LOOK BEFORE YOU LEAP: AUTONOMOUS EXPLORATION FOR LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model based agents often fail in unfamiliar environments due to premature exploitation: a tendency to act on prior knowledge before acquiring sufficient environment-specific information. We identify autonomous exploration as a critical yet underexplored capability for building adaptive agents. To formalize and evaluate this capability, we introduce Exploration Checkpoint Coverage, a verifiable metric that measures how broadly an agent discovers key states, objects, and affordances. Our systematic evaluation reveals that agents trained with standard task-oriented reinforcement learning consistently exhibit narrow and repetitive behaviors that impede downstream performance. To address this limitation, we develop a training strategy that interleaves task-execution rollouts and exploration rollouts, with each type of rollout optimized by its corresponding verifiable reward. We evaluate the resulting agents under the Explore-then-Act paradigm, which decouples information seeking from task execution: an agent first allocates an interaction budget to autonomously acquire grounded knowledge and then leverages this knowledge to solve the specific task. Our results highlight that learning to first understand a new environment is a fundamental prerequisite for building more adaptive, generalizable, and real-world-ready agents.

1 INTRODUCTION

Large language model based agents have remarkable application in realistic scenarios involving multi-step interactions with complex and diverse environments Liu et al. (2024); Zhou et al. (2023); Xie et al. (2024); Barres et al. (2025). With the advancement of Reinforcement Learning with Verifiable Rewards (RLVR), models have made substantial progress in interacting with complex environments to solve multi-step tasks (Zhang et al., 2025; Xi et al., 2025; Feng et al., 2025). Despite this progress, a key aspect remains underexplored: current RLVR approaches primarily optimize for task-completion rewards in known or static distributions, thereby encouraging instrumental behaviors aimed at solving predefined tasks. As a result, they provide limited incentive for developing the autonomous exploration capabilities required to adapt to novel, unfamiliar environments.

Lacking the intrinsic capability to explore, current LLM-based agents often exhibit a pattern of **premature exploitation**. When deployed in an unfamiliar environment, these agents tend to prematurely commit to actions derived from training-time priors, rather than systematically interacting with their surroundings to uncover hidden constraints or identify available tools (Zhou et al., 2026; Chen et al., 2026). This limitation manifests in two recurring failure modes. First, the agent often does not know where to begin, leading to aimless trial and error or confidently executing a poorly informed plan instead of proactively seeking state information. Second, the agent could misinterprets environment-specific semantics, such as specific tool arguments or UI affordances, leading to action-environment mismatches that accumulate into failures.

Prior work has primarily addressed inadequate environment understanding by preparing environment-specific knowledge before deployment, rather than cultivating autonomous exploration as a model-level capability. Several methods construct diverse task sets that broadly cover target environments, encouraging models to internalize environment-specific knowledge during training (Mai et al., 2025; SU et al., 2025; Pahuja et al., 2025); others build external knowledge bases or manuals through complex frameworks that model the environment (Zhou et al., 2026; Huang et al., 2024; Chen et al., 2024). Although these approaches can improve performance in their target environments, they rely on

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.

pre-compiling knowledge offline into model weights or external databases, leaving agents without the ability to autonomously acquire environment knowledge online. This limitation becomes increasingly critical as real-world deployment environments span diverse and dynamically evolving scenarios (He et al., 2026; Song et al., 2026; Wei et al., 2025), where it is infeasible to pre-compile all necessary knowledge. This motivates a shift from pre-deploying environment knowledge to endowing agents with the ability to acquire such knowledge themselves through autonomous online exploration.

In this work, we begin by formalizing environment exploration as an independent, measurable capability and introduce Exploration Checkpoint Coverage (ECC), a verifiable metric that quantifies how broadly an agent discovers key states, objects, and affordances in an unfamiliar environment. Using ECC, we conduct a systematic evaluation of existing models and training paradigms, revealing a notable finding: task-oriented training, including strong RLVR-style optimization for task completion, does not reliably yield autonomous exploration ability. Agents trained under these paradigms often terminate exploration prematurely, cover only a limited portion of the environment, or interact repeatedly with a narrow set of familiar states.

Motivated by this gap, we study how to equip agents with exploration capabilities by explicitly optimizing exploration during training. To achieve this, we introduce an interleaved GRPO training strategy that interleaves task-execution rollouts and exploration rollouts, with each type of rollout optimized by its corresponding verifiable reward. Task-execution rollouts are trained with task-completion rewards, whereas exploration rollouts are trained with the ECC reward to encourage broad coverage of informative states, relevant objects, and available affordances. We evaluate the resulting agents under the *Explore-then-Act* paradigm, in which an exploration-capable agent first allocates an interaction budget to autonomously acquire grounded knowledge about the environment and then uses this knowledge to solve the specified task.

We conduct experiments across four diverse interactive environments: ALFWorld (Shridhar et al., 2021), SciWorld (Xi et al., 2024), TextCraft (Xi et al., 2024), and a challenging ALFWorld variant. Our results show that task-oriented training does not reliably produce meaningful exploration. In contrast, explicitly training agents to explore improves downstream task performance and helps them make better use of an initial exploration budget. These benefits persist under matched interaction budgets and extend to environments excluded from training, suggesting that autonomous exploration can help agents acquire grounded knowledge and adapt to unfamiliar settings.

Our contributions can be summarized as follows:

- We formalize autonomous environment exploration as an independent agent capability and introduce Exploration Checkpoint Coverage (ECC), a verifiable metric for measuring exploration coverage.
- We systematically demonstrate that task-oriented training, including RLVR-style optimization, fails to reliably yield autonomous exploration. To address this limitation, we develop an effective training strategy that optimizes for exploration capabilities through mixed GRPO with an ECC reward.
- We evaluate exploration-trained agents under the Explore-then-Act paradigm, showing improved downstream performance and robustness across diverse environments and challenging variants.

2 Related Work

2.1 LLM-based Agents

Large language models (LLMs) have become foundational components in modern agent systems, owing to their strong instruction-following capabilities, robust planning abilities, and broad generalization across diverse environments. The development of LLM-based agents has evolved through several paradigms. Initial approaches primarily utilized prompt engineering (Yao et al., 2023; Shinn et al., 2023), whereas subsequent methods enhanced agent performance through supervised fine-tuning on curated trajectories (Zeng et al., 2023; Qin et al., 2024; Patil et al., 2024). Nevertheless, these methods are often constrained by the narrow scope of their training data, which limits their generalization to novel settings. More recently, reinforcement learning has emerged as a promising alternative (Zhang et al., 2025; Xi et al., 2025; Feng et al., 2025), wherein agents are optimized via policy-gradient methods based on task-completion rewards. Across all these paradigms, however, a common limitation is that agents are typically optimized solely for task reward, lacking an explicit

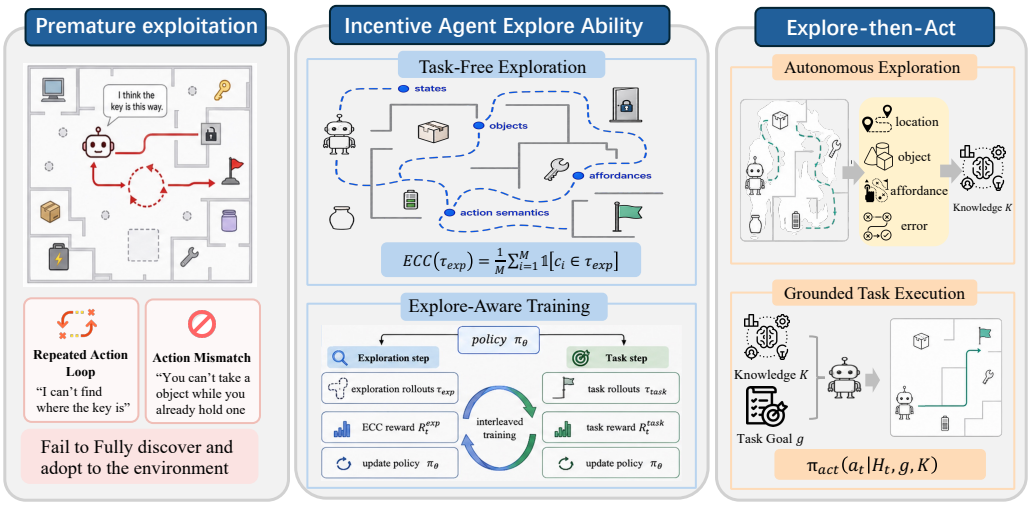


Figure 1: Task-oriented training fails to produce autonomous exploration capabilities, resulting in agents that prematurely exploit familiar patterns and acquire limited environment knowledge. We explicitly optimize for exploration through ECC rewards, enabling agents to systematically discover environment structure, objects, and affordances. The resulting Explore-then-Act paradigm decouples information seeking from task execution: agents first explore to acquire grounded knowledge, then leverage it to solve downstream tasks.

incentive for the information-seeking behavior required in unfamiliar environments. Consequently, they remain susceptible to premature exploitation when subjected to distributional shifts.

2.2 Environment Modeling for Agents

To bridge the discrepancy between the training-time priors of LLM-based agents and the dynamics of unfamiliar environments, existing literature has predominantly formulated environment modeling as an offline engineering or pre-compilation task. One prominent line of research employs heuristic or code-driven pipelines to construct external knowledge bases. For instance, frameworks such as Wall-E Zhou et al. (2026), WESE Huang et al. (2024), and AutoManual Chen et al. (2024) typically rely on traditional search algorithms (e.g., BFS or DFS) or extensive hand-crafted scripts to systematically probe the environment, utilizing the LLM exclusively to parse observations into structured graphs or rules. An alternative trajectory, exemplified by CUES Mai et al. (2025), Learn-by-Interact SU et al. (2025), and Explorer Pahuja et al. (2025), attempts to instill environment knowledge by substantially expanding the diversity of training tasks. This approach effectively compels the model to internalize the constraints of specific environments during the training phase.

3 Methodology

In this work, we investigate autonomous environment exploration as an independent capability of LLM-based agents. Rather than treating exploration as a mere byproduct of task execution, we formalize it as a goal-free, information-seeking process wherein an agent actively probes an unfamiliar environment to uncover intrinsic states, available objects, functional affordances, and action semantics. To rigorously quantify this behavior, we introduce Exploration Checkpoint Coverage (ECC) as a verifiable metric of exploration quality, and we examine methodologies to explicitly optimize agents for this capability. Finally, we demonstrate how the knowledge acquired through autonomous exploration can be systematically leveraged to enhance downstream task execution via an Explore-then-Act protocol.

3.1 Problem Formulation

We begin by formalizing the standard task setting for agents and subsequently define autonomous exploration as a distinct interaction process. The fundamental distinction lies in their operational objectives: task execution is conditioned on specific goals and driven by extrinsic rewards, whereas autonomous exploration is inherently task-agnostic and motivated by active learning and adaptation to the environment.

3.1.1 Agent environmment Interaction

We consider a standard setting where an LLM-based agent interacts with an environment $\mathcal{E}$. The agent's objective is to complete a task specified by a high-level natural language goal, g. The interaction unfolds over a sequence of steps. At each step t, the agent receives an observation $o_t \in \mathcal{O}$ from the environment, which describes the current state. Based on the history of interactions $H_t = (o_1, a_1, \ldots, o_t)$, the agent's policy π generates the next action $a_t \in \mathcal{A}$. The policy is typically conditioned on both the history and the goal: $a_t \sim \pi(\cdot|H_t, g)$.

This multi-step interaction produces a trajectory $\tau = (o_1, a_1, o_2, a_2, \ldots, o_T)$, where T is the episode length. The agent's performance is evaluated by a reward function $R(\tau, g) \in \{0, 1\}$, which assigns a reward of 1 upon task success and 0 otherwise. In this conventional paradigm, the agent follows an exploitative behavioral pattern, with each action instrumentally directed toward maximizing the task-specific reward R.

3.1.2 Autonomous Environment Exploration

In contrast to goal-directed task execution, we define *autonomous exploration* as a proactive, information-seeking process that operates independently of any specific task goal. In this mode, the agent is situated within the environment $\mathcal{E}$ without an assigned task g. Its primary objective shifts to interactively probing the surroundings to build an internal model of the environment's latent transition dynamics $\mathcal{T}(o_{t+1}|o_t, a_t)$, state space (e.g., map layout, available items), and action semantics (e.g., tool arguments, hidden constraints). We formalize this process as an exploration session, which yields a trajectory $\tau_{\text{exp}} = (o_1, a_1, \ldots, a_N, o_{N+1})$, where N denotes the allocated interaction budget. Subsequently, the agent processes τ_{exp} to synthesize a grounded knowledge summary, denoted as $\mathcal{K}$. This knowledge encapsulates the discovered environment-specific characteristics, serving to reconcile the discrepancies between the pre-existing priors of the agent and the actual properties of the environment.

3.2 Measuring Exploration with Exploration Checkpoint Coverage

To evaluate autonomous exploration independently from task success, we introduce *Exploration Checkpoint Coverage* (ECC). For each environment instance, we define a finite set of exploration checkpoints

$$\mathcal{C} = \{c_1, c_2, \ldots, c_M\}. \tag{1}$$

Each checkpoint corresponds to an environment-specific fact or affordance that a competent explorer should be able to discover. Examples include reachable locations, important objects, valid interaction targets, functional states, action-relevant affordances, or environment-specific constraints.

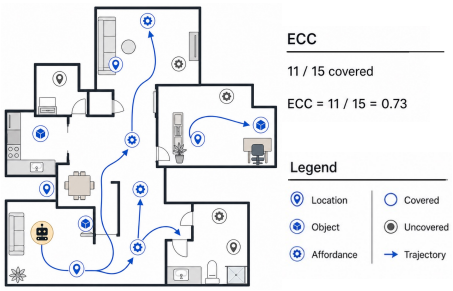


Figure 2: Illustration of Exploration Checkpoint Coverage (ECC).

Given an exploration trajectory τ_{EXP}, we define a binary indicator $\mathbb{1}[c_i \in \tau_{\text{EXP}}]$ that equals 1 if checkpoint c_i is reached, observed, or otherwise verified during exploration. ECC is computed as the fraction of checkpoints covered:

$$\text{ECC}(\tau_{\text{EXP}}) = \frac{1}{M} \sum_{i=1}^{M} \mathbb{1}[c_i \in \tau_{\text{EXP}}]. \tag{2}$$

3.3 TRAINING EXPLORATION-CAPABLE AGENTS

Having formalized autonomous exploration as a measurable capability, we now detail how to explicitly optimize for it during training. We adapt the Group Relative Policy Optimization (GRPO) framework to directly reward exploration and integrate this process into an interleaved training schedule alongside standard task-oriented optimization.

Optimizing for Exploration via GRPO. Our core strategy is to provide a direct learning signal for exploration. For an exploration-focused training step, we define the reward for a rollout τ_{EXP} as its Exploration Checkpoint Coverage:

$$R_{\text{EXP}}(\tau_{\text{EXP}}) = \text{ECC}(\tau_{\text{EXP}}). \tag{3}$$

This reward directly encourages the agent to discover more environment checkpoints. Because ECC is computed from verifiable environment interactions, this reward signal does not require a subjective, open-ended language judge.

To update the policy, we follow the GRPO procedure. For each exploration context x, which consists of an environment instance and a general exploration instruction, we sample a group of G rollouts $\{y^{(i)}\}_{i=1}^{G}$ from the current policy π_θ. We then compute the ECC reward $R^{(i)} = \text{ECC}(\tau_{\text{EXP}}^{(i)})$ for each rollout and normalize these rewards within the group to obtain relative advantages:

$$A^{(i)} = \frac{R^{(i)} - \text{mean}_j(R^{(j)})}{\text{std}_j(R^{(j)}) + \epsilon}. \tag{4}$$

The policy is then updated to increase the likelihood of trajectories with higher relative ECC, regularized by a KL penalty to maintain stability with respect to a reference model:

$$\max_\theta \ \mathbb{E}_x \left[\frac{1}{G} \sum_{i=1}^{G} A^{(i)} \log \pi_\theta(y^{(i)} \mid x) - \beta \text{KL}\left(\pi_\theta(\cdot \mid x) \,\|\, \pi_{\text{REF}}(\cdot \mid x)\right) \right]. \tag{5}$$

Interleaved Training Schedule. To develop both exploration and task-solving abilities, we employ an interleaved training schedule that alternates between exploration-focused and task-focused optimization steps. In an exploration step, we update the policy using the ECC-based GRPO objective described above. In a task-execution step, we revert to the standard GRPO setup, where rollouts are generated for specific downstream tasks and rewarded based on task completion. By alternating between these two objectives, our training process enables the agent to cultivate a robust exploration capability while simultaneously learning to apply the acquired knowledge to solve specific goals. The exploration reward provides explicit supervision for discovering environment structure, while the task reward ensures that this capability is effectively leveraged for downstream performance.

3.4 EXPLORE-THEN-ACT: DECOUPLING INFORMATION SEEKING FROM TASK EXECUTION

Existing LLM agents predominantly operate under a *direct task-execution* paradigm (Yao et al., 2023; Shinn et al., 2023), wherein every interaction is strictly conditioned on a specified goal g and evaluated solely by extrinsic task rewards. A canonical instantiation of this approach is the ReAct-style loop, which interleaves reasoning and actions under a unified goal-directed policy, formalized as

$$a_t \sim \pi_{\text{ACT}}(\cdot \mid H_t, g), \tag{6}$$

thereby lacking an explicit mechanism to allocate an interaction budget for resolving environmental uncertainties. Under the *Explore-then-Act* paradigm, information seeking is decoupled from task execution through a preliminary, goal-free exploration phase. During this initial stage, the agent is deployed in the environment $\mathcal{E}$ without a designated task. It follows an exploration policy for a fixed interaction budget of N steps, generating a trajectory $\tau_{\text{EXP}} = (o_1, a_1, \ldots, o_N, a_N, o_{N+1})$, where

$$a_t \sim \pi_{\text{EXP}}(\cdot \mid H_t). \tag{7}$$

After completing exploration, the agent synthesizes the interaction sequence into a grounded knowledge summary $\mathcal{K} = \text{SUMMARIZE}(\tau_{\text{EXP}})$ which serves as a structured natural-language artifact capturing actionable properties of the environment, including state layouts, object affordances, action preconditions, discovered constraints, and failure cases. In the subsequent goal-conditioned acting

stage, the agent tackles the downstream task using an updated policy that conditions on the current interaction history, the task goal, and the acquired knowledge:

$$a_t \sim \pi_{\text{ACT}}(\cdot \mid H_t, g, \mathcal{K}). \tag{8}$$

In practice, this decoupling is implemented by injecting the synthesized knowledge $\mathcal{K}$ into the prompt after the agent completes exploration, ensuring that downstream decisions are grounded in empirically discovered facts about the environment.

4 EXPERIMENTS

In this section, we comprehensively evaluate our proposed framework. We first introduce the experimental setup, then diagnose the innate exploration deficit in current LLMs, and finally demonstrate how explicit exploration-aware training transforms the *Explore-then-Act* (E-t-A) paradigm into consistent performance gains.

4.1 EXPERIMENTAL SETUP

LLM Backbones. To ensure the robustness of our conclusions across different model scales and families, we evaluate a diverse set of open-source backbones, including Qwen2.5-7B (Yang et al., 2024), Qwen3-4B, and LLaMA3.1-8B (Touvron et al., 2023). we also benchmark frontier proprietary models, including GPT-4.1 (OpenAI, 2023) and Claude-Opus-4.5 (Anthropic, 2025).

Environments. We evaluate our approach across three diverse environments, each requiring agents to acquire environment-specific knowledge for effective decision-making. ALFWorld (Shridhar et al., 2021) involves household navigation and object manipulation under high-level instructions. ScienceWorld (Wang et al., 2022) requires agents to discover and apply scientific rules through interactions with a complex simulated world. TextCraft (Xi et al., 2024) tests resource gathering and multi-step crafting under hidden recipe structures. Together, these environments cover embodied navigation, scientific reasoning, and compositional planning, providing a comprehensive testbed for exploration and task execution.

4.2 DIAGNOSING THE EXPLORATION DEFICIT IN CURRENT LLMS AGENTS

Before evaluating downstream task completion, we must answer a fundamental question: *How thoroughly can current LLMs autonomously discover their environment without explicit task guidance?*

Implementation Details. We deploy each LLM Agent in all three environments, imposing a maximum interaction budget of 100 steps. Crucially, the agents are **not** provided with any specific task instructions. Instead, they are prompted to freely explore and interact with the environment to gather as much useful information as possible. The exploration prompts and checkpoint construction are detailed in Appendices D.5 and E. Mean and standard deviation over three runs, each using four rollouts per task (avg@4), are reported in Appendix C.6.

Evaluation Metrics. We evaluate exploration quality using two metrics: average trajectory length (**Steps**) and Exploration Checkpoint Coverage (**ECC**, %), as defined in Section 3.2. ECC quantifies the fraction of predefined environment checkpoints discovered during the free exploration phase, including critical states, key objects, and distinct locations. We then measure the downstream utility of exploration by reporting the **Performance Gain**, denoted as $\Delta_{\text{Task}} = \text{E-t-A} - \text{Dir.}$, which captures the absolute improvement in task success rate under the Explore-then-Act paradigm (E-t-A) over direct task execution without prior exploration (Dir.). To ensure a fair comparison, all downstream task executions are performed by a fixed agent, Qwen3-4B, thereby isolating the exploration model as the only varying component.

Results. The results, presented in Table 1, reveal a significant exploration deficit in existing models. Our analysis reveals three primary findings:

- Open-source models exhibit limited intrinsic exploratory behavior. Models such as Qwen2.5-7B and Qwen3-4B achieve low average ECC scores (22.2% and 28.5%, respectively), frequently becoming trapped in repetitive loops or terminating their exploration prematurely.

Table 1: **Autonomous exploration capability in task-free environments.** We place each agent in three environments without task instructions and ask it to freely explore within a budget of 100 steps. We report average interaction turns (**Steps**), Exploration Checkpoint Coverage (**ECC**, %), and the downstream task performance change induced by Explore-then-Act, denoted as $\Delta_{\text{Task}} = \text{E-t-A} - \text{Dir.}$. The rightmost columns report the macro-average ECC and Δ_{Task} across all three environments.

Model	ALFWorld			SciWorld			TextCraft			Avg. ECC	Δ_{Task}
	Steps	ECC	Δ	Steps	ECC	Δ	Steps	ECC	Δ		
Open-Source Models											
Qwen2.5-7B	36.8	19.3	−0.3	63.4	32.1	−0.6	50.8	15.2	−1.1	22.2	−0.7
Qwen2.5-7B$_{+\text{GRPO}}$	11.8	11.2	−1.3	7.4	15.4	−0.3	8.7	11.3	−2.1	12.6	−1.2
Qwen3-4B	19.2	35.5	−2.2	87.8	29.3	−0.9	21.9	20.6	−3.4	28.5	−2.2
Qwen3-4B$_{+\text{GRPO}}$	35.5	32.8	−0.5	43.4	12.9	−1.7	14.5	10.8	−0.2	18.8	−0.8
LLaMA3.1-8B	22.5	36.8	−1.6	97.5	33.7	−2.1	65.9	22.1	−1.5	30.9	−1.7
Closed-Source Models											
GPT-4.1	24.8	52.3	+1.9	50.8	38.7	−0.2	31.4	57.6	+4.3	49.5	+2.0
Claude-Opus-4.5	61.9	**96.8**	+6.3	97.8	**89.3**	+11.7	97.3	**82.5**	+7.8	**89.5**	+8.6

- Task-oriented reinforcement learning can impede exploratory capabilities. Fine-tuning these models with task-oriented GRPO *reduces* their exploration coverage, as exemplified by Qwen3-4B, whose average ECC drops from 28.5% to 18.8%. This finding suggests that optimizing for task-completion rewards fosters narrow, instrumental policies at the expense of systematic environment mapping.
- Ineffective exploration can degrade downstream task performance. Consequently, the Explore-then-Act paradigm is not universally beneficial. When the exploration phase is shallow, repetitive, or misaligned with the environment's structure, the collected observations constitute noisy or incomplete context rather than actionable guidance.

4.3 Equipping LLM Agents with Exploration Abilities

Given that optimizing for task-specific rewards is insufficient for fostering exploration, we investigate whether reinforcement learning with explicit exploration-aware objectives can instill autonomous exploratory capabilities and thereby leverage the full potential of the *Explore-then-Act* (E-t-A) inference paradigm.

Implementation Details. We train agents with Group Relative Policy Optimization (GRPO) under three configurations aligned with the formulations in Section 3, using the training split from AgentGym (Xi et al., 2024). All models are trained for up to 300 steps on a cluster of 8 H800 GPUs. GRPO (Task-Only) serves as a conventional goal-directed baseline corresponding to Section 3.1.1, where the agent is optimized solely on task-specific rollouts. GRPO (Explore-Only), corresponding to Section 3.1.2, removes explicit task goals and optimizes the policy with the Exploration Checkpoint Coverage (ECC) reward R_{EXP}, encouraging purely information-seeking behavior. Our main method follows the interleaved training schedule in Section 3.3, alternating between task-focused and exploration-focused updates so that the agent develops both capabilities. Each configuration uses a total of 38,400 rollouts; Interleaved training replaces a portion of the task rollouts with exploration rather than increasing the allocation. Unless otherwise specified, we use a 5:1 task:exploration ratio, which provides the best performance in our ALFWorld ratio ablation (Appendix C.4). Detailed hyperparameters and training configurations are provided in Appendix D. For a fair comparison, we match the total interaction budget: up to 100 execution steps for Direct and 50 exploration plus 50 execution steps for E-t-A.

Results. Table 2 summarize the task success rates comparing Direct Execution (Dir.) with the Explore-then-Act (E-t-A) paradigm. The results yield several key observations.

Exploration-aware training improves performance in both execution paradigms. GRPO (Interleaved) consistently outperforms GRPO (Task-Only) under both Direct Execution and E-t-A. Even

Table 2: Task success rates (%) across three interactive environments for models with and without exploration-aware training. Both inference modes have a total allowance of 100 environment interactions: *Direct Execution* (Dir.; 100 execution steps) and *Explore-then-Act* (E-t-A; 50 exploration and 50 execution steps). Subscripts $\uparrow\Delta$ and $\downarrow\Delta$ mark gains and drops relative to Direct in percentage points. Exploration-aware models consistently benefit from E-t-A in this comparison, whereas task-only models often decline.

Method	ALFWorld		SciWorld		TextCraft		Avg.	
	Dir.	E-t-A	Dir.	E-t-A	Dir.	E-t-A	Dir.	E-t-A
Backbone: Qwen2.5-7B								
Zero-Shot (ReAct)	54.4	54.1 $\downarrow$0.3	4.9	4.3 $\downarrow$0.6	15.4	14.3 $\downarrow$1.1	24.9	24.2 $\downarrow$0.7
Task-Oriented Baselines								
GRPO (Task-Only)	94.4	93.2 $\downarrow$1.2	43.9	42.6 $\downarrow$1.3	66.8	66.5 $\downarrow$0.3	68.4	67.4 $\downarrow$0.9
Exploration-Aware Training								
GRPO (Explore-only)	60.2	65.3 $\uparrow$5.1	10.1	14.2 $\uparrow$4.1	46.7	49.7 $\uparrow$3.0	39.0	43.1 $\uparrow$4.1
GRPO (Interleaved)	**96.9**	**98.5** $\uparrow$1.6	**45.8**	**47.2** $\uparrow$1.4	**70.1**	**73.7** $\uparrow$3.6	**70.9**	**73.1** $\uparrow$2.2
Backbone: Qwen3-4B								
Zero-Shot (ReAct)	30.9	28.7 $\downarrow$2.2	5.2	4.3 $\downarrow$0.9	34.1	30.7 $\downarrow$3.4	23.4	21.2 $\downarrow$2.2
Task-Oriented Baselines								
GRPO (Task-Only)	84.6	84.3 $\downarrow$0.3	54.9	53.1 $\downarrow$1.8	82.2	83.1 $\uparrow$0.9	73.9	73.5 $\downarrow$0.4
Exploration-Aware Training								
GRPO (Explore-only)	49.7	54.4 $\uparrow$4.7	26.1	28.9 $\uparrow$2.8	70.3	73.1 $\uparrow$2.8	48.7	52.1 $\uparrow$3.4
GRPO (Interleaved)	**90.5**	**92.7** $\uparrow$2.2	**55.2**	**56.9** $\uparrow$1.7	**85.9**	**89.0** $\uparrow$3.1	**77.2**	**79.5** $\uparrow$2.3

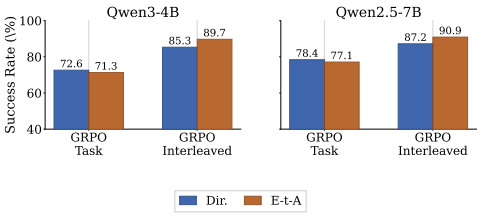


Figure 3: **Task Performance on ALFWorld Task Variants**. Exploration-aware training improves adaptation to ALFWorld variant environments, with E-t-A further enhancing the adaptability of the model.

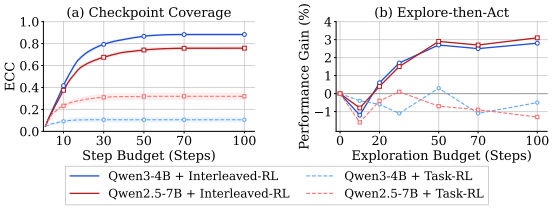


Figure 4: **Exploration efficiency and downstream task performance on ALFWorld.** (a) Environment Checkpoint Coverage (ECC) discovered within a k-step budget. (b) Explore-then-Act performance gains (%) over a Qwen3-4B executor baseline (30.9%) when using different models as explorers under a k-step exploration budget.

GRPO (Explore-Only), which is not explicitly optimized for task execution, improves over the base model. These gains suggest that exploration-aware training enhances task-solving ability even when no separate exploration phase is provided at test time.

Exploration-aware training is crucial for realizing the benefits of the E-t-A paradigm. GRPO (Interleaved) and GRPO (Explore-Only) achieve positive E-t-A gains across all three environments and both backbones. In contrast, GRPO (Task-Only) exhibits minimal or negative gains in most cases. These results suggest that exploration-aware training enables agents to convert an initial exploration budget into useful environment knowledge. With the trained Qwen3-4B policy, E-t-A also outperforms the WESE and WALL-E 2.0 adaptations under the same 50-step exploration and 50-step execution budget on ALFWorld and its Variant (Appendix C.5).

4.4 Analysis

The Intrinsic Benefit of Exploration-Aware Training. To understand why exploration-aware training improves direct execution, we examine the failure-case diagnostics in Table 3. The task-only model frequently repeats ineffective actions, whereas Interleaved training reduces repetition and encourages information seeking and recovery. This difference is consistent with the reward ablation on Qwen3-4B, ALFWorld, where ECC yields higher task success than entropy, count-based novelty, or unstructured state coverage (Appendix C.3). State coverage also reduces looping but does not improve task success over Task-Only, suggesting that the benefit depends on the information acquired rather than simply on avoiding repetition.

Table 3: Direct execution behavior diagnostics.

Metric	Task-Only	Exploration-Aware
Repeated Action Rate ↓	63.4%	**24.9%**
Loop Rate ↓	16.0%	**7.7%**
Info-Seeking Rate ↑	1.0%	**7.5%**
Error Recovery Rate ↑	0.0%	**20.1%**

Robustness to Perturbations. We further investigate robustness under environmental shifts by perturbing object locations, interaction preconditions, and distractors in ALFWorld (Appendix D.4). These perturbations preserve the task structure while testing whether agents can rediscover object locations, verify action preconditions, and distinguish targets from distractors. As shown in Figure 3, Interleaved training with E-t-A achieves the highest success rate and the smallest performance decline. The results suggest that exploration-aware training helps agents adapt when familiar environmental assumptions no longer hold.

Transfer without Target-Environment Training. We train Qwen3-4B solely on ALFWorld and evaluate it on ScienceWorld and ALFWorld-Variant without target-environment training or model adaptation. Each evaluated policy serves as both explorer and executor. Relative to Task-Only, Interleaved training raises E-t-A success from 5.1% to 14.3% on ScienceWorld and from 71.3% to 89.7% on ALFWorld-Variant. These results provide initial evidence of transfer beyond the training environment, although absolute success on ScienceWorld remains limited. Detailed protocols, Direct results, and ECC statistics are provided in Appendix C.2.

Exploration Efficiency and Its Impact on Task Performance. To disentangle exploration quality from task-execution proficiency, we compare Task-Only and Interleaved explorers under a k-step budget while fixing the executor to base Qwen3-4B. Figure 4 shows that Interleaved training achieves higher ECC and larger downstream gains, although both explorers hurt performance at $k = 10$. The association between coverage and task utility also holds across 240 ALFWorld exploration trajectories (Spearman $\rho = 0.73$; Appendix F.1), while high-coverage failures highlight the importance of retaining discovered facts in the summary and using them effectively during execution.

The gains also persist when exploration and execution share a fixed total interaction budget. With the same base executor and 100 interactions per task, allocating 50 steps to the interleaved-trained explorer and 50 to execution improves success over Direct Execution by 2.2 percentage points on ALFWorld and 8.7 on its Variant, and also outperforms two goal-conditioned attempts (Appendix C.1). This supports the value of learning how to explore, beyond simply allowing more interaction.

5 Conclusion

We identify autonomous environment exploration as a missing but essential capability for LLM agents: models optimized primarily for task completion often exhibit premature exploitation, failing to acquire the environment-specific knowledge needed in unfamiliar settings. To study this capability, we formalize exploration as an independent trainable objective and introduce Exploration Checkpoint Coverage (ECC) as a verifiable measure of how broadly agents discover key states, objects, and affordances. We further show that exploration can be instilled through trajectory distillation and interleaved GRPO with ECC-based rewards, enabling agents to first build grounded environment knowledge and then use it for downstream task execution under the Explore-then-Act paradigm. Across diverse interactive environments, our experiments show that naive exploration can hurt performance, whereas exploration-aware training consistently improves both direct task execution and Explore-then-Act performance, highlighting autonomous exploration as a practical path toward more adaptive and generalizable agents.

AI USE STATEMENT

We used generative AI tools to assist with language polishing during manuscript preparation, including improvements to grammar, clarity, and sentence flow while preserving the intended technical meaning. The authors are responsible for reviewing all AI-assisted revisions and ensuring the accuracy and integrity of the manuscript, and take full responsibility for the final content of this work.

ETHICS STATEMENT

This work investigates autonomous exploration in LLM-based agents through experiments in three simulated benchmark environments: ALFWorld, ScienceWorld, and TextCraft. Although exploration can help agents acquire environment-specific knowledge, exploratory actions may themselves have unintended or irreversible consequences in real-world settings. Improved benchmark performance therefore does not establish the safety of unrestricted exploration. Deployment beyond these environments should restrict exploration to authorized actions, protect sensitive information encountered during interaction, and incorporate human oversight for consequential decisions. Any reuse or release of interaction data, models, and software should respect applicable licenses and terms of use.

REPRODUCIBILITY STATEMENT

To support reproducibility, Section 3 defines ECC and describes the training procedure and Explore-then-Act protocol, while Section 4 identifies the evaluated models and environments. Appendix D provides training hyperparameters, rollout accounting, explorer–executor configurations, statistical reporting, ALFWorld-Variant construction, and the exploration prompt. Appendix E describes checkpoint construction and verification, including the distinction between policy inputs and privileged evaluation signals. Appendix C reports additional experiments and controlled comparisons, with expanded mean and standard-deviation statistics in Appendix C.6.

REFERENCES

Anthropic. Introducing claude opus 4.5. https://www.anthropic.com/news/claude-opus-4-5, November 2025. Accessed: 2026-04-29.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation, 2016. URL https://arxiv.org/abs/1606.01868.

Arthur Chen, Zuxin Liu, Jianguo Zhang, Akshara Prabhakar, Zhiwei Liu, Shelby Heinecke, Silvio Savarese, Victor Zhong, and Caiming Xiong. Test-time adaptation for llm agents via environment interaction, 2026. URL https://arxiv.org/abs/2511.04847.

Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. Automanual: Generating instruction manuals by LLM agents via interactive environmental learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=Pwl9n4zlf5.

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

Wei He, Yueqing Sun, Hongyan Hao, Xueyuan Hao, Zhikang Xia, Qi GU, Hui Su, and Xunliang Cai. Vitabench: Benchmarking LLM agents with versatile interactive tasks in real-world applications. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=rtcX9qOBaz.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. Wese: Weak exploration to strong exploitation for llm agents, 2024. URL https://arxiv.org/abs/2404.07456.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating LLMs as agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=zAdUB0aCTQ.

Shinji Mai, Yunpeng Zhai, Ziqian Chen, Cheng Chen, Anni Zou, Shuchang Tao, Zhaoyang Liu, and Bolin Ding. Cues: A curiosity-driven and environment-grounded synthesis framework for agentic rl, December 2025. URL https://arxiv.org/abs/2512.01311.

OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023.

Vardaan Pahuja, Yadong Lu, Corby Rosset, Boyu Gou, Arindam Mitra, Spencer Whitehead, Yu Su, and Ahmed Hassan Awadallah. Explorer: Scaling exploration-driven web trajectory synthesis for multimodal web agents. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 6300–6323, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.326. URL https://aclanthology.org/2025.findings-acl.326/.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=tBRNC6YemY.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=dHng2O0Jjr.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre 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.

Xiaoshuai Song, Haofei Chang, Guanting Dong, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Envscaler: Scaling tool-interactive environments for llm agent via programmatic synthesis, 2026. URL https://arxiv.org/abs/2601.05808.

Hongjin SU, Ruoxi Sun, Jinsung Yoon, Pengcheng Yin, Tao Yu, and Sercan O Arik. Learn-by-interact: A data-centric framework for self-adaptive agents in realistic environments. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=3UKOzGWCVY.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023.

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/.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents, 2025. URL https://arxiv.org/abs/2504.12516.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, Songyang Gao, Lu Chen, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. Agentgym: Evolving large language model-based agents across diverse environments, 2024.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Honglin Guo, Jiaqi Liu, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, Wei He, Yiwen Ding, Guanyu Li, Zehui Chen, Zhengyin Du, Xuesong Yao, Yufei Xu, Jiecao Chen, Tao Gui, Zuxuan Wu, Qi Zhang, Xuanjing Huang, and Yu-Gang Jiang. Agentgym-rl: Training llm agents for long-horizon decision making through multi-turn reinforcement learning, 2025. URL https://arxiv.org/abs/2509.08755.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 2024. doi: 10.48550/ARXIV.2404.07972. URL https://doi.org/10.48550/arXiv.2404.07972.

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

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, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agenttuning: Enabling generalized agent abilities for llms, 2023.

Hanchen Zhang, Xiao Liu, Bowen Lv, Xueqiao Sun, Bohao Jing, Iat Long Iong, Zhenyu Hou, Zehan Qi, Hanyu Lai, Yifan Xu, Rui Lu, Hongning Wang, Jie Tang, and Yuxiao Dong. Agentrl: Scaling agentic reinforcement learning with a multi-turn, multi-task framework, 2025. URL https://arxiv.org/abs/2510.04206.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. *ArXiv*, abs/2307.13854, 2023. URL https://api.semanticscholar.org/CorpusID:260164780.

Siyu Zhou, Tianyi Zhou, Yijun Yang, Guodong Long, Deheng Ye, Jing Jiang, and Chengqi Zhang. WALL-E 2.0: World alignment by neurosymbolic learning improves world model-based LLM agents, 2025. URL https://arxiv.org/abs/2504.15785.

Siyu Zhou, Tianyi Zhou, Yijun Yang, Guodong Long, Deheng Ye, Jing Jiang, and Chengqi Zhang. WALL-e: World alignment by neurosymbolic learning improves world model-based LLM agents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=DorAT49sxj.

A LIMITATIONS AND FUTURE WORK.

Our work takes an initial step toward incentivizing autonomous exploration abilities in LLM-based agents. Looking ahead, we consider following potential limitations and future work. First, this work studies exploration primarily as an initial phase before task execution, providing a clean and controllable setting for isolating, measuring, and training exploration ability. However, real-world environments are often too large and complex to be fully explored upfront. Extending our framework to dynamic, task-conditioned exploration is therefore an important direction for future work. Second, our experiments focus on text-based interactive environments, where language provides clear affordances and enables verifiable coverage metrics for studying exploration. Extending exploration to more open-ended multimodal environments is another promising direction. Overall, we view this work as a foundation for a broader research agenda on exploration-capable agents. Future progress on dynamic and multimodal exploration may further enable agents to acquire grounded environment knowledge online, adapt to evolving conditions, and operate robustly in realistic deployment settings.

B BOARDER IMPACT

This work formalizes autonomous exploration as a measurable capability for LLM agents and introduces training strategies to improve it. By enabling agents to acquire grounded environment knowledge online, our methods may benefit applications such as virtual assistants, web automation, educational tools, and embodied AI systems, especially in unfamiliar or changing environments. While this work is methodological and does not directly deploy real-world agents, stronger exploration ability may indirectly increase agent autonomy. Agents that better discover tools, rules, and affordances could also interact with environments in unexpected ways. Therefore, practical deployment should include appropriate safeguards, such as permission control, monitoring, constrained environments, and human oversight for high-stakes actions. Overall, this work provides evaluation and training tools for building more adaptive and generalizable LLM agents, while highlighting the need for responsible use as autonomous exploration capabilities improve.

C ADDITIONAL QUANTITATIVE RESULTS

These experiments complement the main results with expanded statistics, inference-budget controls, transfer, and training and framework comparisons. The shared protocols are in Appendix D; experiment-specific settings are given with each result.

C.1 EXPLORATION UNDER A FIXED INTERACTION BUDGET

Setup. All rows in Table 4 use the same base Qwen3-4B executor and a maximum total allowance of 100 environment interactions. In E-t-A rows, the explorer is a Qwen3-4B model with the indicated training objective; only its exploration output is supplied to the fixed executor. This design separates exploration quality from the task-solving ability acquired by the explorer during training.

Table 4: Fixed-executor comparison under a 100-interaction allowance. Success is in percent. The step column reproduces the reported average total interaction statistic, including exploration for E-t-A. The reporting does not specify the handling of failed episodes, so these values should not be interpreted as an unconditional expected cost.

Setting / explorer	ALFWorld		ALFWorld-Variant	
	Success	Steps	Success	Steps
Direct (100 Act)	30.9 ± 1.4	77.4 ± 3.6	21.8 ± 1.4	82.7 ± 4.4
Two attempts (50 Act + 50 Act)	31.3 ± 0.9	80.6 ± 2.8	20.3 ± 2.1	87.9 ± 3.3
E-t-A: 50 Explore + 50 Act; fixed base Qwen3-4B executor				
Base explorer	27.9 ± 1.6	78.3 ± 3.9	19.1 ± 0.8	84.2 ± 4.5
Task-Only explorer	30.5 ± 1.3	76.7 ± 3.2	20.6 ± 1.3	80.5 ± 3.8
Explore-Only explorer	32.2 ± 1.5	73.5 ± 2.8	28.4 ± 1.6	77.1 ± 3.4
Interleaved explorer	33.1 ± 1.7	70.4 ± 2.5	30.5 ± 1.2	75.4 ± 3.1

Results. The interleaved-trained explorer improves success over Direct Execution by 2.2 pp on ALFWorld and 8.7 pp on ALFWorld-Variant. It also exceeds the two-attempt baseline by 1.8 and 10.2 pp, respectively. The corresponding reported step statistic decreases by 7.0 and 7.3 relative to Direct Execution. In contrast, base-model exploration reduces success, while task-only exploration provides little benefit. Within this fixed-executor setting, the advantage therefore depends on the exploration policy rather than on simply allocating more interactions.

Distinguishing the protocols. The 30.5% Variant success rate here uses an interleaved-trained explorer with a base executor. The 19.1% rate for the base model in the transfer experiment uses the base model for both roles. The 89.7% rate in that experiment uses the interleaved-trained model for both roles. These values answer different questions and should not be compared as if only the inference budget had changed.

Accounting limits and possible reuse. The reported experiment charges exploration separately for every task. If a summary remains valid across m tasks in the same environment, the nominal interaction overhead of a 50-step exploration allowance would be $50/m$ per task. This is an accounting possibility, not a measured amortization result: changes to object state, layout, or action preconditions can invalidate the summary. A full deployment-cost comparison would additionally measure summary generation, context processing, failed attempts, and any auxiliary extraction or planning calls.

C.2 Transfer without Target-Environment Training

Training and evaluation separation. For this experiment, Qwen3-4B is trained solely in the original ALFWorld environment and evaluated on ScienceWorld and ALFWorld-Variant. No target-environment training data or model adaptation is used. The evaluated model performs both exploration and task execution. This experiment differs from the environment-specific training comparison in Table 2 and from the fixed-executor control in Table 4.

Table 5: Transfer from ALFWorld with no target-environment training or model adaptation. Dir. and E-t-A are success rates (%); ECC is diagnostic checkpoint coverage (%). Target-environment checkpoints are used for evaluation, not supplied to the policy.

Qwen3-4B policy	ScienceWorld			ALFWorld-Variant		
	Dir.	E-t-A	ECC	Dir.	E-t-A	ECC
Base	5.2 ± 0.6	4.3 ± 0.5	29.3 ± 3.7	21.8 ± 1.2	19.1 ± 0.8	32.3 ± 4.7
Task-Only	6.3 ± 1.1	5.1 ± 0.8	10.6 ± 5.1	72.6 ± 1.6	71.3 ± 0.9	19.7 ± 3.2
Interleaved	8.7 ± 0.5	14.3 ± 1.3	67.3 ± 4.9	85.3 ± 0.9	89.7 ± 1.3	81.9 ± 4.1

Results and their scope. Relative to Task-Only training, Interleaved training improves ScienceWorld success by 2.4 pp under Direct Execution and 9.2 pp under E-t-A. On ALFWorld-Variant the

corresponding differences are 12.7 and 18.4 pp. ScienceWorld therefore provides evidence beyond an ALFWorld perturbation, although the absolute transferred E-t-A success rate remains 14.3%. The result supports transfer of useful exploratory behavior under this protocol, without establishing broad cross-domain generalization.

C.3 Alternative Exploration Rewards

Setup. We compare alternative exploration rewards within the Qwen3-4B interleaved training framework on ALFWorld. The supplied experimental description holds the training configuration and rollout allocation fixed while changing the exploration reward. State coverage measures the episode-level fraction of distinct textual observation signatures visited. Count-based novelty assigns a per-step bonus $1/\sqrt{N(s)}$, where $N(s)$ is the observation-visitation count, following the general count-based exploration principle (Bellemare et al., 2016). The entropy alternative adds a per-step action-entropy bonus. ECC measures bounded episode-level coverage of the structured checkpoint set. Task-Only is the reference without exploration rollouts.

Table 6: Alternative exploration rewards on Qwen3-4B, ALFWorld. Success rates and the reported action-loop ratio are percentages. These are instantiated reward alternatives, not reproductions of every method in the corresponding exploration literature.

Exploration reward	Dir.	E-t-A	Action-loop ratio ↓
Task-Only (reference)	84.6 ± 1.2	84.3 ± 1.7	16.0 ± 3.9
Entropy bonus	85.3 ± 2.4	84.1 ± 1.9	15.4 ± 4.6
Count-based novelty	64.7 ± 2.1	65.3 ± 1.1	34.6 ± 5.3
State coverage	79.2 ± 1.8	80.4 ± 1.5	11.2 ± 3.8
ECC	90.5 ± 1.1	92.7 ± 1.2	7.7 ± 3.4

Interpretation. ECC improves Direct and E-t-A success over Task-Only by 5.9 and 8.4 pp, while reducing the action-loop ratio from 16.0% to 7.7%. State coverage also reduces loops but decreases success, so less repetition alone does not explain the benefit. Count-based novelty produces more loops, possibly because revisits retain positive reward. These results favor the tested ECC implementation on ALFWorld; they do not isolate reward scaling or credit assignment as the cause, or establish that generic exploration rewards are generally ineffective.

C.4 Sensitivity to the Task-Exploration Ratio

Setup. We examine how the balance between task-execution and exploration rollouts affects GRPO training. All runs use Qwen3-4B on ALFWorld with the same training budget and hyperparameters, while varying only the composition of rollouts in each training batch. We include two endpoint baselines, Task-Only and Explore-Only, and six mixed task:exploration ratios: 1:10, 1:5, 1:3, 3:1, 5:1, and 10:1. Each trained policy is evaluated under both Direct Execution (Dir.) and Explore-then-Act (E-t-A), where the latter provides an exploration phase before task execution. Results are reported in Table 7.

Table 7: Sensitivity to task:exploration ratio in GRPO training. Evaluated on Qwen3-4B, ALFWorld. Dir. = Direct Execution success rate (%), E-t-A = Explore-then-Act success rate (%).

Task-Exploration Ratio	**Dir. (%)**	**E-t-A (%)**
Explore-Only	49.7 ± 2.4	54.4 ± 2.2
1:10	60.9 ± 2.0	64.3 ± 1.9
1:5	79.5 ± 1.6	73.2 ± 1.8
1:3	82.7 ± 1.4	83.9 ± 1.5
3:1	86.1 ± 1.3	88.1 ± 1.4
5:1	$\mathbf{90.5 \pm 1.1}$	$\mathbf{92.7 \pm 1.2}$
10:1	87.9 ± 1.3	91.0 ± 1.3
Task-Only	84.6 ± 1.2	84.3 ± 1.7

Analysis. Task-Only training preserves strong direct task performance but provides almost no benefit from the additional exploration phase, while Explore-Only training substantially underperforms because it lacks enough task-completion signal. Mixed training improves over both endpoints once task rollouts are sufficiently represented. Performance rises as the ratio shifts from exploration-heavy settings toward task-heavy settings, with 5:1 achieving the best Direct and E-t-A success rates. Increasing the task share further to 10:1 slightly reduces performance, suggesting that too little exploration weakens the transferable environment knowledge used by E-t-A. We therefore use 5:1 as the default ratio in the main GRPO experiments, as it provides the best empirical trade-off between task optimization and exploration-aware behavior.

C.5 Exploration Framework Comparisons

Setup and adaptation. WESE (Huang et al., 2024) and WALL-E 2.0 (Zhou et al., 2025) are adapted to a single episode with 50 exploration and 50 execution steps. Each Qwen3-4B policy serves as both explorer and executor. This matches the environment-interaction allowance across exploration methods, but does not establish equal costs for auxiliary extraction, symbolic processing, or planning. The adaptations do not reproduce the full systems at their original resource allocations.

Table 8: Adapted exploration frameworks. All entries are task success rates (%). Direct is the reported no-exploration reference; the equal-total-interaction Direct control is separately specified in Table 4.

Environment	Policy	Direct	WESE	WALL-E 2.0	E-t-A
ALFWorld	Base	30.9 ± 1.4	27.1 ± 0.7	29.3 ± 1.1	28.7 ± 1.6
	Interleaved	90.5 ± 1.0	90.7 ± 0.8	87.6 ± 1.5	92.7 ± 1.2
Variant	Base	21.8 ± 1.2	18.9 ± 1.3	18.3 ± 2.1	19.1 ± 0.8
	Interleaved	85.3 ± 0.9	86.3 ± 1.4	75.3 ± 0.5	89.7 ± 1.3

Results and comparison limits. No exploration approach exceeds Direct with the base policy. With the trained policy, E-t-A exceeds the WESE adaptation by 2.0 pp on ALFWorld and 3.4 pp on Variant. This supports the tested combination of learned exploration and E-t-A, without establishing superiority over the original systems. AutoManual (Chen et al., 2024) was not evaluated here; its multi-agent, code-based rule accumulation requires a comparison accounting for experience across episodes.

Relation to policy learning. Exploration followed by knowledge reuse is already present in prior agent systems. WESE separates exploration from exploitation and stores acquired information in a knowledge graph (Huang et al., 2024); AutoManual updates environmental rules through interaction (Chen et al., 2024); WALL-E 2.0 aligns a world model using knowledge extracted from exploration trajectories (Zhou et al., 2025). Our contribution is to diagnose and explicitly optimize exploration behavior in the policy using a verifiable reward, and to test the resulting behavior under controlled explorer–executor configurations. It does not require a new policy-gradient algorithm. Learned exploration and external memory or world-model scaffolds may be complementary.

C.6 Expanded Main-Result Statistics

Table 9: Exploration diagnosis over three runs (avg@4). Each cell reports the mean with ± standard deviation below. The exploration allowance is 100 steps. ECC is in percent; Δ_{Task} is E-t-A minus Direct in percentage points for a fixed base Qwen3-4B executor.

Explorer	ALFWorld			ScienceWorld			TextCraft		
	Steps	ECC (%)	Δ (pp)	Steps	ECC (%)	Δ (pp)	Steps	ECC (%)	Δ (pp)
Qwen2.5-7B	36.8 ±4.2	19.3 ±2.8	−0.3 ±0.5	63.4 ±6.5	32.1 ±3.9	−0.6 ±0.3	50.8 ±5.4	15.2 ±2.4	−1.1 ±0.4
Qwen2.5-7B + Task-Only	11.8 ±1.9	11.2 ±2.2	−1.3 ±0.3	7.4 ±1.5	15.4 ±3.1	−0.3 ±0.6	8.7 ±1.7	11.3 ±2.6	−2.1 ±0.5
Qwen3-4B	19.2 ±3.1	35.5 ±4.3	−2.2 ±0.4	87.8 ±7.9	29.3 ±3.7	−0.9 ±0.3	21.9 ±3.4	20.6 ±2.9	−3.4 ±0.6
Qwen3-4B + Task-Only	35.5 ±4.0	32.8 ±4.1	−0.5 ±0.2	43.4 ±5.2	12.9 ±2.3	−1.7 ±0.4	14.5 ±2.6	10.8 ±2.0	−0.2 ±0.2
LLaMA3.1-8B	22.5 ±3.3	36.8 ±4.6	−1.6 ±0.4	97.5 ±1.4	33.7 ±4.0	−2.1 ±0.5	65.9 ±6.3	22.1 ±3.2	−1.5 ±0.4
GPT-4.1	24.8 ±3.5	52.3 ±4.9	+1.9 ±0.4	50.8 ±5.7	38.7 ±4.2	−0.2 ±0.3	31.4 ±3.9	57.6 ±5.3	+4.3 ±0.6
Claude-Opus-4.5	61.9 ±5.5	96.8 ±1.8	+6.3 ±0.7	97.8 ±1.6	89.3 ±3.4	+11.7 ±1.1	97.3 ±2.3	82.5 ±4.1	+7.8 ±0.8

Table 10: Expanded task-performance statistics over three runs (avg@4). Each cell reports the mean with ± standard deviation below. The evaluated policy serves as both explorer and executor. Δ is the within-policy E-t-A change in percentage points. This is the main performance protocol, not the fixed-executor, 100-interaction control.

Training	ALFWorld			ScienceWorld			TextCraft		
	Dir. (%)	E-t-A (%)	Δ (pp)	Dir. (%)	E-t-A (%)	Δ (pp)	Dir. (%)	E-t-A (%)	Δ (pp)
Backbone: Qwen2.5-7B									
Zero-shot	54.4 ±1.5	54.1 ±1.7	−0.3 ±0.2	4.9 ±0.6	4.3 ±0.5	−0.6 ±0.3	15.4 ±1.1	14.3 ±1.2	−1.1 ±0.4
Task-Only	94.4 ±0.9	93.2 ±1.1	−1.2 ±0.4	43.9 ±1.6	42.6 ±1.8	−1.3 ±0.5	66.8 ±1.4	66.5 ±1.5	−0.3 ±0.3
Explore-Only	60.2 ±1.7	65.3 ±1.5	+5.1 ±0.6	10.1 ±0.9	14.2 ±1.0	+4.1 ±0.5	46.7 ±1.6	49.7 ±1.4	+3.0 ±0.5
Interleaved	96.9 ±0.7	98.5 ±0.5	+1.6 ±0.3	45.8 ±1.5	47.2 ±1.4	+1.4 ±0.4	70.1 ±1.3	73.7 ±1.2	+3.6 ±0.5
Backbone: Qwen3-4B									
Zero-shot	30.9 ±1.4	28.7 ±1.6	−2.2 ±0.5	5.2 ±0.6	4.3 ±0.5	−0.9 ±0.3	34.1 ±1.5	30.7 ±1.7	−3.4 ±0.6
Task-Only	84.6 ±1.2	84.3 ±1.7	−0.3 ±0.6	54.9 ±1.7	53.1 ±1.8	−1.8 ±0.5	82.2 ±1.1	83.1 ±1.2	+0.9 ±0.3
Explore-Only	49.7 ±2.4	54.4 ±2.2	+4.7 ±0.5	26.1 ±1.3	28.9 ±1.2	+2.8 ±0.4	70.3 ±1.4	73.1 ±1.3	+2.8 ±0.5
Interleaved	90.5 ±1.1	92.7 ±1.2	+2.2 ±0.6	55.2 ±1.6	56.9 ±1.5	+1.7 ±0.4	85.9 ±1.2	89.0 ±1.0	+3.1 ±0.4

D Experimental Setup and Evaluation Protocols

D.1 Training Configuration and Rollout Accounting

For GRPO training, we follow Section 3.3. Task-execution rollouts receive binary task-success rewards, whereas goal-free exploration rollouts receive ECC rewards. Interleaved training combines these objectives using the reported aggregate 5:1 task:exploration rollout ratio. Table 11 gives the common hyperparameters.

Table 11: Hyperparameters for GRPO training.

Hyperparameter	**Qwen2.5-7B**	**Qwen3-4B**
Learning Rate	1e-6	1e-6
LR Scheduler	constant	constant
Group Size (G)	8	8
Max Training Step	300	300
Max Rollout Length	32k	32k
Batch Size	16	16
Rollouts Per Step	128	128

Matched rollout count. Each method uses 300 optimization steps with 128 rollouts per step, totaling 38,400 rollouts. Task-Only allocates all of them to task execution; the reported aggregate 5:1 Interleaved allocation comprises 32,000 task and 6,400 exploration rollouts. Exploration replaces task rollouts rather than increasing the total. Matching rollout count does not match interaction or token costs when trajectory lengths differ.

Training resources. All experiments are conducted on a single node equipped with $8\times$ NVIDIA H800 (80GB) GPUs. SFT training takes approximately 8 GPU-hours per backbone, while GRPO training requires approximately 192 GPU-hours due to the online rollout generation process. We use DeepSpeed ZeRO Stage-2 for memory-efficient distributed training for SFT.

D.2 Explorer–Executor Configurations

We distinguish exploration quality from the combined capabilities of the explorer and executor.

Exploration diagnosis. In Tables 1 and 9, each explorer receives up to 100 goal-free environment interactions. Steps and ECC characterize the explorer, while downstream gains use a fixed base Qwen3-4B executor. Success-rate changes are expressed in percentage points (pp).

Joint exploration and execution. The model in each row of Table 2 performs both exploration and execution. This also applies to the transfer experiment in Appendix C.2; those results measure both capabilities of the evaluated policy.

Controlled inference budget. Appendix C.1 fixes the executor to base Qwen3-4B and varies the explorer. The 100-interaction allowance is allocated to direct execution, two 50-step goal-conditioned attempts, or 50 goal-free exploration steps followed by 50 execution steps. Exploration is charged to each task. Token, auxiliary-call, and wall-clock costs are not established as equal.

D.3 Statistical Reporting

Tables 9 and 10 report mean and standard deviation over three runs, each using four rollouts per task (avg@4). Avg@4 averages outcomes rather than selecting the best attempt. Paired differences and across-environment averages must be formed within each seed before estimating their standard deviations. The supplied aggregate results lack the underlying per-seed vectors, so we report descriptive statistics without reconstructing paired tests from marginal standard deviations or adding significance claims.

D.4 ALFWORLD VARIANT CONSTRUCTION

To evaluate whether exploration-capable agents can adapt to environment shifts at test time, we construct three perturbed variants of ALFWorld. Each variant modifies a single axis of the environment while preserving the underlying task structure, so that performance degradation can be attributed to the agent's inability to handle the specific perturbation type rather than a fundamentally different task. All variants are derived from the same 274 test instances used in the original ALFWorld evaluation, yielding 274 instances per variant (1,096 total including the original).

Variant 1: Object Relocation. We modify the initial placement of task-relevant objects and receptacles. Specifically, for each task instance, we randomly reassign target objects to different receptacles or rooms while ensuring that the task remains solvable (i.e., all necessary objects are still reachable). For example, a task that originally requires finding a mug on the kitchen counter may now have the mug placed inside a bedroom drawer. This variant tests whether an agent has memorized fixed object–location associations from training or can discover the current object layout through exploration.

Variant 2: Interaction Precondition Changes. We alter the preconditions required to interact with certain objects or receptacles. For example, a container that is normally open by default may now start in a closed state and require an explicit `open` action before the agent can access its contents, or a receptacle that previously accepted objects directly may now require the agent to first clear an existing item. These modifications change the valid action sequences without altering the spatial layout, testing whether the agent can identify and adapt to new affordance constraints through exploratory interaction.

Variant 3: Distractor Injection. We introduce additional distractor objects into the environment that are visually or semantically similar to the task-relevant targets. For instance, in a task requiring the agent to pick up a specific book, we add several additional books in different locations. This variant increases the ambiguity of the environment and tests whether the agent can distinguish the correct target from distractors, a capability that benefits from thorough exploration and environment mapping prior to task execution.

Summary. Table 12 summarizes the statistics of the original and variant ALFWorld test sets.

Table 12: ALFWorld variant statistics.

Setting	# Instances
Original	274
Variant 1: Object Relocation	274
Variant 2: Interaction Precondition	274
Variant 3: Distractor Injection	274
Total	1,096

D.5 PROMPT FOR EXPLORATION

General Exploration Prompt

```
You are an general curious agent. Your goal is to fully understand the
environment through systematic exploration.

Objectives
1. Systematically explore all available actions and observe their
    effects
2. Map out the information structure of the environment
3. Identify reliable patterns and clues
```

```
Exploration Strategy
- Try different actions to understand state transitions
- Note what information is available at each state
- Track which actions are reversible vs irreversible
- Identify key decision points
- Explore different paths and branches

IMPORTANT NOTE
All findings must be grounded in direct interaction with the
    environment, not in prior knowledge or assumptions.
- Do not assume how an action behaves -- execute it and observe the
    actual result.
- Do not skip exploration because something "seems obvious" -- the
    environment may violate common expectations.
- When your expectation conflicts with an observation, always trust the
     observation and update your understanding accordingly.

Output Format
You can output your findings at any time when you feel exploration is
    complete.
Wrap your world modeling analysis in <world_modeling> </world_modeling>
     tags.
Describe freely in natural language:
- How actions change the environment
- Where information is located
- Which clues are reliable
- Key insights about the environment structure
- Any other information you found interesting or new
- ...

Example:
<world_modeling>

</world_modeling>

Environment Description
{environment_description_str}

Response Format
Think through your approach step by step, wrap your thinking process in
<think> </think>
<think>
Your reasoning and analysis here...
</think>
{response_format_str}

You can output world modeling only when you feel exploration is fully
    enough (Thus at the end of the exploration).
Wrap your world modeling in <world_modeling> </world_modeling> tags.
```

E CHECKPOINT CONSTRUCTION AND INFORMATION ACCESS

E.1 CHECKPOINT SCHEMA AND VERIFICATION

As described in Section 3.2, Exploration Checkpoint Coverage (ECC) requires a predefined set of checkpoints $\mathcal{C} = \{c_1, c_2, \ldots, c_M\}$ for each environment instance. Here we detail the construction procedure, which leverages the environment engine's internal state representation to derive verifiable, ground-truth checkpoints without relying on any model-generated annotations.

General Procedure. For each environment instance, we extract checkpoints from three categories: (1) **Locations**: distinct navigable rooms or areas the agent can visit; (2) **Objects**: key interactable entities present in the environment; and (3) **Affordances**: valid actions or state transitions associated with specific objects or locations (e.g., an object that can be opened, a device that can be activated). Algorithm 1 provides pseudocode for this extraction pipeline.

Algorithm 1 ECC Checkpoint Construction

Require: Environment engine $\mathcal{E}$, instance I
Ensure: Checkpoint set $\mathcal{C}$

$\mathcal{C} \leftarrow \emptyset$
$\mathcal{S} \leftarrow$ GETREACHABLESTATES($\mathcal{E}$, I)
for each state $s \in \mathcal{S}$ **do**
  $L \leftarrow$ EXTRACTLOCATIONS(s)
  $O \leftarrow$ EXTRACTOBJECTS(s)
  $A \leftarrow$ EXTRACTAFFORDANCES(s)
  $\mathcal{C} \leftarrow \mathcal{C} \cup L \cup O \cup A$
end for
$\mathcal{C} \leftarrow$ DEDUPLICATE($\mathcal{C}$)
$\mathcal{C} \leftarrow$ FILTERBYRELEVANCE($\mathcal{C}$)
return $\mathcal{C}$

Verification Mechanism. A checkpoint c_i is marked as *covered* if the agent's exploration trajectory contains an observation or action that unambiguously demonstrates awareness of c_i. Specifically, we perform string matching against the environment's textual observations: a location checkpoint is triggered when the agent receives the corresponding room description, an object checkpoint is triggered when the object appears in an observation following an interaction or examination action, and an affordance checkpoint is triggered when the agent successfully executes the associated valid action. This verification is deterministic and does not require any LLM-based judgment.

Environment-Specific Details.

- **ALFWorld.** Checkpoints are derived from the PDDL game state. Locations correspond to navigable rooms (e.g., kitchen, bedroom, bathroom). Objects include all task-relevant items and receptacles. Affordances cover valid pick-up, put, open, close, and toggle actions.
- **ScienceWorld.** Checkpoints are extracted from the environment's object tree and action space. Locations include all accessible rooms in the simulated house and yard. Objects encompass scientific instruments, materials, and containers. Affordances capture valid experimental operations (e.g., heating, mixing, measuring) and state changes (e.g., substance melting, temperature rising).
- **TextCraft.** Checkpoints are derived from the crafting recipe graph. Locations represent distinct resource-gathering zones. Objects include raw materials and intermediate crafted items. Affordances correspond to valid crafting recipes and resource-gathering commands that the agent can execute.

E.2 POLICY INPUTS AND PRIVILEGED EVALUATION SIGNALS

ECC uses simulator metadata to construct and verify checkpoints during training. The policy receives standard textual observations and its interaction history; it does not receive the checkpoint list

or simulator internals as input. At deployment it receives neither ECC feedback nor checkpoint definitions. ScienceWorld checkpoints in the transfer experiment are generated for post-hoc diagnostic scoring only. The lack of privileged policy inputs at deployment does not remove the need for privileged infrastructure when training with ECC or evaluating coverage.

E.3 Automatic Extraction and Engineering Cost

The environment-specific mappings above define an extractor that converts simulator metadata into canonical records, such as `location(kitchen)`, `object(mug_1)`, and `affordance(open, drawer_1)`. Once the interface has been implemented, extraction is automatic across instances and does not require manually listing checkpoints for each task. This is a one-time implementation cost per environment interface, rather than zero engineering cost. The experiments do not quantify implementation effort or verifier runtime.

Metric interpretation. ECC counts each checkpoint at most once and is bounded in $[0, 1]$. Repeating an already covered checkpoint adds no reward, and observations outside the checkpoint set do not increase ECC. These properties limit reward accumulation through repetition. They do not guarantee that an agent retains a discovered fact, understands its implications, or uses it correctly. In particular, triggering many valid checkpoints can still yield high ECC without a useful summary. A verifiable coverage metric is therefore not a direct measure of knowledge retention or task-relevant understanding.

F Exploration Utility and Qualitative Analysis

F.1 ECC, Downstream Utility, and Failure Modes

Association with downstream utility. An additional analysis pairs 240 ALFWorld exploration trajectories with downstream performance using a fixed Qwen3-4B executor. The reported Spearman correlation between trajectory ECC and downstream improvement Δ_{Task} is $\rho = 0.73$ ($p < 0.01$). This positive association is consistent with the budget analysis in Figure 4. It does not by itself establish that coverage causes better execution: exploration budget, task difficulty, and explorer identity can affect both quantities. The supplied aggregate statistic does not resolve those factors or specify the dependence structure among trajectories.

Failure despite high coverage. The additional analysis identifies two qualitative failure modes. First, the exploration can reveal relevant facts while the task still exceeds the executor's reasoning or planning ability. Second, the explorer can observe a decisive fact that is then omitted during summary compression. These observations motivate separating three stages in future evaluation: facts encountered in the trajectory, facts retained in the summary, and facts actually used by the executor. No failure frequencies or controlled summary-intervention results are reported for this analysis.

The bounded coverage definition and its interpretation are discussed in Appendix E. The cases below illustrate differences in exploration and task execution; they do not provide a controlled estimate of causality or a frequency estimate for the high-coverage failure modes above.

F.2 Qualitative Case Studies

We present two qualitative case studies from ALFWorld to illustrate the behavioral differences between exploration-aware and task-oriented agents. All examples are drawn from real model rollouts on the same environment instances.

F.2.1 Task-Oriented versus Exploration-Aware Training

We compare exploration trajectories produced by GRPO (Mixed) (exploration-aware) and GRPO (Task-Only) (task-oriented) on the same ALFWorld bedroom environment (task instance: `pick_and_place_simple`). Both models receive identical goal-free exploration instructions and a budget of 50 interaction steps.

Task-Oriented Model (GRPO Task-Only) — ECC: 0 / Steps: 1

Initial Observation:

```
You are in the middle of a room. Looking quickly around you, you
see a bed 1, a diningtable 1, a drawer 4, a drawer 3, a drawer 2, a
drawer 1, a garbagecan 1, a sidetable 2, and a sidetable 1.
```

Step 1: `done` *(Immediately terminates exploration)*

Failure Mode: Premature Termination. The task-oriented model finds no actionable goal in the exploration setting. Without a task reward signal, it immediately outputs `done` after a single observation, failing to interact with any object or discover any environment dynamics.

Task-Oriented Model: Generated World Model

Environment is a household setting with various locations and objects. The state space includes:

- Locations: bed 1, diningtable 1, drawer 1–4, sidetable 1–2, garbagecan 1
- Objects: various items in receptacles (not explicitly named)

Key structural features:

- The environment is navigable between locations.
- Objects are stored in receptacles and can be picked up, moved, examined, or processed.
- Some objects have specific actions (heat, cool, clean, slice) that modify their state.
- The robot can carry only one object at a time.

I need to explore the environment to learn more about the specific objects, their properties, and the rules governing interactions.

Assessment: Generic prior knowledge only. No grounded observations — objects "not explicitly named," action syntax unknown, no verified constraints. The model *acknowledges* it needs to explore but has already terminated.

Exploration-Aware Model (GRPO Mixed) — ECC: 87% / Steps: 49

Initial Observation: *(Same environment as above)*

Phase 1: Object Interaction Discovery

1. `look`	Observes room layout with all receptacles.
2. `go to bed 1`	Arrives. Sees: book 1, laptop 1, pillow 2, pillow 1.
3. `examine book 1`	"Nothing happens." (learns examine needs holding)
4. `take book 1 from bed 1`	Picks up book 1 successfully.
5. `inventory`	Confirms: "You are carrying: a book 1."
6. `examine book 1`	"There's nothing special about book 1."
7. `take laptop 1 from bed 1`	Fails → *discovers one-object limit.*
8. `move book 1 to bed 1`	Places object back. Learns `move` syntax.

Phase 2: Receptacle Exploration

9. `inventory`	Confirms empty hands.
10. `take laptop 1 from bed 1`	Tests picking different object type.
11. `move laptop 1 to bed 1`	Returns it.
12. `go to diningtable 1`	Sees: cd 2, cellphone 3, cellphone 1, mug 1, . . .
13. `take mug 1 from diningtable 1`	Picks up mug.
14. `go to drawer 1`	"The drawer 1 is closed."
15. `open drawer 1`	Opens drawer → discovers container mechanics.

. . . continues visiting sidetable 1–2, drawer 2–4, testing move . . .

Phase 3: Action Verification

45. `go to drawer 1`	Returns to verify drawer state persistence.
46. `open drawer 1`	Re-confirms open/close mechanics.
47. `close drawer 1`	Tests close action.
48. `go to bed 1`	Final sweep of starting location.
49. `done`	Terminates after exhaustive exploration.

Exploration-Aware Model: Generated World Model

Room Structure: Single room (bedroom type) containing: bed, diningtable, drawers (1–4), garbagecan, sidetables (1–2).

Actions and Their Effects:

- `look` — Shows room description; subsequent uses may return less info.
- `inventory` — Shows held object or "You are not carrying anything."
- `go to (receptacle)` — Moves to receptacle; shows contents on arrival.
- `open/close (receptacle)` — For containers (drawers, fridges, cabinets).
- `take (obj) from (recep)` — Requires: at receptacle, not holding another.
- `move (obj) to (recep)` — Requires: holding object, at target receptacle.
- `examine (object)` — Only works on held objects.
- `heat/cool/clean (obj) with (recep)` — State-changing with specific devices.

Key Constraints (Verified):

1. **Single-item inventory:** Exactly one object at a time; second take fails silently.
2. **Proximity required:** Must be at receptacle for take/move.
3. **Container state:** Drawers must be opened before contents visible.
4. **Examine requires holding:** Cannot examine objects on receptacles.
5. **Error signal:** Invalid actions return "Nothing happens" (no error type).

Discovered Objects: bed 1: book 1, laptop 1, pillow 1–2; diningtable 1: cd 2, cellphone 1/3, mug 1, keychain 1; drawer 1: pencil 2; sidetable 1–2: various objects.

Assessment: Fully grounded knowledge with verified action semantics, correct syntax, discovered constraints, and complete object inventory.

F.2.2 Low-ECC vs. High-ECC Grounded Task Execution

We compare task execution on the same ALFWorld kitchen environment (task: *pick_cool_then_place_in_recep* — pick up a mug, cool it, then place it on the coffeemachine). The exploration-aware model has access to its rich exploration summary (high ECC), while the task-oriented model operates with its shallow summary (zero ECC).

High-ECC Grounded Execution (Exploration-Aware) — ✓ Success / 7 steps

Task: Pick up a mug, cool it with the fridge, then place it on the coffeemachine.

1. `look`	Observes kitchen: cabinet 1–6, coffeemachine 1, countertop 1–3, fridge 1, . . .
2. `go to countertop 1`	Arrives. Sees: glassbottle 2, **mug 1**, papertowelroll 1.
3. `take mug 1 from countertop 1`	Picks up target object.
4. `go to fridge 1`	Navigates directly to cooling device.
5. `cool mug 1 with fridge 1`	Successfully cools the mug.
6. `go to coffeemachine 1`	Navigates to target placement.
7. `move mug 1 to coffeemachine 1`	Places cooled mug. **Task Complete!**

Key: The agent knows the exact syntax (`cool X with fridge 1`), the one-object constraint, and efficiently locates the mug on its first receptacle visit.

Zero-ECC Grounded Execution (Task-Only) — × Failure / 100 steps

Task: *(Same task, same environment as above)*

Phase 1: Aimless search (steps 1–10)

1. `look`	Observes kitchen (same environment).
2. `go to fridge 1`	Goes to fridge first (no plan).
3. `open fridge 1`	Opens fridge. Sees: apple, bowl, egg — *no mug.*
4. `go to cabinet 1`	Starts searching cabinets. Sees: vase 1.
5. `go to cabinet 2`	Opens cabinet 2. Sees: glassbottle, saltshaker, vase.
6–10. `go to cabinet 3-5`	Continues cabinets. Finds plates, saltshaker, soap.

. . . (steps 11–60: exhaustively searches all cabinets, drawers, sinkbasin; eventually finds mug but fails to cool due to incorrect sequencing) . . .

Phase 2: Stuck in loop (steps 96–100)

96. `cool mug 1 with fridge 1`	"Nothing happens." (not at fridge)
97. `cool mug 1 with fridge 1`	"Nothing happens."
98. `cool mug 1 with fridge 1`	"Nothing happens."
99. `cool mug 1 with fridge 1`	"Nothing happens."
100. `cool mug 1 with fridge 1`	"Nothing happens." → **Budget exhausted.**

Failure mode: The agent (1) searches inefficiently without knowing where mugs are located, (2) attempts the correct action but violates the proximity constraint, and (3) perseverates on the same failed action without adapting.

Summary. These cases illustrate the contrast between early termination or repeated ineffective actions and exploration that tests actions, verifies constraints, and records environment-specific facts. The execution example shows how such facts can support planning. These qualitative comparisons illustrate possible mechanisms; they do not isolate exploration quality from executor ability or establish that high coverage guarantees success.