EXPVOYAGER: DIRECT EXPERIENCE NAVIGATION FOR DYNAMIC AGENT SKILL SYNTHESIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning from experience in LLM agents has become a key paradigm for developing self-evolving agents that continuously learn and expand their capabilities. Within this paradigm, synthesizing the *agent skill* has emerged as a promising solution for transforming accumulated experience into reusable *procedural knowledge*, serving as an important layer for the harness system that supplies agents at runtime. Despite its potential, existing approaches largely abstract past experience into fixed procedural knowledge before downstream demands are known, which risks discarding knowledge that later becomes critical while retaining instance-specific details irrelevant to future tasks. In this paper, we reframe agent skill synthesis as a dynamic navigation problem over past experience, where agents actively explore accumulated trajectories on demand for the current task with targeted and fine-grained access to experience knowledge. To this end, we propose **EXPVOYAGER**, a novel framework in which a skill curator navigates raw experience across different views and resolutions, continually identifying reusable procedural knowledge from what it observes while tracking remaining knowledge needs that guide where to navigate next. Extensive experiments demonstrate both the effectiveness and versatility of EXPVOYAGER, showing consistent improvements in downstream task performance, continual gains as the experience space scales, and practical compatibility with existing skills under efficient experience access. **[CODE]**

1 INTRODUCTION

LLMs are rapidly evolving from systems that merely generate responses into long-horizon agents that interact with environments, use tools, and execute complex multi-step tasks (Yao et al., 2023; ichter et al., 2023). As the tasks and environments they operate in become increasingly diverse and complex, the general capabilities of foundation models alone may not be sufficient to reliably provide the knowledge required across different execution contexts, making the *agent harness* (OpenAI, 2026; Lee et al., 2026; Zhang et al., 2026a) an increasingly important layer for supplying agents at runtime.

In response to these needs, the *agent skill* (Anthropic, 2025), typically instantiated as manually authored instructions, procedures, and heuristics that guide agent behavior, has emerged as a promising solution for extending agent capabilities without modifying model weights. Recently, a line of research (Wang et al., 2024; 2025c) has taken a step further by enabling agents to synthesize skills from their own task experience, showing the potential of self-evolving agents (Zhao et al., 2024; Zheng et al., 2025; Ouyang et al., 2026b) that continuously learn and expand their capabilities.

One of the key challenges in this direction is to transform past execution experience from task-specific records into *procedural knowledge* (Wu et al., 2026; Mi et al., 2026) that generalizes to future decision-making. Existing approaches largely address this challenge by abstracting reusable lessons from past experience into fixed procedural knowledge, either summarizing individual trajectory (Ouyang et al., 2026b; Fang et al., 2026) or consolidating patterns across multiple trajectories (Wang et al., 2025d; Ni et al., 2026), and storing the resulting skills for future use. However, as the demands of future tasks are inherently unknown at the time of skill construction, this approach risks discarding knowledge that later becomes critical, while retaining excessive instance-specific details that are irrelevant to their downstream application. This gap leads us to ask a central question: ***how can we enable agents to dynamically synthesize skills on demand for the current task from raw experience?***

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

In this paper, we answer this question by reframing skill synthesis from the fixed abstraction of past experience into a **dynamic navigation problem**, where agents search for procedural knowledge on demand within accumulated trajectories. One straightforward instantiation of this task-time formulation is to retrieve the top-k trajectories most relevant to the current task and synthesize a task-specific skill from them. However, similarity-based access to past experience can struggle to capture procedural relevance to the current task, as irrelevant knowledge is often contained in similar trajectories, while localized critical cues may also appear in superficially dissimilar trajectories. Therefore, instead of relying on such a passive retrieval interface, we define experience navigation as an active decision-making process carried out by the agent itself, in which the agent determines where to inspect and at what resolution to examine past experience targeted to the current needs.

To this end, we introduce **EXPVOYAGER**, a novel framework for dynamic agent skill synthesis in which a skill curator navigates past experience to construct task-specific guidance for a frozen executor. Motivated by the intuition that useful procedural knowledge needed for a target task often resides in different aspects of a trajectory (such as overall execution patterns and local decision contexts), we first equip the curator with a ***Navigable Interface*** that supports access to multiple views of raw experience at different resolutions. Through this interface, the curator actively decides how to access past experience by selecting navigation actions that specify the appropriate level of view in its action arguments for the current investigation, and expand localized records into broader trajectory context when needed. Next, to support the navigation process that evolves with what the curator has observed in previously accessed experience, we also design a ***Navigation State*** for the curator. This state is updated each round by interpreting newly accessed observations in order to decide what procedural knowledge can be reused for the current target and what remains to be investigated, thereby linking experience access and interpretations to continually guide one another.

We conduct extensive experiments on three popular agent benchmarks including household interaction, online shopping, and scientific reasoning. Our evaluation mainly focuses on (1) the *effectiveness* of EXPVOYAGER in improving agent task performance and (2) its *versatility* in supporting scalable and efficient experience reuse. Across all benchmarks, EXPVOYAGER consistently outperforms existing skill-based approaches in task performance and improves base agents across diverse curator–executor configurations. EXPVOYAGER also drives online self-evolution without relying on pre-collected experience, and additional experimens shows that it can scale to larger experience space, progressively widening its performance advantage over existing approaches. We further demonstrate its versatility in practical agent-harness scenarios, showing that EXPVOYAGER can synergize with existing skills through on-demand refinement while also making effective use of experience-access budgets.

We summarize our contributions as follows:

- We reframe agent skill synthesis as a dynamic navigation problem over past experience, enabling targeted and fine-grained access to task-relevant procedural knowledge on demand.
- We propose EXPVOYAGER, a novel framework that supports effective experience navigation with a *Navigable Interface* for accessing raw experience through different views and resolutions, and *Navigation State* that continually connects experience access with target-relevant knowledge.
- We demonstrate both the effectiveness and versatility of EXPVOYAGER, showing consistent improvements in downstream task performance, continual gains as the experience space scales, and practical compatibility with existing skills under efficient experience access.

2 PRELIMINARY ANALYSIS

We conduct two preliminary analyses to better understand the limitations of existing skill construction and retrieval paradigms, asking: (1) how much knowledge useful for future tasks is preserved when skills are constructed in advance from past experience, and (2) how effectively existing retrieval paradigms can surface the knowledge needed for a given target task from past experience.

To empirically examine these questions, we first construct *oracle skills* whose helpfulness for the target task is verified through task execution and annotate the *oracle knowledge* in each source trajectory that can contribute to constructing each skill. Specifically, we first collect multiple successful and failed trajectories from distinct attempts of a frozen executor on each target task and use them to synthesize a candidate skill. We then provide the candidate skill back to the same frozen executor and

retain it only when the executor successfully completes the corresponding task with the skill. After this iterative generate-then-verify process, we collect 80 oracle skills with verified helpfulness on ALFWorld (Shridhar et al., 2021) test tasks. Next, to identify the oracle knowledge available in past experience for constructing each oracle skill, we sample 300 source trajectories from the ALFWorld training set and examine whether each source contains relevant knowledge. Helpful sources are annotated with the relevant oracle knowledge items, while the remaining sources are labeled *none*. Please refer to Appendix B.4 for more detailed experimental setups.

2.1 Preliminary Analysis I: Pre-Constructed Skills Lose the Majority of Procedural Knowledge Useful for Future Tasks.

Setups. We first examine how much knowledge that later becomes useful for a future task is preserved when past experience is abstracted into a skill before its downstream demand is known. Specifically, we compare the knowledge retained in pre-constructed skills of existing methods (Wang et al., 2025d; Ouyang et al., 2026b; Ni et al., 2026) against the oracle knowledge from the corresponding source trajectory. For quantification, we decompose each skill into atomic knowledge claims for semantic matching, following prior claim-level evaluation protocols (Ru et al., 2024). We then measure how much of the oracle knowledge in each source is covered by these claims.

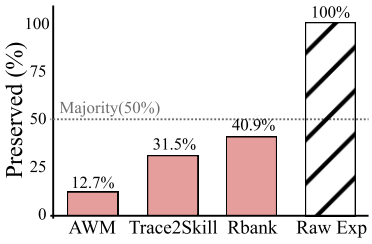


Figure 1: Experiments on how much knowledge useful for future tasks is preserved in skills. Here, we use raw experience as the 100% reference for preservation.

Results. Figure 1 shows that even the best-performing method fails to preserve the majority of oracle knowledge contained in raw experience, suggesting that much of the knowledge that later becomes critical is already lost during pre-construction.

2.2 Preliminary Analysis II: Existing Experience Retrieval Paradigms Provide Limited Access to Target-Relevant Knowledge.

Setups. We next investigate whether existing retrieval paradigms can provide effective access to target-relevant knowledge in past experience. For the analysis, we follow the retrieval setup of each method, where the retrieved sources may consist of pre-constructed skills (Wang et al., 2025d; Ouyang et al., 2026b), or raw trajectories (Wang et al., 2026). At the source level, we measure the recall of helpful sources across different retrieval depths k. Since retrieving a helpful source does not necessarily indicate how much target-relevant knowledge it actually provides, we further conduct a knowledge-level assessment. Specifically, we compare the unique knowledge contained in the retrieved sources against the full set of unique oracle knowledge for the target task, measuring recall as coverage of required knowledge and precision as the fraction of retrieved knowledge relevant to the target task.

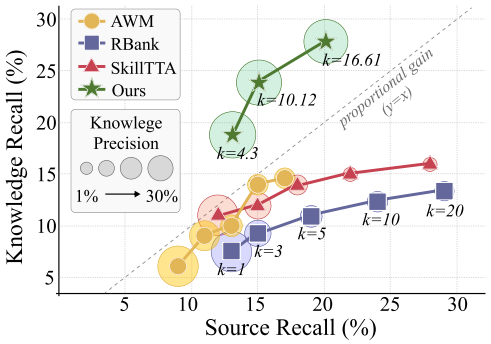


Figure 2: Experiments on how effectively existing experience retrieval methods access target-relevant knowledge across different retrieval depths k. For Ours, k is the average number of unique sources accessed during navigation.

Results. Figure 2 shows that existing methods achieve low knowledge recall, leaving a substantial portion of target-relevant knowledge inaccessible through retrieval. Meanwhile, expanding retrieval with larger k does not fully resolve this issue, as marginal gains in knowledge recall are increasingly outweighed by redundant knowledge and irrelevant information, causing precision to drop rapidly. This indicates that existing retrieval paradigms provide only limited access to the knowledge needed for the target task, motivating a more targeted form of knowledge access in which agents actively seek what is needed as their understanding of the task's procedural needs evolves during retrieval.

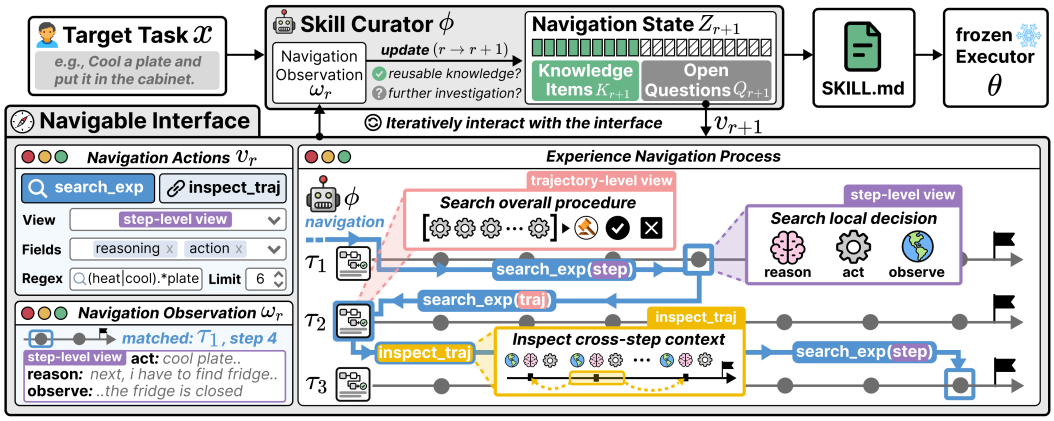


Figure 3: Conceptual overview of EXPVOYAGER.

3 EXPVOYAGER FOR DYNAMIC SKILL SYNTHESIS

Motivated by the insights in Section 2, we propose **EXPVOYAGER**, a framework for dynamic agent skill synthesis via direct experience navigation. We present an overview of EXPVOYAGER in Figure 3.

3.1 PROBLEM FORMULATION

Following prior work on learning from experience in LLM agents (Wang et al., 2025d; Xia et al., 2026; Ouyang et al., 2026a), we study how an agent can reuse its own experience from previous task attempts to solve a new target task. The available agent experience is represented as a collection of trajectories $E = \{\tau_i\}_{i=1}^{N}$, where each trajectory τ_i contains a task instruction, a sequence of observations and actions, and its execution outcome. Given the agent experience E and a target task instance x, the downstream task is to generate actions that successfully complete x through interaction with the environment. While the broader objective is successful task execution, our focus is on building a curator ϕ to generate a target-conditioned skill $S_{E,x}$ from E that provides task-time guidance for completing x and using it as additional context for a frozen executor θ:

$$S_{E,x} = \phi(E, x), \qquad a_t \sim \theta(\cdot \mid x, h_t, S_{E,x})\,. \tag{1}$$

where $h_t = (a_0, o_1, \ldots, a_{t-1}, o_t)$ denotes the executor's interaction history for solving x at step t.

3.2 BUILDING NAVIGABLE INTERFACE OVER AGENT EXPERIENCE

Our goal is to enable more targeted use of past experience by giving the agent active control over how it accesses the relevant knowledge for the current target. To achieve this, we equip the skill curator with a ***Navigable Interface*** over E that exposes past execution through multiple views at different resolutions. This is motivated by the intuition that procedural knowledge needed for a target task often resides in different aspects of a trajectory. For example, an overall action sequence can reveal the high-level procedure followed to complete a task, whereas understanding a failed operation may require examining the recorded reasoning to identify the condition behind the chosen action. Rather than presenting each trajectory as a single retrieval unit, our interface lets the curator inspect evidence at the resolution required by the current investigation and expand to broader context when needed.

Experience Views. To support such access, we organize each trajectory into complementary knowledge views that preserve the relationship between context, decision, and consequence, without modifying the knowledge contained in the underlying raw experience.

- **Trajectory-level views:** Capture the overall execution through the ordered action sequence, execution outcome, and task-specific metadata provided by the environment. These views provide a compact picture of the procedure followed across the episode, helping the curator identify broader strategies, success/failure patterns, or execution stages worth further investigation.
- **Step-level views:** Capture individual decisions through the observation, recorded reasoning, executed action, and immediate result, together with surrounding execution context. These views

expose the local conditions and immediate consequences of a decision, helping the curator identify conditions associated with different action outcomes.

Navigation Actions. We formulate experience navigation over these views as an iterative decision-making process in which the curator selects both an operation and its arguments at each round. Specifically, the curator can invoke **`search_exp`** to locate new experience records across different trajectories by specifying the level of view, fields, a regular-expression pattern, and a result limit as arguments. The selected fields and regex pattern determine where matching occurs, while each match is returned as a complete step- or trajectory-level record to preserve the execution context needed for interpretation. At the step level, this includes the observation, reasoning, action, and result; at the trajectory level, the action sequence, execution outcome, and task metadata. Each returned record also retains a reference to its source trajectory. When broader procedural context is needed, the curator can invoke **`inspect_traj`** on this reference to return the full source trajectory chronologically. At round r, the curator selects an operation and its arguments as a navigation action ν_r through the navigable interface $\mathcal{I}$ over E, receiving the corresponding navigation observation $\omega_r = \mathcal{I}(E, \nu_r)$. Based on ω_r, the curator determines its next operation and arguments, repeatedly moving between localized records and full trajectory context as needed.

3.3 Guiding Experience Navigation via State Management

While the navigable interface gives the agent active control over how it accesses past experience, effective navigation requires more than just observing relevant experience. In particular, the returned experience often capture what happened under source-specific conditions, requiring the curator to interpret what those observations imply for the current target before deciding how to proceed with further navigation. Moreover, knowledge derived from newly accessed records can refine or revise earlier interpretations and thereby change what remains to be investigated. To address this challenge, we incorporate a ***Navigation State*** Z into the curator's navigation process, which is updated after each navigation round to reflect what procedural knowledge has been established for the target and what remains to be investigated, so that it can guide the subsequent navigation action:

$$(Z_{r+1}, \nu_{r+1}) = \phi_{\text{nav}}(Z_r, \omega_r; x), \qquad \text{where } Z_r = (K_r, Q_r). \tag{2}$$

Here, K_r represents the knowledge items curated for the current target, while Q_r represents the open questions to be investigated through further navigation.

Updating Knowledge Curated from Prior Navigation. Rather than simply storing the navigation observations, K_r accumulates target-relevant knowledge across navigation rounds. Given the target x, the current state Z_r, and newly accessed navigation observation ω_r, the curator generates updated knowledge items K_{r+1} by distinguishing what observations remain specific to the source experience, what procedural relations can transfer to the current target, and what conditions still require verification during execution. For example, observing that an object was found at a particular location in a past trajectory may support a procedural knowledge for locating and moving the object, but does not establish that the object occupies the same location in the current environment. The resulting knowledge is organized by its role in execution, while newly accessed ω_r can refine the conditions attached to existing knowledge or revise earlier judgments about what transfers to the current target.

Open Questions for Guiding Further Navigation. Q_r turns gaps in the current knowledge into concrete objectives for further navigation, focusing on uncertainties whose resolution can make the current knowledge more complete. It is initialized with procedural questions derived from x and evolves as navigation proceeds: questions supported by newly accessed records can be resolved or revised, while newly identified gaps can be added for subsequent investigation. The curator selects a useful open question and chooses the navigation action and arguments that can help resolve it.

3.4 Synthesizing Skill for Frozen Executor

After navigation, the curator uses the resulting Z to synthesize a `skill.md` for the frozen executor θ. Starting from Z_0, initialized from x, the curator iterates the navigation process over R rounds until further investigation is no longer useful or the navigation budget is exhausted. The curator then generates the final skill as $S_{E,x} = \phi_{\text{synth}}(Z_R, x)$. The resulting skill turns the accumulated procedural knowledge into tailored execution guidance, retaining unresolved target conditions as execution-time checks and failure lessons as cautions or recovery guidance at relevant stages.

Table 1: Main results on ALFWorld, WebShop, and ScienceWorld, using Qwen3.5-9B as curator ϕ.

Methods	ALFWorld		WebShop			ScienceWorld		
	SR (↑)	Steps (↓)	SR (↑)	Score (↑)	Steps (↓)	SR (↑)	Score (↑)	Steps (↓)
Executor θ: Qwen3.5-9B								
Base (ReAct) (Yao et al., 2023)	$41.4_{1.3}$	34.1	$19.6_{0.5}$	$50.3_{0.4}$	7.1	$27.0_{1.3}$	$37.2_{1.2}$	11.5
+ AWM (Wang et al., 2025d)	$45.6_{1.8}$	35.1	$24.2_{0.4}$	$52.0_{1.3}$	7.9	$32.6_{1.7}$	$42.7_{1.0}$	8.4
+ RBank (Ouyang et al., 2026b)	$43.5_{2.2}$	33.9	$20.3_{1.4}$	$41.6_{1.7}$	9.9	$29.6_{0.6}$	$34.1_{0.6}$	**7.9**
+ Trace2Skill (Ni et al., 2026)	$45.4_{1.7}$	34.3	$22.6_{0.7}$	$45.7_{1.0}$	9.1	$31.9_{1.3}$	$37.6_{0.7}$	9.8
+ SkillTTA (Wang et al., 2026)	$51.1_{2.8}$	30.4	$19.2_{1.3}$	$34.3_{1.8}$	10.9	$29.6_{0.6}$	$40.7_{0.7}$	11.3
+ **ExpVoyager** (Ours)	$\mathbf{63.7}_{2.2}$	**25.9**	$\mathbf{28.7}_{0.6}$	$\mathbf{61.2}_{1.7}$	**6.9**	$\mathbf{37.4}_{1.3}$	$\mathbf{47.0}_{0.9}$	11.1
Executor θ: Gemma4-31B								
Base (ReAct) (Yao et al., 2023)	$53.9_{0.6}$	29.2	$24.7_{0.7}$	$48.4_{0.9}$	7.7	$48.2_{3.4}$	$57.4_{0.5}$	14.9
+ AWM (Wang et al., 2025d)	$57.0_{0.9}$	27.5	$30.6_{0.6}$	$51.8_{0.4}$	7.9	$52.2_{1.1}$	$59.1_{0.3}$	12.7
+ RBank (Ouyang et al., 2026b)	$57.9_{1.5}$	28.8	$29.8_{0.7}$	$51.1_{1.1}$	7.1	$50.0_{1.1}$	$57.7_{1.3}$	12.2
+ Trace2Skill (Ni et al., 2026)	$55.2_{2.4}$	25.4	$29.4_{0.3}$	$50.8_{1.0}$	6.6	$51.9_{0.6}$	$58.4_{0.4}$	11.8
+ SkillTTA (Wang et al., 2026)	$62.5_{1.6}$	24.7	$31.3_{1.2}$	$57.1_{1.7}$	6.8	$49.6_{1.7}$	$56.4_{0.7}$	13.3
+ **ExpVoyager** (Ours)	$\mathbf{69.6}_{0.1}$	**22.0**	$\mathbf{37.9}_{0.8}$	$\mathbf{66.7}_{1.0}$	**6.1**	$\mathbf{55.6}_{1.1}$	$\mathbf{63.7}_{1.2}$	**11.3**

4 Experiment

4.1 Experimental Setup

We conduct experiments on three popular agent benchmarks across diverse domains, including ALFWorld (Shridhar et al., 2021), WebShop (Yao et al., 2022), and ScienceWorld (Wang et al., 2022), covering household interaction, online shopping, and scientific reasoning in interactive environments. We evaluate effectiveness (Success Rate) and efficiency (Steps), with specific metrics varying for each dataset. For comparison, we consider three categories of baselines: **(1) base ReAct** (Yao et al., 2023) **agent without skills**; **(2) pre-constructed skill** baselines, including AWM (Wang et al., 2025d), ReasoningBank (RBank) (Ouyang et al., 2026b), and Trace2Skill (Ni et al., 2026); and **(3) test-time skill synthesis** baseline SkillTTA (Wang et al., 2026). For fair evaluation, all baselines and ExpVoyager use the same backbone models within each configuration for both the downstream task execution and skill construction. Please refer to Appendix for full descriptions of the datasets B.1, baselines and evaluation protocols B.2, implementation details B.3.1, and prompts B.3.2.

4.2 Results: Effectiveness and Versatility of ExpVoyager

ExpVoyager helps agents reuse past experience through dynamic skills. We first evaluate ExpVoyager in an offline setting, where a fixed set of source trajectories is available as past experience, and compare it against the base agent and existing baselines across ALFWorld, WebShop, and ScienceWorld. As shown in Table 1 , ExpVoyager consistently improves the performance of the base agent and outperforms existing approaches across all three benchmarks in terms of task success and benchmark-specific scores. ExpVoyager also reduces the number of execution steps, indicating that the synthesized skills not only improve task completion but also guide the agent toward more efficient execution. These results suggest that dynamically navigating the experience space and synthesizing actionable skills for the current task provides effective guidance for downstream agent execution, allowing past execution experience to be reused in a task-relevant manner.

ExpVoyager is transferable to diverse executor and curator backbone. To validate the transferability of our framework, we evaluate ExpVoyager across diverse curator and executor backbones with different model families and scales. From Table 1 and 2, ExpVoyager consistently improves task performance across different curator–executor combinations, demonstrating its robustness across diverse model configurations. In particular, a smaller curator can improve the

Table 2: Performance comparison of diverse curator backbones with ExpVygr using a Gemma4-31B executor θ.

Methods	Curator ϕ	ALFWorld		WebShop		SciWorld	
		SR	Steps	SR	Steps	SR	Steps
Base θ	–	53.9	29.2	24.7	7.7	48.2	14.9
+ ExpVygr	Qwen3.5-9B	**69.6**	**22.0**	**37.9**	**6.1**	**55.6**	**11.3**
+ ExpVygr	Gemma4-31B	**71.3**	**21.8**	**38.4**	**5.7**	**60.8**	**11.2**
+ ExpVygr	GPT-5.4-mini	**75.9**	**19.7**	**38.1**	**5.5**	**62.1**	**13.1**

performance of a much larger executor, highlighting that guidance on what knowledge to reuse from past experience can be just as important as the agent's intrinsic task-solving capability.

ExpVoyager drives online self-evolution of agents from accumulating experience. While our main experiments assume that a fixed set of past experience is available offline, practical agents may need to continually learn from experience as their experience space is built incrementally during deployment. To evaluate ExpVoyager in this online setting, we start without a pre-collected experience corpus and sequentially execute the test task stream, and allowing each method to reuse only the experience collected from earlier tasks in the current online stream. For fair evaluation, we repeat the experiment over three random task orderings and report the average performance.

As shown in Figure 4, ExpVoyager provides limited gains and can even degrade performance when the available experience is limited. However, as the accumulated experience becomes broader, ExpVoyager continues to improve while baseline methods plateau, overcoming its early disadvantage and progressively widening the performance gap. These results suggest that continual improvement of agent can emerge not only from distilling new experience into a growing store of reusable knowledge for future tasks, but also from accumulating raw experience and navigating it for task-relevant knowledge on demand.

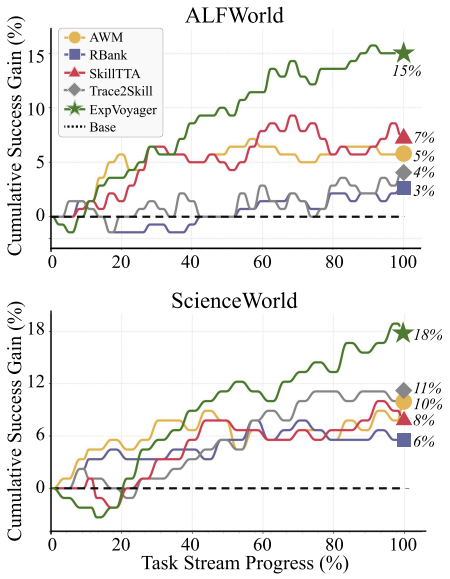


Figure 4: Online evaluation on ALFWorld and ScienceWorld. The x-axis shows task stream progress, and the y-axis reports cumulative success gain over the base agent.

ExpVoyager scales to larger experience space. To further investigate the scaling behavior of ExpVoyager observed in the online setting, we examine how each method scales with increasing amounts of source experience under a controlled offline setup. Specifically, we construct nested subsets of the source trajectories by progressively increasing the number of available trajectories and provide the same subset to all methods at each scale. As shown in Figure 5, baseline performance declines as more source experience becomes available, whereas ExpVoyager consistently improves as the source experience grows, with its advantage becoming more pronounced at larger scales. These results show that simply having more experience available can produce noisier skills for the target task in existing baselines, while ExpVoyager turns a growing experience space into additional gains by effectively identifying task-relevant knowledge from the expanded source experience.

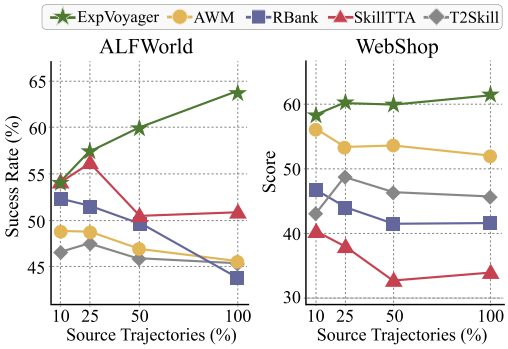


Figure 5: Task performance with increasing source experience pool. All methods use the same nested subsets of 1000 souce trajectories (100%).

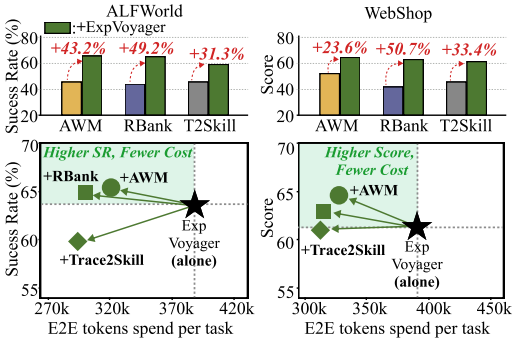


Figure 6: Synergy between ExpVoyager and pre-constructed skills. E2E tokens cover the full task-time pipeline; navigation, and execution.

ExpVoyager can synergize with pre-constructed skills. While the previous experiments evaluate ExpVoyager as a standalone approach for dynamic skill synthesis, practical agents may already maintain pre-constructed skills or memories that can provide useful guidance for future tasks. Such

preconstructed knowledge enables efficient reuse across future tasks, but, as discussed in Section 2, abstracting experience into reusable skills in advance risks discarding information that may later become critical for a particular task. Conversely, synthesizing the on demand skill from raw experience for every task avoids committing to this abstraction in advance, but may repeatedly search for and reconstruct reusable knowledge already captured in pre-constructed skills.

We therefore examine whether EXPVOYAGER can combine the strengths of both by reusing existing skills as a starting point and dynamically complementing them with on demand knowledge navigation. Specifically, we compare each *baseline alone*, its *combination with* EXPVOYAGER, and EXPVOYAGER *alone*, where the combined setting provides the pre-constructed skill as initial knowledge and lets EXPVOYAGER navigate the original experience to refine the given skill. As shown in Figure 6, combining EXPVOYAGER with pre-constructed skills significantly boosts the performance of corresponding baselines. When compared with standalone EXPVOYAGER, the combined variants consistently reduce task-time cost, while also achieving higher task performance with only a single exception. From the perspective of operating an agent harness system, these results suggest that pre-constructed skills are not required to exhaustively encode all variants of knowledge for future tasks, but can instead serve as reusable foundations that are specialized on demand for each target.

EXPVOYAGER harnesses experience-access budget more effectively than agentic search. One might wonder whether the gains of EXPVOYAGER simply come from spending more experience-access budget on repeated access, rather than from its active navigation over the experience space. To disentangle these effects, we compare EXPVOYAGER against iterative extensions of existing top-k retrieval-based skill approaches. Specifically, following the iterative retrieval approach commonly used in agentic search (Jin et al., 2025; Li et al., 2025), each extension repeatedly performs top-k retrieval and refines its experience search query based on the results retrieved in the previous round. For evaluation under comparable experience-access budgets, we vary the navigation rounds of EXPVOYAGER and continue each extension of baselines until it reaches a similar token usage to EXPVOYAGER.

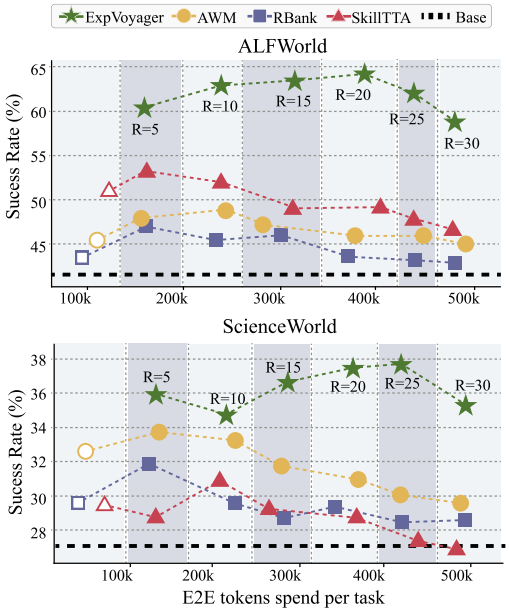


Figure 7: Comparison with agentic search extensions of baselines under matched experience-access budgets. R denotes navigation rounds, and E2E token spend cover the full task-time pipeline.

As shown in Figure 7, EXPVOYAGER consistently outperforms the agentic search extensions under comparable experience-access budgets. Moreover, increasing the budget yields only marginal gains for baselines and can even hurt performance, whereas EXPVOYAGER benefits from additional budget up to 20 rounds before it declines with further navigation. These suggest that effective experience reuse depends not on the amount of access itself, but on targeted access to knowledge that is actually helpful for current task execution.

4.3 ANALYSIS: WHY AND HOW EXPVOYAGER WORKS

Table 3: Ablation study on ALFWorld. We ablate Navigable Interface and Navigation State to assess their individual contributions.

Variants	SR (%)
EXPVYGR (full)	**63.7**
w/o Nav. Interface	55.3
w/o Nav. State	58.9

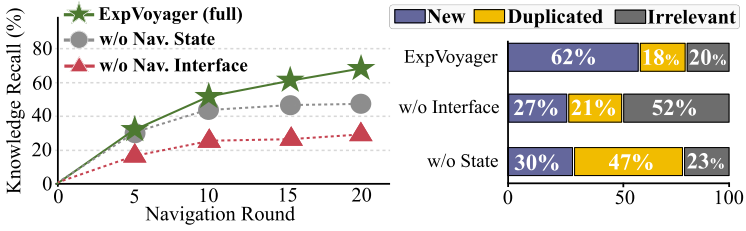


Figure 8: Effects of two key components on knowledge discovery. (Left) Oracle knowledge recall over navigation rounds. (Right) Composition of knowledge type discovered by each variant.

Navigable Interface facilitates knowledge access, while Navigation State sustains further discovery. To better understand why EXPVOYAGER works, we ablate its two main components and analyze the resulting navigation behavior using the oracle knowledge annotations from Section 2. Specifically, we compare the knowledge K_r accumulated by EXPVOYAGER at each navigation round to the oracle knowledge, measuring Knowledge Recall (Figure 8, Left) and classifying each knowledge as one of three types: *new knowledge*, *duplicated knowledge*, or *irrelevant knowledge* (Figure 8, Right). In Table 3, removing either component substantially degrades performance. In Figure 8, removing the Navigable Interface lowers Knowledge Recall and increases irrelevant knowledge, whereas removing Navigation State slows recall growth by repeatedly visiting duplicated knowledge. These results suggest that both components play crucial yet distinct roles in effective experience navigation.

EXPVOYAGER integrates knowledge across diverse sources by exploring experience at different resolutions. To provide an in-depth analysis of how EXPVOYAGER navigates past experience, we analyze its experience-access behavior during navigation. Specifically, we report (a) the proportion of selected navigation actions, (b) the types of knowledge sought by each action, and (c) source diversity during navigation. As shown in Figure 9 (a,b), the curator selects actions according to its information needs, using trajectory-view for procedure overview, step-view for local execution records, and trajectory inspection to understand context across steps. Figure 9 (c) further shows that EXPVOYAGER explores diverse procedural knowledge distributed across multiple source trajectories. These results show that EXPVOYAGER flexibly accesses experience at different resolutions and across diverse sources as its needs evolve during navigation. We provide detailed setups in Appendix B.7, and case study in Appendix C.2.

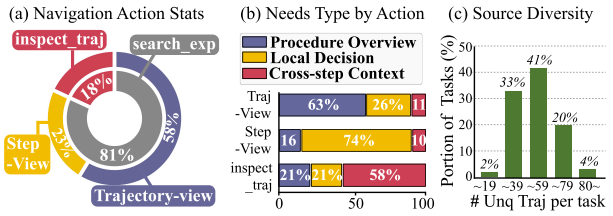


Figure 9: Experience-access behavior of EXPVOYAGER. For (b), knowledge types are classified from the open question Q_r at each navigation round.

5 RELATED WORK

Learning from Experience for Self-Evolving Agents. Recent work studies how agents reuse past experience to improve future decision-making. Most approaches abstract past trajectories into reusable procedural guidance (Wang et al., 2025d; Ouyang et al., 2026b; Ni et al., 2026), while more recent works further learn to curate such guidance from experience (Xia et al., 2026; Ouyang et al., 2026a). Despite these advances, these approaches still construct reusable knowledge before the demands of future tasks are known, which risks discarding knowledge that later becomes critical for a particular task. While an alternative approach retrieves trajectories and synthesizes a skill for the current task (Wang et al., 2026), top-k relevance can struggle to capture procedural relevance, as localized cues may also appear in superficially dissimilar trajectories. In contrast, we reframe skill synthesis as an on demand navigation process that determines what parts of raw experience to inspect and at what resolution, enabling more targeted and fine-grained access to task-relevant knowledge.

Agentic Search and Corpus Interaction. Recently, search paradigms have been shifting from relying on passive retrieval interface toward more active agentic workflows. Most approaches iteratively refine their search queries based on intermediate reasoning and previously retrieved evidence (Jin et al., 2025; Li et al., 2025; Chen et al., 2025), while more recent work extends agent control to corpus exploration itself, enabling finer inspection of evidence in web corpus (Li et al., 2026; Salemi et al., 2026). However, agent experience carries an execution-specific structure where procedural knowledge emerges from relations among context, decision, and consequence. EXPVOYAGER exposes theses relations through an interface tailored for experience navigation, allowing agents to move between complementary views both within and across trajectories to access distinct execution semantics.

6 CONCLUSION

We propose EXPVOYAGER, a novel framework that reframes agent skill synthesis as dynamic navigation over past experience for targeted and fine-grained access to procedural knowledge on demand. Experimental results show that EXPVOYAGER consistently improves downstream task performance, scales with larger experience spaces, and remains compatible with existing skills under efficient experience access. We believe EXPVOYAGER provides a promising foundation for building self-evolving agents that continuously turn accumulated experience into reusable capabilities.

AI USE STATEMENT

We used LLMs to assist with manuscript writing and polishing for clarity and readability, as well as with implementing code for our experiments. The authors reviewed all AI-assisted text and manually verified all AI-generated code before using it in the experiments. The authors take full responsibility for the final content of this work, including all AI-assisted text and code.

REFERENCES

Anthropic. Introducing agent skills. https://claude.com/blog/skills, October 2025.

Mingyang Chen, Linzhuang Sun, Tianpeng Li, sunhaoze, ZhouYijie, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. Research: Learning to reason with search for LLMs via reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=OuGAwwAT8G.

Runnan Fang, Yuan Liang, Xiaobin Wang, Jialong Wu, Shuofei Qiao, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. Memp: Exploring agent procedural memory. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17490–17502, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.866. URL https://aclanthology.org/2026.findings-acl.866/.

brian ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J. Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. Do as i can, not as i say: Grounding language in robotic affordances. In Karen Liu, Dana Kulic, and Jeff Ichnowski (eds.), *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/ichter23a.html.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Rwhi91ideu.

Jiazheng Kang, Mingming Ji, Zhe Zhao, and Ting Bai. Memory os of ai agent, 2025. URL https://arxiv.org/abs/2506.06326.

Hyunseo Kim, Sangam Lee, Kwangwook Seo, and Dongha Lee. BESPOKE: Benchmark for search-augmented large language model personalization via diagnostic feedback. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=Qr8Zvwoi88.

Yoonho Lee, Roshen Sanjay Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=tmbOUyFx3R.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5420–5438, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.276. URL https://aclanthology.org/2025.emnlp-main.276/.

Zhuofeng Li, Haoxiang Zhang, Cong Wei, Pan Lu, Ping Nie, Yi Lu, Yuyang Bai, Shangbin Feng, Hangxiao Zhu, Ming Zhong, Yuyu Zhang, Jianwen Xie, Yejin Choi, James Zou, Jiawei Han, Wenhu Chen, Jimmy Lin, Dongfu Jiang, and Yu Zhang. Beyond semantic similarity: Rethinking retrieval for agentic search via direct corpus interaction, 2026. URL https://arxiv.org/abs/2605.05242.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn LLM agents. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=4S8agvKjle.

Qirui Mi, Zhijian Ma, Mengyue Yang, Haoxuan Li, Yisen Wang, Haifeng Zhang, and Jun Wang. Skill-pro: Learning reusable skills from experience via non-parametric PPO for LLM agents. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=9kJQjx2B80.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. Trace2skill: Distill trajectory-local lessons into transferable agent skills, 2026. URL https://arxiv.org/abs/2603.25158.

OpenAI. Harness engineering: Leveraging codex in an agent-first world. https://openai.com/index/harness-engineering/, February 2026.

Siru Ouyang, Jun Yan, Yanfei Chen, Rujun Han, Zifeng Wang, Bhavana Dalvi Mishra, Rui Meng, Chun-Liang Li, Yizhu Jiao, Kaiwen Zha, Maohao Shen, Vishy Tirumalashetty, George Lee, Jiawei Han, Tomas Pfister, and Chen-Yu Lee. Skillos: Learning skill curation for self-evolving agents, 2026a. URL https://arxiv.org/abs/2605.06614.

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

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Dongyu Ru, Lin Qiu, Xiangkun Hu, Tianhang Zhang, Peng Shi, Shuaichen Chang, Cheng Jiayang, Cunxiang Wang, Shichao Sun, Huanyu Li, Zizhao Zhang, Binjie Wang, Jiarong Jiang, Tong He, Zhiguo Wang, Pengfei Liu, Yue Zhang, and Zheng Zhang. RAGChecker: A fine-grained framework for diagnosing retrieval-augmented generation. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=J9oefdGUuM.

Alireza Salemi, Chang Zeng, Atharva Nijasure, Jui-Hui Chung, Razieh Rahimi, Fernando Diaz, and Hamed Zamani. Grepseek: Training search agents for direct corpus interaction, 2026. URL https://arxiv.org/abs/2605.29307.

Kwangwook Seo and Dongha Lee. P-check: Advancing personalized reward model via learning to generate dynamic checklist. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 43447–43471, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.2011. URL https://aclanthology.org/2026.acl-long.2011/.

Kwangwook Seo, Donguk Kwon, and Dongha Lee. MT-RAIG: Novel benchmark and evaluation framework for retrieval-augmented insight generation over multiple tables. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23142–23172, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1128. URL https://aclanthology.org/2025.acl-long.1128/.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. {ALFW}orld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Zhen Tan, Jun Yan, I-Hung Hsu, Rujun Han, Zifeng Wang, Long Le, Yiwen Song, Yanfei Chen, Hamid Palangi, George Lee, Anand Rajan Iyer, Tianlong Chen, Huan Liu, Chen-Yu Lee, and Tomas Pfister. In prospect and retrospect: Reflective memory management for long-term personalized dialogue agents. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8416–8439, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.413. URL https://aclanthology.org/2025.acl-long.413/.

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*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=ehfRiF0R3a.

Jingxing Wang, Chenyu Zhou, Zhihui Fu, Jun Wang, Weiwen Liu, Weinan Zhang, and Jianghao Lin. Skills on the fly: Test-time adaptive skill synthesis for llm agents, 2026. URL https://arxiv.org/abs/2605.16986.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. ScienceWorld: Is your agent smarter than a 5th grader? In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.775. URL https://aclanthology.org/2022.emnlp-main.775/.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Daniel Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. Openhands: An open platform for AI software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=OJd3ayDDoF.

Yu Wang, Ryuichi Takanobu, Zhiqi Liang, Yuzhen Mao, Yuanzhe Hu, Julian McAuley, and Xiaojian Wu. Mem-α: Learning memory construction via reinforcement learning, 2025b. URL https://arxiv.org/abs/2509.25911.

Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. Inducing programmatic skills for agentic tasks. In *Second Conference on Language Modeling*, 2025c. URL https://openreview.net/forum?id=lsAY6fWsog.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Forty-second International Conference on Machine Learning*, 2025d. URL https://openreview.net/forum?id=NTAhi2JEEE.

Di Wu, Devendra Singh Sachan, Wen tau Yih, and Mingda Chen. Procedural knowledge at scale improves reasoning, 2026. URL https://arxiv.org/abs/2604.01348.

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

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for LLM agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=FiM0M8gcct.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z. Pan, Hinrich Schuetze, Volker Tresp, and Yunpu Ma. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12805–12825, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.583. URL https://aclanthology.org/2026.acl-long.583/.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=mXpq6ut8J3.

Shunyu Yao, Howard Chen, John Yang, and Karthik R Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=R9KnuFlvnU.

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*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

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

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-harness: Harnesses that improve themselves, 2026a. URL https://arxiv.org/abs/2606.09498.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=eC4ygDs02R.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i17.29936. URL https://doi.org/10.1609/aaai.v38i17.29936.

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

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 19724–19731, 2024.

Contents of Appendix

A Extended Related Work **15**

B Experimental Details **15**

B.1 Datasets 15

B.2 Baselines 16

B.3 Implementation Details 17

B.3.1 Details of ExpVoyager 17

B.3.2 Prompts 19

B.4 Experimental Details for Preliminary Analyses 19

B.5 Experimental Details for Comparison to Agentic Search 21

B.6 Experimental Details for Analysis on why ExpVoyager works 21

B.7 Experimental Details for Analysis on How ExpVoyager works 22

C Additional Results and Analysis **23**

C.1 Detailed Results 23

C.2 Case Study 24

C.2.1 Success Cases 24

C.2.2 Failure Cases 26

APPENDIX

A EXTENDED RELATED WORK

Memory Management for Conversational Agents Recent work studies how conversational agents preserve information from past user interactions to support consistent and personalized responses across sessions. Early approaches introduce persistent memory for retaining conversational information (Packer et al., 2023; Zhong et al., 2024), while subsequent work organizes accumulated information into structured or interconnected memories that can be updated as new interactions arrive (Xu et al., 2025; Kang et al., 2025). More recent approaches further learn how memories should be constructed, revised, and utilized from downstream feedback (Tan et al., 2025; Wang et al., 2025b; Yan et al., 2026), or enable query-dependent access to user histories at different levels of abstraction (Seo & Lee, 2026; Kim et al., 2026). Whereas this line of work primarily uses interaction histories to maintain knowledge about users and prior conversations, EXPVOYAGER draws on an agent's own task-execution experience to guide new tasks. Rather than optimizing the memory management policy itself, EXPVOYAGER navigates past execution records to identify reusable procedural knowledge and compose it into task-specific guidance for the current execution.

Agent Harness Systems. As LLM agents tackle increasingly long-horizon and tool-intensive tasks, recent work has emphasized the surrounding agent harness that equips foundation models with the interfaces, context, and external resources required for effective execution (OpenAI, 2026; Lee et al., 2026). One line of work focuses on execution interfaces and environments, designing agent-facing tools and interaction mechanisms that enable models to operate more effectively in complex software and interactive settings (Yang et al., 2024; Wang et al., 2025a). Beyond tool access, recent systems increasingly manage the information supplied to the model during execution, maintaining compact context, persistent state, or evolving playbooks to support coherent behavior over long horizons (Zhang et al., 2026b). Reusable agent skills provide another form of harness support by packaging instructions, procedures, and auxiliary resources that can be supplied to the model when relevant to the current task (Anthropic, 2025). More recent work further makes the harness itself adaptive, optimizing how execution context is constructed or even searching over the implementation of the harness itself using feedback from previous executions (Ye et al., 2026; Lee et al., 2026). Whereas these approaches broadly improve the infrastructure and mechanisms surrounding agent execution, EXPVOYAGER focuses on the procedural knowledge supplied to a given executor for the current task. Rather than redesigning the execution interface or optimizing the harness itself, EXPVOYAGER provides a complementary mechanism for adapting the knowledge available to a fixed executor without modifying its underlying harness configuration, allowing the harness to draw on raw experience as an on-demand source of knowledge for agent execution.

B EXPERIMENTAL DETAILS

Default Configuration. Unless otherwise specified, we use Qwen3.5-9B as both the skill curator ϕ and frozen executor θ, set the navigation budget to $R = 20$ rounds, and report results averaged over three random runs. Except for Tables 1 and 2, which explicitly vary the curator–executor configuration, all experiments follow this default setting. For fair comparison, EXPVOYAGER and all baselines use the same backbone models and source pool within each configuration.

B.1 DATASETS

ALFWorld. ALFWorld (Shridhar et al., 2021) evaluates household task execution through text-based environments constructed by aligning TextWorld with ALFRED. It comprises six task categories—Pick & Place, Examine in Light, Clean & Place, Heat & Place, Cool & Place, and Pick Two & Place—across 120 rooms covering kitchens, bedrooms, bathrooms, and living rooms. Given a natural-language goal, an agent navigates the environment and manipulates objects through high-level textual commands, receiving observations after each action. We use 1,000 source trajectories for offline evaluation setting, retaining both successful and unsuccessful attempts as past experience. For evaluation, we follow the official split of ALFWorld and use all 140 instances as our test set, which are separate from the source instances. Each evaluation episode is limited to 50 execution steps.

WebShop. WebShop (Yao et al., 2022) is an interactive shopping benchmark built from approximately 1.18 million Amazon products. Agents use `search[query]` and `click[button]` actions to browse products, inspect their details, select options, and complete a purchase that satisfies the user's requirements. Each episode receives a reward between 0 and 1 based on the purchased item's agreement with the requested product type, attributes, options, and price constraint. To construct our source trajectory pool, we sample 1,000 training instructions without replacement using a fixed random seed of 42 and collect one agent rollout per instruction. We retain trajectories with both full and partial or unsuccessful outcomes and follow the official test split, evaluating on all 500 held-out test instructions. Each evaluation episode is limited to 15 execution steps.

ScienceWorld. ScienceWorld (Wang et al., 2022) assesses scientific reasoning through interactive experiments in a text-based simulated environment. Its 30 task types span 10 science topics, including changes of state, electrical conductivity, chemical mixtures, and plant growth. In the provided environment, agents should translate a goal instruction into a sequence of actions, such as navigating between locations, manipulating materials, and using scientific instruments, while interpreting feedback from the environment. Following the official setting of AgentBoard (Ma et al., 2024), we adopt its fixed evaluation set of 90 ScienceWorld instances. Each evaluation episode is limited to 30 execution steps. We measure performance using the environment's official score, which rewards partial credit for completing required and optional subgoals, normalized to $[0, 1]$ and averaged across evaluation instances. We construct our source pool by sampling 500 distinct task–variation pairs from the official training split using a fixed random seed of 42, approximately preserving the task-family proportions of the evaluation set subject to availability.

B.2 Baselines

We compare against four representative baselines that reuse agent experience through workflow retrieval (Wang et al., 2025d), reasoning memory (Ouyang et al., 2026b), global skill consolidation (Ni et al., 2026), and task-specific skill synthesis (Wang et al., 2026). For fair evaluation, all methods use the same source trajectory pools and are evaluated on the same held-out tasks. Each method constructs its memory or skills from the source experience according to its own procedure.

AWM. Agent Workflow Memory (AWM) (Wang et al., 2025d) extracts reusable workflows from successful agent trajectories and provides them as procedural guidance for subsequent tasks. For a fair comparison, we follow the retrieval-based AWM setting used in ReasoningBank (Ouyang et al., 2026b), retrieving workflows from a preconstructed memory bank rather than inducing a new workflow for each target task. We use Qwen3-Embedding-8B across all three benchmarks to embed the target's task type and objective, as well as the stored workflow content. Candidate workflows are ranked by cosine similarity across the entire workflow bank for the corresponding benchmark. The top-3 workflows are provided to the executor as additional context.

ReasoningBank. ReasoningBank (Ouyang et al., 2026b) distills reusable reasoning strategies from both successful and failed trajectories, and stores them in a searchable memory bank. We follow the official implementation for memory extraction and retrieval, using the source trajectories to construct the memory bank. For all benchmarks, we use Qwen3-Embedding-8B for embedding-based similarity search. Given a target task, the method retrieves the most relevant experience entry and injects its associated reasoning memories into the executor's context. We use top-1 retrieval, which achieved the best performance in the retrieval-size ablation reported in the original paper.

Trace2Skill. Trace2Skill (Ni et al., 2026) consolidates lessons from agent trajectories into a reusable skill through a hierarchical map–reduce procedure. Starting from a frozen seed skill, we independently process each source trajectory in the MAP stage to propose structured edits, extracting reusable procedures from successes and corrective checks from failures. These proposals are then merged hierarchically with a fan-in of five, resolving conflicts and removing redundant guidance at each level. The final merged edits are applied deterministically to the seed, producing a single `SKILL.md` for each benchmark. This skill is shared across all evaluation tasks within the benchmark. To use the same source experience as the other baselines, we reuse the existing source trajectories rather than collecting new rollouts conditioned on the seed skill.

SkillTTA. SkillTTA (Wang et al., 2026) synthesizes a temporary skill for each target task from retrieved source trajectories. Specifically, we encode source metadata and the visible target context with Qwen3-Embedding-8B across all three benchmarks and retrieve the top-3 source trajectories by cosine similarity for each target, considering both successful and unsuccessful attempts. The retrieved trajectories and target context are used to synthesize a task-specific `SKILL.md` containing applicability conditions, possible failure modes, procedures, and a verification checklist.

B.3 Implementation Details

B.3.1 Details of ExpVoyager

Models and local inference. We serve Qwen3.5-9B and Gemma4-31B locally using vLLM with an OpenAI-compatible Chat Completions interface on NVIDIA RTX A6000 GPUs with 48 GB memory. In the six-GPU configuration, Qwen3.5-9B uses six independent single-GPU replicas with tensor parallelism of one, while Gemma4-31B uses three replicas with tensor parallelism of two. Both local models run in BF16 with a context window of 32,768 tokens, a temperature of 1.0, and thinking enabled. We use model-specific reasoning parsers and the Hermes and Gemma4 tool-call parsers for Qwen and Gemma, respectively. For experiments using GPT-5.4-mini as the curator (Table 2), we access the model through the OpenAI Responses API with reasoning effort set to medium, without explicitly specifying sampling parameters. The corresponding Gemma4-31B executor remains locally served, using two replicas with tensor parallelism of two.

Experience representation. We retain source trajectories as chronological execution records and materialize trajectory-level and step-level views in separate JSONL indexes. Trajectory-level records contain the ordered action sequence, episode-level execution outcome. Step-level records contain the observations (before and after action), executor reasoning, executed action, and immediate result. Both views retain a reference to the full source trajectory.

Navigable Interface. The curator invokes `search_exp` by specifying the view level, fields, regular-expression pattern, and result limit. We implement case-insensitive regular-expression matching in Python over the selected fields. Matching records are returned in stored order up to the requested limit. Each match returns the complete stored view record and its source reference, even when matching uses only a subset of the fields. Trajectory-level searchable fields comprise the action sequence, and outcome; step-level searchable fields comprise the observation, reasoning, action, and result. The curator invokes `inspect_traj` on a source reference to access the chronological trajectory. The underlying reader supports line offsets and limits, with continuation information for longer records.

Native tool calling and navigation control. Navigation actions are implemented through native function calling. Tool descriptions and JSON-schema argument specifications are supplied through the request's `tools` field. The runtime parses the returned `tool_calls` and dispatches the corresponding Python function. We set `parallel_tool_calls=False` to execute one navigation operation per action-selection request. Navigation requests use automatic tool selection, whereas initialization and state-update requests explicitly select their respective state-management functions through `tool_choice`. The curator may return `FINISH` after incorporating valid navigation observations into its state. Navigation otherwise continues until the default budget of 20 rounds is exhausted.

Navigation State. We represent $Z_r = (K_r, Q_r)$ as a structured JSON object initialized from the target context. Each knowledge item records a source-specific observation, a transferable procedural relation, and conditions that remain to be verified during execution. Items are organized by their role in execution, while open questions retain unresolved procedural uncertainties and their priorities. After receiving a navigation observation, a separate curator request assesses its relevance and updates the knowledge items and open questions. The runtime validates the structured update and maintains references to supporting observations. Subsequent navigation requests receive the target context, current Navigation State, active experience records, and navigation feedback. After the final navigation round, the curator synthesizes `skill.md` from the resulting state and target context. Figures 10 and 11 illustrate an example of Navigation State update and a generated `skill.md`, respectively.

Navigation State update **Target:** put a newspaper in sofa

Observation ω_r 00213 · step 15

```
... { "observation": "You arrive at armchair 2. On the armchair
  2, you see a newspaper 3, and a newspaper 2.",
  "action": "take newspaper 2 from armchair 2",
  "result": "You pick up the newspaper 2 from the armchair 2.",
    ... } ...
```

Curator reasoning

... The armchair newspaper locations provide useful guidance - they're common surfaces - but don't guarantee presence without coffeetable. ...

Navigation State Z_r

```
{ "K": [{
1  "source_observation": "Trajectory 00213 found
     newspaper 2 on armchair 2 after coffeetable
     checks; ...",
2  "transferable_rule": "If coffeetable 1 checks
     return empty, ... check armchair 2 as
     secondary newspaper source; ...",
3  "target_unknown": "Whether armchair 2 exists
     and contains newspaper(s) in current
     instance",
   ... }, ...],
  "Q": [{
4  "question": "If sofa 1 is already the target
     for placement, does inspecting it for
     existing newspapers help determine
     relocation sources?",
    "priority": "high", ... }]
}
```

→

Navigation State Z_{r+1}

```
{ "K": [{
1  "source_observation": "Trajectory 00213 step
     15 found newspaper 3 and newspaper 2 on
     armchair 2",
2  "transferable_rule": "... check armchair(s)
     accessible in current observation before
     sofa placement",
3  "target_unknown": "Whether armchair(s) exist
     and contain newspaper(s) in current
     instance",
   ... }, ...],
  "Q": [{
+   "question": "Which furniture surfaces serve
     as reliable newspaper sources when
     coffeetable is absent from observation?",
    "priority": "critical", ... }, {
4  "question": "During newspaper search, should
     armchair examination occur before or after
     sofa check, given papers may already exist
     on sofa?",
    "priority": "high", ... }]
}
```

Figure 10: **Example of a Navigation State update in ALFWorld.** The curator uses navigation observation ω_r to revise knowledge K_r and open questions Q_r, while newspaper availability in the target remains unverified. Orange and green highlight prior and updated content, respectively; matching numbers link corresponding content, and + marks a new question.

`skill.md` **Target:** put a newspaper in sofa

```
# SKILL.md

## Procedure
1. Identify and navigate to a reliable newspaper source location. Prioritize checking
`coffeetable` (e.g., `coffeetable 1`) for newspapers based on high-yield experience. If not
found or unavailable, systematically inspect secondary sources such as `armchair 1` or
`sidetable 1-5`. Do not cycle through the same furniture repeatedly.
2. Once a newspaper is located, execute `take newspaper [ID] from [Furniture]` to acquire
the item.
3. Navigate to the target furniture using `go to sofa 1`.
4. Execute `move newspaper [ID] to sofa 1` to complete the placement.

## Checks and Recovery
- **Source Verification**: If navigation to `coffeetable` fails or inventory remains empty,
immediately broaden search to adjacent or nearby furniture (`armchair`, `sidetable`,
`dresser`) to prevent `incomplete_systematic_search`.
- **Identifier Accuracy**: Confirm `sofa` identifier matches current observation (`sofa 1`)
before executing move commands to avoid `move_command_valid` failures.
- **Obstruction Handling**: If `move` fails, verify current inventory and ensure the
newspaper was successfully taken into pocket/inventory before attempting placement again.
```

Figure 11: **Generated `skill.md` for the same ALFWorld task.** Synthesized after the final navigation round, the skill provides a procedure for locating and placing a newspaper, with checks for source availability, identifier accuracy, and placement recovery. The full generated Markdown is shown.

B.3.2 PROMPTS

We present the curator prompts for Navigation Action Selection, Navigation State Update, and Final Skill Synthesis. The system and user prompts for Navigation Action Selection (Tables 7 and 8) guide the curator to select an open question, choose the appropriate experience view and search fields, and determine when to finish navigation. The native function schemas for `search_exp` and `inspect_traj` (Tables 9 and 10) specify the arguments for searching experience records and inspecting a source trajectory, respectively. The system and user prompts for Navigation State Update (Tables 11 and 12) guide the curator to determine whether the navigation observation answers the current question, distinguish transferable knowledge from facts requiring verification during execution, and update the knowledge items and open questions. The Final Skill Synthesis prompt (Table 13) instructs the curator to generate `skill.md` from Navigation State of the final round, preserving supported procedures, conditional guidance, and relevant checks and recovery strategies.

B.4 EXPERIMENTAL DETAILS FOR PRELIMINARY ANALYSES

Oracle skill construction. We define *oracle skill* as a target-specific skill whose helpfulness for the corresponding task has been verified through repeated execution, rather than as a unique or exhaustive description of all knowledge that could solve the task. We use GPT5.6 Sol for candidate skill generation, knowledge annotation, and claim-level evaluation, and Qwen3.5-9B for trajectory collection and execution-based verification. For each target task, we initially collect ten independent execution rollouts and collect additional rollouts as needed to obtain at least four successful and four failed trajectories, ensuring sufficient evidence for both effective procedures and failure-specific checks or recovery strategies. Given the target instruction, initial observation, and collected trajectories with their outcomes, the model synthesizes a candidate skill containing actionable procedures and relevant checks or recovery strategies. We accept a candidate as an oracle skill only after verifying its helpfulness for the target task through execution. Specifically, we provide each candidate skill to the same frozen executor and conduct three independent executions of the corresponding task. To reduce the influence of execution randomness, we retain a candidate only if all three executions succeed. Otherwise, failed execution trajectories are added to the evidence pool before generating the next candidate. We allow up to 20 candidate versions per target and obtain 80 oracle skills through this iterative generate-then-verify process. Although repeated execution-based verification reduces the influence of execution randomness, successful outcomes can still arise from task-specific shortcuts or other spurious factors. We therefore conduct a manual review of each retained oracle skill and its execution trajectories to confirm that the target requirements are genuinely satisfied and that the skill provides useful procedural guidance without effectively substituting for execution through an overly instance-specific solution.

Source sampling and oracle knowledge annotation. We sample 300 source trajectories from the ALFWorld training split, including both successful and unsuccessful attempts. For each of the 80 oracle skills, we examine every source trajectory. The annotation model receives the target information, the oracle skill, and the source trajectory, including its recorded reasoning, actions, observations, and outcome information. The oracle skill identifies the knowledge needed for the target, while the source trajectory provides the experience supporting that knowledge. An item is annotated only when supporting evidence can be identified in the source itself. A source is labeled helpful when it contains non-trivial knowledge that can contribute to constructing the corresponding oracle skill. Such knowledge includes actionable procedures, state transitions, preconditions, ordering constraints, failure causes, recovery strategies, and completion checks. Generic interface knowledge, task restatements, superficial object-name overlap, and unsupported speculation are excluded. Failed trajectories can therefore be helpful when they reveal valid partial progress or a concrete failure lesson, while successful trajectories can also receive a *none* label when they contribute no relevant knowledge. Each annotated item is linked to a supporting excerpt from the source, and sources without qualifying items are labeled *none*. After annotation, manually review all knowledge annotations, including retained knowledge items and *none* labels, to verify that annotated items are supported by the cited source evidence and that the above inclusion and exclusion criteria are applied consistently.

Claim extraction and semantic matching. Following the claim-level evaluation protocol of Ru et al. (2024); Seo et al. (2025), we separate evaluation into claim extraction and semantic matching. We adapt this principle to procedural knowledge in skills and agent experience, assessing the degree

of semantic correspondence on a five-point Likert scale. First, we decompose each evaluated artifact into atomic knowledge claims. An atomic claim expresses one independently assessable procedure, condition, state transition, or verification rule. Statements containing several separable instructions are split, while conditions essential to an instruction's meaning are preserved. Oracle knowledge is hidden during this extraction stage so that claims are extracted from the artifact independently of the desired matches. Each extracted claim is linked to supporting text in the artifact. Next, we rate the semantic correspondence between each extracted claim and the target's oracle knowledge on a 1–5 Likert scale. The judgment considers how closely the actionable knowledge and its required conditions, ordering constraints, causal relations, and verification requirements align, rather than relying on surface-level textual similarity. Differences in wording or object names are permitted when the claims express the same transferable knowledge, whereas shared keywords or task categories alone are insufficient. A score s is normalized to $[0, 1]$ using $(s - 1)/4$.

For aggregation, each item contributes only its highest matching score against the comparison set. When measuring oracle knowledge coverage, we use the score of the best-matching claim for each oracle item. When measuring the precision of retrieved knowledge, we use the score of the best-matching oracle item for each retrieved claim. Scores from multiple matches are not summed, so each item contributes at most once to the corresponding metric.

Analysis I: Knowledge preservation in pre-constructed skills. We evaluate the pre-constructed skills or memory artifacts of AWM, ReasoningBank, and Trace2Skill. For each target, we compare their extracted atomic claims with the oracle knowledge annotated in the corresponding source trajectories. We record which source trajectories are used to construct each artifact and use the oracle knowledge annotated in those sources as its reference. For methods that consolidate multiple trajectories, such as AWM (Wang et al., 2025d) and Trace2Skill (Ni et al., 2026), we compare the consolidated artifact against the oracle knowledge from its contributing sources without duplicating the artifact for each source. All methods are evaluated against oracle knowledge annotated over the same source pool. Knowledge preservation is measured by taking the highest normalized matching score between each oracle item and the artifact's claims, then averaging these scores. This captures how faithfully the source knowledge is retained in the pre-constructed artifacts. We compute preservation scores separately for each target and then macro-average across targets.

Analysis II: Source-level evaluation. We evaluate AWM, ReasoningBank, and SkillTTA at retrieval depths $k \in \{1, 3, 5, 10, 20\}$, following each method's retrieval setup. AWM retrieves pre-constructed workflows, ReasoningBank retrieves memory entries, and SkillTTA retrieves raw source trajectories. We use Qwen3-Embedding-8B for embedding-based retrieval. Candidate pools are restricted to the sampled source trajectories or artifacts constructed from them, and oracle annotations are used only for evaluation.

For target t, let H_t denote the set of source trajectories labeled helpful for that target in the preceding helpfulness annotation. $R^k_{m,t}$ denotes the original source trajectories associated with the top-k artifacts retrieved by method m. We identify these sources from the records of which trajectories were used to construct each artifact and count repeated source IDs only once. Source recall is defined as

$$\text{SourceRecall}\,@k = \frac{\text{Number of retrieved helpful sources}}{\text{Total number of sources annotated as helpful}}.$$

Here, k counts each method's native retrieval units. For AWM, which consolidates multiple source trajectories into each workflows, one retrieved workflow may correspond to several source trajectories.

Analysis II: Knowledge-level evaluation. For each target t, we construct a unique knowledge set K_t by removing semantically similar or duplicate items expressing the same knowledge from the oracle annotations over the 300 source trajectories. Items that differ only in wording are represented once, while distinct conditions, ordering requirements, or recovery rules remain separate. We then extract atomic claims from the actual retrieved content: workflow text for AWM, memory text for ReasoningBank, and raw trajectory content for SkillTTA. Semantically duplicate claims are removed from the retrieved collection to obtain a unique claim set. We apply the same 1–5 scoring and normalization procedure described above. Let r_i denote the highest normalized matching score between oracle item i and the retrieved claims, and let p_j denote the highest normalized matching

score between retrieved claim j and the oracle knowledge. With N oracle items and L unique retrieved claims, we calculate

$$\text{KnowledgeRecall}\,@k = \frac{1}{N}\sum_{i=1}^{N} r_i, \qquad \text{KnowledgePrecision}\,@k = \frac{1}{L}\sum_{j=1}^{L} p_j.$$

Both r_i and p_j lie in $[0, 1]$. Recall measures how faithfully the target's oracle knowledge is covered by the retrieved material, while precision measures how closely the retrieved knowledge aligns with the target's oracle knowledge. Retrieving a helpful source does not automatically credit all knowledge associated with that source. Matching scores must be supported by the content actually retrieved. Both metrics are computed per target and macro-averaged across targets at each retrieval depth.

B.5 Experimental Details for Comparison to Agentic Search

To examine whether ExpVoyager's gains come simply from additional experience access, we compare it against agentic search extensions of retrieval-based baselines. Starting from the target task, each extension repeatedly retrieves top-k skills or trajectories and refines its search query based on the retrieved results. Processing these results and reasoning about subsequent retrieval queries incur additional tokens, allowing us to compare agentic search through a passive retrieval interface with active navigation under comparable experience-access budgets. Specifically, we operationalize the experience-access budget as the token usage of the entire task-time process before downstream execution, including experience retrieval or navigation, reasoning over the accessed experience, and any method-specific step used to construct the final guidance. For ExpVoyager, this includes navigation-action selection, processing of accessed experience, Navigation State updates, and final skill synthesis. For the baselines, we follow their native task-time pipelines: SkillTTA includes target-specific skill synthesis after retrieval, whereas AWM and ReasoningBank directly reuse their pre-constructed workflows or memories and therefore incur no additional synthesis stage. We vary ExpVoyager's navigation rounds and continue iterative retrieval for each baseline until this pre-execution token usage approximately matches that of ExpVoyager. Downstream task execution is excluded from the matched budget and is included only when reporting the full end-to-end token cost. From the results, ExpVoyager consistently outperforms these extensions under comparable budgets, suggesting that active navigation makes more effective use of the available task-time computation. However, its performance also declines beyond approximately 20–25 rounds, possibly because continued navigation after sufficient useful experience has been collected introduces redundant or less relevant experience that can interfere with task execution. We present the full results in Table 5.

B.6 Experimental Details for Analysis on why ExpVoyager works

To better understand the distinct roles of the Navigable Interface and Navigation State, we ablate each component while preserving the remaining ExpVoyager pipeline and navigation setup (Section 4.3).

w/o Navigable Interface. We retain the Navigation State and the same iterative navigation procedure, but remove the experience-specific views and access operations provided by the Navigable Interface. Specifically, similar to Li et al. (2026), we expose the raw source trajectories as chronological files and allow the curator to interact with them through general-purpose terminal operations (e.g., grep/rg) over the serialized trajectory content. Unlike the full ExpVoyager, this interface does not explicitly expose the execution-specific structure of agent experience through complementary views. Instead, distinct execution semantics, such as episode-level procedural patterns and local context–decision–consequence relations, remain embedded in the raw trajectory content rather than being surfaced through dedicated views and view-specific fields for targeted access. The matched portions of source trajectories, with their surrounding context determined by the curator's search and read operations, are then passed to the same Navigation State updater as in the full ExpVoyager, which guides subsequent navigation.

w/o Navigation State. We retain the full Navigable Interface and the same iterative navigation procedure, but remove the explicit Navigation State. Specifically, instead of maintaining $Z_r = (K_r, Q_r)$ across rounds, the curator conditions each navigation decision on the target context and the accumulated navigation history, including previous navigation actions and the experience records

accessed through the interface. To keep the interaction context bounded across rounds, earlier navigation history is compressed into a simple running summary that preserves the information needed for subsequent decisions. Thus, evidence from earlier rounds remains available in the interaction context, but is not explicitly interpreted and consolidated into target-relevant procedural knowledge and open questions that are revised after each navigation round. This variant therefore preserves the same experience-access operations and navigation process while removing the explicit state management that connects the interpretation of previously accessed experience with what to investigate next.

B.7 Experimental Details for Analysis on How ExpVoyager works

We analyze ExpVoyager's experience-access behavior on ALFWorld, Webshop, and ScienceWorld with three runs per task. Both the curator and executor use Qwen3.5-9B, and navigation is limited to 20 rounds. The remaining settings follow the main experiments. We collect the navigation actions, open questions, curator reasoning, and accessed source trajectories to examine action selection, information needs, and source diversity.

For Figure 9(a), we group navigation actions into trajectory-view search, step-view search, and trajectory inspection. We aggregate the number of calls across all tasks and runs and compute the proportion of each action. The inner ring shows the proportions of `search_exp` and `inspect_traj`, while the outer ring further divides `search_exp` into trajectory-view and step-view searches.

For Figure 9(b), we classify the information sought at each navigation round using the open question Q_n together with the curator's reasoning for that navigation decision. GPT-5.6 Sol assigns one primary category to each round. *Procedure Overview* covers questions about the overall procedure or ordering of subgoals. *Local Decision* covers questions about a specific action, its execution conditions, or command details. *Cross-step Context* covers questions about dependencies between actions or state changes across multiple steps. When a question involves multiple needs, the primary category is determined by the main information sought, as expressed in the question and accompanying reasoning. We then group these labels by the selected action and report the proportion of each category within each action.

For Figure 9(c), we measure source diversity by counting the unique source trajectories accessed during navigation for each task and run. This includes trajectories accessed through trajectory-view search, step-view search, and trajectory inspection. Multiple records or repeated accesses originating from the same trajectory count as a single source. We group the resulting counts into intervals of 0–19, 20–39, 40–59, 60–79, and 80 or more trajectories, and report the percentage of tasks in each interval averaged over the three runs.

Table 4: Results on ALFWorld across different subtask types, using Qwen3.5-9B as the curator ϕ.

Methods	Pick	Look	Clean	Heat	Cool	Pick2	Avg. SR	Steps
Executor θ: Qwen3.5-9B								
Base (ReAct) (Yao et al., 2023)	$70.5_{1.6}$	$48.7_{4.4}$	$32.1_{2.1}$	$22.9_{3.6}$	$21.3_{2.3}$	$52.8_{2.4}$	$41.4_{1.3}$	34.1
+ AWM (Wang et al., 2025d)	$67.6_{3.3}$	$48.7_{4.4}$	$49.4_{2.2}$	$35.4_{3.6}$	$32.0_{4.0}$	$40.3_{2.4}$	$45.6_{1.8}$	35.1
+ RBank (Ouyang et al., 2026b)	$83.8_{1.6}$	$41.0_{4.4}$	$34.6_{2.2}$	$18.8_{6.3}$	$41.3_{2.3}$	$41.7_{4.2}$	$43.5_{2.2}$	33.9
+ Trace2Skill (Ni et al., 2026)	$65.7_{2.9}$	$38.5_{0.0}$	$55.6_{3.7}$	$25.0_{6.2}$	$33.3_{2.3}$	$54.2_{0.0}$	$45.4_{1.7}$	34.3
+ SkillTTA (Wang et al., 2026)	$81.0_{1.7}$	$30.8_{7.7}$	$67.9_{4.3}$	$29.2_{3.6}$	$\mathbf{54.7}_{2.3}$	$43.1_{2.4}$	$51.1_{2.8}$	30.4
+ **ExpVoyager** (Ours)	$\mathbf{88.6}_{2.9}$	$\mathbf{74.4}_{4.4}$	$\mathbf{72.9}_{4.3}$	$\mathbf{47.9}_{3.6}$	$42.7_{2.3}$	$\mathbf{55.6}_{2.4}$	$\mathbf{63.7}_{2.2}$	**25.9**
Executor θ: Gemma4-31B								
Base (ReAct) (Yao et al., 2023)	$91.4_{2.9}$	$56.4_{4.5}$	$39.5_{2.1}$	$52.1_{3.6}$	$53.3_{2.3}$	$30.6_{2.4}$	$53.9_{0.6}$	29.2
+ AWM (Wang et al., 2025d)	$87.6_{1.7}$	$59.0_{4.4}$	$54.3_{4.3}$	$\mathbf{54.2}_{3.6}$	$50.7_{2.3}$	$36.1_{2.4}$	$57.0_{0.9}$	27.5
+ RBank (Ouyang et al., 2026b)	$92.4_{1.7}$	$64.1_{4.4}$	$51.9_{3.7}$	$52.1_{7.2}$	$50.7_{4.6}$	$36.1_{6.4}$	$57.9_{1.5}$	28.8
+ Trace2Skill (Ni et al., 2026)	$86.7_{4.4}$	$53.8_{7.7}$	$50.6_{2.1}$	$47.9_{9.5}$	$54.7_{4.6}$	$37.5_{4.2}$	$55.2_{2.4}$	25.4
+ SkillTTA (Wang et al., 2026)	$92.4_{1.7}$	$76.9_{7.7}$	$60.5_{4.3}$	$\mathbf{54.2}_{7.2}$	$50.7_{4.6}$	$40.3_{4.8}$	$62.5_{1.6}$	24.7
+ **ExpVoyager** (Ours)	$\mathbf{94.3}_{2.9}$	$\mathbf{84.6}_{7.7}$	$\mathbf{82.7}_{2.1}$	$\mathbf{54.2}_{3.6}$	$\mathbf{57.3}_{2.3}$	$\mathbf{44.4}_{2.4}$	$\mathbf{69.6}_{0.1}$	**22.0**

Table 5: Detailed results on the ALFWorld and ScienceWorld in Figure 7. Tokens denote End-to-end tokens per task across the full task-time pipeline. B01–B06 index budget groups, and R is the number of ExpVoyager navigation rounds. Baselines use iterative retrieval in B01–B06. Following each works original setup, one-shot settings are top-3 for AWM and SkillTTA and top-1 for ReasoningBank. Bold marks the highest SR within each budget group for each benchmark.

Budget (R)	ExpVoyager		AWM		ReasoningBank		SkillTTA	
	Tokens ↓	SR ↑	Tokens ↓	SR ↑	Tokens ↓	SR ↑	Tokens ↓	SR ↑
ALFWorld	No-skill: 51,880.50 tokens/task; SR 41.4%.							
One-shot	–	–	110,392.94	45.6	94,361.06	43.5	123,427.34	51.1
B01 ($R = 5$)	159,389.51	**60.4**	156,245.11	47.9	161,216.50	47.0	162,389.29	53.1
B02 ($R = 10$)	239,787.24	**62.9**	244,231.09	48.8	233,414.12	45.5	239,236.17	51.9
B03 ($R = 15$)	315,366.21	**63.4**	282,331.08	47.2	300,416.35	46.0	313,346.31	48.9
B04 ($R = 20$)	385,314.17	**63.7**	378,191.30	45.9	371,782.17	43.6	405,290.17	49.1
B05 ($R = 25$)	439,619.44	**62.0**	448,131.31	45.9	440,619.44	43.1	439,009.04	47.7
B06 ($R = 30$)	480,549.49	**58.8**	491,139.12	45.0	480,117.07	42.9	478,132.18	46.5
ScienceWorld	No-skill: 40,722.28 tokens/task; SR 27.0%.							
One-shot	–	–	45,629.32	32.6	36,728.71	29.6	69,037.24	29.6
B01 ($R = 5$)	131,518.27	**35.9**	135,123.11	33.7	123,118.30	31.9	131,238.36	28.7
B02 ($R = 10$)	216,308.67	**34.7**	226,174.18	33.2	226,311.35	29.6	208,909.92	30.8
B03 ($R = 15$)	286,391.10	**36.6**	279,311.10	31.7	282,121.41	28.7	265,191.10	29.2
B04 ($R = 20$)	363,279.73	**37.4**	369,239.24	31.0	341,272.13	29.4	367,991.21	28.7
B05 ($R = 25$)	419,819.22	**37.7**	418,817.12	30.1	421,123.15	28.5	439,111.42	27.3
B06 ($R = 30$)	494,121.38	**35.3**	487,872.37	29.6	492,998.98	28.6	483,331.17	26.8

C Additional Results and Analysis

C.1 Detailed Results

We provide detailed results for three experiments reported in the main text: (1) the ALFWorld subtask-wise breakdown of the main results in Table 1, with detailed results in Table 4; (2) the experience-access budget comparison in Figure 7, with detailed results in Table 5; and (3) the synergy between ExpVoyager and pre-constructed skills in Figure 6, with detailed results in Table 6.

Table 6: Comparison of standalone EXPVOYAGER and its combinations with pre-constructed skills, corresponding to the lower panels of Figure 6. Performance denotes success rate (%) on ALFWorld and score on WebShop, with higher values indicating better performance. E2E tokens are reported per task and cover the full task-time pipeline, including navigation and execution. Gray rows denote combinations with pre-constructed skills. Bold indicates improvements over standalone EXPVOYAGER within each benchmark: higher performance or lower token cost.

Benchmark	Method	SR / Score ↑	E2E tokens/task ↓
ALFWorld	EXPVOYAGER (alone)	63.7	385,314
	AWM + EXPVOYAGER	**65.3**	**320,103**
	ReasoningBank + EXPVOYAGER	**64.9**	**298,582**
	Trace2Skill + EXPVOYAGER	59.6	**293,893**
WebShop	EXPVOYAGER (alone)	61.2	390,997
	AWM + EXPVOYAGER	**64.24**	**329,760**
	ReasoningBank + EXPVOYAGER	**62.7**	**317,538**
	Trace2Skill + EXPVOYAGER	60.95	**311,924**

C.2 CASE STUDY

To better understand how EXPVOYAGER navigates past experience and synthesizes target-conditioned skills, we conduct a qualitative analysis of successful and failed target executions across ALFWorld, WebShop, and ScienceWorld. The *cherry-picked* **success cases** provide a closer look at the navigation process , showing how the curator moves across experience views, follows evolving knowledge needs, and turns the observations into skill for the target task. The *lemon-picked* **failure cases** focus on the remaining challenges of experience navigation, examining where the navigation process can still fail to produce effective guidance for the target task. Each figure presents the target task, example navigation rounds, the corresponding navigation observations, and updated navigation state.

C.2.1 SUCCESS CASES

Distinguishing Source-Specific Observations from Target-Relevant Knowledge. Figure 12 illustrates how the curator distinguishes a source-specific object location from a procedure that can transfer to the current target. The ALFWorld task requires *placing a newspaper on a sofa*. An initially accessed source trajectory obtains the *newspaper from a coffee table*, but the target's initial observation lists *no coffee table*. This discrepancy redirects navigation from the placement action to alternative newspaper locations. Using `search_exp` in the step-level view, the curator locates a record in which *a newspaper is obtained from an armchair*, and `inspect_traj` expands this record into the surrounding search and placement sequence. The curator interprets these navigation observations as support for a reusable search-and-placement procedure, rather than evidence that the newspaper occupies the same location in the target environment. The synthesized skill retains an execution-time check of the expected source and recovery guidance to broaden the search when it is empty or unavailable. The frozen executor ultimately finds *a newspaper on a side table* and completes the task.

Connecting Actions to Their Observed Results. Figure 13 demonstrates how the curator uses local execution records and their surrounding context to establish a required action and the observations needed to check its effect. The ALFWorld task requires *cleaning a soap bar and placing it in a toilet*. Through `inspect_traj`, the curator examines observations of *the same soap bar before and after cleaning*, connecting the action to the observed change in object state. A further `search_exp` call over the reasoning field in the step-level view accesses a record explaining that *the agent already holds the soap bar* and should therefore proceed with cleaning even though *the sink appears empty*. Together, these navigation observations clarify the condition under which the action can proceed and the result that should be checked afterward. The synthesized skill requires cleaning before placement and retains verification of the cleaning result as an execution-time check. With this guidance, the executor completes the task, whereas the base agent without skills never issues a *cleaning action*.

Preserving Action Ordering through Cross-Step Context. Figure 14 shows how expanding local records into broader trajectory context helps the curator identify the ordering required for the target task. The ALFWorld task requires *cooling a plate and placing it in a cabinet*. An initial source trajectory provides the cabinet-placement action, while a later `inspect_traj` call exposes a sequence in which *an early placement is followed by cooling and another placement*. A subsequent `search_exp` call in the step-level view supplies a concrete cooling command. This cross-step context distinguishes an intermediate action from the final placement that follows the required change in object state. The curator interprets these navigation observations together to establish that cooling must precede the final placement. The synthesized skill turns this procedural knowledge into an ordered sequence: *acquire the plate, cool it with a refrigerator, and then place it in the cabinet*. The frozen executor follows this ordering and completes the task.

Retaining Target Conditions as Execution-Time Checks. Figure 15 illustrates how the curator interprets past decisions to identify conditions that still require verification during target execution. The WebShop task requires *black jeans in a specified size*, but the relevant attributes may appear either as selectable options or as product text. Using `search_exp` over recorded reasoning in the step-level view, the curator accesses a record warning that *an immediate purchase could leave the product in an unintended default variant*. Through `inspect_traj`, the curator also examines *a purchase made without selecting the requested options after opening a new product*. These navigation observations support a procedural rule that depends on how an attribute is presented: selectable options require explicit interaction, whereas descriptive attributes require checking the product text. The synthesized skill retains these conditions as execution-time checks for the current product. The frozen executor *selects black and large before purchasing*, applying the procedure to the options available in the target environment.

Checking Target Conditions against Available Options. Figure 16 demonstrates how procedural knowledge from past experience guides verification of the options available in the current environment. The WebShop task requires *grey memory-foam slippers*. The curator uses `search_exp` in the step-level view to access recorded reasoning about checking product details, then inspects a source trajectory in which *the agent leaves an unsuitable candidate and selects an option on a replacement product*. These navigation observations support execution-time checks of the requested attributes and recovery guidance for replacing a candidate that cannot satisfy them. During target execution, the first product mentions *"Black-Grey"* in its description, but its selectable colors are *pink and lake blue*. Guided by the synthesized skill, the frozen executor *moves to another candidate, selects its grey option, and completes the purchase*. The transferred knowledge is therefore a verification-and-recovery procedure, rather than an assumption that related wording in a product description establishes the availability of a requested option.

Identifying Required Conditions across Source Trajectories. Figure 17 shows how the curator combines navigation observations from different source trajectories to identify a condition needed for the current target. The ScienceWorld task requires *making a peanut-butter sandwich*. Through `inspect_traj`, the curator examines a failed sandwich attempt in which *acquiring the ingredients and then mixing an ingredient produces no result*. A `search_exp` call over the reasoning field in the step-level view accesses a record from a mixed-nuts task that connects mixing to *ingredients already sharing a container*. Further trajectory inspection clarifies that *the sandwich ingredients had remained in inventory rather than being transferred into a common container*. Interpreting these observations together, the curator identifies the shared-container requirement as procedural knowledge that can transfer across recipes. The synthesized skill turns this condition into execution guidance: *transfer the ingredients into a common container before mixing its contents*. The frozen executor *transfers the ingredients into a cup and successfully produces the sandwich.*

Interpreting Recorded Reasoning for the Current Target. Figure 18 illustrates how recorded reasoning helps the curator interpret why an action was taken and what part of that decision can transfer to the current target. The ScienceWorld task requires *identifying the animal with the longest lifespan and then the one with the shortest lifespan*. Using `search_exp` over the reasoning field in the step-level view, the curator accesses a record explaining that *examining another animal serves to compare the available candidates*, even though the associated observation contains no numerical lifespan information. Through `inspect_traj`, the curator also examines a source sequence that

establishes the required order of the two focus actions, while additional navigation observations expose different candidate sets. The curator separates these source-specific candidates from the reusable procedure for comparing the animals present and applying the required selection order. The synthesized skill guides the executor to identify the available animals and *focus on the longest-lived candidate before the shortest-lived one*. The frozen executor completes the task by focusing on *the tortoise egg and then the dragonfly*.

C.2.2 Failure Cases

Losing Action Ordering during Skill Synthesis. Figure 19 illustrates a failure to retain ordering constraints from navigation observations in the final skill. The ALFWorld task requires *placing two remote controls on an ottoman*. Through `inspect_traj`, the curator examines a source trajectory that handles two objects sequentially: *acquire the first, place it, and then repeat for the second*. A further `search_exp` match in the step-level view explicitly records the reasoning that *the currently held remote should be placed before searching for another*. Despite these supporting records, the synthesized skill requires *acquiring both remotes before beginning placement*. The frozen executor consequently *continues searching while holding one remote and never issues a placement action*. This case suggests that, even when the relevant ordering constraints are successfully identified during navigation, experience navigation can still fail to preserve them when multiple observations are consolidated into a single executable skill.

Failing to Recheck Target Conditions during Execution. Figure 20 presents a failure involving conditions that must be checked again after the executor revisits a product. The WebShop task requires *a purple, XX-Large, officially licensed Batman shirt*. During navigation, the curator accesses records concerning explicit option selection and recorded reasoning about uncertainty when option clicks leave observations visually similar. During target execution, the frozen executor *selects the requested color and size but subsequently leaves and reopens the product*. After its final return, it *purchases without selecting those options again*, and the purchase record contains *an empty options map*. Thus, earlier option-selection actions do not establish that the required selections remain active at the moment of purchase. This case reveals a remaining challenge in adapting knowledge as the execution state changes. In particular, more execution-state-aware navigation and knowledge management could help determine when previously satisfied conditions should be checked again before consequential actions such as purchase.

Translating Required Conditions into Execution Guidance. Figure 21 demonstrates a gap between procedural knowledge established during navigation and the actions specified in the synthesized skill. The ScienceWorld task requires *making a peanut-butter-and-jam sandwich*. By examining failed recipe trajectories and a successful mixing example, the curator identifies that *the ingredients must share a container before mixing* and retains this condition in the Navigation State. However, the final skill incorrectly proposes a mixing command, such as `mix kitchen`, as a way to *transfer the ingredients into a container*. The frozen executor *gathers the ingredients but repeatedly attempts to mix the kitchen without first transferring them into a common container*. In contrast to the successful sandwich case in Figure 17, the required condition is identified, but the final skill does not specify the actions needed to establish it. This case shows that identifying the correct precondition is not sufficient when the skill cannot determine the valid action sequence needed to satisfy it. A remaining challenge is therefore to extend navigation beyond identifying the required precondition toward finding the concrete actions needed to establish that state in the target environment.

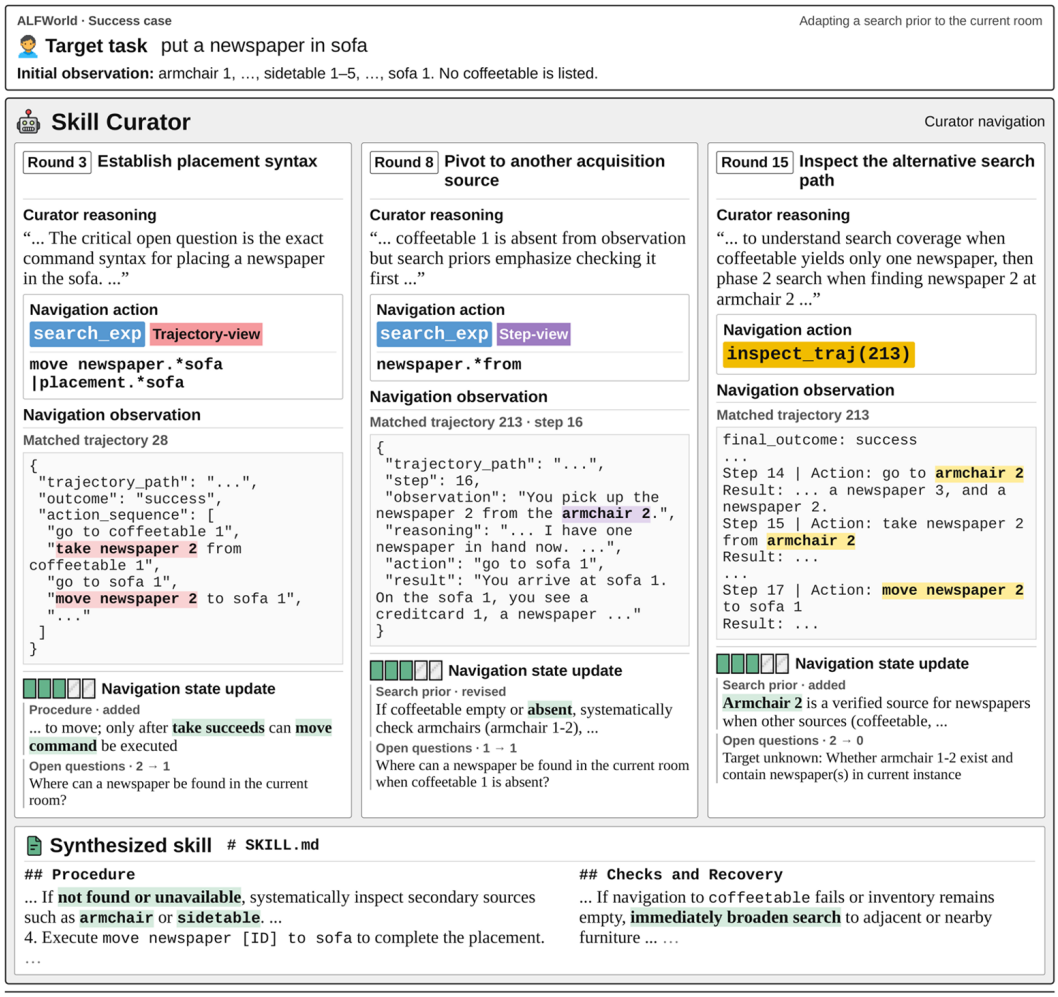


Figure 12: **Successful example (ALFWorld): distinguishing source-specific observations from target-relevant knowledge.** The absence of a coffee table in the target's initial observation redirects navigation from the placement action to alternative newspaper locations. The curator interprets a step-level `search_exp` match together with the cross-step context returned by `inspect_traj`, distinguishing a reusable search-and-placement procedure from a source-specific object location. The final skill retains a location check and fallback search as execution guidance.



Figure 13: **Successful example (ALFWorld): connecting actions to their observed results.** Through `inspect_traj`, the curator connects observations of the same soap bar before and after cleaning, while recorded reasoning in a step-level match clarifies when the cleaning action can proceed. The synthesized skill preserves cleaning before placement and retains verification of its result as an execution-time check. The frozen executor follows this ordering and completes the task.

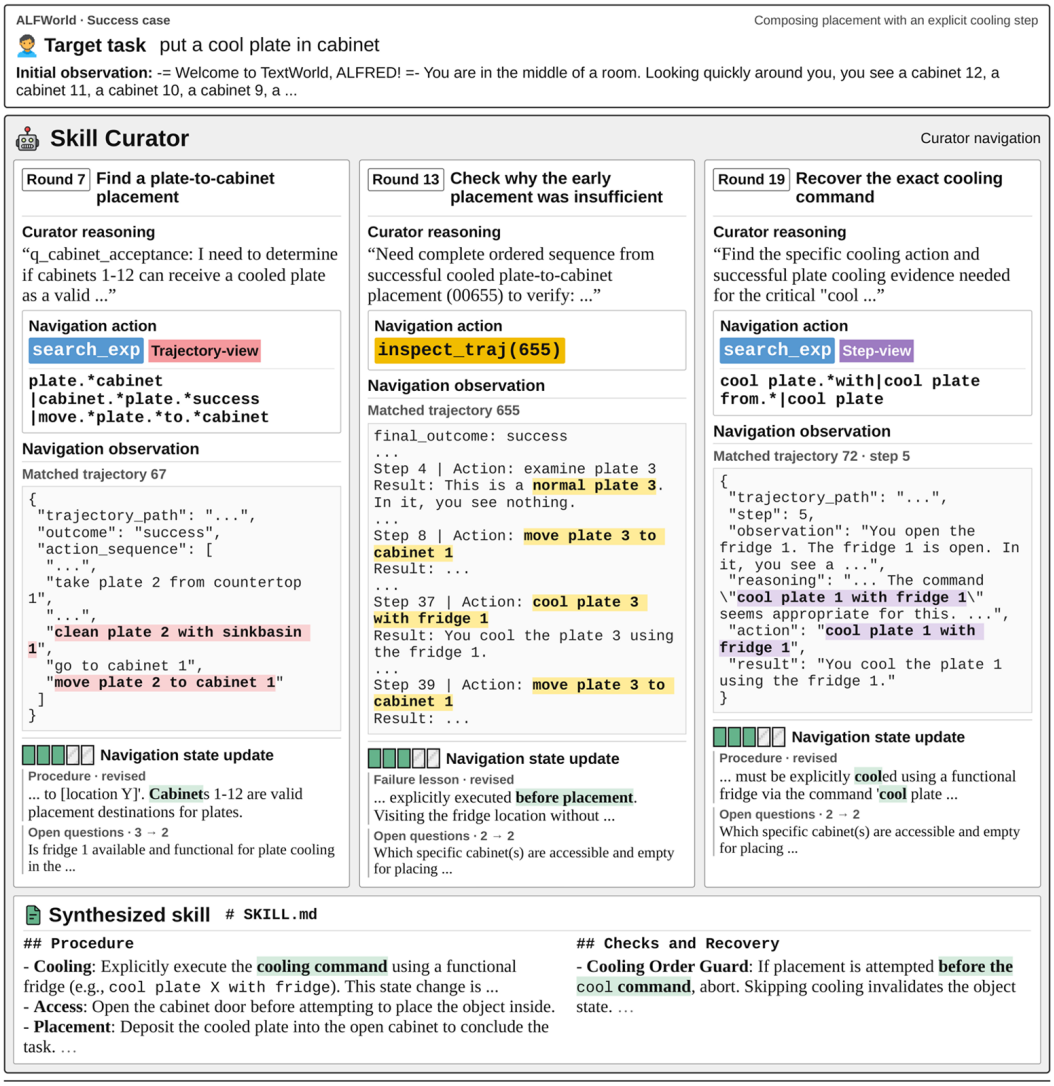


Figure 14: **Successful example (ALFWorld): preserving action ordering through cross-step context.** The cross-step context returned by `inspect_traj` distinguishes an early placement from the final placement after cooling. A step-level `search_exp` match supplies a concrete cooling command. The curator interprets these navigation observations together and retains the required cooling-before-placement ordering in the synthesized skill.

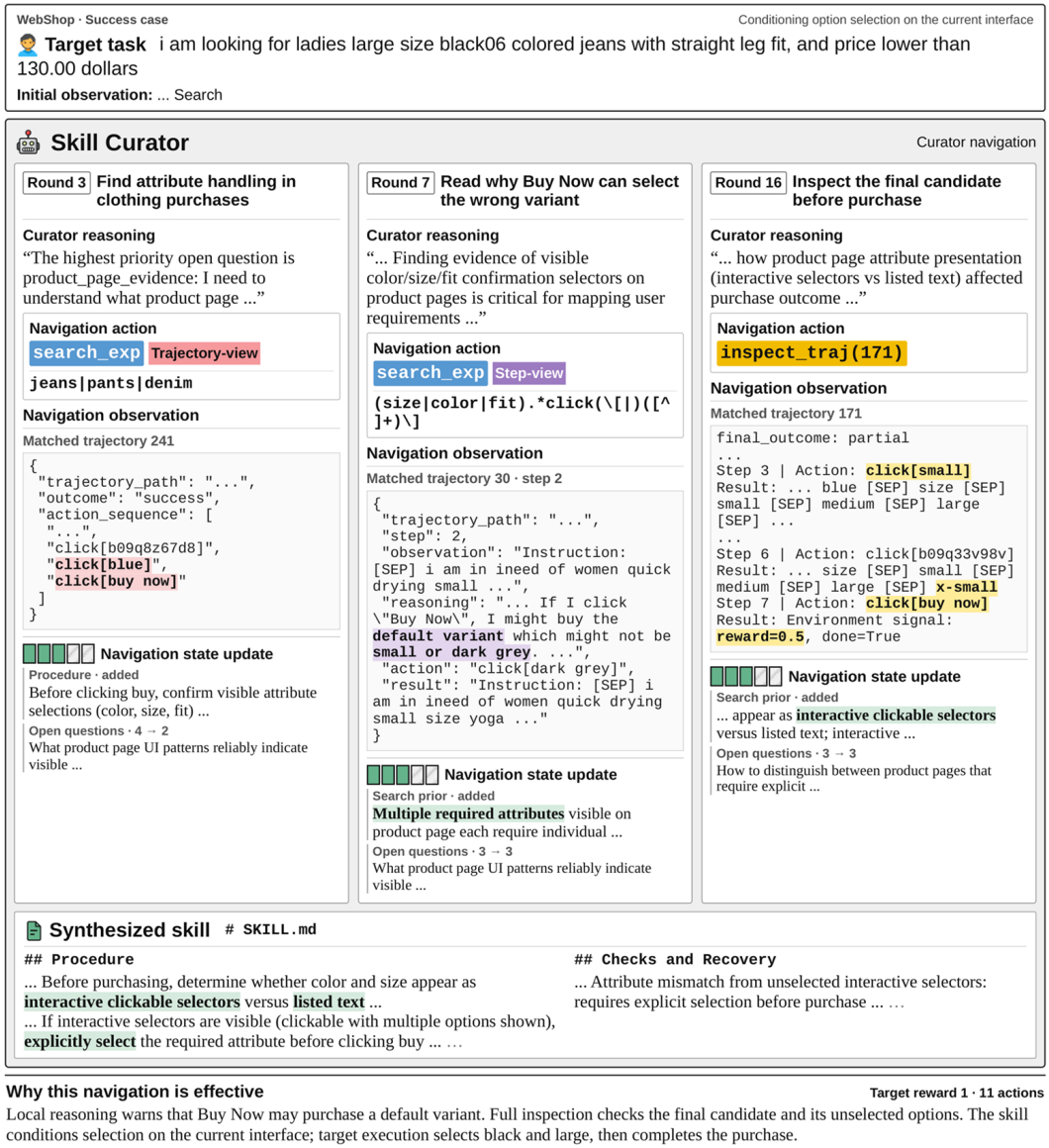


Figure 15: **Successful example (WebShop): retaining target conditions as execution-time checks.** Recorded reasoning accessed through `search_exp` warns against purchasing a default variant, while `inspect_traj` exposes a purchase without option selection after opening a new product. The synthesized skill distinguishes selectable options from descriptive attributes and retains the corresponding execution-time checks. The frozen executor selects the requested color and size before purchasing.

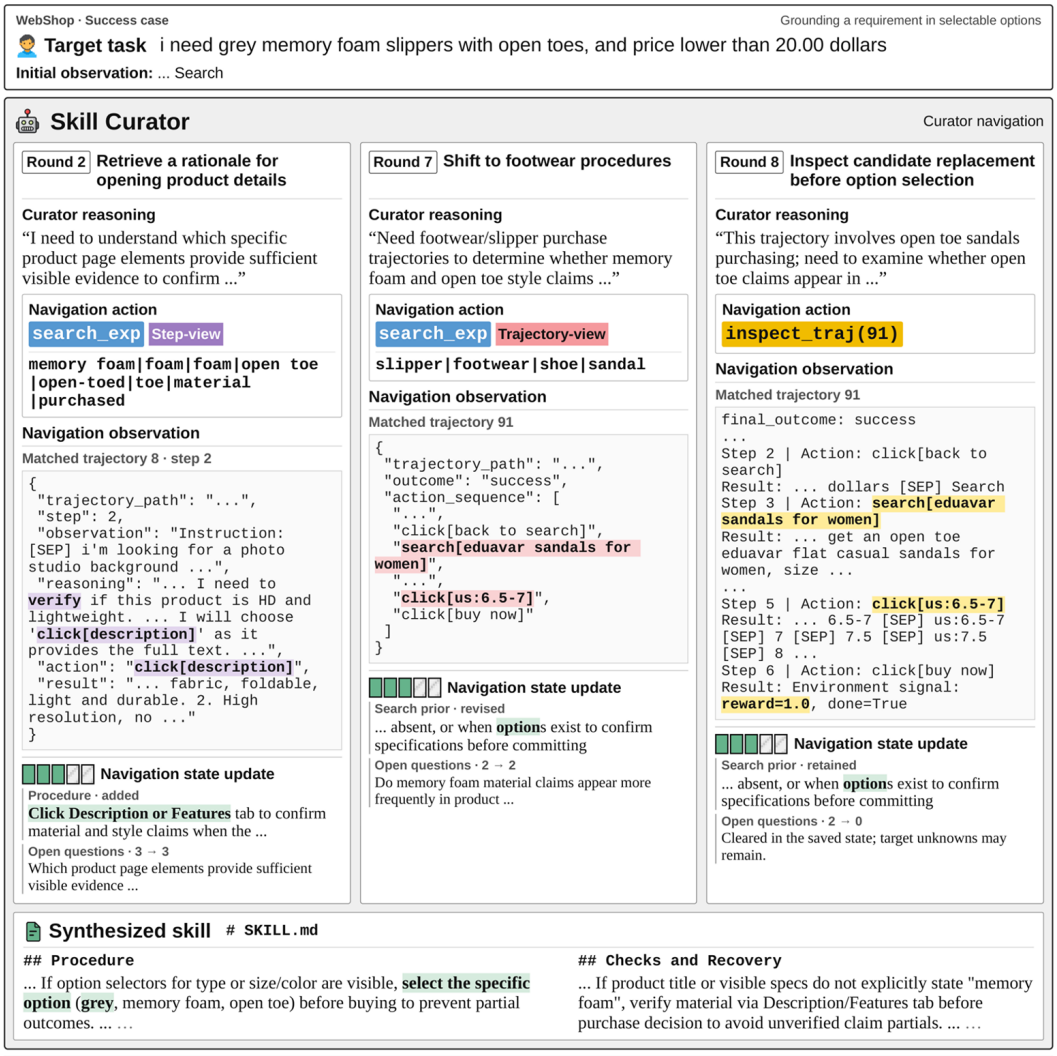


Figure 16: **Successful example (WebShop): checking target conditions against available options.** Navigation observations support execution-time checks of required attributes and recovery guidance for replacing an unsuitable product. During target execution, grey appears in the first product's description but is unavailable among its selectable colors. The frozen executor switches to another candidate and selects its grey option before purchasing.

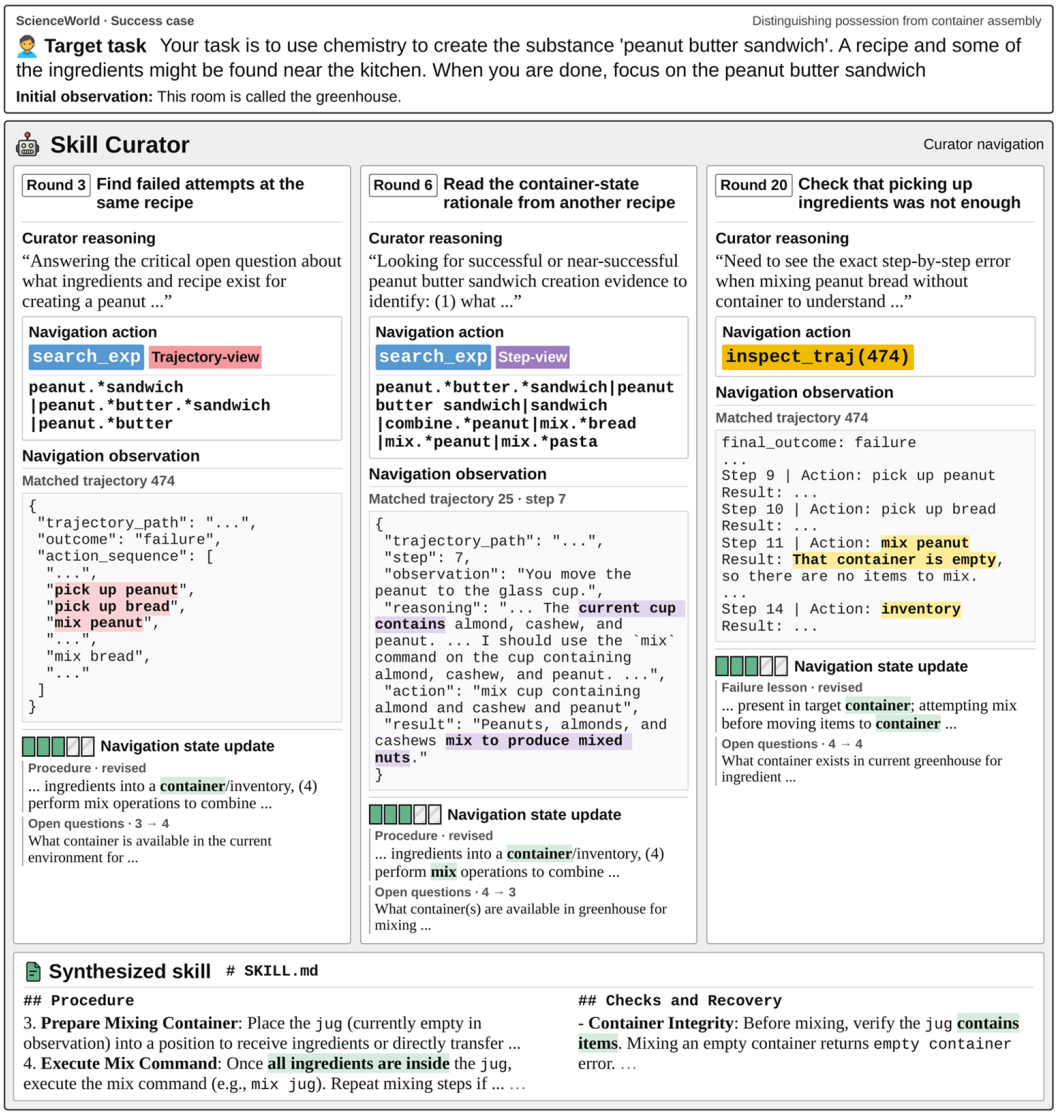


Figure 17: **Successful example (ScienceWorld): identifying required conditions across source trajectories.** The curator interprets a failed sandwich trajectory together with recorded reasoning from another recipe to identify that the ingredients must share a container before mixing. The synthesized skill turns this procedural knowledge into explicit ingredient-transfer and mixing actions. The frozen executor moves the ingredients into a cup and successfully produces the sandwich.

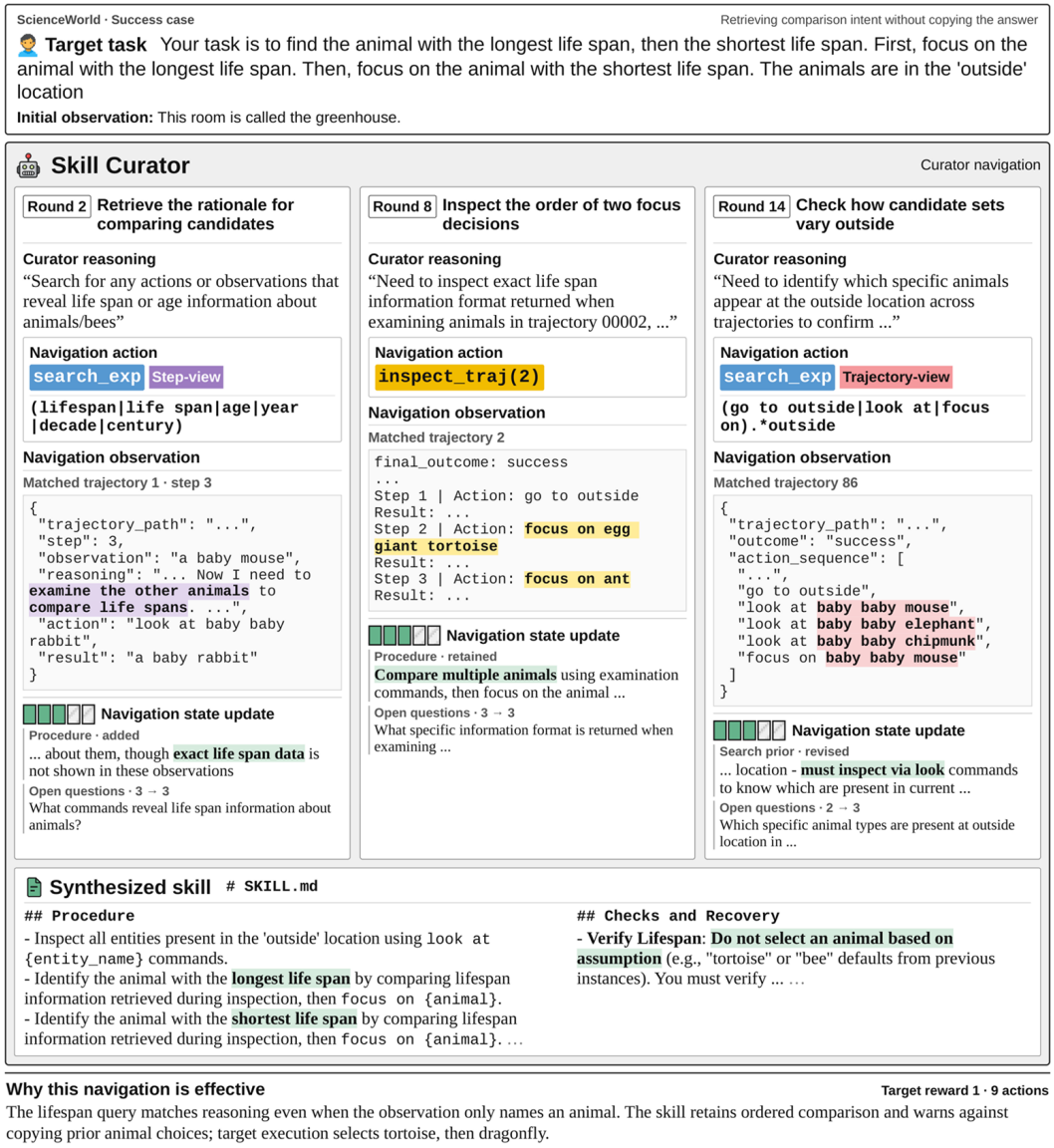


Figure 18: **Successful example (ScienceWorld): interpreting recorded reasoning for the current target.** A `search_exp` match over the reasoning field in the step-level view exposes the comparison purpose behind examining additional animals, while `inspect_traj` establishes the required focus-action order. The curator separates source-specific candidates from the reusable comparison procedure. The synthesized skill guides the frozen executor to apply this procedure to the animals available in the current environment.

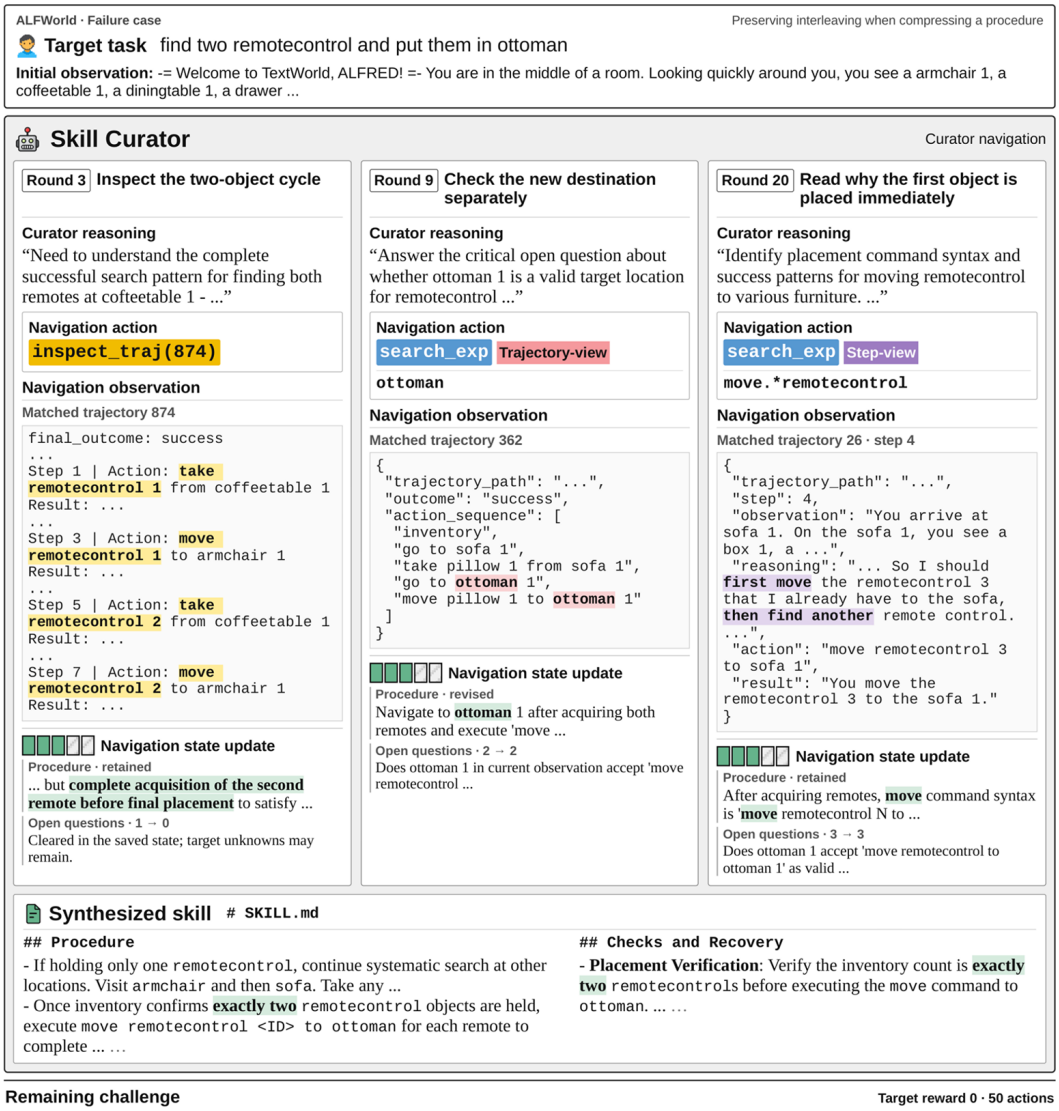


Figure 19: **Failure example (ALFWorld): losing action ordering during skill synthesis.** Navigation observations show that acquisition and placement should alternate, but the final skill requires both remote controls to be acquired before placement. The frozen executor continues searching while holding one remote and never issues a placement action.

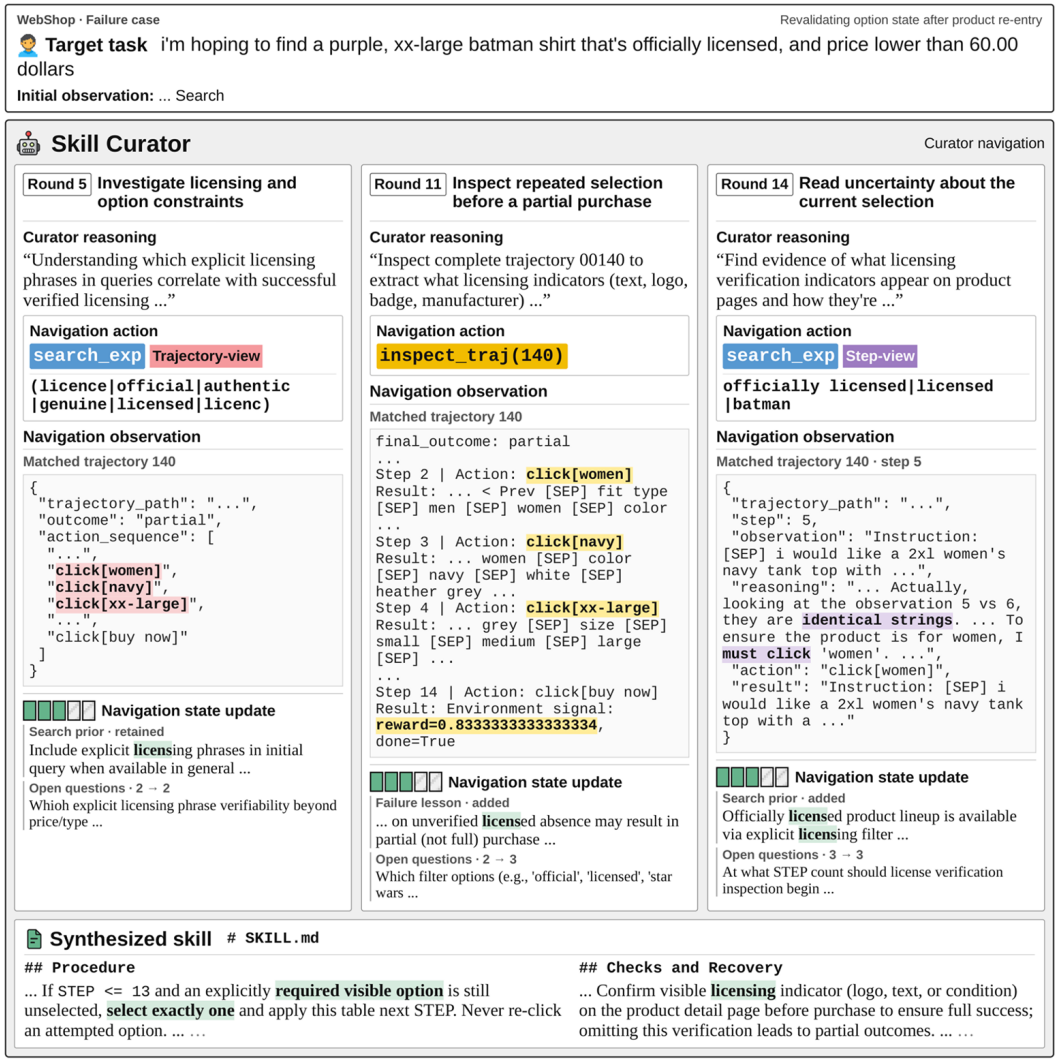


Figure 20: **Failure example (WebShop): failing to recheck target conditions during execution.** The frozen executor selects the requested color and size but later purchases after reopening the product without reselecting those options. The empty options map in the purchase record shows that earlier selections do not establish the required state at purchase time.

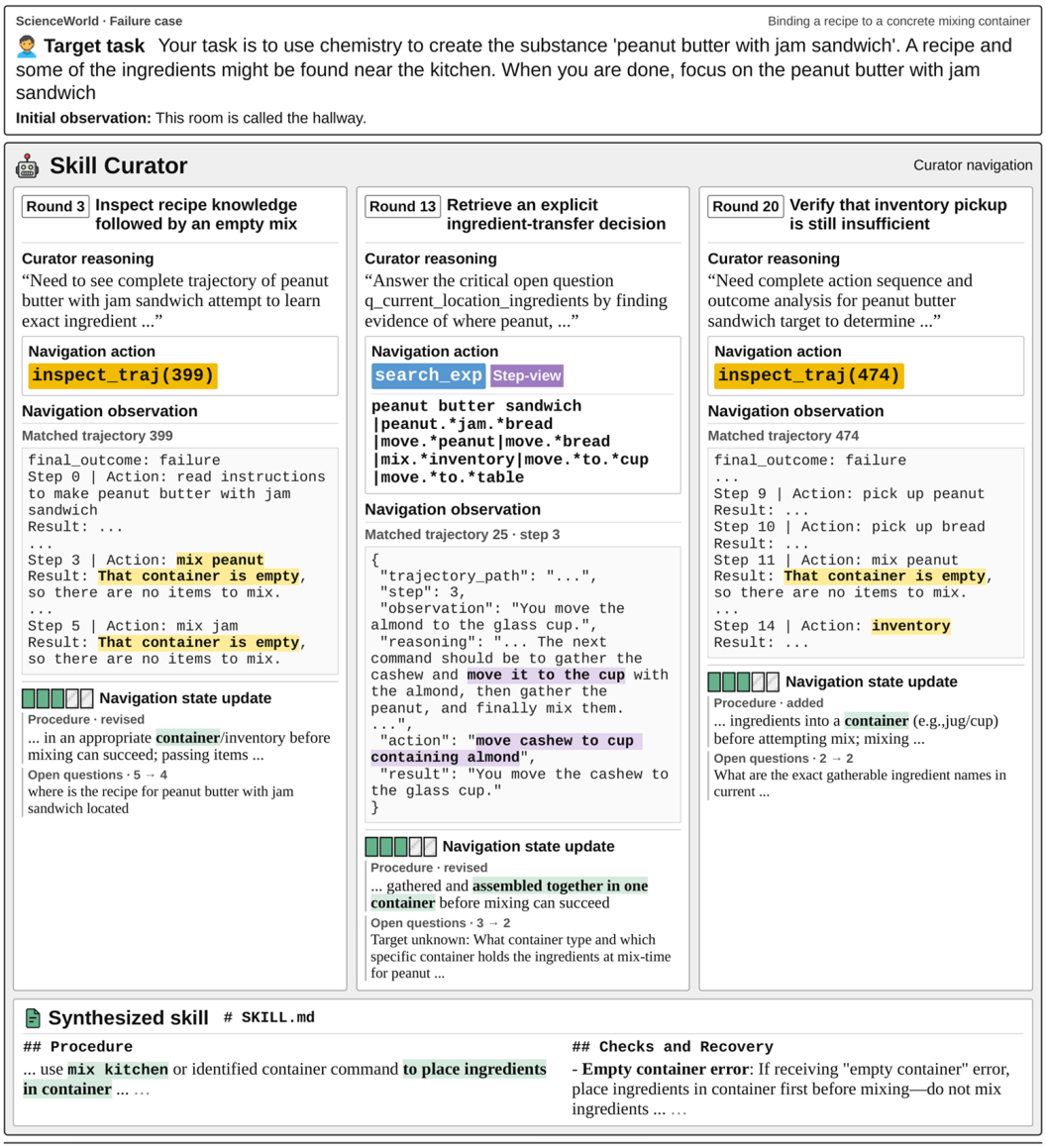


Figure 21: **Failure example (ScienceWorld): translating required conditions into execution guidance.** The Navigation State retains the requirement that ingredients share a container, but the final skill incorrectly proposes `mix kitchen` as an ingredient-transfer action. The frozen executor repeatedly attempts to mix without first moving the ingredients into a common container. The required condition is identified during navigation but is not translated into actions that establish it.

Table 7: System prompt for Navigation Action Selection.

Navigation Action Selection: system message

You investigate an outcome-aware trajectory corpus using one role-separated Navigation State as the sole persistent memory of the investigation.

The goal and its requirements come only from the current target. The remaining State distinguishes established procedures, evidence-informed search priors, failure lessons, and open questions. Choose the highest-priority open question whose answer would most improve the final executable skill. Do not draft or revise a Working Skill during search.

The corpus exposes two evidence levels. trajectory_index.jsonl contains one episode-level outcome judgment. Use it for high-level success causes, failure causes, recovery patterns, and reusable strategy. step_index.jsonl contains raw local transitions. Use it for exact preconditions, executed actions, state changes, and environment feedback. There is no fixed order between the two indexes; choose the level needed by the current question.

Exact command, ordering, or state-change claims require a step row or linked chronological trajectory. Source-model reasoning explains intent or hypothesis; observation, executed action, and result establish what locally happened.

search_exp returns complete raw rows rather than a model-authored summary. Treat returned rows as an unranked evidence set. Read an exact trajectory_path when whole-episode ordering, consequences, recovery, or grounding is needed. If returned evidence does not answer the stated question, let the State updater reject it rather than extracting a convenient but unrelated pattern.

The current target and observation are authoritative. A source episode cannot establish the current instance's location, id, or state.

Before calling a tool, state the open question it should answer and why that answer matters to the final skill. Call exactly one search_exp/inspect_traj tool, or respond exactly with FINISH when the State contains enough grounded knowledge to synthesize a useful skill and another retrieval is unlikely to improve it. Do not write skill.md on a search turn.

Table 8: User-message template for Navigation Action Selection after a corpus search. Before the first search, the active-evidence block is `Active evidence set:` followed by (`none; search the evidence level needed by the current question`). The last-action feedback block is included only when feedback is available.

Navigation Action Selection: user message

Target:
{target_task}

Current observation:
{current_observation}

Admissible commands:
{admissible_commands}

Navigation State:
{current_state}

Corpus evidence levels:
- `trajectory_index.jsonl`: one row per source episode with `trajectory_path`, `outcome`, and exact ordered `action_sequence`. Search `action_sequence` for multi-action ordering.
- `step_index.jsonl`: one row per raw transition with outcome context and exact local evidence (`observation`, `reasoning`, `action`, `result`).
- `trajectory_path`: the complete source episode, including its outcome judgement and chronological Action -> Result path. It remains the source of truth for whole-path grounding.

Navigation rules:
- Search exactly one index and only the fields needed by the current Navigation State question; every match is returned as the complete stored row.
- Compare multiple trajectory candidates through `outcome`, and `action_sequence` when the question concerns procedure ordering, search coverage, failure, recovery, or completion.
- Use a step match as complete local evidence. Read its exact `trajectory_path` only when the whole path is needed.
- Use `action_sequence` to locate an observed action ordering, not as proof of effects; validate effects in step rows or the linked raw trajectory.
- Choose the evidence level and fields from the current question; no trajectory-first or step-first order is mandatory.

Active evidence set (latest corpus search):
Search query:
{latest_search_query}
Candidate inspection status:
{candidate_inspection_status}
Returned evidence rows:
{returned_evidence_rows}

Last action feedback:
{navigation_feedback}

Choose an open Navigation State question whose answer could most improve the final executable skill, taking critical items before high or optional items. First use an active candidate that can answer it; issue a new query only when the active set cannot. Do not repeat a query unless evidence has changed.

For a tool call, provide arguments in this order:
1. `reason`: the Navigation State question to answer and why its answer matters to the final skill.
2. For search_exp, choose the evidence-level `index`, a focused `query`, the fields to search, and a bounded result limit. For inspect_traj, use an exact returned `trajectory_path`.

Respond exactly with FINISH only when the latest valid evidence is stored in the Navigation State, the State supports a useful skill, and no further corpus action is likely to improve it.

Table 9: Native function-schema excerpt for search_exp. Numeric bounds on limit are omitted.

Navigation tool: search_exp schema

```
{
  "type": "function",
  "function": {
    "name": "search_exp",
    "description": "Search one corpus JSONL index in selected evidence fields and
    return complete matching evidence rows. An exact returned trajectory_path may be
    opened with inspect_traj when the full episode is needed.",
    "parameters": {
      "type": "object",
      "properties": {
        "reason": {"type": "string", "description": "Why this query should answer
        the active question"},
        "index": {
          "type": "string",
          "enum": ["trajectory_index.jsonl", "step_index.jsonl"],
          "description": "Choose one evidence index and use only its allowed fields.
          trajectory_index.jsonl allowed fields: action_sequence, outcome.
          step_index.jsonl allowed fields: observation, reasoning, action, result."
        },
        "query": {"type": "string", "description": "Case-insensitive regular
        expression matched against the selected evidence fields"},
        "fields": {
          "type": "array",
          "items": {
            "type": "string",
            "enum": ["action_sequence", "outcome", "observation", "reasoning",
            "action", "result"]
          },
          "description": "One or more exact fields allowed for the chosen index; do
          not mix fields from different indexes",
          "minItems": 1
        },
        "limit": {"type": "integer", "description": "Maximum number of complete
        rows to return"}
      },
      "required": ["reason", "index", "query", "fields", "limit"]
    }
  }
}
```

Table 10: Native function schema for `inspect_traj`. Only `reason` and `path` are required.

Navigation tool: `inspect_traj` schema

```
{
  "type": "function",
  "function": {
    "name": "inspect_traj",
    "description": "Read one exact trajectory_path returned by corpus evidence to
    inspect the complete source episode. Step rows are already returned in full. Do
    not guess a path.",
    "parameters": {
      "type": "object",
      "properties": {
        "reason": {"type": "string", "description": "Why this evidence path should
        answer the active question"},
        "path": {"type": "string"},
        "offset": {"type": "number"},
        "limit": {"type": "number"}
      },
      "required": ["reason", "path"]
    }
  }
}
```

Table 11: System prompt for Navigation State Update.

Navigation State Update: system message

Role

Maintain the role-separated Navigation State for an outcome-aware task-time corpus investigation. You receive the current target context, current Navigation State, retrieval question, and one new corpus observation.

The State is the knowledge that a later skill may responsibly use for the current target. It is not a record of a source episode. Interpret the source evidence internally, then project only what transfers to the current target.

Update procedure

1. Check relevance

Decide whether the new observation actually answers the retrieval question. The retrieval question is only the investigation focus; it is not evidence.

If the observation does not answer the question, set evidence_relevant=false and leave the complete Navigation State unchanged. Do not manufacture a conclusion merely because a retrieval returned text.

2. Establish what happened in the source

Use each evidence type only at the level it supports:

- step_index and chronological trajectory evidence may support exact commands, local transitions, and their ordering.
- A failed attempt may support a failure lesson. It does not prove that the attempted action is a successful procedure.

Record what the evidence directly shows in source_observation. Do not yet treat that observation as knowledge about the current instance.

3. Project the source finding to the current target

For each candidate lesson, ask whether it would still be valid if the source episode's identifiers, locations, and other instance-specific details were different while the current target stayed the same.

- If it would remain valid, write that invariant or conditional lesson in transferable_rule.
- If it depends on a source-specific detail, do not copy that detail into transferable_rule. Either express the lesson using information to be checked in the current observation, or do not retain it as target-usable knowledge.
- A specific detail may appear in transferable_rule only when the target or current observation explicitly confirms it for the current instance.
- Put facts that can only be learned during current execution in target_unknown. They are not questions for further corpus search.

The target and current observation are the only authorities for current-instance facts. Source repetition or source success does not make an identifier, location, or local state true in the current instance.

4. Assign the projected lesson one role

- confirmed_procedures: a target-applicable operation or causal relation whose procedural relation is established by the evidence;
- search_priors: a target-applicable candidate to check against current observation, stated without assuming a source-instance value;
- failure_lessons: a target-applicable failure cause, guard, or recovery;
- open_questions: uncertainties still worth another corpus investigation.

Confirmed means that the procedural relation is supported. It never means that a source episode's concrete value is confirmed for the current instance.

5. Maintain the complete State

Remove an open question when the updated State no longer needs it. Retain it only when a materially different corpus retrieval could answer it. Add a new question only when the evidence exposes such a procedural gap. Do not create or retain an open question whose answer depends on future observation of the current instance.

Because you return the complete State, also repair or remove an existing item when it contains a source-specific claim that the target or current observation does not support. Do not preserve a bad projection merely because it already has an id.

6. Check consistency

Before returning, compare every transferable_rule with its target_unknown. A rule must not assert a concrete value or condition that the same item says is unknown for the current target. Rewrite or remove any such item.

Output contract

Return the complete updated State through update_search_state. Do not store, write, or reproduce a draft skill.md or a source-episode summary. Call update_search_state exactly once and emit no separate answer.

Table 12: User-message template for Navigation State Update.

Navigation State Update: user message

Navigation State Update

Inputs

Current target context

Target:
{target_task}

Current observation:
{current_observation}

Admissible commands:
{admissible_commands}

Current Navigation State

{current_state}

Retrieval question

{retrieval_question_and_reason}

New evidence type

{evidence_kind}

New corpus evidence

{navigation_observation}

Required update

1. Set evidence_relevant first.
2. If it is false, return the complete unchanged Navigation State with every evidence_used=false.
3. If it is true, return complete replacements for confirmed_procedures, search_priors, failure_lessons, and open_questions.
4. Set evidence_used=true only for entries created or changed by this evidence.

Classify what the evidence supports; do not force it into an executable rule. For every knowledge entry, use source_observation only for what happened in the source, use transferable_rule only for the target-projected lesson, and use target_unknown for facts that current execution must resolve. Ensure the rule does not assert anything declared unknown. Repair unsupported projections already present in the complete State instead of carrying them forward. An answered question may be removed, and a new question may be added when the evidence exposes a procedural uncertainty that another corpus retrieval can answer. Do not use open_questions for hidden current-instance facts. The target-derived Goal is maintained outside this update and cannot be rewritten here.

Table 13: System prompt and user-message template for Final Skill Synthesis.

Final Skill Synthesis: messages

System message

You write one final task-time skill.md from the current target context and its role-separated Navigation State. The target defines what must be achieved; the State distinguishes what corpus evidence established from what it only suggested or warned against. Return only the complete # skill.md.

User message

Final Skill Synthesis

Current target context

Target:
{target_task}

Current observation:
{current_observation}

Admissible commands:
{admissible_commands}

Final Navigation State

{final_state}

Synthesis rules

- Preserve the Goal and every target requirement.
- Use confirmed_procedures as the grounded procedural core, adapting their scope to the current target rather than copying a source episode.
- Use search_priors only as candidates for the executor to check against live observation and feedback. Do not present them as current facts.
- Turn relevant failure_lessons into concise guards or recovery guidance.
- Open questions mark what the corpus did not establish. Keep those choices conditional on runtime evidence rather than inventing answers.
- Specific details are allowed when the target or live context supports them; otherwise retain their uncertainty and function.
- Produce concise end-to-end guidance, not an evidence report, search history, or restatement of the State.

Use this shape, omitting Checks and Recovery when it adds no useful information: `# skill.md`

```
## Procedure
- Actionable guidance for the current target.

## Checks and Recovery
- Include this section only when it adds useful guidance. Otherwise omit it.
```

Return only the complete # skill.md.