

R-IDEAL: Reinforcing Insight-Driven Experience for Agent Learning

Wentao Hong¹, Xin Zhou², Yukai Xu¹, Jianing Yu¹,
Sheng Yang^{2*}, Sheng Gao¹, Weiran Xu^{1*}

¹Beijing University of Posts and Telecommunications

²Li Auto Inc.

{hwt, xuweiran}@bupt.edu.cn

Abstract

Learning from experience is essential for agents in real-world applications. However, most existing methods rely on static imitation of successful expert trajectories, overlooking the informative signals exposed from failed experiences and lacking a interactive feedback loop with the environment. In this paper, we propose R-IDEAL, a unified framework for insight-driven experience generation and optimization. R-IDEAL first perform contrastive analysis between successful and failed trajectories to automatically synthesize structured insights, consisting of *positive plans* and *negative lessons*, which guide agents not only on *what to do*, but also on *what to avoid*. These synthesized insights serve as supervision to train a lightweight insight model that generates task-specific guidance. While this supervision stabilizes insight generation, it does not guarantee the effectiveness of the insights due to distribution shifts and execution discrepancies across agents. Therefore, we further optimize the insight model via reinforcement learning using execution feedback from heterogeneous agents in real environments, ensuring that the generated insights effectively improve agent behavior. Experiments on AlfWorld and SciWorld show that R-IDEAL substantially outperforms existing methods, demonstrating strong robustness in out-of-distribution scenarios and serving as a lightweight, plug-and-play module for both closed-source and large-scale models.

1 Introduction

In recent years, large language model (LLM)-driven agents have demonstrated strong potential in long-horizon interactive tasks (Shridhar et al., 2020; Wang et al., 2022). Such tasks require agents to understand environmental feedback over multiple interaction steps and choose executable actions (Xi et al., 2025; Wang et al., 2024a). However,

compared with humans, existing agents still lack a crucial capability: abstracting discrete interaction trajectories into reusable and transferable experiences, and leveraging experiences to consistently guide future decision making (Fu et al., 2024a). In practice, this limitation often leads to inefficient trial-and-error exploration, poor robustness to out-of-distribution (OOD) perturbations (Fu et al., 2025), and error accumulation in long-horizon reasoning.

To address this challenge of learning from interaction experience, existing approaches can be broadly categorized into two paradigms. The first paradigm encodes experience implicitly into model parameters, through supervised fine-tuning or preference optimization on trajectories (Zeng et al., 2024; Song et al., 2024). While this paradigm is effective at improving a specific model’s performance on the tasks it is trained on, the learned experience is tightly coupled to the model weights, making it difficult to interpret, reuse, or transfer across models and tasks. The second paradigm aims to extract explicit textual guidance from experience and inject it into the agent’s context to shape its behavior.

Although such guidance is interpretable and transferable, existing methods primarily extract guidance on what to do from successful trajectories (Sun et al., 2023), while lacking systematic consolidation of what not to do from failures (Zhang et al., 2024; Zhu et al., 2025a). Moreover, most such guidance is derived from offline data and relies on external experience pools for test-time retrieval without grounding in real environment interaction, often leading to plausible yet non-executable, hallucinated behaviors.

We argue that effective guidance for long-horizon interactive tasks should be formulated as *structured insights* that capture both successful workflows and failure-aware constraints. Concretely, each insight comprises two components:

*Corresponding authors.

Plan, which specifies the key steps required to accomplish the task, thereby reducing uninformed exploration; and *Lessons*, which summarize common failure modes, invalid actions, and critical pre-conditions identified from unsuccessful trajectories, enabling the agent to explicitly avoid error-prone branches and improve robustness to OOD scenarios. Moreover, effective insights should meaningfully influence agent behavior in practical environments, lead to measurable performance improvements, and generalize across different models.

Following the above perspective, we propose R-IDEAL, a unified framework for insight generation and optimization. Instead of relying on expert trajectories, R-IDEAL learns directly from the target agent’s trajectories collected in real environments. We introduce a fully automated pipeline to extract structured insights from raw interaction data. By summarizing successful trajectories into positive *Plans* and contrasting successful and failed trajectories to derive negative *Lessons*, R-IDEAL constructs transferable, task-level guidance that encodes both what to do and what to avoid. To make such insight reusable, we train a lightweight *Insight Model* to generate structured insights conditioned on a task instruction. R-IDEAL adopts a two-stage training procedure. We first apply supervised fine-tuning to teach the Insight Model to stably produce high-quality, structured insights from task instructions. However, supervision alone cannot guarantee that the generated insights are executable or effective under distribution shifts and diverse agent behaviors. We further optimize the Insight Model via reinforcement learning (Shao et al., 2024) using feedback from real environments, ensuring that the generated insights are aligned with real environment and effectively guide agent behavior. Moreover, we design a hybrid reward function that aggregates feedback from format, length and heterogeneous agents, which substantially improves the generalization of the learned insights.

At inference time, R-IDEAL works as a plug-and-play module: the Insight Model generates structured insights from the task instruction and injects them into the agent’s context to guide its behavior. Because insight generation is decoupled from the downstream agent, R-IDEAL is compatible with both open-source and closed-source models, and allows a lightweight Insight Model to guide larger agents and improve long-horizon performance without additional fine-tuning, which is often costly and impractical for large-scale or

proprietary models.

We conduct extensive experiments on AlfWorld and SciWorld. The results show that R-IDEAL significantly outperforms existing baselines in both success rate and execution efficiency, and exhibits stronger robustness under out-of-distribution (unseen) settings. Notably, an Insight Model trained with a smaller backbone consistently improves the performance of a wide range of agents, including proprietary models and larger open-source models, demonstrating strong cross-model transferability of the learned insights.

Our contributions are as follows. (1) We propose a structured insight representation centered on Plan and Lessons, enabling insights to provide both an action backbone and constraints derived from failures. (2) We introduce an end-to-end framework for insight generation and optimization, using SFT to learn to generate structured insights and GRPO to align insights with real environmental utility, while leveraging feedback from heterogeneous models to improve cross-model generalization. (3) Finally, we achieve substantial performance gains on AlfWorld and SciWorld, showing that R-IDEAL can serve as a plug-and-play insight module for cross-model transfer and for guiding larger or proprietary models with a smaller Insight Model.

2 Related work

Experiential learning The ability to continuously evolve through interactive experience is a cornerstone for language agents aiming for general-purpose capabilities (Zhang et al., 2025; Zhao et al., 2024). Early approaches (Shinn et al., 2023) focused on utilizing natural language feedback for self-correction. Subsequent research (Zhu et al., 2023; Wang et al., 2024b; Gao et al., 2025) has further investigated mechanisms for explicit experience formalization and cross-task knowledge transfer. Notably, retrieval-based methods such as ExpeL (Wang et al., 2024b) and ERL (Allard et al., 2026) extract textual rules into an external experience pool for runtime retrieval. However, maintaining an expanding experience pool incurs non-trivial retrieval overhead, and reflection detached from real environments easily produces "guidance hallucinations." In contrast, R-IDEAL directly internalizes insights into a lightweight generator, completely eliminating the need for an external database. To leverage experience more profoundly, extensive research has centered on the

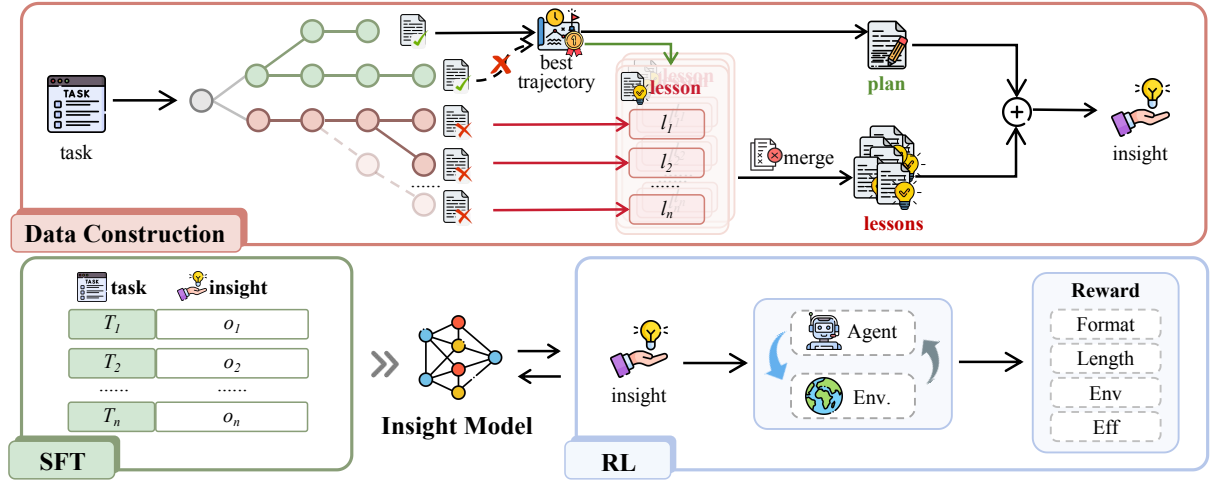


Figure 1: Overview of the R-IDEAL framework. The pipeline is composed of three parts: (1) Data Construction, which synthesizes structured insights containing “Plans” and “Lessons” via trajectory contrast; (2) Supervised Fine-Tuning (SFT), used for Insight Model initialization; and (3) Reinforcement Learning (RL), which utilizes the GRPO algorithm to optimize the Insight Model based on execution feedback from agents in real environments.

parametric optimization of policy weights. This includes imitation learning via Supervised Fine-Tuning (SFT) (Chen et al., 2023; Zeng et al., 2024; Chen et al., 2024b, 2025) and reinforcement learning methods ranging from offline preference optimization (Song et al., 2024; Jung et al., 2025; Putta et al., 2024) to end-to-end online exploration (Wei et al., 2025). However, this implicit internalization paradigm tightly couples experience with specific model weights. Such coupling not only hinders the transferability of knowledge across heterogeneous architectures but also restricts the direct application of these techniques to large-scale closed-source systems that are ineligible for fine-tuning.

Planning in LLM Agents Large Language Models have demonstrated significant potential in tackling complex interactive tasks (Liu et al., 2023). Early research (Yao et al., 2022) leveraged prompt engineering to stimulate planning capabilities within environments, yet continues to face stability challenges in long-horizon scenarios. Recent efforts have focused on decoupling planning from underlying execution by constructing explicit guidance mechanisms (Qiao et al., 2024; Zhu et al., 2025b; Xiong et al., 2025) or incorporating reinforcement learning techniques to enhance execution quality. Concurrently, works such as Voyager (Wang et al., 2023), AutoManual (Chen et al., 2024a), and AutoGuide (Fu et al., 2024b) explore the automatic generation of structured manuals through environmental interaction. Although these methods establish the paradigm of explicit guid-

ance, existing optimization trajectories predominantly focus on the static distillation of successful workflows and lack real-time constraints for defining “error boundaries”. Furthermore, the absence of dynamic optimization tailored to heterogeneous execution feedback often renders generated plans prone to overfitting the specific reasoning patterns of individual models.

3 Methodology

3.1 Overview

R-IDEAL is motivated by an experiential learning paradigm that progresses from “imitation” to “generalization.” We explicitly represent experience as Structured Insights ($o = [P; L]$), generated by an Insight Model (π_ϕ). As illustrated in Figure 1, the training pipeline proceeds in two phases. First, through Data Construction and Supervised Fine-Tuning (SFT), we synthesize high-quality insights via trajectory contrast to obtain an initialized π_ϕ . Second, we employ Reinforcement Learning (RL) with the GRPO algorithm to further optimize the insight model based on execution feedback, aligning the generated insights with downstream task success.

3.2 Phase I: Insight Synthesis

To construct high quality insights, for each task instruction g we collect a set of N trajectories $\mathcal{T}_g = \{\tau_1, \tau_2, \dots, \tau_N\}$. Each trajectory τ contains a sequence of state-action pairs and a final reward $r(\tau)$.

Data Construction We construct a seed dataset $\mathcal{D}_{seed}$ containing a plan P and lessons L via the following steps.

Optimal Path Selection We denote the set of successful trajectories (i.e., $r(\tau) = 1$) as $\mathcal{T}_{succ}$. The optimal trajectory τ^* is defined as the most efficient (fewest-step) trajectory in $\mathcal{T}_{succ}$:

$$\tau^* = \arg \min_{\tau \in \mathcal{T}_{succ}} |\tau|. \quad (1)$$

Based on the selected optimal trajectory τ^* , we prompt the teacher model to induce a task-general plan P .

Negative Lesson Extraction To extract lessons, we compare τ^* with each trajectory in the failure set $\mathcal{T}_{fail} = \mathcal{T}_g \setminus \mathcal{T}_{succ}$ and remove duplicated trajectory pairs. For each pair $(\tau^*, \tau_{fail}^{(i)})$, the teacher model generates a specific lesson l_i . Finally, we use an aggregation prompt p_{agg} to perform semantic deduplication and consolidation over all lessons:

$$L = M_T \left(\{M_T(\tau^*, \tau_{fail}^{(i)}, p_{compare})\}_i, p_{agg} \right). \quad (2)$$

Supervised Fine-Tuning (SFT) Subsequently, we perform supervised fine-tuning using the constructed dataset. Since the agent cannot access any reference trajectories during evaluation, we constrain the model to generate insights $o = [P; L]$ solely from the task instruction g . This process equips the model with an initial capability to produce standardized insights. Concretely, we fine-tune the Insight Model π_ϕ on the dataset $\mathcal{D}_{seed} = \{(g, o) \mid o = [P; L]\}$ by minimizing the following autoregressive loss:

$$\mathcal{L}_{SFT}(\phi) = -\mathbb{E}_{\mathcal{D}_{seed}} [\log \pi_\phi(o \mid g)]. \quad (3)$$

The model trained in this stage is denoted as π_{SFT} . In the subsequent second stage, it plays two roles simultaneously: serving as the initialization of the GRPO policy model π_θ (i.e., $\theta_{init} = \phi$), and acting as the reference policy π_{ref} for computing the KL-divergence constraint.

3.3 Phase II: Utility-Driven Policy Refinement

While SFT endows the model with the formatting standards for generating structured insights, the model often remains at the level of “formal imitation” of training data, struggling to generate truly effective guidance for unseen, complex instructions.

To transform static insights into dynamic problem-solving capability, we introduce the Group Relative Policy Optimization (GRPO) algorithm.

Unlike traditional PPO algorithms that rely on an additional Value Network, GRPO significantly reduces training overhead by calculating relative advantages over a group of outputs generated from the same input. In the R-IDEAL framework, the core objective of this phase is to optimize the Insight Model π_θ so that its generated insights o maximize the execution utility of the downstream agent. The specific training process is as follows:

Group Sampling: For each task instruction g , the Insight Model samples a group of G distinct outputs $\{o_1, o_2, \dots, o_G\}$ based on the current policy π_θ . Each output o_i contains the Plan and Lessons in a complete XML language format.

Heterogeneous Rollout: We construct a validation environment containing heterogeneous models (see Section 3.4). For each generated insight o_i , we inject it as context into different agents for parallel execution and calculate the comprehensive reward r_i based on the execution results.

Advantage Estimation: Based on the mean and standard deviation of the intra-group rewards, we calculate the advantage A_i for each output to identify insights that are superior to the average level:

$$A_i = \frac{r_i - \text{mean}(\{r_1, \dots, r_G\})}{\text{std}(\{r_1, \dots, r_G\})}. \quad (4)$$

Policy Update: We update the policy parameters θ by maximizing the following objective function, while introducing a KL divergence constraint to prevent the model from deviating too far from the SFT initial policy π_{ref} :

$$\mathcal{L}(\theta) = \mathbb{E}_{g \sim \mathcal{D}, o \sim \pi_\theta} \left[\frac{1}{G} \sum_{i=1}^G \min \left(r_i(\theta) A_i, \text{clip} \left(r_i(\theta), 1 - \epsilon, 1 + \epsilon \right) A_i \right) - \beta_{KL} \mathbb{D}_{KL}(\pi_\theta \parallel \pi_{ref}) \right], \quad (5)$$

where $r_i(\theta) = \frac{\pi_\theta(o_i|g)}{\pi_{old}(o_i|g)}$. Through this process, the model gradually learns to suppress hallucinations that “sound reasonable but are ineffective,” reinforcing valid insights that tangibly help the agent avoid risks and shorten paths.

3.4 Reward Function Design

To guide the model in generating insights that are not only normative and efficient but also possess Cross-model Generalization, we design a hybrid reward function incorporating format gating, heterogeneous utility, and length penalty. For a given instruction g and generated insight o , the total reward $R(o, g)$ is defined as follows:

Format Gating We enforce strict format verification: the insight o must contain complete `<workflow>` and `<lessons>` tags. If format parsing fails, we immediately apply a fixed penalty of -1.0 and terminate the reward computation.

Heterogeneous Utility Reward If the format is valid, we evaluate the actual contribution of the insight to the downstream task. To avoid overfitting the generated insight to the reasoning bias of a single model, we introduce a heterogeneous model set $\mathcal{M} = \{\text{Llama-3.1-8B}, \text{Qwen-2.5-7B}\}$. We inject o into these models respectively and execute them in the environment to calculate the total heterogeneous utility reward:

$$R_{utility} = \sum_{m \in \mathcal{M}} \left(r_{env}^{(m)} + \alpha \cdot r_{eff}^{(m)} \right). \quad (6)$$

Here, $r_{env}^{(m)}$ represents the reward returned by the environment. For the efficiency reward $r_{eff}^{(m)}$, we encourage generating insights that shorten the execution path. This term is calculated only when the task reward $r_{env}^{(m)} = 1.0$; otherwise, it is 0. The formula is:

$$r_{eff}^{(m)} = \mathbb{I}(r_{env}^{(m)} = 1) \cdot \left(1 - \frac{T_{steps}^{(m)}}{T_{max}} \right), \quad (7)$$

where $T_{steps}^{(m)}$ denotes the actual steps consumed by model m to complete the task, and T_{max} is the maximum interaction steps allowed by the environment. In our experiments, we set the weight coefficient $\alpha = 0.25$.

Length Penalty To prevent the model from generating verbose and redundant text in pursuit of coverage, we introduce a dynamic length penalty term $r_{len}(o)$ based on token count. We define this as a normalized piecewise function designed to apply a linear penalty to redundant outputs exceeding

L_{soft} :

$$r_{len}(o) = \begin{cases} 0 & \text{if } L < L_{soft} \\ \frac{L - L_{soft}}{L_{hard} - L_{soft}} & \text{if } L_{soft} \leq L \leq L_{hard} \\ 1 & \text{if } L > L_{hard} \end{cases}, \quad (8)$$

where L is the token length of the generated text. In our experiments, we set the soft threshold $L_{soft} = 512$ and the hard threshold $L_{hard} = 2048$.

Final Reward Formula In summary, given a valid format, the final reward is the heterogeneous utility reward minus the length penalty:

$$R(o, g) = R_{utility} - \beta_{len} \cdot r_{len}(o). \quad (9)$$

We set $\beta_{len} = 0.25$ in our experiments. This design ensures that high-scoring insights must simultaneously satisfy: (1) guiding different models to complete the task; (2) reducing execution steps; and (3) being concise.

3.5 Inference Phase

Distinct from implicit methods that internalize experience into model parameters, R-IDEAL achieves a full decoupling between the Insight Model and the Agent. Specifically, during inference, given any test-task instruction g , the Insight Model first generates guiding insights o . These insights are then concatenated to the end of the original instruction, forming an augmented input $[g; o]$, which is fed into any target agent to guide subsequent trajectory generation.

This text-concatenation-based decoupling mechanism endows the framework with plug-and-play inference capabilities, enabling both closed-source model compatibility and small-to-large generalization: since the Insight Model outputs structured natural-language text, its insights can be directly injected as additional context to improve the performance of closed-source models as well as open-source models; moreover, the insight-generation policy can be trained with a smaller model (e.g., Llama-3.1-8B-Instruct) and, at inference time, the concise insights produced by the small model can guide larger models to overcome reasoning bottlenecks in specific long-horizon tasks, avoiding expensive fine-tuning of large models.

4 Experiments

In this section, we benchmark our method, R-IDEAL, on datasets involving household tasks (Alf-World) and scientific experiments (SciWorld). We

Method	AlfWorld		SciWorld		Average
	Seen	Unseen	Seen	Unseen	
Agents w/o Training					
Llama-3.1-8B-Instruct (Dubey et al., 2024)	22.9	28.4	47.7	42.2	35.3
Llama-3.1-8B-Instruct + MPO (Xiong et al., 2025)	50.0	52.2	56.5	55.5	53.6
Llama-3.1-8B-Instruct + R-IDEAL (Ours)	94.0	93.5	82.1	75.1	86.2
GPT-4o (Achiam et al., 2023)	78.6	83.6	60.0	56.0	69.6
GPT-4o + MPO (Xiong et al., 2025)	89.3	93.3	67.3	67.8	79.4
GPT-4o + R-IDEAL (Ours)	98.3	98.5	81.7	78.6	89.3
Llama-3.1-70B-Instruct (Dubey et al., 2024)	78.6	73.9	72.6	70.2	73.8
Llama-3.1-70B-Instruct + MPO (Xiong et al., 2025)	85.7	86.6	80.4	79.5	83.1
Llama-3.1-70B-Instruct + R-IDEAL (Ours)	99.0	99.5	84.0	81.1	90.9
Agents w/ Training					
Llama-3.1-8B-Instruct + KnowAgent (Zhu et al., 2025b)	80.0	74.9	81.7	69.6	76.6
Llama-3.1-8B-Instruct + WKM (Qiao et al., 2024)	82.1	76.5	77.1	78.2	78.5
Llama-3.1-8B-Instruct-SFT (Zeng et al., 2024)	79.3	71.6	65.3	57.0	68.3
Llama-3.1-8B-Instruct-SFT + MPO	80.7	81.3	70.2	65.9	74.5
Llama-3.1-8B-Instruct-SFT + R-IDEAL (Ours)	98.6	98.3	82.3	79.6	89.7
Llama-3.1-8B-Instruct-ETO (Song et al., 2024)	77.1	76.4	81.3	74.1	77.2
Llama-3.1-8B-Instruct-ETO + MPO	85.0	79.1	83.4	80.8	82.1
Llama-3.1-8B-Instruct-ETO + R-IDEAL (Ours)	99.0	98.8	85.7	82.6	91.5

Table 1: Performance comparison on AlfWorld and SciWorld benchmarks. Average reward (%) are reported for seen and unseen test sets. R-IDEAL consistently outperforms baselines across both untrained agents (top) and trained agents (bottom).

validate the effectiveness of our method by answering the following research questions:

RQ1: How does R-IDEAL perform compared to other experience utilization methods? **RQ2:** Why is the additional Insight Model necessary? **RQ3:** What are the contributions of the individual components of R-IDEAL? **RQ4:** In what ways is R-IDEAL effective regarding efficiency, error reduction, and specific cases?

4.1 Experimental Setup

Datasets We evaluate R-IDEAL on AlfWorld (Shridhar et al., 2020) and SciWorld (Wang et al., 2022). AlfWorld is a text-based household task benchmark, while SciWorld is a text-based scientific simulation benchmark. For detailed information about the datasets, please refer to Appendix A.

Baselines We compare our method against a range of baselines: (1) Open-source and Proprietary Models: In this setting, models directly

execute tasks using the ReAct framework. We test GPT-4o (Achiam et al., 2023) as the proprietary model. For open-source models, we select models of various sizes, including Llama-3.1-8B-Instruct and Llama-3.1-70B-Instruct. (2) Implicit Experience Utilization: These methods enhance agent capabilities via parameter updates. Agent-Tuning (Zeng et al., 2024) uses expert trajectories for SFT. ETO (Song et al., 2024) uses positive and negative trajectories for DPO (Rafailov et al., 2023) training to enhance failure experience utilization. (3) Explicit Experience Utilization: These methods formulate experience as explicit knowledge to guide the agent. KnowAgent (Zhu et al., 2025b) and WKM (Qiao et al., 2024) collect task-related knowledge and integrate it into training. MPO (Xiong et al., 2025) trains a planner providing global plans via SFT and DPO on human-curated data. These methods also provide additional guidance, and therefore are not compatible with our approach.

Method	AlfWorld		Average
	Seen	Unseen	
Baseline (w/o insight)	22.9	28.4	25.7
GPT-5-mini	31.0	28.4	29.7
GPT-5	39.3	38.3	38.8
R-IDEAL(Ours)	94.0	93.5	93.8

Table 2: Success rates (%) on AlfWorld using insights generated by different methods. All insights are evaluated with Llama-3.1-8B-Instruct as the execution agent.

Implementation Details We select Llama-3.1-8B-Instruct as the backbone for the Insight Model. In the SFT phase, we sample 4 times on the training set using Llama-3.1-8B-Instruct with a temperature of 0.8 to construct the contrastive trajectory repository. We perform full fine-tuning using LlamaFactory (Zheng et al., 2024) with a batch size of 32, a learning rate of 1×10^{-5} , a warm-up ratio of 0.03, and a cosine scheduler. For the RL phase, we use the Verl (Sheng et al., 2025) to further fine-tune the SFT model, following the algorithm and reward design in Sections 3.3 and 3.4. We employ Llama-3.1-8B-Instruct and Qwen2.5-7B-Instruct as heterogeneous action Models, with a group sampling size of 8, a batch size of 64, and a learning rate of 1×10^{-6} .

Evaluation During evaluation, we set the decoding temperature of both the Insight Model and Action Model to 0 to ensure reproducibility. Following MPO’s setting, we use zero-shot prompting for insight generation and 1-shot in-context prompting for task trajectory generation. We report the average reward for each test, and all reported results are averaged over three runs to reduce variance.

4.2 Main Results

As shown in Table 1, R-IDEAL significantly improves the performance of various base models across different settings, demonstrating broad applicability. Specifically, the agent based on Llama-3.1-8B-Instruct achieves a substantial increase in average reward to 86.2, outperforming the previous state-of-the-art baseline MPO by 32.6. This fully validates the synergistic effectiveness of insights (containing negative lessons) and the dynamic optimization mechanism based on heterogeneous execution feedback.

Crucially, the experiments demonstrate significant “Small-to-Large” generalization: using an

Configuration	AlfWorld		Average
	Seen	Unseen	
R-IDEAL (Full Model)	94.0	93.5	93.8
<i>Inference-time</i>			
w/o Lesson	92.4	89.6	91.0
w/o Plan	73.1	67.4	70.3
<i>Training-time</i>			
w/o Lesson	93.1	86.3	89.7
w/o Plan	88.1	80.8	84.5

Table 3: Ablation study of Plan and Lesson components on AlfWorld. We compare removing each component at inference time versus excluding it during training.

Insight Model trained only on an 8B model, R-IDEAL serves as a general plug-and-play module that consistently enhances the reasoning performance of closed-source models and large open-source models without additional parameter fine-tuning. Particularly in challenging *Unseen* scenarios, the structured insights generated by R-IDEAL maintains high quality and applicability, boosting Llama-3.1-70B-Instruct’s average reward on AlfWorld Unseen tests by 25.6, reaching a near-perfect 99.5. These data strongly suggest that R-IDEAL not only generates generally useful insights through heterogeneous feedback but also effectively solves the adaptability problem in unfamiliar environments, significantly enhancing robustness and generalization.

4.3 Directly Generating Insights with GPT-5

A natural question is: given that state-of-the-art large language models such as GPT-5 demonstrate strong reasoning capabilities, do we still need to train a dedicated Insight Model? Is it sufficient to directly ask SOTA models to generate insights? To answer this question, we conduct a comparative experiment. We fix Llama-3.1-8B-Instruct as the agent’s backbone model, and use R-IDEAL (trained from Llama-3.1-8B-Instruct), as well as prompt-engineered GPT-5-mini and GPT-5, to generate guiding insights.

As shown in Table 2, although GPT-5 has top-tier general reasoning ability, its directly generated insights improves the Agent’s success rate only to 39.3% on Seen tasks and 38.3% on Unseen tasks. While this outperforms the unguided baseline, it remains fundamentally behind R-IDEAL, which achieves 94.0% and 93.5%. We attribute this discrepancy primarily to insufficient ground-

Method	Average Score
R-IDEAL (Ours)	97.3
- w/o Efficiency Reward	94.4
- w/o Heterogeneous Utility Reward	92.8

Table 4: Ablation of RL components on AlfWorld. Average success rates (%) across four models (Llama-3.1-8B/70B, Qwen-2.5-7B/72B) on seen/unseen splits.

ing: without feedback from environment interaction, SOTA models often produce hallucinated suggestions that sound plausible but are not executable, such as attempting to manipulate non-interactive objects or ignoring necessary preconditions. In contrast, R-IDEAL leverages environment feedback to refine its strategy, highlighting the value of targeted training: even an 8B model, once equipped with environment-specific insights, can substantially outperform an untuned ultra-large model in guidance effectiveness.

4.4 Ablation Studies

4.4.1 Module Ablation

To investigate the independent contributions of the two core cognitive components, “Plan” and “Lessons,” within the R-IDEAL framework, we designed ablation experiments at two levels: (1) Inference-time Ablation, where specific modules are selectively removed from the context via masking on the fully trained model; and (2) Training-time Ablation, where specific modules are completely removed throughout the entire pipeline—from data construction and SFT to RL—and the model is retrained. The results are summarized in Table 3.

At inference time, removing Lessons leads to a small performance drop, with a larger decline on Unseen tasks, indicating that Lessons improve the agent’s robustness. Removing Plan causes a substantial degradation on both Seen and Unseen tasks, showing that Plan provides critical guidance for task solving. Compared with inference-time context masking, the absence of Lessons during training results in a more severe loss of generalization ability. Overall, Plan provides guidance and reduces the search space, while Lessons further highlight where errors are likely to occur and help reduce failures; together they complement each other and achieve the best performance.

Method	AlfWorld		Average
	Seen	Unseen	
Baseline	22.9	28.4	25.7
w/o trajectory contrast	28.6	20.9	24.8
w/ trajectory contrast	53.6	62.9	58.3

Table 5: Ablation study on contrastive trajectory analysis in data construction. Insight models are evaluated after SFT without RL refinement.

4.4.2 RL Method Ablation

To analyze the contribution of each component in the R-IDEAL reward function, we conduct a progressive ablation study, with results reported in Table 4.

We first establish an RL baseline using a single model (Llama-3.1-8B-Instruct) with binary rewards (denoted as w/o Heterogeneous Utility Reward), achieving an average success rate of 92.8%. We then add Qwen2.5-7B-Instruct as a second verification model (denoted as w/o Efficiency Reward). Simply introducing heterogeneous feedback increases the average success rate to 94.4% (+1.6%). This gain mainly comes from the complementarity of heterogeneous models: the generated insights must satisfy two distinct reasoning patterns, which filters out “shortcuts” that only work for a specific architecture and improves cross-model robustness.

On top of dual-model feedback, we further incorporate the efficiency reward, boosting the average success rate from 94.4% to 97.3% (+2.9%). This suggests that the efficiency reward does more than reduce the number of steps; by penalizing redundant actions, it strengthens the logical constraints of the plan, leading to more precise and direct solution paths and substantially reducing failures caused by error accumulation in long-horizon reasoning.

Overall, heterogeneous feedback improves generalizability, while the efficiency reward improves precision. Both are indispensable and together underpin R-IDEAL’s performance advantage.

4.4.3 Ablation on Trajectory Contrast

To rigorously verify that the performance gains come from our trajectory-contrastive data construction strategy—rather than simply from distilling knowledge from a strong teacher model—we conducted a controlled ablation study. Using the same teacher model (Qwen3-235B-A22B-Instruct-2507), we generated SFT data in two ways: (1) Direct gen-

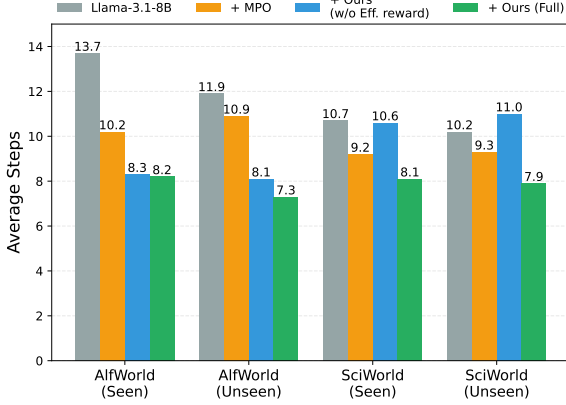


Figure 2: Average execution steps for successful episodes on AlfWorld and SciWorld.

eration (w/o trajectory comparison): generating insights solely from the task description, without using any execution trajectories. (2) Ours (w/ trajectory comparison): extracting insights by contrasting successful trajectories with failed ones.

Table 5 shows that direct generation achieves a 28.6% success rate on Seen tasks, only slightly higher than the unguided baseline (22.9%). On Unseen tasks, its performance drops to 20.9%, even lower than the unguided baseline (28.4%). This indicates that direct generation is prone to a “hallucination trap”: without real interaction trajectories as grounding, it tends to produce suggestions that sound plausible but are not executable, which can mislead the agent—especially in unfamiliar environments. In contrast, our method is grounded in real trajectories and substantially reduces hallucinations, suggesting that high-quality supervision should come from environmental interaction rather than purely semantic reasoning.

4.5 Analysis

Efficiency Analysis Beyond success rate, execution efficiency is also a key indicator of an agent’s practical utility. As shown in Figure 2, R-IDEAL significantly reduces the number of steps required to execute tasks on both AlfWorld and SciWorld. Moreover, the efficiency reward yields more pronounced gains on complex tasks such as SciWorld. Although R-IDEAL incurs one additional model call to generate insights, it reduces the number of agent–environment interaction steps, thereby lowering overall runtime. In addition, this mechanism requires no retraining when guiding different models, further demonstrating its cost-effectiveness and generalizability in practical deployment.

Method	AlfWorld		Average
	Seen	Unseen	
Llama-3.1-8B-Instruct	44.60	54.51	49.56
+ MPO	36.95	41.87	39.41
+ Ours (w/o Plan)	41.72	48.26	44.99
+ Ours (w/o Lesson)	16.76	19.24	18.00
+ Ours (Full)	13.35	14.70	14.03

Table 6: Analysis of Invalid Action Rates on AlfWorld. All methods use Llama-3.1-8B-Instruct as the agent’s backbone model.

Error Step Analysis To analyze how R-IDEAL improves decision quality, we measure the frequency of “Invalid Actions” produced on AlfWorld. Invalid actions mainly fall into two categories: errors that violate environment interaction rules (with “Nothing happens”) and syntactic errors that violate formatting requirements (with feedback such as “Error Input...”). These errors reflect the agent’s limited understanding of environmental dynamics and insufficient instruction-following ability. As shown in Table 6, the unguided agent exhibits a high invalid-action rate. With R-IDEAL’s insights guidance, the error rate is substantially reduced and clearly outperforms MPO. Overall, R-IDEAL enhances the effectiveness of agent–environment interaction through the synergy of Plan, which establishes a correct trajectory, and Lesson, which prunes erroneous branches.

5 Conclusion

In this paper, we propose R-IDEAL, a unified framework for insight generation and optimization. R-IDEAL synthesizes structured insights via contrastive analysis, producing positive Plans and negative Lessons that explicitly guide agents on both what to do and what to avoid. To ensure the generated insights are effective and generalizable, we optimize the lightweight Insight Model via reinforcement learning using execution feedback from heterogeneous agents. Experiments on AlfWorld and SciWorld demonstrate that R-IDEAL substantially outperforms baselines and exhibits strong robustness in out-of-distribution scenarios. As a plug-and-play module, R-IDEAL proves that a lightweight model can effectively steer large-scale and closed-source agents without costly fine-tuning, offering a scalable paradigm for enhancing agent reasoning through transferable experiences.

Limitations

Despite its strong performance, R-IDEAL still has limitations that merit further investigation. First, R-IDEAL is primarily validated in text-based, discrete decision-making environments. For more complex tasks with substantially larger action spaces (e.g., Software Engineering (Jimenez et al., 2024) and Open-ended Web Research (Mialon et al., 2023)), how to define and generate effective insights remains an open problem. Second, in the current framework, insights become relatively fixed after training; however, in highly stochastic novel environments, the initial insights may become invalid. Enabling the Insight Model to dynamically update insights according to real-time feedback is an exciting direction for future work.

Ethical Considerations

This work aims to improve the reliability and execution efficiency of large language model (LLM)-based agents in long-horizon interactive tasks by distilling interaction trajectories from simulated environments into structured insights and optimizing them with environment feedback. Our trajectory data are automatically sampled from simulated environments and contain no human annotations or real user data. Potential risks are similar to those of other LLM-based methods: when faced with malicious or harmful inputs, the model may generate inappropriate content, and the increased capability of agents could be misused. We recommend thoughtful deployment practices, such as adding input/output filtering and restricting application settings.

Acknowledgments

The computational resources used in this work were provided by Li Auto Inc. We would also like to thank the anonymous reviewers and area chairs for their insightful comments and constructive suggestions.

References

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Marc-Antoine Allard, Arnaud Teinturier, Victor Xing, and Gautier Viaud. 2026. Experiential reflective learning for self-improving llm agents. *arXiv preprint arXiv:2603.24639*.
- Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023. Fireact: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*.
- Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. 2024a. Automanual: Constructing instruction manuals by llm agents via interactive environmental learning. *Advances in Neural Information Processing Systems*, 37:589–631.
- Yihan Chen, Benfeng Xu, Xiaorui Wang, Yongdong Zhang, and Zhendong Mao. 2025. Training llm-based agents with synthetic self-reflected trajectories and partial masking. *arXiv preprint arXiv:2505.20023*.
- Zehui Chen, Kuikun Liu, Qiuchen Wang, Wenwei Zhang, Jiangning Liu, Dahua Lin, Kai Chen, and Feng Zhao. 2024b. Agent-flan: Designing data and methods of effective agent tuning for large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 9354–9366.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.
- Dayuan Fu, Keqing He, Yejie Wang, Wentao Hong, Zhuoma Gongque, Weihao Zeng, Wei Wang, Jinggang Wang, Xunliang Cai, and Weiran Xu. 2025. Agentrefine: Enhancing agent generalization through refinement tuning. *arXiv preprint arXiv:2501.01702*.
- Dayuan Fu, Biqing Qi, Yihuai Gao, Che Jiang, Guanting Dong, and Bowen Zhou. 2024a. Msi-agent: Incorporating multi-scale insight into embodied agents for superior planning and decision-making. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 643–659.
- Yao Fu, Dong-Ki Kim, Jaekyeom Kim, Sungryull Sohn, Lajanugen Logeswaran, Kyunghoon Bae, and Honglak Lee. 2024b. Autoguide: Automated generation and selection of context-aware guidelines for large language model agents. *Advances in Neural Information Processing Systems*, 37:119919–119948.
- Jinglong Gao, Xiao Ding, Lingxiao Zou, Bibo Cai, Bing Qin, and Ting Liu. 2025. Expetrans: Llms are experiential transfer learners. *arXiv preprint arXiv:2505.23191*.
- Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? In *12th International Conference on Learning Representations, ICLR 2024*.

- Yeonsung Jung, Trilok Padhi, Sina Shaham, Dipika Khullar, Joonhyun Jeong, Ninareh Mehrabi, and Eunho Yang. 2025. Co-evolving agents: Learning from failures as hard negatives. *arXiv preprint arXiv:2511.22254*.
- Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.
- Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. *Gaia: a benchmark for general ai assistants*. Preprint, arXiv:2311.12983.
- Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. 2024. Agent q: Advanced reasoning and learning for autonomous ai agents. *arXiv preprint arXiv:2408.07199*.
- Shuofei Qiao, Runnan Fang, Ningyu Zhang, Yuqi Zhu, Xiang Chen, Shumin Deng, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. 2024. Agent planning with world knowledge model. *Advances in Neural Information Processing Systems*, 37:114843–114871.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741.
- Zhihin Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.
- Guangming Sheng, Chi Zhang, Zilinfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pages 1279–1297.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652.
- Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*.
- Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024. Trial and error: Exploration-based trajectory optimization of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7584–7600.
- Haotian Sun, Yuchen Zhuang, Ling kai Kong, Bo Dai, and Chao Zhang. 2023. Adapllanner: Adaptive planning from feedback with language models. *Advances in neural information processing systems*, 36:58202–58245.
- Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.
- Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024a. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.
- Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. 2022. Scienceworld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11279–11298.
- Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. 2024b. Agent workflow memory. *arXiv preprint arXiv:2409.07429*.
- Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, et al. 2025. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. *arXiv preprint arXiv:2505.16421*.
- Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2025. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101.
- Weimin Xiong, Yifan Song, Qingxiu Dong, Bingchan Zhao, Feifan Song, Xun Wang, and Sujian Li. 2025. Mpo: Boosting llm agents with meta plan optimization. *arXiv preprint arXiv:2503.02682*.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.
- Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2024. Agenttuning: Enabling generalized agent abilities for llms. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3053–3077.
- Tianjun Zhang, Aman Madaan, Luyu Gao, Steven Zheng, Swaroop Mishra, Yiming Yang, Niket Tandon, and Uri Alon. 2024. In-context principle learning from mistakes. *arXiv preprint arXiv:2402.05403*.
- Zeyu Zhang, Quanyu Dai, Xiaohe Bo, Chen Ma, Rui Li, Xu Chen, Jieming Zhu, Zhenhua Dong, and Ji-Rong Wen. 2025. A survey on the memory mechanism of large language model-based agents. *ACM Transactions on Information Systems*, 43(6):1–47.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. 2024. 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*, pages 19632–19642.

Yaowei Zheng, Richong Zhang, Junhao Zhang, YeYanhan YeYanhan, and Zheyang Luo. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 400–410.

Kunlun Zhu, Zijia Liu, Bingxuan Li, Muxin Tian, Yingxuan Yang, Jiaxun Zhang, Pengrui Han, Qipeng Xie, Fuyang Cui, Weijia Zhang, et al. 2025a. Where llm agents fail and how they can learn from failures. *arXiv preprint arXiv:2509.25370*.

Yuqi Zhu, Shuofei Qiao, Yixin Ou, Shumin Deng, Shiwei Lyu, Yue Shen, Lei Liang, Jinjie Gu, Hua-jun Chen, and Ningyu Zhang. 2025b. Know-agent: Knowledge-augmented planning for llm-based agents. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 3709–3732.

Zhaocheng Zhu, Yuan Xue, Xinyun Chen, Denny Zhou, Jian Tang, Dale Schuurmans, and Hanjun Dai. 2023. Large language models can learn rules. *arXiv preprint arXiv:2310.07064*.

A Dataset Details

In this section, we provide detailed information about the two benchmarks used in our experiments: AlfWorld and SciWorld.

AlfWorld (Shridhar et al., 2020) is a text-based household task benchmark. It requires agents to achieve goals through a sequence of actions, such as “put object X in location Y.” The agent receives a reward of 1 for achieving the goal, and 0 otherwise. The main challenge lies in leveraging the common sense of Large Language Models (LLMs) and their understanding of environmental interactions. We evaluate on all 140 *seen* evaluation tasks and 134 *unseen* evaluation tasks under task-specific settings.

SciWorld (Wang et al., 2022) is a text-based scientific simulation benchmark. It simulates a complex interactive laboratory environment, requiring agents to complete tasks involving chemistry, biology, and physics, such as “boil water using a heater” or “mix specific chemicals to create a reaction.” SciWorld provides dense rewards ranging from 0 to 1. Its main challenges include scientific reasoning, causal understanding, and multi-step

planning in a dynamic environment with a vast state space. It contains 194 *seen* and 211 *unseen* evaluation tasks.

B Case Study

To further validate the specific roles of Plan and Lessons in the R-IDEAL framework, we analyze concrete cases under the guidance of MPO and R-IDEAL, as shown in Figure 3. In AlfWorld’s task “put some peppershaker on drawer”, MPO’s plan lacks robustness: during execution, when encountering a semantically similar saltshaker, the agent makes an incorrect substitution decision (“saltshaker is similar to peppershaker”), which leads to an erroneous loop. In contrast, R-IDEAL’s plan provides more robust guidance and explicitly warns about this error in the Lessons module in advance. In SciWorld’s long-horizon task “Grow a orange”, MPO’s plan shows insufficient understanding of environment interaction details; the agent repeatedly attempts an undefined action, “use shovel to move soil,” resulting in task failure. R-IDEAL’s plan clearly specifies the action order and, through the Lessons module, consolidates historical trial-and-error into actionable procedural knowledge: it directly leverages the environment-supported “pour” action to transfer substances between containers, thereby avoiding the erroneous loop in MPO caused by repeatedly attempting undefined actions.

C Prompts Used in Our Work

We present the full prompts used in our experiments below.

Prompt Template: Workflow Extractor

You are an expert Task Planner specializing in agent tasks. Your goal is to reverse-engineer a GENERALIZABLE WORKFLOW from a successful execution trajectory.

Core Philosophy: "The No-Oracle Rule" The input trajectory contains "Oracle-view" information (i.e., known answers). However, the generated general workflow MUST NOT assume that the specific details required for exploration are known beforehand. You must convert these "specific actions" into "SEARCH AND SOLVING STRATEGIES".

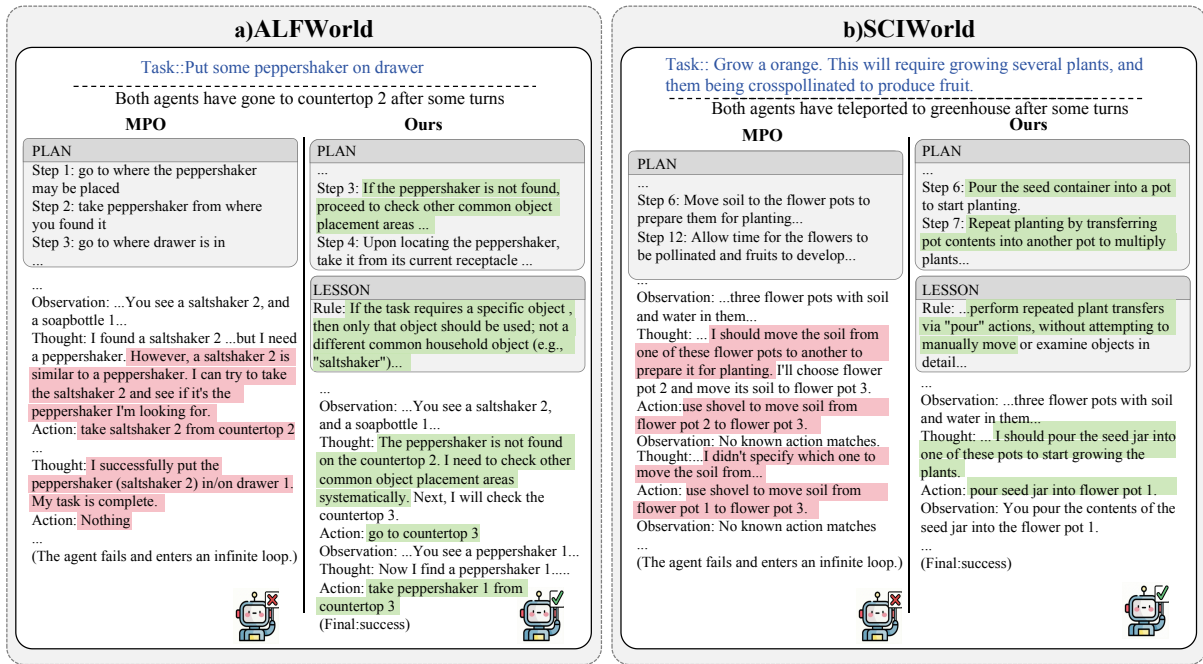


Figure 3: Case study illustrating the effectiveness of the Plan and Lessons modules in R-IDEAL.

Requirements:

Entity Generalization: Convert specific IDs, coordinates, or names into abstract category descriptions.

Preserve Task Entities: Use the specific object names defined in the task goal.

Abstract Instance IDs: Remove specific instance numbers.

Strategize Actions: Transform "specific paths/actions" into "exploration and solving strategies".

Maintain Robustness: The workflow should remain valid even if environmental details change. Avoid assuming specific object locations or states.

Noise Filtering: Ignore execution errors, navigation corrections, or irrelevant observations unless they trigger a necessary state change. Focus only on the successful logic flow.

Concise Imperative Style: Use short, direct imperative sentences. Avoid fluff like "Begin by...", "It is important to...", or "The agent should...".

Input Data:

```
<task_description>
{task_description}
</task_description>
```

```
<success_trajectory>
{success_trajectory}
</success_trajectory>
```

Output Format:

```
<workflow>
Step 1: ...
Step 2: ...
...
</workflow>
```

Prompt Template: Lessons Extractor

You are an expert Agent Trajectory Analyst. Your goal is to extract a high-value "General Lesson" (Rule, Constraint, or Skill) by analyzing the differences between a complete successful trajectory and a complete failed trajectory.

Core Philosophy: "Comparative Analysis"
You are provided with two complete execution trajectories:

Success Trajectory: The full sequence of actions that successfully completed the task.

Failure Trajectory: The full sequence of actions that failed to complete the task.

Your task is to analyze both trajectories, identify critical decision points where they

differ, and extract the Abstract Rule that governs successful execution. Explain why the failure trajectory failed and why the success trajectory succeeded, focusing on the underlying principles.

Input Data:

```
<task_description>
{task_description}
</task_description>
```

```
<success_trajectory>
{success_trajectory}
</success_trajectory>
```

```
<failure_trajectory>
{failure_trajectory}
</failure_trajectory>
```

Analysis Requirements:

Comparative Analysis: Compare the success and failure trajectories step by step.

Critical Decision Points: Identify the moments that determine success vs failure.

Generalization: Formulate rules that apply to similar situations, not just this specific instance.

Output Format:

```
<analysis>
1. Trajectory Comparison: ...
2. Critical Factors: ...
3. Success Logic: ...
</analysis>
```

```
<lessons>
(One concise, generalizable rule.)
Rule: ...
</lessons>
```

Prompt Template: Lessons Merger

You are an expert Data Analyst specializing in semantic deduplication. Your ONLY goal is to remove redundant information from the given list of lessons.

INPUT DATA:

```
<raw_lessons_list>
{lessons}
</raw_lessons_list>
```

STRICT GUIDELINES: Semantic Deduplication Only: Compare the lessons one by

one. If two or more lessons convey the exact same logic (even if phrased differently), merge them into one representative rule.

No Content Filtering: Do NOT evaluate the quality, validity, or importance of the lessons. As long as a lesson is unique, you must keep it.

Preserve Original Meaning: When merging duplicates, you may polish the grammar, but you must NOT change the original meaning or scope of the rule.

List All Unique Rules: Do not over-summarize or abstract.

OUTPUT FORMAT:

Output the final result within <lessons></lessons> tags.

```
<lessons>
Rule:...
</lessons>
```

Instruction Prompt for ALFWorld

Interact with a household to solve a task. Imagine you are an intelligent agent in a household environment and your target is to perform actions to complete the task goal. At the beginning of your interactions, you will be given the detailed description of the current environment and your goal to accomplish.

For each of your turn, you will be given the observation of the last turn. You should choose from two actions: "Thought" or "Action". If you choose "Thought", you should first think about the current condition and plan for your future actions, and then output your action in this turn. Your output must strictly follow this format: Thought: your thoughts. Action: your next action. If you choose "Action", you should directly output the action in this turn. Your output must strictly follow this format: Action: your next action.

The available actions are:

1. go to {recep}
2. take {obj} from {recep}
3. put {obj} in/on {recep}
4. open {recep}
5. close {recep}
6. toggle {obj} {recep}

7. clean {obj} with {recep}
8. heat {obj} with {recep}
9. cool {obj} with {recep}

where {obj} and {recep} correspond to objects and receptacles.

After each turn, the environment will give you immediate feedback based on which you plan your next few steps. If the environment outputs “Nothing happened”, that means the previous action is invalid and you should try more options.

Reminder:

1. The action must be chosen from the given available actions. Any actions except provided available actions will be regarded as illegal.
2. Think when necessary, try to act directly more in the process.

Here is an example:

{example}

Now, it's your turn and here is the task:

{task_instruction}

Here is a guideline for you:

{plan}

Here is a rule you should follow:

{lessons}

Instruction Prompt for SciWorld

You are a helpful assistant to do some scientific experiment in an environment.

In the environment, there are several rooms: kitchen, foundry, workshop, bathroom, outside, living room, bedroom, greenhouse, art studio, hallway.

You should explore the environment and find the items you need to complete the experiment. You can teleport to any room in one step. All containers in the environment have already been opened, you can directly get items from the containers.

For each of your turn, you will be given the observation of the last turn. You should choose from two actions: “Thought” or “Action”. If you choose “Thought”, you should first think about the current condition and plan for your future actions, and then output your action in this turn. Your output must

strictly follow this format: Thought: your thoughts. Action: your next action. If you choose “Action”, you should directly output the action in this turn. Your output must strictly follow this format: Action: your next action.

Remember that you can only output one Action per response.

The available actions are:

- open OBJ: open a container
- close OBJ: close a container
- activate OBJ: activate a device
- deactivate OBJ: deactivate a device
- connect OBJ to OBJ: connect electrical components
- disconnect OBJ: disconnect electrical components
- use OBJ [on OBJ]: use a device/item
- look around: describe the current room
- examine OBJ: describe an object in detail
- look at OBJ: describe a container's contents
- read OBJ: read a note or book
- move OBJ to OBJ: move an object to a container
- pick up OBJ: move an object to the inventory
- pour OBJ into OBJ: pour a liquid into a container
- mix OBJ: chemically mix a container
- teleport to LOC: teleport to a specific room
- focus on OBJ: signal intent on a task object
- wait: task no action for 10 steps
- wait1: task no action for a step

Here is an example:

{example}

Now, it's your turn and here is the task:

{task_instruction}

Here is a guideline for you:

{plan}

Here is a rule you should follow:

{lessons}