SKILLRL: EVOLVING AGENTS VIA RECURSIVE SKILL-AUGMENTED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Model (LLM) agents have shown stunning results in complex tasks, yet they often operate in isolation, failing to learn from past experiences. Existing memory-based methods primarily store raw trajectories, which are often redundant and noise-heavy. This prevents agents from extracting high-level, reusable behavioral patterns that are essential for generalization. In this paper, we propose SKILLRL, a framework that bridges the gap between raw experience and policy improvement through automatic skill discovery and recursive evolution. Our approach introduces an experience-based distillation mechanism to build a hierarchical skill library SKILLBANK, an adaptive retrieval strategy for general and task-specific heuristics, and a recursive evolution mechanism that allows the skill library to co-evolve with the agent's policy during reinforcement learning. These innovations significantly reduce the token footprint while enhancing reasoning utility. Experimental results on ALFWorld and WebShop benchmarks demonstrate that SKILLRL achieves state-of-the-art performance, outperforming strong baselines over 14% and maintaining robustness as task complexity increases.

1 INTRODUCTION

Large language model (LLM) agents (Yao et al., 2022b; Shinn et al., 2023) have demonstrated remarkable capabilities across various sophisticated tasks, such as web navigation (Google, 2025; OpenAI, 2025c) and deep research (OpenAI, 2025b; Google, 2024; Team et al., 2025), by interacting with complex environments through natural language. Despite these advances, each task execution remains largely episodic. Current LLM agents operate in isolation, unable to learn from past successes or failures (Zhang et al., 2025b), which significantly hinders their evolution. Consequently, a fundamental challenge remains: *how can agents efficiently learn from experience and transfer that knowledge to other tasks?*

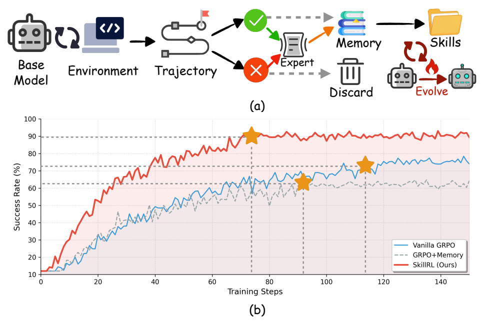


Figure 1: (a) Overview of the SKILLRL pipeline. Unlike previous methods (gray dashed lines) that store raw trajectories and discard failures, SKILLRL employs an experience-based distillation mechanism to transform diverse experiences into structured skills. (b) Performance on ALFWorld validation set. SKILLRL achieves faster convergence and superior success rates.

The existing memory-based methods for LLM agents primarily involve saving raw trajectories directly into external databases during the sampling process to serve as references for similar future tasks (Shinn et al., 2023; Zhao et al., 2024). While intuitive, these raw trajectories are often lengthy and contain significant redundancy and noise (Chhikara et al., 2025), making it difficult for the model to extract critical information. Recent work has attempted to compress trajectories and update the memory bank via online training (Zhang et al., 2025b; 2026), improving memory efficiency. However, these methods merely mimic past solutions and they fail to distill core principles or adapt the agent's internal policy to leverage memory for guided decision-making. As depicted in the dashed flow of Figure 1(a), such approaches often struggle with the trade-off between information density and noise, leading to sub-optimal performance or even degradation as shown in Figure 1(b).

We argue that these approaches miss a crucial insight: effective experience transfer requires *abstraction*. Human experts do not memorize every action in every situation; instead, they develop *skills* (Anthropic, 2024), compact and reusable strategies that capture the essence of how to accomplish specific subtasks. Inspired by this observation, we propose SKILLRL, a framework that bridges the gap between raw experience and efficient policy improvement through automatic skill discovery and recursive skill evolution.

SKILLRL first introduces an experience-based skill distillation mechanism, which gathers diverse trajectories from environment rollouts and applies differential processing: successful episodes are preserved as demonstrations, while failed ones are synthesized into concise failure lessons to mitigate context noise. Secondly, we transform these experiences into a hierarchical skill library SKILLBANK, differentiating between *general skills* for universal strategic guidance and *task-specific skills* for task-level heuristics. This abstraction allows the agent to adaptively retrieve relevant skills during decision-making, significantly reducing the token footprint while enhancing reasoning utility. Lastly, SKILLRL incorporates a recursive skill evolution mechanism during reinforcement learning (RL), where the skill library is treated as a dynamic component rather than a static knowledge source. By analyzing failure modes after each validation epoch to generate new skills or refine existing ones, our approach ensures the skill library and the agent's policy co-evolve, maintaining robustness as task complexity increases. As demonstrated in Figure 1(b), SKILLRL achieves substantially faster convergence and higher asymptotic performance.

The primary contribution is SKILLRL, a framework that enables LLM agents to bridge the gap between raw experience and policy improvement through automatic skill discovery and recursive evolution. By distilling redundant trajectories into a hierarchical SKILLBANK, our method abstracts general and task-specific skills to guide decision-making efficiently. Furthermore, we introduce a recursive evolution mechanism that ensures the skill library and agent policy co-evolve during reinforcement learning. Empirical results on ALFWorld and WebShop benchmarks demonstrate that SKILLRL achieves state-of-the-art performance with 14% improvements, significantly outperforming current memory-based agent-tuning baselines in both task success and reasoning utility.

2 PRELIMINARIES

LLM Agents. We consider an agent operating in an interactive environment $\mathcal{E}$. At each timestep t, the agent observes a state $o_t \in \mathcal{O}$, selects an action $a_t \in \mathcal{A}$, and receives a reward r_t and next observation o_{t+1}. A trajectory $\tau = (o_0, a_0, r_0, \ldots, o_T, a_T, r_T)$ captures one episode of interaction. Tasks are specified by natural language descriptions d. An LLM-based agent parameterized by θ implements a policy $\pi_\theta(a_t | o_{\leq t}, d, c)$ where c represents additional context (e.g., skills, demonstrations). Our goal is to learn a policy that maximizes expected return $\max_\theta \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \gamma^t r_t\right]$ subject to context length constraints $|c| \leq L_{\max}$.

Group Relative Policy Optimization (GRPO). GRPO (Shao et al., 2024) is a reinforcement learning method that avoids training a critic by using intra-group relative rewards to optimize the policy. For each query x, the model samples G responses $\{y^{(1)}, \ldots, y^{(G)}\}$, which are scored to obtain rewards $\{R_1, \ldots, R_G\}$. GRPO computes normalized advantages and updates the policy with a PPO-style clipped objective (Schulman et al., 2017):

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E}_{x,\{y_i\}}\left[\frac{1}{G}\sum_{i=1}^{G} \min\left(r_i A_i, \text{clip}(r_i, 1-\epsilon, 1+\epsilon)A_i\right) - \beta D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})\right] \tag{1}$$

where $r_i = \frac{\pi_\theta(y_i|x)}{\pi_{\text{old}}(y_i|x)}$ is the importance ratio, $A_i = \frac{R_i - \text{mean}(\{R_j\}_{j=1}^G)}{\text{std}(\{R_j\}_{j=1}^G)}$ is the normalized advantage, ϵ and β are hyperparameters, and π_{old} is the policy before the current update. GRPO enables scalable policy learning using only relative rewards, without requiring a learned critic.

3 SKILLRL

In this section, as illustrated in Figure 2, we propose SKILLRL, a framework designed to bridge the gap between raw interaction experience and policy improvement through automatic skill discovery and recursive evolution. SKILLRL consists of three core components. First, we develop

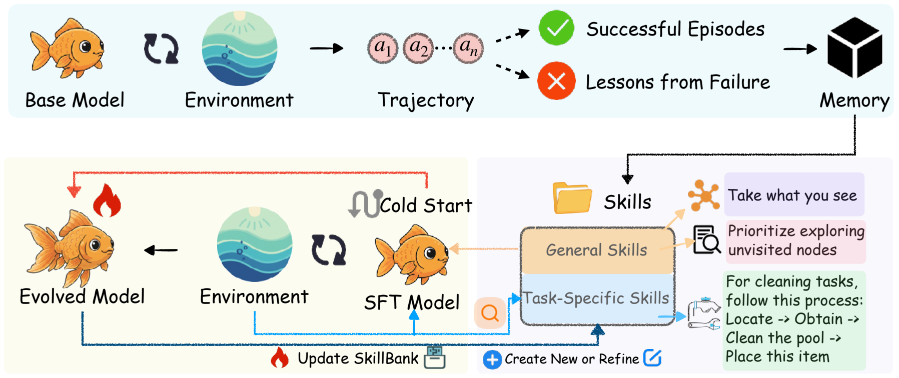


Figure 2: Overview of the SkillRL framework. We collect trajectories using a base model, distill them into a hierarchical skill library, perform cold-start SFT to enable skill utilization, and then conduct RL training with dynamic skill evolution based on validation failures.

an experience-based skill distillation mechanism to transform redundant trajectories into concise, actionable knowledge. Second, we organize these distilled experiences into a hierarchical skill library $\mathcal{S}$, enabling efficient retrieval of general and task-specific expertise. Lastly, we introduce a recursive skill evolution mechanism that leverages RL to dynamically refine the skill library in tandem with the agent's policy. We detail these components as follows:

3.1 Experience-based Skill Distillation

Raw trajectories τ collected from environment interactions are verbose, containing exploratory actions, backtracking, and redundant steps that obscure the critical decisions leading to success or failure. To transform these experiences into actionable knowledge, we employ a teacher model $\mathcal{M}_T$ to distill trajectories into compact, reusable skills.

Specifically, we first deploy a base LLM agent π_{base} in the target environment $\mathcal{E}$ to collect diverse trajectories. Unlike prior approaches that retain only successful episodes, we deliberately preserve both successful trajectories $\mathcal{T}^+ = \{\tau_i : r(\tau_i) = 1\}$ and failed trajectories $\mathcal{T}^- = \{\tau_i : r(\tau_i) = 0\}$, where $r(\tau)$ denotes the binary task success indicator. Failed trajectories reveal failure modes and boundary conditions, i.e., information difficult to infer from successes alone.

We apply differential processing based on trajectory outcomes. For *successful trajectories* $\tau^+ \in \mathcal{T}^+$, we extract the strategic patterns that led to task completion:

$$s^+ = \mathcal{M}_T(\tau^+, d). \tag{2}$$

The teacher model identifies critical decision points, the reasoning behind correct actions, and generalizable patterns that transfer beyond the specific task instance.

For *failed trajectories* $\tau^- \in \mathcal{T}^-$, direct inclusion in context is infeasible due to their length and noise. Instead, we synthesize concise failure lessons:

$$s^- = \mathcal{M}_T(\tau^-, d). \tag{3}$$

The analysis identifies: (1) the point of failure, (2) the flawed reasoning or action, (3) what should have been done, and (4) general principles to prevent similar failures. This transforms verbose failed episodes into actionable avoidance guidance.

3.2 Hierarchical Skill Library (SkillBank) Construction

Following the design principles of Agent Skills (Anthropic, 2024), we organize the distilled knowledge into a hierarchical skill library SkillBank that enables efficient retrieval of relevant expertise during decision-making.

Skill Organization. We structure SkillBank into two levels: 1) *General Skills* $\mathcal{S}_g$ capture universal strategic principles applicable across all task types within an environment. These typically include exploration strategies (e.g., systematic search patterns, prioritizing unvisited locations), state management principles (e.g., verifying preconditions before actions), and goal-tracking heuristics (e.g., maintaining progress counters, terminating only upon verified completion). General skills provide foundational guidance that transfers across different task categories. 2) *Task-Specific Skills* $\mathcal{S}_k$ encode specialized knowledge for task category k. These capture domain-specific action sequences, task-particular preconditions and constraints, common failure modes unique to the task type, and optimized procedures that exploit task structure. By organizing trajectories by task type during collection, we enable extraction of fine-grained, category-specific strategies that complement the broader general skills.

The complete skill library SKILLBANK is $\mathcal{S}_g \cup \bigcup_{k=1}^{K} \mathcal{S}_k$. Each skill $s \in$ SKILLBANK is structured with: a concise name (e.g., systematic exploration), a principle describing the strategy, and when_to_apply conditions specifying applicability. This format enables efficient retrieval while providing clear guidance for application.

Skill Retrieval. At inference, given a task description d, the agent retrieves relevant skills to augment its context. General skills $\mathcal{S}_g$ are always included as foundational guidance. Task-specific skills are retrieved via semantic similarity:

$$\mathcal{S}_{\text{ret}} = \text{TopK}\left(\{s \in \mathcal{S}_k : \text{sim}(e_d, e_s) > \delta\}, K\right), \tag{4}$$

where e_d, e_s are embeddings of the task description and skill respectively, δ is a similarity threshold, and K controls the number of retrieved skills. The policy then conditions on the retrieved skills:

$$a_t \sim \pi_\theta(a_t | o_{\leq t}, d, \mathcal{S}_g, \mathcal{S}_{\text{ret}}). \tag{5}$$

Notably, skill distillation achieves 10–20× token compression compared to raw trajectories while enhancing rather than degrading the utility of the original experience. This compression allows the agent to leverage rich experiential knowledge within limited context windows.

3.3 RECURSIVE SKILL EVOLUTION VIA REINFORCEMENT LEARNING

A static skill library cannot anticipate all scenarios the agent will encounter. As the policy improves and explores new state regions, it faces situations where existing skills provide insufficient guidance. We introduce recursive skill evolution during reinforcement learning to address this limitation, enabling the skill library and agent policy to co-evolve.

Cold-Start Initialization. Before RL training, we address a critical challenge: the base agent has not learned how to effectively utilize skills. Simply providing skills to an unchanged model yields limited benefit (Guo et al., 2025). We therefore perform a cold-start supervised fine-tuning (SFT) stage (Ouyang et al., 2022), where the teacher model $\mathcal{M}_T$ generates N skill-augmented reasoning traces $\mathcal{D}_{\text{SFT}} = \{(d_i, \mathcal{S}_i, \tau_i^*)\}_{i=1}^{N}$ demonstrating how to retrieve, interpret, and apply skills during decision-making. The base model is then fine-tuned on these demonstrations:

$$\theta_{\text{sft}} = \arg\min_\theta \mathcal{L}_{\text{CE}}(\mathcal{D}_{\text{SFT}}; \theta), \tag{6}$$

where $\mathcal{L}_{\text{CE}}$ denotes the cross-entropy loss. The resulting model $\pi_{\theta_{\text{sft}}}$ serves as both the starting point for RL training and the reference policy π_{ref} for KL regularization.

Recursive Skill Evolution. A static skill library cannot anticipate all scenarios the agent will encounter. As the policy improves and explores new state regions, it faces situations where existing skills provide insufficient guidance. We introduce recursive skill evolution to address this limitation. The process begins with an initial skill library containing baseline task-action principles.

After each validation epoch, we monitor the success rate $Acc(C)$ for each task category C. To ensure targeted growth, the evolution is triggered only for categories where $Acc(C) < \delta$. We then collect failed trajectories $\mathcal{T}_{\text{val}}^- = \{\tau_j : r(\tau_j) = 0\}_{j=1}^{M}$ using a diversity-aware stratified sampling strategy: trajectories are grouped by category, prioritized by the severity of failure (negative rewards), and selected via round-robin sampling to maintain categorical entropy. Then we will analyze these samples to identify gaps:

$$\mathcal{S}_{\text{new}} = \mathcal{M}_T(\mathcal{T}_{\text{val}}^-, \text{SKILLBANK}). \tag{7}$$

The teacher model is prompted to: (1) identify failure patterns not addressed by current skills, (2) propose new skills to cover these gaps, and (3) suggest refinements to existing skills that proved ineffective. The library is then updated: $\textsc{SkillBank} \leftarrow \textsc{SkillBank} \cup \mathcal{S}_{\text{new}}$.

This creates a virtuous cycle: as the agent improves, it encounters new challenges, which drive skill library expansion, which enables further improvement.

RL-based Policy Optimization. We optimize the skill-augmented policy using GRPO. For each task with description d, the agent first retrieves relevant skills and then samples G complete trajectories $\{\tau^{(1)}, \dots, \tau^{(G)}\}$ from the current policy π_θ. Each trajectory $\tau^{(i)}$ receives a binary reward $R_i = r(\tau^{(i)}) \in \{0, 1\}$ indicating task success or failure. The normalized advantage for each trajectory is computed as:

$$A_i = \frac{R_i - \text{mean}(\{R_j\}_{j=1}^G)}{\text{std}(\{R_j\}_{j=1}^G)}. \tag{8}$$

The policy is updated according to:

$$\mathcal{J}(\theta) = \mathbb{E}_{d,\{\tau^{(i)}\}}\left[\frac{1}{G}\sum_{i=1}^{G} \min\Big(\rho_i A_i, \text{clip}(\rho_i, 1-\epsilon, 1+\epsilon)A_i\Big) - \beta D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})\right], \tag{9}$$

where $\rho_i = \frac{\pi_\theta(\tau^{(i)}|d, \mathcal{S}_g, \mathcal{S}_{\text{ret}})}{\pi_{\text{old}}(\tau^{(i)}|d, \mathcal{S}_g, \mathcal{S}_{\text{ret}})}$ is the importance ratio computed over the skill-augmented context. The KL penalty anchored to $\pi_{\text{ref}} = \pi_{\theta_{\text{sft}}}$ ensures that RL optimization preserves the learned skill utilization capabilities while improving task performance.

4 Experiments

Table 1: Performance on ALFWorld and WebShop. For ALFWorld, we report the average success rate (%) for each subtask as well as the overall result. For WebShop, we report both the average score and the average success rate (%). * denotes the results replicated from (Feng et al., 2025). The best results and second best results are highlighted in red and blue, respectively.

Method	ALFWorld							WebShop	
	Pick	Look	Clean	Heat	Cool	Pick2	All	Score	Succ.
Closed-source LLMs									
GPT-4o	75.3	60.8	31.2	56.7	21.6	49.8	48.0	31.8	23.7
Gemini-2.5-Pro	92.8	63.3	62.1	69.0	26.6	58.7	60.3	42.5	35.9
Qwen2.5-7B-Instruct									
Qwen2.5	33.4	21.6	19.3	6.9	2.8	3.2	14.8	26.4	7.80
Prompt-based Agentic or Memory-based Methods									
ReAct*	48.5	35.4	34.3	13.2	18.2	17.6	31.2	46.2	19.5
Reflexion*	62.0	41.6	44.9	30.9	36.3	23.8	42.7	58.1	28.8
Mem0	54.0	55.0	26.9	36.4	20.8	7.69	33.6	23.9	2.00
ExpeL	21.0	67.0	55.0	52.0	71.0	6.00	46.3	30.9	11.2
MemP	54.3	38.5	48.1	56.2	32.0	16.7	41.4	25.3	6.40
RL-based Methods									
RLOO*	87.6	78.2	87.3	81.3	71.9	48.9	75.5	80.3	65.7
GRPO*	90.8	66.1	89.3	74.7	72.5	64.7	77.6	79.3	66.1
Memory-Augmented RL-based Methods									
MemRL	62.8	38.5	22.2	12.5	8.00	0.00	21.4	29.5	9.20
EvolveR	64.9	33.3	46.4	13.3	33.3	33.3	43.8	42.5	17.6
Mem0+GRPO	78.1	54.8	56.1	31.0	65.0	26.9	54.7	58.1	37.5
SkillRL	97.9	71.4	90.0	90.0	95.5	87.5	89.9	85.2	72.7

We evaluate SkillRL on two challenging benchmarks for LLM agents: ALFWorld and WebShop. Our experiments address the following questions: 1) How does SkillRL compare to state-of-the-art methods? 2) What is the contribution of each component? 3) How does the skill library evolve during training? 4) Does skills accelerate model convergence?

4.1 Experimental Setup

Environments. ALFWorld (Shridhar et al.) is a text-based game aligned with the ALFRED embodied AI benchmark. Agents must complete household tasks by navigating and interacting with objects through text commands. WebShop (Yao et al., 2022a) simulates web shopping. Agents navigate a realistic web interface to find and purchase products matching user specifications.

Baselines. We compare SkillRL against three types of methods. 1) (1) Closed-source LLMs: GPT-4o (OpenAI, 2024) and Gemini-2.5-Pro (Comanici et al., 2025). 2) Prompt-based agentic or memory-based methods, including ReAct (Yao et al., 2022b), Reflexion (Shinn et al., 2023), Mem0 (Chhikara et al., 2025), ExpeL (Zhao et al., 2024), MemP (Fang et al., 2025). 3) RL-based methods, including PPO (Schulman et al., 2017), RLOO (Ahmadian et al., 2024) and GRPO (Shao et al., 2024). 4) Memory-augmented RL-based methods, including EvolveR (Wu et al., 2025), MemRL (Zhang et al., 2026) and Mem0+GRPO.

Implementation Details. We use Qwen2.5-7B-Instruct (Bai et al., 2023) as our base model and OpenAI o3 (OpenAI, 2025a) as the teacher model for skill distillation and SFT data generation. For RL training, we use GRPO with learning rate 1×10^{-6}, batch size 16, group size 8, and 4 gradient accumulation steps. We set $K = 6$ for task-specific skill retrieval and $\delta = 0.4$ for the collection of failed trajectories. For more detailed information on training hyperparameters, please see Appendix C.1.

4.2 Main Results

Comparison with Baselines. We compare SkillRL with baseline methods across two benchmarks as shown in Table 1. Our method consistently outperforms all baselines, with key observations as follows:

1) *Significant Gains over Prompt-based Methods*. SkillRL achieves a 89.9% success rate on ALFWorld and 72.7% on WebShop, outperforming the best prompt-based baselines by a large margin. This gap suggests that while in-context learning can leverage past experiences, it often fails to distill actionable knowledge from verbose trajectories or fundamentally adapt the agent's policy.

2) *Superiority over Vanilla RL*. RL training brings substantial gains, yet SkillRL consistently surpasses standard RL baselines. Compared to PPO, RLOO, and GRPO, SkillRL achieves the best overall performance. Notably, since SkillRL utilizes GRPO as its base optimizer, the 12.3% absolute improvement over GRPO on ALFWorld (from 77.6% to 89.9%) is directly attributable to our skill-augmentation mechanism rather than algorithmic variance. In complex subtasks like *Cool* and *Pick2*, SkillRL outperforms GRPO by 23.0% and 22.8% respectively, proving that structured skill priors effectively accelerate and enhance policy learning in sparse-reward environments.

3) *Advantage over Memory-Augmented RL*. SkillRL substantially outperforms existing memory-augmented RL frameworks, which differ in how they manage and update experience. MemRL, which uses RL solely to update its memory bank while keeping the policy frozen, fails to adapt to complex environments, yielding only 21.4% on ALFWorld. EvolveR, which jointly updates the policy and memory bank, shows improvement (43.8%) but remains limited by its reliance on rough trajectory storage. To provide a more competitive baseline, we implemented Mem0+GRPO, which combines a state-of-the-art prompt-based memory mechanism with an optimized policy model. While this hybrid approach improves performance to 54.7% on ALFWorld and 37.5% on WebShop, it still trails SkillRL by a wide margin (about 35.2% absolute success rate gap). These results validate our core hypothesis: effective experience transfer requires high-level skill abstraction and a co-evolving library rather than simple trajectory compression or prompt-based memory retrieval.

Comparison with Closed-Source Models. Remarkably, SkillRL with Qwen2.5-7B-Instruct significantly outperforms much larger closed-source models, as shown in Table 1. On ALFWorld, our method exceeds GPT-4o (OpenAI, 2024) by 41.9% and Gemini-2.5-Pro (Comanici et al., 2025) by 29.6%. This demonstrates that effective skill learning can compensate for model scale, enabling smaller open-source models to achieve superior task performance through structured experiential knowledge.

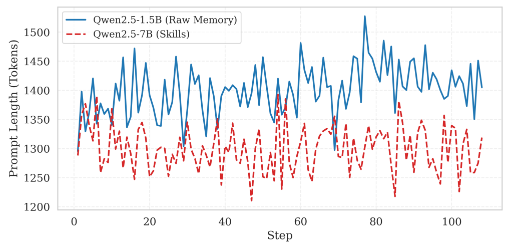


Figure 4: Comparison of prompt length (tokens) between raw memory retrieval and our distilled skill abstraction. SKILLRL consistently reduces context overhead while maintaining reasoning utility.

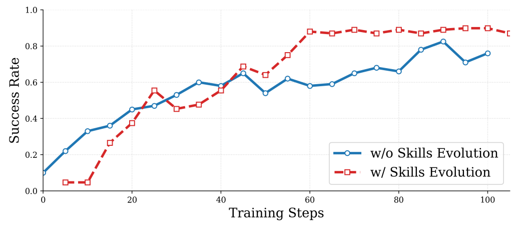


Figure 5: Success rate on ALFWorld validation set. The recursive skill evolution significantly accelerates convergence and enhances the overall performance ceiling.

4.3 ANALYSIS

In this section, we provide detailed analysis of each module's effectiveness and the skill evolution dynamics.

Table 2: Ablation study results. We report average success rate (%) for two datasets.

Method	ALFWorld	WebShop
SKILLRL	**89.9**	**72.7**
Skill Library Ablations		
w/o Hierarchical Structure	76.8	61.4
w/o Skill Library (Raw Trajectories)	61.7	50.2
Training Pipeline Ablations		
w/o Cold-Start SFT	65.2	46.5
w/o Dynamic Evolution	84.4	70.3

Ablation Studies. We conduct ablation experiments to evaluate each component's contribution, with results in Table 2. According to the results: (1) Removing hierarchical structure (i.e., task-specific skills only) decreases performance by 13.1% on ALFWorld and 11.3% on WebShop, indicating universal strategic principles provide essential foundational guidance. (2) Replacing the skill library with raw trajectories causes the largest degradation (up to 25%), which directly supports our motivation that abstraction is superior to memorization. Raw experiences introduce significant redundancy and noise that hinder effective knowledge transfer. (3) Cold-start SFT proves critical (20% drop without it), confirming that the base model requires an initial explicit demonstration phase to learn how to adaptively retrieve and utilize the abstracted skills before entering the RL stage. (4) Dynamic evolution contributes a 5.5% improvement by ensuring the skill library is a dynamic component rather than a static database. This co-evolution allows the agent to iteratively refine its internal policy by addressing emergent failure modes that were not covered by the initial skill set.

Per-Task Analysis on ALFWorld. Table 1 breaks down ALFWorld performance by task type. The largest gains are on PickTwo (+23%), Cool (+22%) and Heat (+15%), which are among the most challenging tasks requiring multi-step planning and state tracking. Task-specific skills are particularly valuable here, capturing strategies like "when picking two objects, verify the first is secured before searching for the second" that address common failure modes.

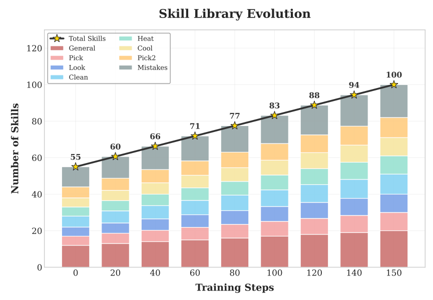
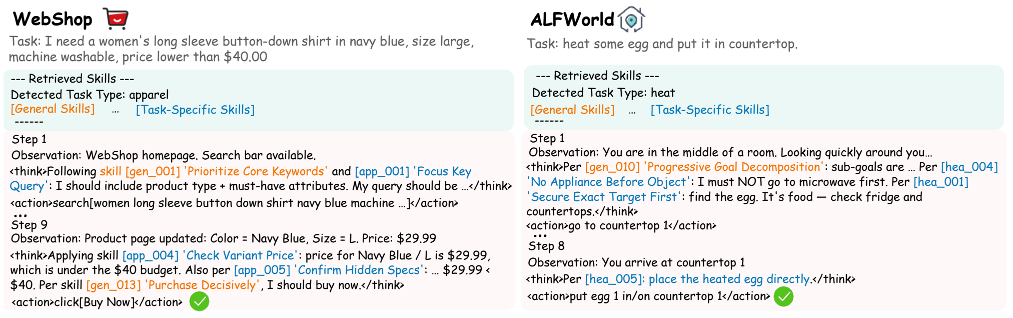


Figure 3: Evolution of skill library size during RL training. Dynamic skill evolution adds skills at validation checkpoints.

Skill Library Growth. Figure 3 shows how the skill library evolves during training. The initial skill library contains 55 skills (12 general, 43 task-specific). Through dynamic evolution, this grows to 100 skills by the end of training (Step 150). The growth is predominantly driven by task-specific skills (increasing from 43 to 80), while general skills show a steadier increase (from 12 to 20). Notably, we observe a balanced expansion across various task categories, ensuring the agent develops specialized expertise for each environment rollout. This overall expansion reflects the agent's increasing ability to refine its repertoire and tackle diverse scenarios within specific task types.



```
WebShop
Task: I need a women's long sleeve button-down shirt in navy blue, size large,
machine washable, price lower than $40.00
--- Retrieved Skills ---
Detected Task Type: apparel
[General Skills]   ...   [Task-Specific Skills]
 ------
Step 1
Observation: WebShop homepage. Search bar available.
<think>Following skill [gen_001] 'Prioritize Core Keywords' and [app_001] 'Focus Key
Query': I should include product type + must-have attributes. My query should be ...</think>
<action>search[women long sleeve button down shirt navy blue machine ...]</action>
...
Step 9
Observation: Product page updated: Color = Navy Blue, Size = L. Price: $29.99
<think>Applying skill [app_004] 'Check Variant Price': price for Navy Blue / L is $29.99,
which is under the $40 budget. Also per [app_005] 'Confirm Hidden Specs': ... $29.99 <
$40. Per skill [gen_013] 'Purchase Decisively', I should buy now.</think>
<action>click[Buy Now]</action> ✓
```

```
ALFWorld
Task: heat some egg and put it in countertop.
 --- Retrieved Skills ---
 Detected Task Type: heat
[General Skills]   ...   [Task-Specific Skills]
 ------
Step 1
Observation: You are in the middle of a room. Looking quickly around you...
<think>Per [gen_010] 'Progressive Goal Decomposition': sub-goals are ... Per [hea_004]
'No Appliance Before Object': I must NOT go to microwave first. Per [hea_001]
'Secure Exact Target First': find the egg. It's food — check fridge and
countertops.</think>
<action>go to countertop 1</action>
...
Step 8
Observation: You arrive at countertop 1
<think>Per [hea_005]: place the heated egg directly.</think>
<action>put egg 1 in/on countertop 1</action> ✓
```

Figure 6: Case studies of SKILLRL on WebShop and ALFWorld. The examples illustrate how the agent adaptively retrieves and integrates General Skills and Task-Specific Skills within its reasoning process to achieve precise and efficient task execution.

Context Efficiency. To evaluate the impact of skill abstraction on inference overhead, we compare the average prompt length of SKILLRL with a memory-augmented baseline using raw trajectories (Qwen2.5-7B with Raw Memory) in Figure 4. The results reveal that while the raw memory approach suffers from a high and fluctuating token footprint (averaging $\sim$1,450 tokens), SKILLRL maintains a significantly leaner prompt (averaging $<$1,300 tokens), achieving approximately a 10.3% reduction in context length. This efficiency stems from our distillation mechanism, which compresses verbose environment interactions into high-density, actionable skills. Notably, SKILLRL requires less context than the memory-based baseline to achieve superior performance, demonstrating that skill abstraction effectively mitigates the context-bloat problem common in traditional memory-based agents.

Evolution Dynamics. Figure 5 illustrates the reinforcement learning training curves with and without the recursive skill evolution mechanism. We observe that while SKILLRL without evolution shows steady improvement, SKILLRL with skill evolution exhibits a notably higher learning rate and superior asymptotic performance. Specifically, SKILLRL achieves a success rate of over 80% within 60 training steps, whereas the baseline requires approximately 90 steps to reach a lower peak. This acceleration in convergence suggests that the dynamic introduction of new skills and refinement of existing ones effectively provide the agent with timely strategic guidance to overcome local optima. Furthermore, the higher performance ceiling validates that the co-evolution of the skill library and the policy allows the agent to adapt to increasingly complex task scenarios that static memory methods fail to resolve.

Qualitative Analysis. To further investigate how SKILLRL utilizes the learned knowledge, we visualize the reasoning process on ALFWorld and WebShop in Figure 6. The case studies demonstrate that our trained agent can effectively retrieve and execute relevant skills from the SKILLBANK to guide its decision-making. For instance, in the WebShop task, the agent invokes general strategies like *"Prioritize Core Keywords"* alongside task-specific heuristics *"Focus Key Query"* to ensure the product meets all constraints within a limited budget. Similarly, in ALFWorld, the agent coordinates hierarchical skills, i.e., using *"Progressive Goal Decomposition"* for high-level planning and *"No Appliance Before Object"* to avoid common logical pitfalls. This seamless integration of general and specific skills confirms that the agent does not merely memorize trajectories, but rather develops a structured understanding of task logic, allowing for more robust and efficient problem-solving.

5 CONCLUSION

We introduced SKILLRL, a framework for skill-augmented reinforcement learning in LLM agents. By distilling raw trajectories into compact, reusable skills and enabling dynamic skill evolution during training, SKILLRL achieves state-of-the-art performance on ALFWorld and WebShop while using substantially less context than memory-based approaches. Our work demonstrates that the abstraction from experience to skill is a powerful principle for building capable, sample-efficient agents.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, 2024.

Anthropic. The claude 3 model family: Opus, sonnet, haiku, 2024. URL `https://www.anthropic.com/news/claude-3-family`.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

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

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, et al. A survey on in-context learning. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 1107–1128, 2024.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. *arXiv preprint arXiv:2508.06433*, 2025.

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

Huan-ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, et al. A survey of self-evolving agents: On path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 2025.

Google. Try deep research and our new experimental model in gemini, your ai assistant, 2024. URL `https://blog.google/products/gemini/google-gemini-deep-research/`.

Google. Introducing the gemini 2.5 computer use model, 2025. URL `https://blog.google/technology/google-deepmind/gemini-computer-use-model/`.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Yuyang Hu, Shichun Liu, Yanwei Yue, Guibin Zhang, Boyang Liu, Fangyi Zhu, Jiahang Lin, Honglin Guo, Shihan Dou, Zhiheng Xi, et al. Memory in the age of ai agents. *arXiv preprint arXiv:2512.13564*, 2025.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008, 2023.

Jiaqi Liu, Kaiwen Xiong, Peng Xia, Yiyang Zhou, Haonian Ji, Lu Feng, Siwei Han, Mingyu Ding, and Huaxiu Yao. Agent0-vl: Exploring self-evolving agent for tool-integrated vision-language reasoning. *arXiv preprint arXiv:2511.19900*, 2025.

OpenAI. Gpt-4o system card, 2024. `https://openai.com/index/gpt-4o-system-card/`.

OpenAI. Introducing o3 and o4-mini, 2025a. `https://openai.com/index/introducing-o3-and-o4-mini/`.

OpenAI. Openai deep research system card, 2025b. URL `https://openai.com/index/introducing-deep-research/`.

OpenAI. Openai computer-using agent, 2025c. URL `https://openai.com/index/computer-using-agent/`.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*, 2025.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks*, 113:54–71, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*.

Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, Peng Xia, Fang Wu, He Zhu, et al. Agent kb: Leveraging cross-domain experience for agentic problem solving. *arXiv preprint arXiv:2507.06229*, 2025.

Tongyi DeepResearch Team, Baixuan Li, Bo Zhang, Dingchu Zhang, Fei Huang, Guangyu Li, Guoxin Chen, Huifeng Yin, Jialong Wu, Jingren Zhou, et al. Tongyi deepresearch technical report. *arXiv preprint arXiv:2510.24701*, 2025.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*.

Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. Mem-{\alpha}: Learning memory construction via reinforcement learning. *arXiv preprint arXiv:2509.25911*, 2025.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024.

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

Peng Xia, Kaide Zeng, Jiaqi Liu, Can Qin, Fang Wu, Yiyang Zhou, Caiming Xiong, and Huaxiu Yao. Agent0: Unleashing self-evolving agents from zero data via tool-integrated reasoning. *arXiv preprint arXiv:2511.16043*, 2025.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022b.

Guibin Zhang, Muxin Fu, Guancheng Wan, Miao Yu, Kun Wang, and Shuicheng Yan. G-memory: Tracing hierarchical memory for multi-agent systems. *arXiv preprint arXiv:2506.07398*, 2025a.

Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. Memevolve: Meta-evolution of agent memory systems. *arXiv preprint arXiv:2512.18746*, 2025b.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yutao Qi, et al. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024.

APPENDIX

A RELATED WORK

LLM Agents. The emergence of capable LLMs has catalyzed rapid development in autonomous agent systems. ReAct (Yao et al., 2022b) interleaves reasoning and acting, enabling chain-of-thought style planning during interaction, while Reflexion (Shinn et al., 2023) introduces verbal reinforcement through self-reflection on past failures. Frameworks like AutoGen (Wu et al., 2024) and CAMEL (Li et al., 2023) demonstrate general-purpose multi-agent capabilities, featuring automated orchestration and diverse tool integration. While initial efforts focused on constrained tasks like coding or basic arithmetic, these approaches primarily rely on in-context learning (ICL) (Dong et al., 2024). However, these agents struggle to scale as tasks become more complex, as they treat every interaction as an isolated event and must start each new task from scratch without any prior knowledge.

Memory Mechanisms in Agents. To overcome the limitations of finite context windows and the inability of agents to learn from experience, external memory architectures have become a cornerstone of agent design (Hu et al., 2025). Early systems primarily utilized a static RAG paradigm or stored raw trajectories as few-shot examples (Wang et al.; Chhikara et al., 2025; Zhang et al., 2025a; Wang et al., 2024). However, raw trajectories are often token-heavy and contain significant redundancy and noise, which can lead to performance degradation. Current research has moved toward self-improving memory, distilling interactions into higher-level insights or procedural tips (Tang et al., 2025; Fang et al., 2025; Zhao et al., 2024; Ouyang et al., 2025). While some recent work explores updating memory banks via online training to improve efficiency (Zhang et al., 2025b; 2026), many existing methods still struggle to distinguish high-value experiences from noise or fail to distill core principles that can guide internal decision-making.

Evolution of Agentic Skills and Reinforcement Learning. The development of agentic skills (Anthropic, 2024), which are compact, reusable strategies that capture the essence of subtasks, is increasingly viewed through the lens of Continual Learning (CL) and Reinforcement Learning (RL). Traditional CL (Parisi et al., 2019) focuses on knowledge preservation in predefined tasks, but self-evolving agents (Gao et al., 2025; Xia et al., 2025; Liu et al., 2025) aim for active skill acquisition in open-ended environments (Fang et al., 2025; Wang et al., 2025). While RL is widely used to align LLMs (Schulman et al., 2017; Ouyang et al., 2022), or improve reasoning via rule-based verifiers (Shao et al., 2024), applying it to agentic skills remains challenging due to sparse rewards and long horizons. Unlike previous memory-augmented RL which treats memory as a static or auxiliary source, recent trends suggest that the key to efficient experience transfer lies in abstraction (Wu et al., 2025). Our work builds on this by treating the skill library as a dynamic component that co-evolves with the agent's policy, utilizing RL to refine structured skills through recursive failure analysis.

B PROMPTS

In this section, we provide the full prompt templates used throughout the different phases of our framework. These templates are designed to ensure consistent agent behavior and structured data generation across various environments.

B.1 AGENT EXECUTION PROMPTS

The following prompts are used during the online inference phase. These templates provide the agent with the current task description, a history of previous interactions, and a set of retrieved skills (experiences) to guide its decision-making process. The prompts explicitly enforce a Chain-of-Thought (CoT) reasoning step before action selection.

Prompt A.1: ALFWorld Agent Execution with Skills

System Prompt:
You are an expert agent operating in the ALFRED Embodied Environment. Your task is to: {task_description}

## Retrieved Relevant Experience
{retrieved_memories}

## Current Progress
Prior to this step, you have already taken {step_count} step(s). Below are the most recent {history_length} observations and the corresponding actions you took: {action_history}
You are now at step {current_step} and your current observation is: {current_observation}
Your admissible actions of the current situation are: [{admissible_actions}].

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 A.2: WebShop Agent Execution with Skills

System Prompt:
You are an expert autonomous agent operating in the WebShop e-commerce environment. Your task is to: {task_description}.

## Retrieved Relevant Experience
{retrieved_memories}

## Current Progress
Prior to this step, you have already taken {step_count} step(s). Below are the most recent {history_length} observations and the corresponding actions you took: {action_history}
You are now at step {current_step} and your current observation is: {current_observation}
Your admissible actions of the current situation are: [{available_actions}].

Now it's your turn to take one action for the current step. You should first reason step-by-step about the current situation, then think carefully which admissible action best advances the shopping goal. 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.

B.2 SKILL GENERATION AND DISTILLATION PROMPTS

These prompts are utilized during the skill discovery and library initialization phases. They guide a high-capability teacher model to analyze interaction trajectories, identify failure modes, and distill reusable, actionable skills into a structured JSON format.

Prompt B.1: Dynamic Skill Discovery from Failures

Analyze these failed {`env_description`} agent trajectories and suggest NEW skills to add.

FAILED TRAJECTORIES: {`failure_examples`}
EXISTING SKILL TITLES: {`existing_titles`}

Generate 1-3 NEW actionable skills that would help avoid these failures. Each skill must have: `skill_id`, `title` (3-5 words), `principle` (1-2 sentences), `when_to_apply`. The `skill_id` should be unique and follow the pattern: ”dyn_001”, ”dyn_002”, etc.

Return ONLY a JSON array of skills, no other text.

Prompt B.2: Initial Skill Distillation (ALFWorld)

You are an expert at distilling agent behavior patterns into concise, actionable skills. Analyze these successful and failed trajectories from an embodied AI agent operating in household environments (ALFWorld).

SUCCESSFUL TRAJECTORIES: {`success_patterns`}
FAILED TRAJECTORIES: {`failure_patterns`}

Generate 8-12 GENERAL SKILLS that apply across ALL task types. These should be: 1. **Concise**; 2. **Actionable**; 3. **Transferable**; 4. **Failure-aware**. Focus on: Navigation, object manipulation, state tracking, error recovery, and container interaction rules.

Return ONLY the JSON array, no other text.

Prompt B.3: Initial Skill Distillation (WebShop)

You are an expert at distilling agent behavior patterns into concise, actionable skills. Analyze these successful and failed trajectories from an AI agent operating in an online shopping environment (WebShop).

SUCCESSFUL TRAJECTORIES: {`success_patterns`}
FAILED TRAJECTORIES: {`failure_patterns`}

Generate 10-15 GENERAL SKILLS. Focus on: Search query formulation, product selection heuristics, option configuration (size, color, etc.), constraint verification, navigation patterns, and price handling.

Return ONLY the JSON array, no other text.

B.3 Cold-start Trajectory Generation Prompts

To bridge the gap between a base model and the target performance, we use the following prompts to generate high-quality synthetic trajectories for Supervised Fine-Tuning (SFT). These prompts instruct the teacher model to solve tasks while explicitly demonstrating the application of specific skills, thereby providing a clear learning signal for the student model.

Prompt C.1: Synthetic Trajectory Generation (ALFWorld)

You are an expert agent in the ALFRED embodied environment. You will be given a task and relevant skills to apply. Your goal is to generate a successful trajectory that demonstrates proper use of these skills.

You should generate a step-by-step trajectory that:
1. Uses the provided skills appropriately;
2. Takes realistic actions in the environment;
3. Completes the task successfully;
4. Demonstrates good planning and systematic exploration.

For each step, you should:
• Think through the current situation using `<think></think>` tags.
• Choose an appropriate action using `<action></action>` tags.
• The action should be a simple command like ”go to cabinet 1”, ”open drawer 2”, ”take apple 1”, ”put apple 1 in/on countertop 1”.

Generate a complete trajectory from start to finish. Stop when the task is complete.

Prompt C.2: Synthetic Trajectory Generation (WebShop)

You are an expert shopping agent in the WebShop e-commerce environment. You will be given a shopping task and relevant skills to apply. Your goal is to generate a successful trajectory that demonstrates proper use of these skills.

You should generate a step-by-step trajectory that:
1. Uses the provided skills appropriately;
2. Takes realistic actions in the WebShop environment;
3. Successfully finds and purchases the requested product;
4. Demonstrates good search strategies and product evaluation.

For each step, you should:
• Think through the current situation using `<think></think>` tags.
• Choose an appropriate action using `<action></action>` tags.
• Actions can be: `search[query]`, `click[element]`, or `buy now`.

Generate a complete trajectory from start to finish. Stop when the purchase is complete.

C ADDITIONAL EXPERIMENTAL DETAILS

C.1 HYPERPARAMETERS

C.2 COMPUTE RESOURCES

All experiments were conducted on a cluster with 8 NVIDIA H100 80GB GPUs. Training times:

- Trajectory collection: 3 hours
- Skill distillation: 0.5 hours
- Cold-start SFT: 2 hour
- RL training: 24 hours

Total wall-clock time: approximately 30 hours per experiment.

D ILLUSTRATION OF SKILL LIBRARY

In this section, we provide some example catalog of distilled skills and error taxonomies for both the ALFWorld and WebShop environments. Tables 4 and 6 detail the general skills distilled for embodied manipulation and web-based shopping, respectively, highlighting the actionable principles required for systematic exploration and constraint satisfaction. Furthermore, we provide a structured analysis of failure cases in Table 5 and Table 7, which categorizes common mistakes, ranging from spatial

Table 3: Hyperparameters for SKILLRL.

Hyperparameter	Value
Cold-Start SFT	
Learning rate	1×10^{-4}
Batch size	16
Epochs	3
SFT examples	7,500 (AlfWorld) / 2,400 (WebShop)
RL Training	
Learning rate	1×10^{-6}
Batch size	64
KL loss Coef	0.01
Invalid Action Penalty Coef	0.1
Max Prompt Length	6,000
Max Response Length	1,024
Epoch	150
Skill Retrieval	
Top-K retrieval	6
Validation interval	5 Steps
Update Threshold δ	0.4
Max failures analyzed	10 (SR $<$ 0.4) / 5 (SR $>$ 0.4)
Max new skills per evolution	3

Table 4: Example distilled skills from SKILLBANK for ALFWorld (Shridhar et al.). This table summarizes general patterns and application logic derived from raw trajectories.

ID	**Skill Title**	**Principle (Actionable Pattern)**	**When to Apply**
General Exploration & Acquisition Skills			
gen_001	Systematic Exploration	Search every plausible surface or container exactly once before revisiting; prioritize unseen locations.	Anytime the goal count is not met and unexplored areas remain.
gen_002	Immediate Acquisition	As soon as a required object becomes visible and reachable, take it immediately.	Upon first visual confirmation of a goal-relevant object.
gen_003	Destination First Policy	After picking up a goal object, navigate directly to the known target receptacle and place it.	Holding any goal object while target location is identified.
State-Changing & Spatial Relation Skills			
gen_005	Use State-Changing Tools Early	Acquire the object, then immediately use the nearest suitable appliance (heat/cool/clean) before placement.	After picking up an object requiring temperature or cleanliness change.
gen_006	Establish Spatial Relations	First locate the reference object, adjust its state if needed, then search or place in the specified region.	Tasks containing prepositions like “under”, “inside”, or “on”.
Reliability & Error Recovery			
gen_014	Loop Escape Trigger	If the last 3–5 actions do not change the state, switch to an untried search branch or action type.	After several consecutive no-progress observations.
gen_015	Pre-Action Sanity Check	Confirm prerequisites (hand free, capacity, power) before executing manipulative commands.	Right before issuing any command that could legally fail.

reasoning loops in ALFWorld to price-shift oversights in WebShop, alongside their root causes and proposed mitigation strategies.

Table 5: Common Agent Failures and Mitigation Strategies for ALFWorld.

ID	Failure Description	Root Cause (Why it happens)	Mitigation (How to avoid)
err_001	Redundant Revisit	Lacks explicit memory of explored areas; strategy degenerates into local loops.	Maintain an exploration map; prioritize unvisited candidates.
err_006	Skipping State Changes	Conflates object presence with goal satisfaction; omits cleanliness/temp checks.	Integrate state precondition checks into the planner before placement.

Table 6: Example distilled skills for WebShop Navigation (Yao et al., 2022a). These skills represent the strategic patterns used by the agent to handle large-scale product search and constraint satisfaction.

ID	Skill Title	Principle (Actionable Pattern)	When to Apply
Search & Query Engineering			
gen_001	Prioritize Core Keywords	Include product type, 1-2 functional attributes, and hard constraints; omit secondary descriptors.	Before issuing the first search or refining over-specific queries.
gen_002	Iterative Refinement	Adjust keywords or apply site filters instead of repeating the same failed query.	When results are irrelevant or repeat despite multiple searches.
Product Evaluation & Verification			
gen_003	Scan Before You Click	Read titles, thumbnails, and prices in results to ensure plausibility before opening a link.	On search results pages when choosing the next product to inspect.
gen_004	Verify Early, Abort Fast	Immediately check category, attributes, and price on the product page; leave if any constraint is violated.	Within the first observation on every product detail page.
gen_006	Confirm Hidden Attributes	Open Description/Features sections to ensure non-visible specs (e.g., material) meet constraints.	When constraints are not evident from the title or variant list.
Configuration & Transaction			
gen_005	Set Mandatory Variants	Always select required options (size, color, etc.) before evaluating price or purchasing.	After confirming product match but before any purchase action.
gen_007	Check Variant Pricing	For price ranges, select the exact variant combination to verify the specific price is within budget.	Whenever price changes with variant selection or shows as a range.
gen_013	Purchase Decisively	Execute 'Buy Now' immediately once all constraints and prices are confirmed on a variant.	After validating every constraint on the current product variant.

Table 7: Common Failures in Web-based Shopping Tasks.

ID	Failure Description	Root Cause	Mitigation Strategy
err_001	Missing Constraints in Query	Omits size or price caps, leading to overwhelming or irrelevant result sets.	Assemble full requirement list first; ensure every hard constraint is in the query string.
err_004	Price Shift Oversight	Fails to notice price changes after selecting a specific size or color variant.	Re-read the price element after every option change before proceeding to checkout.
err_005	Premature Purchase	Clicks "Buy Now" without setting mandatory variants, leading to errors or wrong items.	Validate that every required drop-down/radio option is explicitly selected before buying.
err_009	Ignoring Stock Status	Attempts to purchase out-of-stock items by ignoring disabled buttons or stock labels.	Verify that the 'Add to Cart' button is enabled and no 'Out of Stock' message is present post-selection.
err_011	Sponsored Link Distraction	Clicks loosely matched ads, diverting the workflow from organic, suitable products.	Implement ad-label detection; prioritize organic listings for higher constraint reliability.

E Additional Cases

Case 1: WebShop Footwear (Functional Constraints)

Task: I want a pair of men's black slip-resistant work shoes, size 10, rubber sole, price lower than $50.00.

Skills	**[foo_002]** Verify features in description; **[foo_004]** Confirm price after variant selection.
Avoid	**[err_001]** Omitting price cap; **[err_003]** Selecting wrong product category.

Agent Trajectory with Skill-Guided Reasoning

Step 1	**Obs:** WebShop homepage. **Think:** Per [gen_001], I'll include key constraints (type, function, color, price) in the query. **Action:** `search[men black slip resistant work shoes rubber sole under 50]`
Step 2	**Obs:** Results: [1] Chef Shoes $38.99; [2] Kitchen Shoes $42.50; [3] Steel Toe Boots $55.99. **Think:** [gen_003] Item [1] is under budget and matches function. Item [3] is over budget (skip). **Action:** `click[Item 1]`
Step 3	**Obs:** Product page. Price: $35.99-$45.99. Colors: Black, White. Sizes: 7-12. **Think:** [foo_002] Description confirms rubber sole. Now selecting Black and Size 10 [foo_003]. **Action:** `click[Black]` $\rightarrow$ `click[10]`
Step 5	**Obs:** Selected: Black, Size 10. Price updated to $38.99. **Think:** [foo_004] $38.99 ¡ $50. All constraints (color, size, function, price) are satisfied. **Action:** `click[Buy Now]`

Result: SUCCESS — Purchased at $38.99. **Key Skills:** [foo_002] (Verification), [foo_004] (Price Check).

Case 2: Electronics (Speaker with Abort-and-Retry)

Task: I need a portable bluetooth speaker, waterproof, black color, price lower than $30.00.

Skills	**[ele_001]** Constraint-rich search; **[ele_004]** Bail on mismatch fast; **[gen_002]** Iterative refinement.
Avoid	**[err_002]** Overly specific variant names in search; **[ele_006]** Avoid click loops.

Agent Trajectory: Handling Missing Constraints

Step 1	**Obs:** WebShop homepage. **Think:** [ele_001] Packing all mandatory attributes: portable, bluetooth, waterproof, black, ¡$30. **Action:** `search[portable bluetooth speaker waterproof black under 30]`
Step 2	**Obs:** Results: [1] Waterproof Speaker $28.99; [2] Mini LED Speaker $19.99; [3] 20W Speaker $35.50. **Think:** [ele_005] Item [1] matches most keywords and price. **Action:** `click[Item 1]`
Step 3	**Obs:** Product page (Item 1). Colors: Blue, Red, Green, Army Green. **Black unavailable.** **Think: [gen_004] Verify Early, Abort Fast:** No black option! [ele_004] Bail immediately to find another candidate. **Action:** `click[Back to Search]`
Step 4	**Obs:** Back on search results. **Think:** [gen_002] Iterative refinement. Item [2] $19.99 is the next best candidate. **Action:** `click[Item 2]`
Step 5	**Obs:** Product page (Item 2). Colors: Black, Blue, Red. Specs: IPX5 Waterproof. **Think:** [ele_002] Black is available! IPX5 confirms waterproof. Selecting Black. **Action:** `click[Black]`
Step 6	**Obs:** Black selected. Final price: $19.99. **Think:** [ele_003] $19.99 ¡ $30. All specs (black, portable, waterproof) verified. **Action:** `click[Buy Now]`

Result: SUCCESS — Efficiently handled attribute mismatch via early exit. **Key Skills Applied:** [gen_004] (Abort Fast), [ele_004] (Bail on Mismatch).

Case 1: ALFWorld Clean Task (Lettuce $\rightarrow$ Sink $\rightarrow$ Countertop)

Task: Clean some lettuce and put it in countertop.

Core Skills	**[cle_001]** Phase-ordered plan (Locate $\rightarrow$ Clean $\rightarrow$ Place); **[cle_006]** Use location priors (Fridge for food).
Mistakes Avoided	**[err_002]** Skipping closed containers; **[err_003]** Ignoring visible target objects.

Step 1-3	**Obs:** Fridge 1 is closed. **Action:** `open fridge 1` $\rightarrow$ `take lettuce 1` **Reasoning:** Per **[cle_006]**, food is likely in the fridge. **[gen_002]** Grab lettuce immediately on sight.
Step 4-6	**Obs:** Arrive at sinkbasin 1. **Action:** `clean lettuce 1 with sinkbasin 1` **Reasoning:** Per **[cle_001]** Phase 2: bring to water source. **[cle_003]** Go straight to sink.
Step 7	**Obs:** Arrive at countertop 1. **Action:** `put lettuce 1 in/on countertop 1` **Reasoning:** Per **[gen_003]** Destination First: place cleaned object immediately to finish.

Result: SUCCESS (7 Steps) — Skills used: [gen_010] (Decomposition), [cle_003] (Sink First).

Case 2: ALFWorld Heat Task (Egg → Microwave → Countertop)

Task: Heat some egg and put it in countertop.

Core Skills	**[hea_001]** Secure exact target first; **[hea_003]** Open-Place-Heat sequence; **[hea_004]** No appliance before object.

Step 1-3	**Obs:** Countertop 1 (no egg) → Countertop 2 (egg found). **Action:** `take egg 1` **Reasoning: [hea_004]** Avoid microwave until object is held. **[hea_002]** Systematic search of surfaces.
Step 4-6	**Obs:** Microwave 1 is closed. **Action:** `open microwave 1` → `heat egg 1` **Reasoning: [hea_003]** Correct sequence: open door first, then initiate state change.
Step 7-8	**Obs:** Egg is heated. **Action:** `go to countertop 1` → `put egg 1` **Reasoning: [hea_005]** Direct Post-Heat Placement: move to target without detours.

Result: SUCCESS (8 Steps) — Skills used: [hea_001] (Target First), [hea_004] (Sequence Control).

Case 3: ALFWorld Examine Task (Pencil ⊕ Desklamp → Success)

Task: Look at pencil under the desklamp.

Core Skills	**[loo_001]** Seek lamp surfaces (Desk/Sidetable); **[loo_002]** Switch lamp on; **[exa_001]** Pair objects early (Target & Tool co-location).
Mistakes Avoided	**[err_003]** Ignoring reachable targets; **[loo_005]** Preventing action loops (Toggle only once).

Agent Trajectory: High-Efficiency Co-location Handling

Step 1	**Obs:** Mid-room. Visible: desk 1/2, drawer 1-3, sidetable 1, etc. **Reasoning: [loo_001]** Desklamps usually sit on desks. Prioritizing `desk 1` to find both pencil and lamp in one go. **Action:** `go to desk 1`
Step 2	**Obs:** On `desk 1`: **alarmclock 1, desklamp 1, pencil 1**. **Reasoning: [exa_001]** Successful co-location! Both objects are present. **[exa_002]** Pick up target first before interacting with the tool. **Action:** `take pencil 1 from desk 1`
Step 3	**Obs:** Holding pencil 1. **Reasoning: [loo_002]** Toggle the lamp state to satisfy the ”under light” condition. **[exa_004]** Single toggle rule ensures no repeated actions. **Action:** `use desklamp 1`

Result: SUCCESS (Optimal 3 Steps) — Demonstrates superior spatial reasoning and zero-waste navigation. **Key Skills:** [exa_001] (Object Pairing), [loo_001] (Spatial Priors).