill_category": "string"
  },
  "skill_overview": ["string"],
  "when_to_use": ["string"],
  "inputs_to_identify": ["string"],
  "workflow": [
    {
      "step": 1,
      "goal": "string",
      "action": "string",
      "tools_or_strategy": "string",
      "completion_condition": "string"
    }
  ],
  "decision_rules": ["string"],
  "error_handling": [
    {
      "error": "string",
      "cause": "string",
      "response": "string"
    }
  ],
  "output_requirements": ["string"]
}

Every field must be present. The workflow should be executable and concrete;
decision rules must add conditional guidance rather than repeat the workflow;
and error handling should address risks supported by the trajectory. Do not
include one-off object instances, paths, or accidental details unless they
represent a stable pattern. If evidence is insufficient, retain the field but
use a conservative empty value rather than inventing content.
```

The two solver variants share the same turn template; the skill-conditioned system message additionally inserts the reference skill and requests an applicability check. Each new observation is appended to the complete interaction history.

ALFWorld Per-Turn Interaction Prompt

```
[System]
You are an intelligent agent in the household environment. Complete the task
by taking actions step by step. At each step, reason inside <reasoning> and
</reasoning> tags, then choose exactly ONE admissible action and output it
inside <action> and </action> tags. Copy the action verbatim from the list.
[Only in the w/ skills condition]
A relevant skill from a similar task is provided below. In your reasoning,
assess whether it applies to the current state and adapt it when necessary.
Reference Skill: {skill}
[User]
Your task is to: {task_description}
[Repeated at step t]
[Step {t}]
Observation: {observation}
Admissible actions: ['{admissible_action_1}', '{admissible_action_2}', ...]
[Assistant]
<reasoning>{reasoning grounded in the current state and history}</reasoning>
<action>{one admissible action copied verbatim}</action>
```

F Limitations and Future Work

Evolving-RL focuses on a general learning framework for jointly optimizing experience extraction and utilization. To keep the framework broadly applicable and to isolate the effects of this joint optimization, we do not incorporate specialized skill-management components such as persistent skill banks, hierarchical organization, skill consolidation, or task-specific retrieval and update strategies. As a result, our current study primarily addresses how the underlying model can better extract and exploit reusable experience, rather than how accumulated experience should be managed over long deployment horizons. Extending Evolving-RL with more sophisticated mechanisms for skill

organization, selection, consolidation, and continual updating is an important direction for future work.