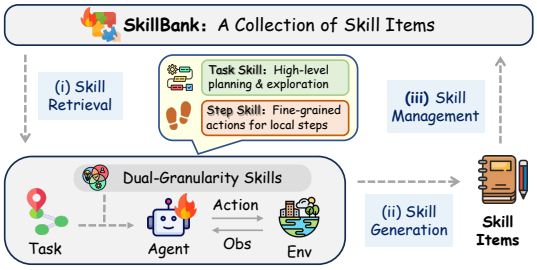
Dynamic Dual-Granularity Skill Bank for Agentic RL

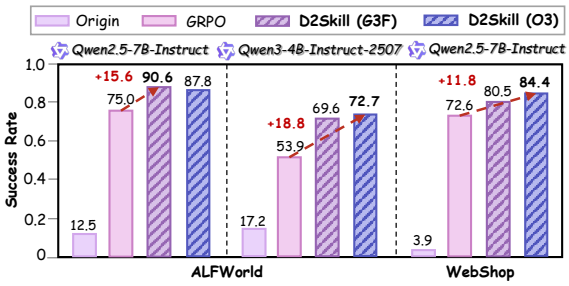
Anonymous ACL submission

Abstract

Agentic RL can benefit substantially from reusable experience, but existing skill-based methods mostly focus on trajectory-level guidance and provide limited support for the principled evaluation and maintenance of an evolving skill memory. We propose **D2Skill**, a **dynamic dual-granularity skill bank** for agentic RL that organizes reusable experience into **task skills** for high-level guidance and **step skills** for fine-grained decision support and error correction. D2Skill jointly trains the policy and skill bank through paired baseline and skill-injected rollouts under the same policy, using their performance gap to derive hindsight utility signals for both skill updating and policy optimization. Built entirely from training-time experience, the skill bank is continuously expanded through reflection and maintained with utility-aware retrieval and pruning. Experiments on ALFWORLD, WEBSHOP, and SEARCH-AUGMENTED QA tasks show that D2Skill substantially improves performance over skill-free baselines across models of different scales. Further ablations and analyses show that both dual-granularity skill modeling and dynamic skill maintenance are critical to these gains, while the learned skills exhibit higher utility, transfer across evaluation settings, and introduce only modest training overhead. **Code Page:** https://anonymous.4open.science/r/D2Skill-AgenticRL-FE80.



(a) D2Skill Framework



(b) Main Results

Figure 1: **Overview of D2Skill.** (a) The dynamic dual-granularity skill bank with retrieval, reflection-driven generation, and management. (b) Overall results on ALFWORLD and WEBSHOP.

1 Introduction

Agentic reinforcement learning (RL) has emerged as a promising paradigm for training language-based agents on long-horizon decision-making tasks, including interactive environments (Jiang et al., 2026), web search (Zhang et al., 2025b), and research scenarios (Tu et al., 2026). In these settings, agents act through textual interfaces based on task descriptions and limited interaction history, making the problem severely partially observable and credit assignment difficult over long horizons (Zhang et al., 2025a). Under sparse rewards and large action spaces, effective learning therefore requires reusable knowledge that can transfer across tasks (Feng et al., 2025).

Recent work addresses these challenges by introducing additional supervision signals for agentic RL. Some methods use outcome-based credit assignment to provide process rewards (Feng et al., 2025), while others derive hindsight supervision from completed trajectories (Yu et al., 2025). Another line of work enables agents to accumulate and refine experience across tasks during training (Zhai et al., 2025; Cai et al., 2025b,a). Within this line, reusable skills have emerged as an effective ab-

straction of past experience and have shown strong empirical gains in agentic RL (Wang et al., 2026). For example, SkillRL (Xia et al., 2026) builds a skill bank from past trajectories and retrieves relevant skills to guide policy interaction, improving exploration efficiency in long-horizon tasks.

However, existing skill-based and reflection-driven frameworks remain limited in two respects. First, many existing methods derive skills from complete trajectories and emphasize task-level reflection, which provides high-level guidance but is less effective for correcting fine-grained errors at individual interaction steps (Xia et al., 2026; Zhang et al., 2026a). Second, as training progresses, the skill bank grows continuously, making retrieval increasingly sensitive to redundancy and skill quality. Without principled mechanisms for skill evaluation and pruning, ineffective or redundant skills can degrade retrieved guidance and hinder policy optimization (Zhou et al., 2025, 2026).

To address these limitations, we propose **D**ynamic **D**ual-Granularity **Skill** Bank (**D2Skill**) for agentic RL, a framework that maintains reusable skills at both the task and step granularities throughout training. As illustrated in Figure 1a, D2Skill combines task skills for high-level guidance with step skills for local decision support, and contrasts skill-injected and non-injected rollouts under the same policy to estimate hindsight skill utility for retrieval, maintenance, and policy optimization. Figure 1b previews the main benchmark result: on ALFWORLD and WEBSHOP, D2Skill consistently outperforms strong skill-free and skill-augmented baselines. It also generalizes favorably to SEARCH-AUGMENTED QA, while ablations validate the roles of dual-granularity skills and dynamic memory management.

The main contributions are as follows:

1. We present **D2Skill**, a dynamic dual-granularity skill bank framework for agentic RL that organizes reusable experience into *task skills* for high-level guidance and *step skills* for local interaction support.
2. We develop a **joint training paradigm** in which the policy and skill bank co-evolve: skills are expanded through **reflection** and maintained with **utility-aware retrieval and pruning**, enabling more effective and efficient memory use throughout training.

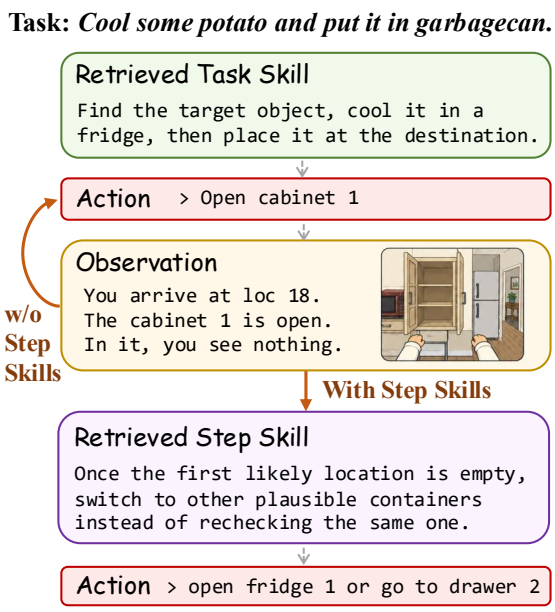


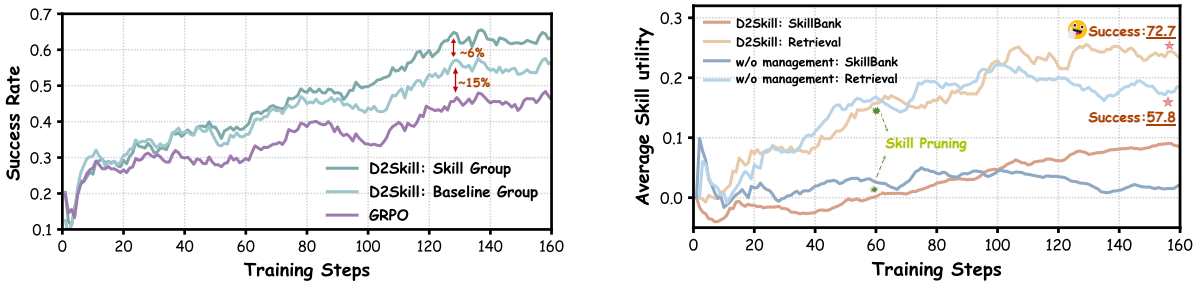
Figure 2: **Step skills correct local search failures.** Without step skills, the agent can repeat the same failed search after a negative observation. A retrieved step skill redirects it to another plausible location.

3. We evaluate D2Skill on ALFWORLD, WEBSHOP, and SEARCH-AUGMENTED QA using QWEN2.5-7B-INSTRUCT and QWEN3-4B-INSTRUCT. D2Skill **consistently outperforms skill-free baselines**, with ablations validating its core design.

2 Motivation

2.1 Local Errors Need Fine-Grained Skills

A limitation of task-level skills is that they mainly provide global guidance, while many failures in long-horizon agentic interaction arise from *local* errors after the overall plan is already reasonable. Once the agent reaches a concrete decision point, it must react to the current observation rather than rely only on a trajectory-level plan. Figure 2 illustrates this gap: a task skill can correctly prescribe the high-level workflow for cooling an object and placing it at the destination, yet it does not specify how to recover when the agent opens a plausible container and finds it empty. What is missing at that point is a short, reusable rule for local correction, such as switching to another plausible location instead of repeating the same failed search. This motivates introducing **step skills**, which provide observation-conditioned, fine-grained corrective guidance that complements task-level planning.



(a) Training Curves of Success Rate

(b) Skill Bank Dynamics of Utility

Figure 3: **Training and skill-bank dynamics motivating D2Skill.** (a) ALFWORLD training curves for the D2Skill skill group, the paired baseline group, and GRPO. (b) Skill-bank dynamics with and without management, measured by average skill utility and retrieval statistics.

We focus on two levels of abstraction because they map naturally to the two dominant failure modes in agentic RL: global planning and local recovery.

2.2 Growing Skill Banks Benefit from Dynamic Management

A limitation of static skill-bank reuse is that the memory keeps growing during training, while not all generated skills remain equally useful. Some skills are overly specific to a single failure, some become redundant as stronger skills are acquired later, and some continue to occupy retrieval and prompt budget without improving decisions. This issue is especially relevant to methods such as SkillRL, which retrieve stored memories but do not explicitly estimate their utility during training. As a result, simply accumulating more skills does not guarantee better guidance. Figure 3a shows that retrieved skills can provide measurable benefit, while Figure 3b shows that utility-aware management preserves higher-quality retrieved skills over time. These observations motivate **dynamic skill management**: instead of treating the bank as a static memory, D2Skill explicitly ranks, retains, and prunes skills according to the estimated utility.

3 Method

3.1 Overall Framework

D2Skill augments RL with a dynamic skill bank, as illustrated in Fig. 4. The framework has three tightly coupled parts.

RL training with skill injection. Each training group contains paired baseline and skill-injected rollouts under the same policy. Their performance gap provides hindsight signals for policy optimization and skill utility estimation (Section 3.2).

Reflection-driven skill generation. When performance is poor, a reflection module summarizes representative trajectories into reusable *task skills* and *step skills*, which are normalized and added to the bank.

Skill retrieval and bank management. During interaction, relevant skills are retrieved from the bank and injected into the policy context. Skill utilities are updated online, and the bank is periodically pruned to maintain a compact and effective memory (Section 3.3).

3.2 RL Training with Skill Injection and Hindsight Optimization

Rollout with skill injection. For each task g, we sample a group of N parallel trajectories, denoted by $\mathcal{G}_g$. The group is evenly divided into a ***skill group*** $\mathcal{G}_g^{\text{skill}}$ and a ***baseline group*** $\mathcal{G}_g^{\text{base}}$, each containing $N/2$ trajectories. Let $b_i \in \{0, 1\}$ denote the group indicator for trajectory $i \in \mathcal{G}_g$, where $b_i = 1$ indicates $i \in \mathcal{G}_g^{\text{skill}}$ and $b_i = 0$ indicates $i \in \mathcal{G}_g^{\text{base}}$. Trajectories in the skill group retrieve skills from the skill bank during interaction, while those in the baseline group follow the same policy without skill injection. Let $Y_i \in \{0, 1\}$ denote the terminal success indicator of trajectory i. For each task g, the baseline success rate and skill-group success rate are $\bar{Y}_g^{\text{base}} = \frac{1}{|\mathcal{G}_g^{\text{base}}|} \sum_{i \in \mathcal{G}_g^{\text{base}}} Y_i$ and $\bar{Y}_g^{\text{skill}} = \frac{1}{|\mathcal{G}_g^{\text{skill}}|} \sum_{i \in \mathcal{G}_g^{\text{skill}}} Y_i$.

Hindsight signals and utility updates. We use the performance gap between the skill group and the baseline group to construct hindsight signals for updating skill utilities. For each task g, the task-level hindsight signal Δ_g^{task} and the trajectory-level credit c_i for step skills retrieved along skill-injected

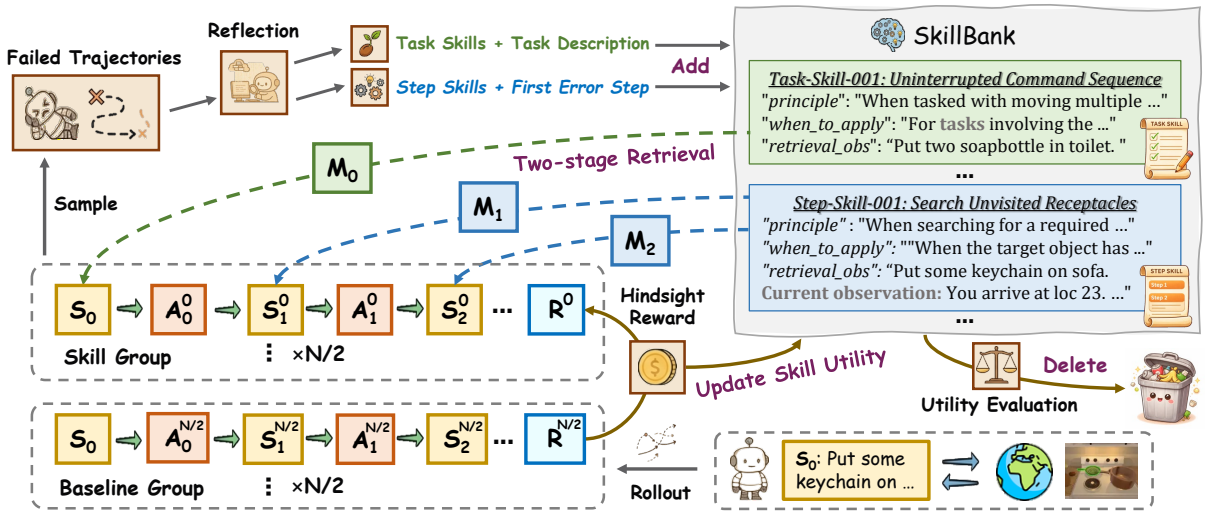


Figure 4: **Overall framework of D2Skill.** D2Skill couples RL with a dynamic dual-granularity skill bank. For each task, training rollouts are divided into a baseline group and a skill group, whose performance gap yields hindsight signals for policy optimization and skill utility estimation. When performance is poor, reflection on representative failed trajectories produces *task skills* for high-level guidance and *step skills* for local error correction. Skills are stored with retrieval keys, reused during interaction, and periodically pruned by utility-based bank management.

trajectory i are defined as

$$\Delta_g^{\text{task}} = \bar{Y}_g^{\text{skill}} - \bar{Y}_g^{\text{base}}, \quad c_i = Y_i - \bar{Y}_g^{\text{base}}. \quad (1)$$

Each skill m maintains a utility u_m updated using an exponential moving average. For a given task g, all retrieved task skills share the same signal Δ_g^{task}, since the task context is identical for the whole group. In contrast, multiple step skills may be retrieved at different steps and from different trajectories, and each retrieved step skill is updated using the credit of the trajectory in which it appears. The updates are defined as

$$\begin{aligned} u_m &\leftarrow (1-\beta_{\text{task}})u_m + \beta_{\text{task}}\Delta_g^{\text{task}}, \\ u_m &\leftarrow (1-\beta_{\text{step}})u_m + \beta_{\text{step}}c_i, \end{aligned} \quad (2)$$

where the first rule is applied to task skills retrieved in task g, and the second rule is applied to each step skill retrieved along skill-injected trajectory i.

Hindsight intrinsic reward shaping. To encourage effective use of retrieved skills, we introduce a hindsight intrinsic reward for trajectories in the skill group. For each skill-injected trajectory $i \in \mathcal{G}_g^{\text{skill}}$, the hindsight intrinsic reward is defined as

$$R_i^{\text{int}} = \lambda\big(Y_i - \bar{Y}_g^{\text{base}}\big), \quad (3)$$

where λ controls the strength of the shaping signal. This term measures performance gain over the baseline and encourages effective skill usage. The hindsight intrinsic reward is applied at the end of each skill-injected trajectory and included in the policy optimization.

Policy optimization with skill-augmented returns. The policy is optimized on the full samples. For each task g, trajectories in the skill group $\mathcal{G}_g^{\text{skill}}$ are generated under skill-augmented context and receive an additional reward R^{int}. Let R_i denote the origin return of trajectory i. For skill-injected trajectories, the return is augmented with R_i^{int}, and advantages are computed by group normalization over the whole trajectory group:

$$\begin{aligned} \tilde{R}_i &= \begin{cases} R_i + R_i^{\text{int}}, & i \in \mathcal{G}_g^{\text{skill}}, \\ R_i, & i \in \mathcal{G}_g^{\text{base}}, \end{cases} \\ A_i &= \frac{\tilde{R}_i - \text{mean}(\{\tilde{R}_j\}_{j\in\mathcal{G}_g})}{\text{std}(\{\tilde{R}_j\}_{j\in\mathcal{G}_g})}. \end{aligned} \quad (4)$$

We optimize the policy with a GRPO-style clipped objective using the hindsight-augmented advantages above; the full loss is given in Appendix A.2.

3.3 Skill Generation, Retrieval, and Bank Management

Reflection and skill generation. Reflection is triggered only for task groups with low performance, i.e., when $\bar{Y}_g^{\text{skill}} < \tau_{\text{ref}}$, where τ_{ref} is a reflection threshold. For each such task g, we sample one failed trajectory τ_g^- from the skill group and, **when available, one successful trajectory τ_g^+ from either the skill or the baseline group, and use them for skill generation**. The reflector produces at most one task skill and one step skill

for each task group, formalized as

$$m_g^{\text{task}} = f_{\text{reflect}}^{\text{task}}(g, \tau_g^-, \tau_g^+),$$
$$(m_g^{\text{step}}, o_j) = f_{\text{reflect}}^{\text{step}}(g, \tau_g^-, \tau_g^+), \quad (5)$$

where f_{reflect} denotes an external reflector LLM used for skill generation, and o_j denotes the observation at the earliest failure step j identified from the sampled failed trajectory.

For each skill m, we define a retrieval key k_m that determines when the skill is applicable. For $m \in \mathcal{M}_{\text{task}}$, the key is defined as $k_m = g$. For $m \in \mathcal{M}_{\text{step}}$, the key is defined as $k_m = (g, o_j)$. New skills are inserted into the skill bank after deduplication and participate in subsequent retrieval and utility updates.

Two-stage skill retrieval. When interacting with environment, skills are retrieved from the skill bank by matching the current query key with the retrieval key k_m of each skill. For task-level retrieval, the query key is $q = g$, while for step-level retrieval the query key is $q_t = (g, o_t)$, where g denotes the task identifier and o_t is the observation at step t.

In the first stage, we retrieve the ***top-m*** candidate skills from the pool $\mathcal{M} \in \{\mathcal{M}_{\text{task}}, \mathcal{M}_{\text{step}}\}$ according to cosine similarity between the embedding of q and k_m. A minimum similarity threshold τ_{sim} is applied, and only skills satisfying $\text{sim}(q, k_m) \geq \tau_{\text{sim}}$ are retained.

In the second stage, the candidates are ranked using a combination of semantic similarity and utility-based exploration. For each skill $m \in \mathcal{M}$, we define the selection score

$$\text{score}(m) = \alpha\, \widehat{\text{sim}}(m, q) + (1-\alpha)\left(u_m + \eta\sqrt{\frac{\log(1+N_r)}{1+n_m}}\right), \quad (6)$$

where $\widehat{\text{sim}}(m, q) \in [0, 1]$ is the normalized cosine similarity, u_m is the utility of skill m, n_m is the number of times the skill has been retrieved, and $N_r = \sum_{m' \in \mathcal{M}} n_{m'}$ is the total retrieval count in the active pool. The second term corresponds to a UCB-style bonus that encourages exploration of skills with low retrieval counts. The ***top-k*** (<top-m) skills ranked by this score are injected into the policy context.

Skill pruning by utility. To prevent unbounded growth of the skill bank, we periodically prune each skill pool $\mathcal{M}$ after validation intervals. Each pool is associated with a capacity limit $N_{\max}$. If $|\mathcal{M}| > N_{\max}$, each skill $m \in \mathcal{M}$ is assigned an eviction score

$$\text{evict}(m) = u_m + \eta\sqrt{\frac{\log(1+N_r)}{1+n_m}}. \quad (7)$$

Then, skills are sorted by $\text{evict}(m)$ in ascending order, and the lowest-scoring ones are removed until $|\mathcal{M}| \leq N_{\max}$. Skills created within the last T_{prot} training steps, i.e., $t - t_m^{\text{create}} < T_{\text{prot}}$, are excluded from eviction to allow sufficient evaluation.

4 Experiments

We evaluate D2Skill on three representative agentic settings: the interactive benchmarks ALFWORLD (Shridhar et al., 2020) and WEBSHOP (Yao et al., 2022), and SEARCH-AUGMENTED QA (Jin et al., 2025). Across these settings, we compare D2Skill against skill-free RL baselines and prior methods that augment agents with memory, skills, or search. Detailed training protocols and baseline descriptions are deferred to Appendix C. Our experiments address three questions:

1. **Main Performance:** Does D2Skill outperform standard RL and existing baselines on agentic tasks? (Section 4.1)
2. **Ablation:** What is the contribution of each major component to the overall gains? (Section 4.2)
3. **Analysis:** How does the dynamic skill bank affect skill utility, training dynamics, computational overhead, and policy generalization without skill-bank access at evaluation time? (Section 4.3)

4.1 Main Performance

Tables 1 and 2 show that **D2Skill consistently outperforms skill-free baselines across all three benchmarks**. Across the strongest D2Skill variants in Table 1, it reaches **90.6** overall success on ALFWORLD and **84.4** success on WEBSHOP under *Qwen2.5-7B-Instruct*. The same trend holds for the smaller *Qwen3-4B-Instruct-2507* (Yang et al., 2025), where D2Skill improves ALFWORLD overall success from **53.9** with GRPO to **72.7**, and remains stronger than GRPO on WEBSHOP. Compared with the most relevant skill-based baseline, D2Skill also yields clear gains on WEBSHOP and

Table 1: Performance on ALFWORLD and WEBSHOP. For ALFWORLD, we report the average success rate (%) for each subtask and the overall success rate. For WEBSHOP, we report the average score and average success rate (%). Evaluation details are given in Appendix C.1. * denotes results replicated from (Feng et al., 2025) and (Xia et al., 2026). The best and second-best results are highlighted in red and blue, respectively.

Method	ALFWorld							WebShop	
	Pick	Clean	Cool	Look	Heat	Pick2	All	Score	Success
Closed-source LLMs									
Gemini-3-Flash	96.4	57.1	96.2	85.7	72.2	95.3	85.2	14.1	16.5
O3	64.3	19.1	23.1	64.3	33.3	61.9	43.8	5.8	4.7
Base Model: *Qwen2.5-7B-Instruct*									
Origin	17.9	4.8	3.8	64.3	0.0	5.3	12.5	16.6	3.9
GRPO	88.3	73.3	76.0	83.3	81.3	40.0	75.0	86.0	72.6
Mem0+GRPO*	78.1	56.1	65.0	54.8	31.0	26.9	54.7	58.1	37.5
SimpleMem+GRPO*	89.5	60.0	64.9	36.3	50.0	26.3	62.5	67.8	46.9
SkillRL(*O3*)*	94.3	90.6	92.0	83.3	93.7	80.0	89.1	85.2	72.7
D2Skill(*Gemini-3-Flash*)	97.1	100.0	75.0	87.5	100.0	78.6	90.6	91.1	80.5
D2Skill(*O3*)	93.8	94.7	95.5	77.8	95.0	72.0	87.8	90.1	84.4
D2Skill(*Self*)	94.3	93.8	73.1	75.7	75.0	65.0	82.8	89.9	81.3
Base Model: *Qwen3-4B-Instruct-2507*									
Origin	50.0	9.5	0.0	2.1	11.1	4.8	17.2	0.0	0.0
GRPO	73.5	46.6	48.0	61.1	62.5	20.0	53.9	87.0	74.2
SkillRL(*O3*)	90.0	92.3	52.0	63.6	42.9	40.9	67.2	89.4	75.7
D2Skill(*Gemini-3-Flash*)	88.6	75.0	54.2	66.7	60.0	52.6	69.6	90.0	82.8
D2Skill(*O3*)	89.4	72.4	66.7	54.5	60.0	50.0	72.7	89.1	75.8
D2Skill(*Self*)	85.7	79.2	57.7	60.0	73.3	35.0	62.5	89.6	77.4
Base Model: *Qwen3-4B-Instruct-2507 + SFT*									
Origin	53.6	28.6	46.2	71.4	55.5	38.1	47.7	65.6	53.1
GRPO(*40-Steps*)	89.7	77.8	85.7	91.6	86.7	69.6	83.6	77.4	67.2
GRPO(*120-Steps*)	100.0	95.2	80.8	88.9	78.6	88.3	92.9	88.2	79.9
D2Skill(*40-Steps*)	92.9	100.0	95.2	80.0	90.9	86.7	92.2	84.1	71.9
D2Skill(*120-Steps*)	97.6	95.8	100.0	88.9	90.0	91.7	95.3	89.2	81.3

remains competitive or stronger on ALFWORLD. Baseline settings are summarized in Appendix C.3.

D2Skill also transfers effectively to SEARCH-AUGMENTED QA. Under *Qwen2.5-7B-Instruct*, it achieves **44.9** average accuracy and outperforms both skill-free and skill-augmented baselines overall, with especially clear gains on **multi-hop QA** such as HOTPOTQA and 2WIKI. We view this pattern as consistent with the role of step skills: multi-hop search requires a sequence of locally correct decisions, making fine-grained guidance particularly valuable. The same trend holds for *Qwen3-4B-Instruct-2507*, where average accuracy improves from **36.0** with Search-R1 to **42.8**.

We further evaluate D2Skill in the *Self* setting, where skills are generated without closed-source reflectors. Even under self-feedback alone, D2Skill still improves over GRPO, reaching **82.8** overall success on ALFWORLD and **81.3** success on WEBSHOP under *Qwen2.5-7B-Instruct*. The same pattern also appears for *Qwen3-4B-Instruct-2507*. These results suggest that the core framework gain does not depend on privileged closed-source supervision, although stronger external reflectors can still further improve trajectory diagnosis and skill abstraction.

Finally, we evaluate D2Skill under a stronger initialization setting by starting from a teacher-initialized policy based on SFT. In this setting, D2Skill improves **both training efficiency and final performance**: after only **40** training steps, it reaches **92.2** on ALFWORLD, already close to GRPO trained for 120 steps, and after 120 steps it further improves to **95.3** while continuing to outperform GRPO under the same budget. A similar trend also appears on WEBSHOP, where D2Skill remains better than GRPO under the same training budget. Together, these results show that D2Skill is effective not only in standard RL training, but also in self-feedback and stronger-initialization regimes.

Table 2: Performance on SEARCH-AUGMENTED QA. We report accuracy (%) on single-hop and multi-hop datasets.

Method	Single-Hop QA			Multi-Hop QA			Avg.
	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	
Base Model: *Qwen2.5-7B-Instruct*							
Origin	11.6	35.6	1.20	16.4	22.2	4.80	15.3
Search-R1	39.3	61.0	39.7	37.0	40.1	14.6	38.6
ZeroSearch	43.6	61.8	51.5	34.6	35.2	18.4	40.9
StepSearch	37.7	54.7	38.6	37.0	40.9	34.6	40.6
EvolveR	43.5	63.4	44.6	38.2	42.0	15.6	41.2
SkillRL(*O3*)	45.9	63.3	45.9	43.2	40.3	20.2	43.1
D2Skill(*O3*)	48.7	63.4	44.9	47.5	43.7	21.0	44.9
D2Skill(*Self*)	46.7	64.0	42.3	43.9	45.9	20.0	43.8
Base Model: *Qwen3-4B-Instruct-2507*							
Origin	18.4	45.5	28.4	22.8	18.0	6.3	23.2
Search-R1	39.5	62.6	41.3	34.9	24.8	12.9	36.0
Dr.Zero	41.3	61.6	44.0	38.4	29.8	14.4	38.3
D2Skill(*O3*)	44.7	61.4	42.6	42.2	46.5	19.7	42.8
D2Skill(*Self*)	44.9	60.2	44.8	41.6	44.7	16.7	42.1

Table 3: Ablation Study on ALFWORLD.

Method	Train		Val
	Skill	Baseline	
D2Skill	**70.9**	**65.8**	**72.7**
w/o task skills	59.1	53.7	62.7
w/o step skills	57.8	55.8	60.2
w/o skill management	60.0	57.4	57.8
w/o baseline group	63.9	-	68.8
w/o utility retrieval	61.4	51.8	64.8
w/o utility module	60.3	-	62.5
w/o skills (GRPO)	-	50.6	53.9

4.2 Ablation Study

We conduct ablations on ALFWORLD with QWEN3-4B-INSTRUCT-2507 to assess the contribution of each component in D2Skill. During training, we report the peak success rates of the skill and baseline groups, measured by the maximum 10-step moving average, and during validation we report the best held-out success rate. We consider six ablated variants: ***(i) w/o task skills***, removing task-level skills; ***(ii) w/o step skills***, removing step-level skills; ***(iii) w/o skill management***, disabling skill pruning and retaining all accumulated skills; ***(iv) w/o baseline group***, removing paired baseline rollouts and training with absolute rewards only; ***(v) w/o utility retrieval***, removing utility-aware ranking and using similarity-only retrieval; ***(vi) w/o utility module***, removing the utility mechanism, including baseline utility estimation and updates; and ***(vii) w/o skills (GRPO)*** as a skill-free reference.

The ablation results in Table 3 reveal three main findings. First, **removing either task skills or step skills consistently reduces performance**, indicating that both high-level task guidance and fine-grained step support are important to D2Skill. The fact that *w/o step skills* remains below SkillRL should not be over-interpreted, since SkillRL additionally relies on a privileged skill bank constructed from held-out trajectories, whereas this ablation remains a training-only variant within the D2Skill framework. Second, **the larger degradation caused by removing skill management** highlights the importance of dynamic bank maintenance in discarding ineffective skills and retaining compact, high-utility knowledge for reuse. Third, removing the baseline group or utility estimation results in smaller but still clear drops, suggesting that these components primarily **enhance credit assignment and skill valuation**, thereby improving optimization and retrieval quality, rather than driving the main gains directly.

4.3 Additional Analysis

Utility and transferability of the skill bank. As shown in Figure 3b, enabling skill management yields a skill bank and retrieved skills with consistently higher average utility, indicating that utility-aware maintenance improves memory and retrieval quality by filtering ineffective skills. Figure 5 further shows that the learned skills are **transferable**. Even without a skill bank at evaluation time, the policy trained with D2Skill remains competitive

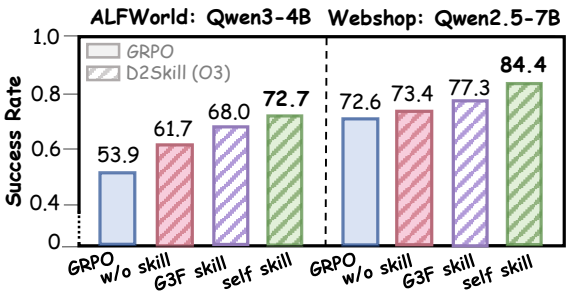


Figure 5: Eval with Different Skills.

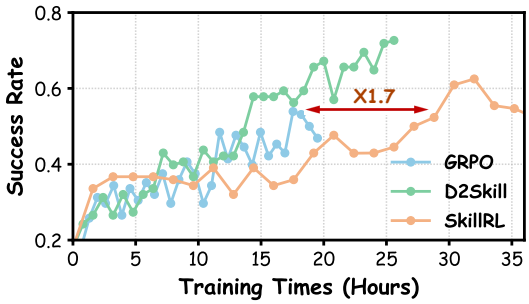


Figure 6: Val Success Dynamics.

with, or outperforms, GRPO, suggesting that part of the gain from skill augmentation has been **internalized into the policy** during training. Moreover, using the *Gemini-3-Flash*-generated skill bank from the corresponding training setting at evaluation time still yields clear gains over the no-skill variant in both ALFWORLD and WEBSHOP, while the self-generated skill bank remains the most effective. This suggests that D2Skill learns reusable skills that retain utility beyond the specific skill bank used during training.

Training cost. On ALFWORLD with *Qwen3-4B-Instruct-2507*, measured on 8×H100 GPUs, D2Skill takes 25.6 wall-clock training hours, remaining **close to GRPO** (20.8 hours) while being **substantially cheaper than SkillRL** (49.2 hours). Figure 6 further shows that D2Skill reaches strong evaluation performance much earlier in wall-clock time, making it about **1.7× faster than SkillRL** in practice. This low overhead mainly comes from an efficient retrieval pipeline: retrieval is executed with batched embedding queries and skill embeddings are updated incrementally, so only newly added skills need to be encoded after each bank update. As a result, D2Skill remains close to GRPO in training cost despite introducing dynamic skill retrieval and management. Further implementation details are available in our open-source codebase.

5 Related Works

5.1 Agent Evolution with Memory

Recent work studies agent evolution through external memory and reusable experience, enabling LLM agents to adapt beyond parameter updates (Zhang et al., 2024; Gao et al., 2025; Du, 2026). Existing methods explore memory retention, structured updating, retrieval-aware optimization, and experience abstraction into reusable lessons, strategies, or workflows (Chhikara et al., 2025; Xu et al., 2025; Zhou et al., 2025; Zhao et al., 2024; Cai et al., 2025b). D2Skill follows this direction, but focuses on RL-time skill valuation and maintenance through paired skill/base rollouts rather than only accumulating reusable textual memories.

5.2 Memory-augmented Agentic RL

Memory-augmented agentic RL treats external memory as a non-parametric complement to policy optimization, storing useful successful or failed experiences and retrieving them to guide future decisions (Liu et al., 2026b; Zhou et al., 2026). Recent methods increasingly organize such experience into reusable guidance for planning and action selection, with skills emerging as a particularly effective abstraction for long-horizon agentic tasks (Zhai et al., 2025; Cai et al., 2025a; Wang et al., 2026; Xia et al., 2026). D2Skill is most closely related to this line, but differs by jointly modeling task- and step-level skills and by dynamically valuing and managing them during training; further connections to hindsight learning, hierarchical abstraction, and contemporaneous methods are discussed in Appendix D.

6 Conclusion

We presented **D2Skill**, a dynamic dual-granularity skill bank framework for agentic RL that integrates task-level guidance, step-level correction, and utility-aware skill management within a joint training loop. Across interactive benchmarks and search-augmented QA, D2Skill consistently improves over strong baselines, while ablations and analyses confirm the importance of both dual-granularity skill modeling and dynamic bank maintenance. Despite these results, our study is still limited to a small set of benchmark environments, and the strongest settings continue to rely on external reflector models. Extending D2Skill to broader agentic domains while reducing this dependence remains an important direction for future work.

Limitations

We note several limitations of the current study:

- **Sensitivity of skill construction.** The quality of constructed skills still depends on the supervision source and training setup used during reflection and initialization. Improving the stability of skill construction under weaker or fully self-generated supervision remains an important direction for future work.
- **Benchmark scope.** We evaluate D2Skill on ALFWORLD, WEBSHOP, and SEARCH-AUGMENTED QA, which cover representative interactive and search-based settings, but do not yet capture the full diversity and noise of more open-ended real-world environments.
- **Skill quality and retrieval robustness.** The quality of generated skills and their utility estimates still depends on the underlying policy and reflector. In more complex environments, noisy, overly specific, or outdated skills may still affect retrieval quality and memory management.

Ethics Statement

This work uses public benchmarks only; it involves no human subjects, private data, or personally identifiable information, and we anticipate no direct harmful societal impact.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight experience replay. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30.

Yuzheng Cai, Siqi Cai, Yuchen Shi, Zihan Xu, Lichao Chen, Yulei Qin, Xiaoyu Tan, Gang Li, Zongyi Li, Haojia Lin, and 1 others. 2025a. Training-free group relative policy optimization. *arXiv preprint arXiv:2510.08191*.

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

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*.

Pengfei Du. 2026. Memory for autonomous llm agents: Mechanisms, evaluation, and emerging frontiers. *arXiv preprint arXiv:2603.07670*.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2025. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*.

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, and 8 others. 2025. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*.

Yiding Jiang, Shixiang Shane Gu, Kevin P. Murphy, and Chelsea Finn. 2019. Language as an abstraction for hierarchical deep reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32.

Zhennan Jiang, Shangqing Zhou, Yutong Jiang, Zefang Huang, Mingjie Wei, Yuhui Chen, Tianxing Zhou, Zhen Guo, Hao Lin, Quanlu Zhang, and 1 others. 2026. Wovr: World models as reliable simulators for post-training vla policies with rl. *arXiv preprint arXiv:2602.13977*.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

Jiaqi Liu, Yaofeng Su, Peng Xia, Siwei Han, Zeyu Zheng, Cihang Xie, Mingyu Ding, and Huaxiu Yao. 2026a. Simplemem: Efficient lifelong memory for llm agents. *arXiv preprint arXiv:2601.02553*.

Zeyuan Liu, Jeonghye Kim, Xufang Luo, Dongsheng Li, and Yuqing Yang. 2026b. Exploratory memory-augmented llm agent via hybrid on-and off-policy optimization. *arXiv preprint arXiv:2602.23008*.

Dilxat Muhtar, Jiashun Liu, Wei Gao, Weixun Wang, Shaopan Xiong, Ju Huang, Siran Yang, Wenbo Su, Jiamang Wang, Ling Pan, and Bo Zheng. 2026. Complementary reinforcement learning. *arXiv preprint arXiv:2603.17621*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*.

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. 2025. Zerosearch: Incentivize the search capability of llms without searching. *arXiv preprint arXiv:2505.04588*.

Songjun Tu, Yiwen Ma, Jiahao Lin, Qichao Zhang, Xiangyuan Lan, Junfeng Li, Nan Xu, Linjing Li, and Dongbin Zhao. 2026. Paperaudit-bench: Benchmarking error detection in research papers for critical automated peer review. *arXiv preprint arXiv:2601.19916*.

Yinjie Wang, Xuyang Chen, Xiaolong Jin, Mengdi Wang, and Ling Yang. 2026. Openclaw-rl: Train any agent simply by talking. *arXiv preprint arXiv:2603.10165*.

Ziliang Wang, Xuhui Zheng, Kang An, Cijun Ouyang, Jialu Cai, Yuhang Wang, and Yichao Wu. 2025. Stepsearch: Igniting llms search ability via stepwise proximal policy optimization. *arXiv preprint arXiv:2505.15107*.

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

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, and 1 others. 2026. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, and 1 others. 2025. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent. *arXiv preprint arXiv:2507.02259*.

Zhenrui Yue, Kartikeya Upasani, Xianjun Yang, Suyu Ge, Shaoliang Nie, Yuning Mao, Zhe Liu, and Dong Wang. 2026. Dr. zero: Self-evolving search agents without training data. *arXiv preprint arXiv:2601.07055*.

Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, and 1 others. 2025. Agentevolver: Towards efficient self-evolving agent system. *arXiv preprint arXiv:2511.10395*.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, and 1 others. 2025a. The landscape of agentic reinforcement learning for llms: A survey. *arXiv preprint arXiv:2509.02547*.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Zhuo Li, Yujie Zheng, Weinan Zhang, Ying Wen, Zhiyu Li, and 1 others. 2026a. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*.

Tianjun Zhang, Xuezhi Wang, Denny Zhou, Dale Schuurmans, and Joseph E. Gonzalez. 2023. The wisdom of hindsight makes language models better instruction followers. In *International Conference on Machine Learning (ICML)*, pages 41414–41428.

Xiaoying Zhang, Zichen Liu, Yipeng Zhang, Xia Hu, and Wenqi Shao. 2026b. Retroagent: From solving to evolving via retrospective dual intrinsic feedback. *arXiv preprint arXiv:2603.08561*.

Yaocheng Zhang, Haohuan Huang, Zijun Song, Yuanheng Zhu, Qichao Zhang, Zijie Zhao, and Dongbin Zhao. 2025b. Criticsearch: Fine-grained credit assignment for search agents via a retrospective critic. *arXiv preprint arXiv:2511.12159*.

Zeyu Zhang, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Quanyu Dai, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2024. A survey on the memory mechanism of large language model based agents. *arXiv preprint arXiv:2404.13501*.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. Expel: Llm agents are experiential learners. *arXiv preprint arXiv:2308.10144*.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, and 1 others. 2025. Memento: Fine-tuning llm agents without fine-tuning llms. *arXiv preprint arXiv:2508.16153*.

Huichi Zhou, Siyuan Guo, Anjie Liu, Zhongwei Yu, Ziqin Gong, Bowen Zhao, Zhixun Chen, Menglong Zhang, Yihang Chen, Jinsong Li, and 1 others. 2026. Memento-skills: Let agents design agents. *arXiv preprint arXiv:2603.18743*.

A Preliminaries

A.1 Agentic RL as a History-Augmented Decision Process

We consider agentic RL in long-horizon environments modeled as a Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $s_t \in \mathcal{S}$, $a_t \in \mathcal{A}$, and $(s_{t+1}, r_t, d_t) \sim P(\cdot \mid s_t, a_t)$. Unlike classical RL, the policy does not directly observe the environment state. Instead, the agent interacts through a textual interface that provides a partial description of the task and the interaction history.

For a task instance g, let τ_g denote the task specification, o_t the textual observation at step t, and $\mathcal{H}_t^L = \{(o_{t-L}, a_{t-L}), \ldots, (o_{t-1}, a_{t-1})\}$ the most recent L observation–action pairs retained in the prompt. Let $\mathcal{A}_t^{\mathrm{adm}} \subseteq \mathcal{A}$ be the admissible action set. The policy acts on the effective context $x_t = (\tau_g, \mathcal{H}_t^L, o_t, \mathcal{A}_t^{\mathrm{adm}})$, and selects actions according to $\pi_\theta(a_t \mid x_t)$. Although the underlying dynamics are Markovian in (s_t, a_t), the context x_t is a fixed-window summary of past interactions and is generally not a sufficient statistic of the latent state, so the resulting MDP can be viewed as a history-augmented partially observable MDP.

A.2 Skill Bank as an External Knowledge Store

In addition to the sliding window $\mathcal{H}_t^L$, we maintain a persistent skill bank $\mathcal{M}$, where each skill $m \in \mathcal{M}$ stores language guidance for decision making. At step t, a retrieval operator selects a small set of relevant skills $m_t \subseteq \mathcal{M}$ conditioned on the current context $x_t = (\tau_g, \mathcal{H}_t^L, o_t, \mathcal{A}_t^{\mathrm{adm}})$, and the policy acts on the augmented context $\tilde{x}_t = (x_t, m_t)$.

Taking GRPO (Shao et al., 2024) as the RL algorithm for example. For each task g, a group of N trajectories is sampled and advantages are computed by normalizing returns within the group. Under the skill-augmented context, the GRPO objective is $\bar{r}_i(\theta) = \mathrm{clip}(r_i(\theta), 1-\epsilon, 1+\epsilon)$:

$$\mathcal{L}_{\mathrm{GRPO}}(\theta) = \mathbb{E}_i\Big[\min\Big(r_i(\theta)\hat{A}_i, \bar{r}_i(\theta)\hat{A}_i\Big) - \beta D_{\mathrm{KL}}\big(\pi_\theta(\cdot \mid \tilde{x}_i) \| \pi_{\mathrm{ref}}(\cdot \mid \tilde{x}_i)\big)\Big].$$

Here $\tilde{x}_t^i$ denotes the context augmented with retrieved skills, $\hat{A}_{i,t}$ denote the normalized advantage and $r_{i,t}(\theta) = \frac{\pi_\theta(a_{i,t}|\tilde{x}_{i,t})}{\pi_{\theta_{\mathrm{old}}}(a_{i,t}|\tilde{x}_{i,t})}$ be the likelihood ratio. The policy is optimized under skill-augmented observations, while the objective remains the same as in standard RL.

B Overall Training Algorithm

For clarity, we provide the detailed training procedure of **D2Skill** in Algorithm 1. The algorithm summarizes how **paired rollout construction**, **dual-granularity skill retrieval**, **hindsight-based optimization**, and **reflection-driven skill updating** are integrated within a unified training loop.

C Experimental Details

C.1 Benchmarks and Evaluation Protocol

This subsection provides the supplementary evaluation details for Table 1 unless otherwise stated. For ALFWORLD, we report the average success rate (%) for each subtask and the overall success rate. For WEBSHOP, we report the average score and average success rate (%).

For the two interactive benchmarks, all methods are trained for 160 training steps, evaluated every 5 training steps on 128 validation tasks by default, and reported by their best performance over the full training run unless otherwise specified. Validation is conducted on held-out tasks with the skill bank frozen, where the agent only retrieves from $\mathcal{M}_{\mathrm{task}}$ and $\mathcal{M}_{\mathrm{step}}$ without reflection or bank updates. For SEARCH-AUGMENTED QA, models are trained for 200 steps and evaluated by their test-set performance at the final 200th training step.

For model initialization, we follow SkillRL and use an SFT-initialized model for *Qwen2.5-7B-Instruct* to ensure reliable instruction following under skill-augmented prompts. For *Qwen3-4B-Instruct-2507*, we use the original public instruct checkpoint by default. One special case is WEBSHOP with *Qwen3-4B-Instruct-2507*: we find that the raw origin model almost never completes tasks successfully and yields near-zero scores regardless of subsequent RL training. Therefore, for this setting we first SFT the model on the SkillRL SFT dataset so that it acquires basic task-completion ability before RL. In addition, for each *Self* setting, we further SFT the open model on about 1,000 reflection examples collected with O3 so that it has basic self-reflection ability before generating task and step skills on its own.

For the environment setup, we use a fixed seed and a short interaction history window of two observation–action pairs. Episodes are capped at 50 steps for ALFWORLD and 15 for WEBSHOP, and invalid actions are penalized during training. For WEBSHOP, we adopt the small split

with non-human goals. ALFWORLD is run with the TextWorld backend `AlfredTWEnv`.

C.2 Models and Implementation Details

This subsection summarizes the implementation details of D2Skill itself. Hyperparameters follow Table 4. The rollout group size is N=8, instantiating the paired baseline/skill rollout construction in Section 3.2. Retrieval hyperparameters $(\tau_{\text{sim}}, m, \alpha, \eta)$ correspond to the similarity filtering and utility-aware ranking in Equation (6), sharing the same UCB-style exploration term as the eviction rule in Equation (7). The number of injected skills k is environment-dependent to control context length.

Reflection is triggered by $\bar{Y}_g^{\text{skill}} < \tau_{\text{ref}}$ (Section 3.3), and skill eviction follows Equation (7) under constraints $N_{\max}$ and T_{prot}. Reflection is implemented with external LLMs (Gemini-3-Flash or O3) unless the *Self* setting is used. For the stronger teacher-initialized setting, we collect 300 successful trajectories per environment using strong external agents (O3 for ALFWORLD, Gemini-3-Pro for WEBSHOP) and perform SFT before RL.

Training is conducted with distributed RL and vLLM rollouts, and reported wall-clock time corresponds to runs on 8× H100 GPUs.

C.3 Baselines

For the interactive benchmarks, we compare against three categories of baselines. **Origin** denotes the base instruct model without RL updates. **GRPO** (Feng et al., 2025) is the skill-free RL baseline. **Mem0+GRPO** (Chhikara et al., 2025) and **SimpleMem+GRPO** (Liu et al., 2026a) augment RL with external memory, while **SkillRL** (Xia et al., 2026) is the most relevant prior skill-bank method. We also report direct rollout performance of strong closed-source LLMs as reference-only anchors, without RL training.

For SEARCH-AUGMENTED QA, we compare against representative search-agent baselines. **Search-R1** (Jin et al., 2025) trains search-capable agents with RL. **ZeroSearch** (Sun et al., 2025) improves search behavior without requiring actual search during training. **StepSearch** (Wang et al., 2025) applies step-wise PPO to search decisions. **EvolveR** (Wu et al., 2025) studies experience-driven self-evolving agents, and **Dr.Zero** (Yue et al., 2026) develops self-evolving search agents without supervised training data. We additionally include **SkillRL**(*O3*) as a skill-based baseline adapted to this setting.

C.4 Prompts

Overview. This section documents the prompt templates used for policy–environment interaction and reflection. Across ALFWORLD and WEBSHOP, we keep the environment-facing prompts largely aligned with the original benchmark formats. For skill-augmented rollouts, the only additional component is the inclusion of retrieved task-level and step-level experiences. The templates below are presented with minimal modification to preserve their original form.

Policy–environment prompts. For both ALFWORLD and WEBSHOP, we use two layouts: a baseline layout without retrieved skills and a skill-augmented layout with retrieved experiences. The policy prompt includes the task description, recent interaction history, current observation, and admissible actions. The model outputs reasoning enclosed in `<think>` tags and exactly one admissible action enclosed in `<action>` tags. These fields are parsed by the environment during rollout and evaluation.

Reflection prompt. When reflection is triggered, we construct a single prompt from one failed trajectory and, when available, one successful trajectory from the same task. The reflector identifies the first error step and produces one step-level reflection and one task-level reflection in structured form. These outputs are parsed and converted into skill entries for subsequent retrieval.

C.5 Examples of Skill Bank

To illustrate the behaviors captured by the proposed dual-granularity skill bank, we present representative examples of both *task skills* and *step skills* from ALFWORLD, WEBSHOP, and SEARCH-AUGMENTED QA.

Task skills. Task skills encode reusable high-level strategies that provide global guidance throughout the trajectory, structuring exploration and mitigating systematic errors.

Step skills. Step skills encode fine-grained, context-dependent guidance at specific decision points, especially after failure, and provide targeted corrections for local mistakes. Together with task skills, they complement global planning with local refinement.

D Additional Related Work and Comparisons

D.1 Agent Evolution and Memory Management

Recent work studies how language agents can evolve through external memory and reusable experience beyond pure parameter updates (Zhang et al., 2024; Gao et al., 2025; Du, 2026). Existing approaches examine memory retention, structured updating, retrieval-aware optimization, and experience abstraction into reusable lessons, strategies, or workflows (Chhikara et al., 2025; Xu et al., 2025; Zhou et al., 2025; Zhao et al., 2024; Cai et al., 2025b). Relative to these methods, D2Skill focuses on *RL-time* skill valuation and maintenance: skills are not only accumulated, but also scored through paired rollouts and managed throughout training.

D.2 Memory-Augmented Agentic RL and Skill Abstraction

Memory-augmented agentic RL treats external memory as a non-parametric complement to policy optimization, storing useful successful or failed experiences and retrieving them to guide future decisions (Liu et al., 2026b; Zhou et al., 2026). More recent methods increasingly organize such experience into reusable guidance for planning and action selection, with skills emerging as a particularly effective abstraction for long-horizon agentic tasks (Zhai et al., 2025; Cai et al., 2025a; Wang et al., 2026; Xia et al., 2026). ExpeL (Zhao et al., 2024), for example, distills trajectory-level textual insights and retrieves them as general reusable guidance. D2Skill is most closely related to this line, but differs from prior skill-bank methods in two key respects: it jointly models **task-level** and **step-level** skills, and it updates, values, and prunes them **during training** rather than treating the bank as largely static memory.

D.3 Hindsight and Hierarchical Abstraction

D2Skill is also related to hindsight learning and hierarchical abstraction in reinforcement learning. Hindsight Experience Replay reuses trajectories by relabeling goals to provide learning signals under sparse rewards (Andrychowicz et al., 2017), and hindsight instruction relabeling adapts a related idea to language-model instruction following (Zhang et al., 2023). Our hindsight signal is different in that it compares paired skill-injected and baseline rollouts under the same policy to estimate the utility of retrieved skills. The use of language skills as reusable abstractions is also related to hierarchical RL and language-mediated subtask abstraction (Jiang et al., 2019); D2Skill adopts a lightweight two-granularity design, separating task-level planning guidance from step-level decision support in agentic interaction.

D.4 Comparison with Contemporaneous Work

Contemporaneous works such as RetroAgent (Zhang et al., 2026b) and Complementary RL (Muhtar et al., 2026) are related in spirit to our approach, showing that self-evolving experience can substantially improve agentic RL performance. However, their results rely on more elaborate prompting pipelines for retrospection and experience extraction, which may increase system complexity and prompt dependence. SkillRL (Xia et al., 2026) is the most closely related prior work to D2Skill. Although SkillRL also distinguishes between two task types, this mainly reflects task categorization rather than different skill granularities, and its guidance remains task-level: each task retrieves skills once and uses them throughout the trajectory. In contrast, D2Skill maintains both task skills and step skills, enabling high-level guidance and fine-grained support with retrieval at each interaction step. Moreover, D2Skill performs skill generation and management during training, rather than relying on privileged validation information for skill construction.

Algorithm 1. D2Skill: Dynamic Dual-Granularity Skill Bank for Agentic RL

Require: Policy π_θ; task set $\mathcal{D}$; skill banks $\mathcal{M}_{\text{task}}, \mathcal{M}_{\text{step}}$; group size N; reflection threshold τ_{ref}

Initialize π_θ, $\mathcal{M}_{\text{task}} \leftarrow \emptyset$, $\mathcal{M}_{\text{step}} \leftarrow \emptyset$
for training step $t = 1, \ldots, T$ **do**
  Sample a batch of tasks $\{g\} \sim \mathcal{D}$
  for each task g **do**
    Sample a trajectory group $\mathcal{G}_g$ with N rollouts
    Split $\mathcal{G}_g$ into a **baseline group** $\mathcal{G}_g^{\text{base}}$ and a **skill group** $\mathcal{G}_g^{\text{skill}}$
    Roll out $\mathcal{G}_g^{\text{base}}$ under the original context

$$x_s = (\tau_g, \mathcal{H}_s^L, o_s, \mathcal{A}_s^{\text{adm}})$$

    Roll out $\mathcal{G}_g^{\text{skill}}$ with **dual-granularity skill retrieval**, where task skills are retrieved by key g, step skills are retrieved by key (g, o_s), and injected as

$$\tilde{x}_s = (x_s, m_s^{\text{task}}, m_s^{\text{step}})$$

    Obtain trajectory returns $\{R_i\}_{i \in \mathcal{G}_g}$ and terminal success indicators $\{Y_i\}_{i \in \mathcal{G}_g}$
    Compute group success rates $\bar{Y}_g^{\text{base}}$ and $\bar{Y}_g^{\text{skill}}$, then form the **task-level hindsight signal**

$$\Delta_g^{\text{task}} = \bar{Y}_g^{\text{skill}} - \bar{Y}_g^{\text{base}}$$

    for each skill trajectory $i \in \mathcal{G}_g^{\text{skill}}$ **do**
      Compute the **step-level credit** and hindsight intrinsic reward

$$c_i = Y_i - \bar{Y}_g^{\text{base}}, \qquad R_i^{\text{int}} = \lambda\big(Y_i - \bar{Y}_g^{\text{base}}\big)$$

      Augment the return by $\tilde{R}_i \leftarrow R_i + R_i^{\text{int}}$
      Update utilities of retrieved skills:

$$u_m \leftarrow (1 - \beta_{\text{task}}) u_m + \beta_{\text{task}} \Delta_g^{\text{task}},$$
$$u_m \leftarrow (1 - \beta_{\text{step}}) u_m + \beta_{\text{step}} c_i$$

    end for
    For each $i \in \mathcal{G}_g^{\text{base}}$, set $\tilde{R}_i \leftarrow R_i$
    Compute group-normalized advantages $\{A_i\}_{i \in \mathcal{G}_g}$ from $\{\tilde{R}_i\}_{i \in \mathcal{G}_g}$
    if $\bar{Y}_g^{\text{skill}} < \tau_{\text{ref}}$ **then**
      Trigger **reflection-driven skill generation** using one failed trajectory τ_g^- and, when available, one successful trajectory τ_g^+
      Generate at most one task skill m_g^{task} and one step skill m_g^{step}
      Deduplicate and insert them into $\mathcal{M}_{\text{task}}$ and $\mathcal{M}_{\text{step}}$
    end if
  end for
  Update π_θ on all collected trajectories using GRPO with group-normalized advantages
  Periodically perform **utility-aware skill bank management** by pruning $\mathcal{M}_{\text{task}}$ and $\mathcal{M}_{\text{step}}$ under capacity constraints
end for
return Trained policy π_θ, task-skill bank $\mathcal{M}_{\text{task}}$, and step-skill bank $\mathcal{M}_{\text{step}}$

Table 4: Hyperparameters and training protocol for main experiments.

Category	Symbol / field	Value
Benchmark / prompt context		
ALFWORLD max steps		50
WEBSHOP max steps		15
History length L	`env.history_length`	2
Environment seed	`env.seed`	0
Invalid-action penalty coef.	`invalid_action_penalty_coef`	1.0
Batch / schedule		
Training tasks per step	`train_batch_size`	16
Validation tasks	`val_batch_size`	128
Val frequency	`test_freq`	every 5 steps
Training steps *	`total_epochs`	160
Prompt max tokens	`max_prompt_length`	4096
Generation max tokens	`max_response_length`	512
GRPO / PPO optimization		
Group size per task	N (`env.rollout.n`)	8
Skill vs. baseline split		$N/2$ each (50%)
Learning rate		1×10^{-6}
PPO clip range	ϵ	0.2
KL penalty coef.	β_{KL}	0.01 (low-var. KL)
KL in reward		disabled
Discount	γ	1.0
PPO epochs per update		1
PPO mini-batch size		128
Rollout (vLLM)		
TP size		1
Val sampling temperature		0.4
Val `do_sample`		True
D2Skill — retrieval		
Similarity threshold	τ_{sim}	0.7
Stage-1 shortlist size	m (`retrieval_top_2k`)	10
Rerank mixing weight	α (`retrieval_alpha`)	0.1
UCB scale (ranking)	η (`retrieval_ucb_c`)	0.05
Injected task skills k (ALFWORLD)	`top_k_task`	3
Injected step skills k (ALFWORLD)	`top_k_step`	3
Injected task / step k (WEBSHOP)		1 / 1
D2Skill — utility & shaping		
Task-skill EMA	β_{task}	0.5
Step-skill EMA	β_{step}	0.5
Hindsight intrinsic coef.	λ	1.0
Credit vs. baseline		enabled
D2Skill — reflection & management		
Reflection threshold	τ_{ref}	0.5 (skill-group success)
Traj. cap for reflection		≤ 10
Max task skills	$N_{\max}$ (`eviction_max_task_skills`)	300
Max step skills	$N_{\max}$ (`eviction_max_step_skills`)	300
Protection window	T_{prot}	10 (training steps)

Prompt 1. ALFWORLD policy template with retrieved skills.

```
You are an expert agent operating in the ALFRED embodied environment.

Your goal is to complete the following task:
{task_description}

## Current Progress
You have already taken {step_count} step(s).

Recent interaction history (observation → action):
{action_history}

Current step: {current_step}

Current observation:
{current_observation}

Admissible actions at this step:
[{admissible_actions}]

## Relevant Experience
Below are past experiences retrieved from memory. They may include:

- **Task-level experience**: for this kind of task as a whole (what to aim for, what to avoid).
- **Step-level experience**: for the current situation (what to do at this step when the
↪  observation is similar).

You should review both before deciding your next action.

When reasoning, you may:

- Use task-level experience to guide your overall plan and avoid known pitfalls
- Use step-level experience when the current observation matches the described situation
- Reuse successful actions if the situations are similar; avoid actions that previously led to
↪  failure

Warning: These lessons may be outdated. Use them only if they align with your current observation.

Retrieved experiences:
{retrieved_memories}

## Instructions
For the current step, you should follow this process:

1. Analyze the current observation
2. Review the retrieved experiences and think about whether any past experience applies
3. Reason step-by-step and choose the best admissible action

Now it's your turn to take an action.
You should first reason step-by-step about the current situation. This reasoning process MUST be
↪  enclosed within <think> </think> tags.
Once you've finished your reasoning, you should choose an admissible action for current step and
↪  present it within <action> </action> tags.
```

Prompt 2. WEBSHOP policy template with retrieved skills.

```
You are an expert autonomous agent operating in the WebShop e-commerce environment.

Your goal is to complete the following task:
{task_description}

## Current Progress
You have already taken {step_count} step(s).

Recent interaction history (observation → action):
{action_history}

Current step: {current_step}

Current observation:
{current_observation}

Admissible actions at this step:
[{available_actions}]

## Relevant Experience
Below are past experiences retrieved from memory. They may include:

- **Task-level experience**: for this kind of task as a whole (what to aim for, what to avoid).
- **Step-level experience**: for the current situation (what to do at this step when the
↪  observation is similar).

You should review both before deciding your next action.

When reasoning, you may:

- Use task-level experience to guide your overall plan and avoid known pitfalls
- Use step-level experience when the current observation matches the described situation
- Reuse successful actions if the situations are similar; avoid actions that previously led to
↪  failure

Warning: These lessons may be outdated. Use them only if they align with your current observation.

Retrieved experiences:
{retrieved_memories}

## Instructions
For the current step, you should follow this process:

1. Analyze the current observation
2. Review the retrieved experiences and think about whether any past experience applies
3. Reason step-by-step and choose the best admissible action

Now it's your turn to take an action.
You should first reason step-by-step about the current situation. This reasoning process MUST be
↪  enclosed within <think> </think> tags.
Once you've finished your reasoning, you should choose an admissible action for current step and
↪  present it within <action> </action> tags.
```

Prompt 3. Reflection template for skill construction.

```
Output the following in order (use the exact section headers):

1) FIRST_ERROR_STEP: N
   Where N is the 1-based turn number in the *failed* trajectory where the agent first went wrong (e.g. Turn 1, Turn 2). Use
   ↪  0 if unclear.

2) STEP_REFLECTION (one step-level experience for that step only):
   Output a JSON object with exactly: "title", "principle", "when_to_apply".
   This should be a concise experience for what to do at that specific step/situation (e.g. "At this step you should ...").

Example: {"title": "Check object location first", "principle": "Before picking, verify the object is in the expected
↪  receptacle.", "when_to_apply": "When the observation mentions an object but its location is unclear"}

3) TASK_REFLECTION (one task-level skill for the whole task):
   Output a JSON object with exactly: "title", "principle", "when_to_apply".
   This summarizes for the *whole task*: what to avoid and how to succeed in this kind of task (will be retrieved by task
   ↪  description).

Example: {"title": "Plan object location before acting", "principle": "For this task type, first identify where the object
↪  is, then plan the sequence of actions.", "when_to_apply": "When the task involves finding or moving a specific object"}

Output format (use these exact labels):
FIRST_ERROR_STEP: N

STEP_REFLECTION:
<single JSON object>

TASK_REFLECTION:
<single JSON object>
```

An Example of Task Skills (ALFWorld)

```
"title": "Plan the temperature-change workflow",
"principle": "For cooling tasks: 1) locate and take the target object, 2) move directly to a
↪  fridge/freezer and execute the 'cool OBJECT with FRIDGE' action, 3) carry the cooled object
↪  to the required destination and place it. Minimize wandering and avoid unrelated actions.",
"when_to_apply": "At the outset of any task that involves cooling (or heating) an object before
↪  placing it elsewhere",
"retrieval_obs": "cool some pan and put it in stoveburner."
```

An Example of Task Skills (WebShop)

```
"title": "Systematically narrow down products by constraints",
"principle": "For complex find-and-buy tasks, first craft a query capturing the main attributes,
↪  then use on-page filters (color, size, price) before purchasing, verifying every required
↪  feature in the product page.",
"when_to_apply": "Whenever the task requires purchasing an item that must satisfy multiple
↪  explicit constraints (fabric, color code, size, price, care, etc.).",
"retrieval_obs": "Find me hand wash women's sweaters with long sleeve, stretch fabric, polyester
↪  spandex for teen girls, daily wear with color: xnj-tshirt345-black, and size: large, and
↪  price lower than 50.00 dollars"
```

An Example of Task Skills (Search-Augmented QA)

```
"title": "Multi-hop entity verification",
"principle": "Break the question into two stages: first resolve the intermediate entity described
↪  in the query, then search for the specific attribute requested about that entity using
↪  evidence from the first hop.",
"when_to_apply": "When a search question refers to the target indirectly through another work,
↪  person, or entity.",
"retrieval_obs": "How many hardcovers appeared in the number-one position of \"The New York
↪  Times\" Bestseller List by the author of \"Night Chills\"?"
```

An Example of Step Skills (ALFWorld)

```
"title": "Broaden the search after negative result",
"principle": "Once you verify the target object is not in the first place you looked, switch to
↪  other likely locations instead of looping over the same empty containers.",
"when_to_apply": "Right after opening the fridge and seeing it contains no potato.",
"retrieval_obs": "cool some potato and put it in garbagecan.\n\nCurrent observation: You arrive
↪  at loc 18. The cabinet 1 is open. In it, you see nothing."
```

An Example of Step Skills (WebShop)

```
"title": "Verify option is present before selecting",
"principle": "Only issue a click when the desired option (e.g., size, color) is explicitly listed
↪  in the current observation; otherwise locate the correct menu first.",
"when_to_apply": "When the goal requires choosing a variant (size, color, etc.) and the
↪  observation does not yet display those variants.",
"retrieval_obs": "Find me loose fit, day comfort, hand wash women's tops, tees & blouses with
↪  short sleeve, polyester spandex for teen girls with color: b17-army green, and size: medium,
↪  and price lower than 40.00 dollars"
```

An Example of Step Skills (Search-Augmented QA)

```
"title": "Identify target attributes in multi-hop questions",
"principle": "When a question links two entities, first resolve the intermediate entity, then
↪  launch a focused follow-up search for the final attribute requested by the question instead
↪  of answering from the first retrieved document.",
"when_to_apply": "When an initial search reveals the missing link in a multi-hop question but not
↪  the final target attribute.",
"retrieval_obs": "In what year was the game console for which Kowloon\'s Gate was released in
↪  1997 released in Japan?\n\nCurrent observation: <information>{\"result\": \"Doc 1:
↪  \\\"Kowloon\'s Gate\\\" ...\"}"
```