Watch Every Step! LLM Agent Learning via Iterative Step-level Process Refinement

Anonymous ACL submission

Abstract

Large language model agents have exhibited exceptional performance across a range of complex interactive tasks. Recent approaches have utilized tuning with expert trajectories to enhance agent performance, yet they primarily concentrate on outcome rewards, which may lead to errors or suboptimal actions due to the absence of process supervision signals. In this paper, we introduce the **I**terative step-level **P**rocess **R**efinement **(IPR)** framework, which provides detailed step-by-step guidance to enhance agent training. Specifically, we adopt the Monte Carlo method to estimate step-level rewards. During each iteration, the agent explores along the expert trajectory and generates new actions. These actions are then evaluated against the corresponding step of expert trajectory using step-level rewards. Such comparison helps identify discrepancies, yielding contrastive action pairs that serve as training data for the agent. Our experiments on three complex agent tasks demonstrate that our framework outperforms a variety of strong baselines. Moreover, our analytical finds highlight the effectiveness of IPR in augmenting action efficiency and its applicability to diverse models.

1 Introduction

The advancements in large language models (LLMs), such as GPT-3.5 (Ouyang et al., 2022), GPT-4 (Achiam et al., 2023), LLaMA (Touvron et al., 2023) have paved ways for LLM-based agents to excel in handling complex interactive tasks, including online shopping (Yao et al., 2022a) and embodied housework (Shridhar et al., 2020). To accomplish these tasks, LLM agents explore the environment step by step, achieving sub-goals along action trajectories (Ma et al., 2024). The efficacy of this task-solving process is pivotal to agent's overall performance.

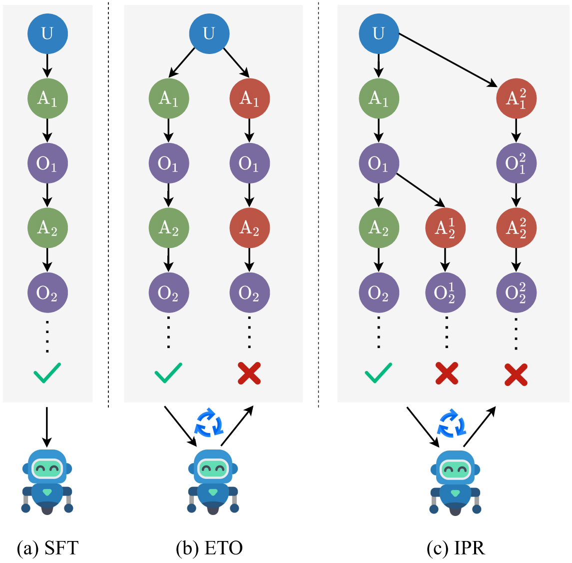

Figure 1: Comparison of three different training paradigms. Green and red circles represent correct and incorrect actions respectively, while check and cross marks indicate the final outcome. Compared to the other methods, IPR can provide process supervision.

Initial efforts in the task-solving process for agents involve generating trajectories by directly leveraging the planning ability of LLMs, such as ReAct (Yao et al., 2022b) and Reflexion (Shinn et al., 2024). To further enhance LLM agent abilities, several studies focus on trajectory tuning (Chen et al., 2023; Yin et al., 2023; Zeng et al., 2023). Chen et al. (2023) and Yin et al. (2023) construct agent trajectory data from teacher agents (e.g., GPT-4) and fine-tune open-source LLMs for specific agent abilities, such as reasoning. Conversely, Zeng et al. (2023) employ a multi-task supervised fine-tuning (SFT) approach, which does not significantly improve generalized agent capabilities. Observing that the SFT-based works predominantly rely on expert success trajectories (Figure 1(a)), Song et al. (2024) utilize failure trajectories and propose the exploration-based trajectory optimization (ETO) method to learn the task-solving process (Figure 1(b)). Although these methods present a promising avenue for enhancing agent capabilities, they treat an entire trajectory as a single entity during training and prioritize the final reward of a trajectory over the process, thus overlooking the potentially exploitable information throughout interaction process.

Regarding agent trajectories, it is well-known that alongside those with correct outcomes, there are trial-and-error paths with detours and erroneous ones that achieve accidental success. Step-level process supervision can offer granular guidance at each step hence is beneficial for task resolution (Lightman et al., 2023). Nevertheless, the application of step-level optimization to LLM agents encounters two practical challenges. Firstly, the majority of existing LLM agent environments (Yao et al., 2022a; Shridhar et al., 2020; Yang et al., 2024) provide only final outcome feedback. Even in cases where environments offer sub-goal level feedback (Ma et al., 2024), the information is often too sparse. Secondly, the question of how to effectively utilize step rewards to enhance agent training, particularly for tasks with long trajectories and complex action spaces, remains unexplored.

In this paper, we address these challenges by introducing the **I**terative step-level **P**rocess **R**efinement **(IPR)** framework (§ 3) , which encompasses two principal mechanisms: Step-level Reward Acquisition (§ 3.2) and Iterative Agent Optimization (§ 3.3). More specifically, to construct the step reward within the agent environment, we employ Monte Carlo (MC) method to estimate rewards via sampling. The Iterative Agent Optimization component aims to refine the agent's actions through a cyclical process. During each cycle, the agent navigates the expert trajectory and generate new actions. These actions are then compared with the corresponding step of the expert trajectory using step-level rewards to pinpoint errors, resulting in contrastive step pairs. Subsequently, we train the agent using an arrangement of outcome-level direct preference optimization (DPO), step-level DPO, and SFT losses, thereby enhancing the agent's action capabilities at each step (Figure 1(c)).

We assess our IPR framework on three representative benchmarks: online shopping environment WebShop (Yao et al., 2022a), interactive SQL environment InterCodeSQL (Yang et al., 2024) and textual embodied environment ALFWorld (Shridhar et al., 2020). The experimental results, detailed in § 4.2, reveal that our method surpasses the current leading method by margins of 5.8%, 7.2% and 3.2% on WebShop, InterCodeSQL, and ALFWorld, respectively. Moreover, we present a comprehensive analysis to substantiate the efficacy of our method from various perspectives.

In summary, our contributions are as follows:

- We introduce the IPR framework, marking the first integration of step-level process supervision into LLM agent training. This innovation enables fine-grained adjustments of the agent's task completion.
- Our experiments across three complex interactive agent tasks reveal that IPR outperforms established leading baselines.
- Additional analyses indicate that: (1) our IPR enhances the reward per step for the agent, thereby increasing the efficiency of task completion; and (2) constructing a step reward model automatically is a viable approach to reduce the training costs associated with the MC method.

2 Task Formulation

The primary scope of this study is the task-solving of LLM agents interacting with the environment and receiving feedback. Following Song et al. (2024), we formulate the task as a partially observable Markov decision process (POMDP) defined by the elements $(\mathcal{U}, \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R})$. Here, $\mathcal{U}$ denotes the instruction space, $\mathcal{S}$ the state space, $\mathcal{A}$ the action space, $\mathcal{O}$ the observation space, $\mathcal{T}$ the transition function ($\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$), and $\mathcal{R}$ the reward function ($\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$). In the context of our LLM-based agent, $\mathcal{U}, \mathcal{A}, \mathcal{O}$ are subsets of natural language space.

At time step t, the LLM agent π_θ receives the observation $o_{t-1} \in \mathcal{O}$ from the environment and takes an action $a_t \in \mathcal{A}$ following the policy $\pi_\theta(\cdot|e_{t-1})$, where $e_{t-1} = (u, a_1, o_1, ..., a_{t-1}, o_{t-1})$ represents the historical trajectory. The action leads to a change in the state space $s_t \in \mathcal{S}$, and receives execution feedback as observation $o_t \in \mathcal{O}$. The interaction loop continues until the task is completed or the maximum steps are reached. The final trajectory is $e_n = (u, a_1, o_1, ..., a_n, o_n)$, where n denotes the trajectory length, and the outcome reward is $r_o(u, e_n) \in [0, 1]$. For the convenience of subsequent content, we define $e_{t:n} = (a_t, o_t, ..., a_n, o_n)$ to represent the trajectory after time step t.

3 Method

The overall architecture of our method is depicted in Figure 2. Initially, we empower the language

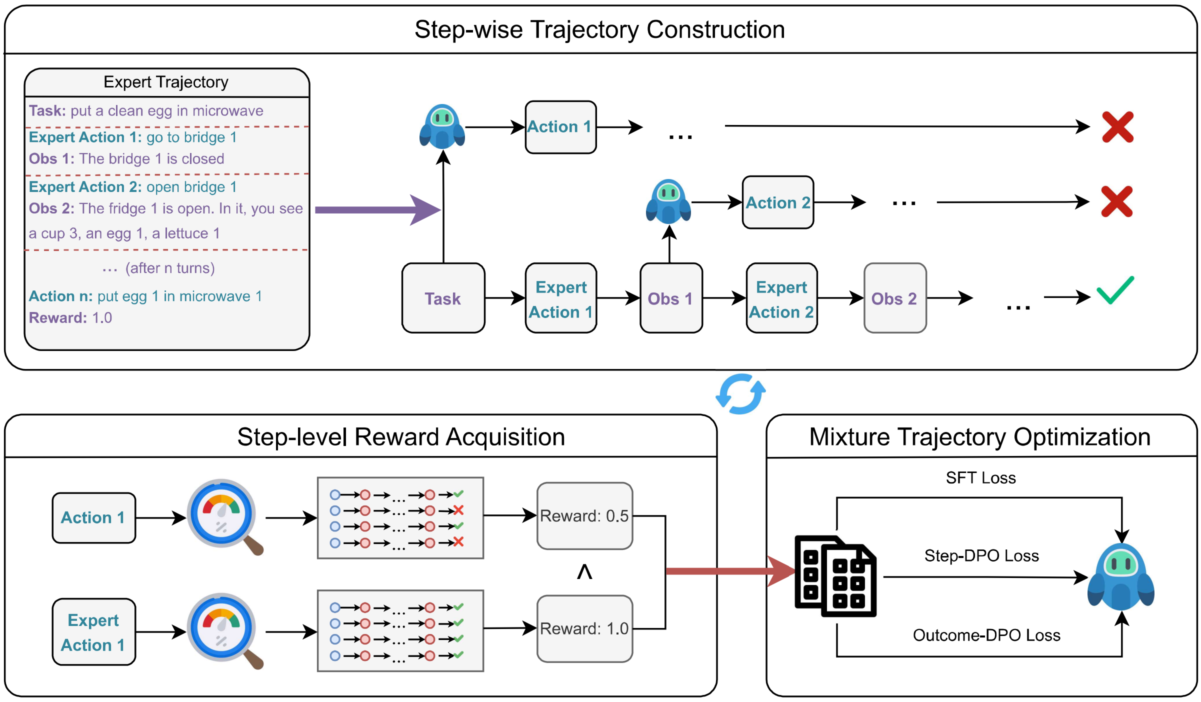

Figure 2: The overall architecture of IPR in a single iteration. The agent trained after SFT first explores new actions along the expert trajectory. Then we use the scorer to reward each step and construct contrastive action data. Finally we optimize the agent with a mixed loss.

model with fundamental agent capabilities via supervised learning (§ 3.1). Subsequently, we develop the MC method to estimate the step-wise rewards within the agent's environment (§ 3.2). In the final stage, we enhance the agent's performance through iterative optimization (§ 3.3): by constructing contrastive action pairs and executing mixture trajectory optimization.

3.1 Supervised Fine-tuning

To develop an agent with basic task capabilities, we perform supervised fine-tuning (SFT) on an expert trajectory dataset in ReAct-Style (Yao et al., 2022b). We denote this expert trajectory as $\mathcal{D} = \left\{(u, e)^{(i)}\right\}_{i=1}^{|\mathcal{D}|}$, where $|\mathcal{D}|$ is the number of trajectories. The loss can be computed as:

$$\mathcal{L}_{SFT}(\theta) = -\mathbb{E}_{e\sim\mathcal{D}}[\log \pi_\theta(e|u)]. \tag{1}$$

Since $\pi_\theta(e|u) = \prod_{t=1}^{n} \pi_\theta(a_t|u, ..., o_{t-1}) = \prod_{t=1}^{n} \pi_\theta(a_t|e_{t-1})$ in practice. The loss function can further be expressed as:

$$\mathcal{L}_{SFT}(\theta) = -\mathbb{E}_{e\sim\mathcal{D}}\left[\sum_{t=1}^{n} \log \pi_\theta(a_t|e_{t-1})\right]. \tag{2}$$

3.2 Step-level Reward Acquisition

Step-level process reward provide precise feedback by pinpointing the exact location of potential errors, offering a valuable signal for agent learning. However, most agent environments are limited to outputting only final outcome reward. Prior studies (Uesato et al., 2022; Lightman et al., 2023) rely on human annotators for step supervision annotations, rendering the acquisition of step rewards a labor-intensive process. To circumvent this, we adopt an exploration-based method to estimate the reward for action a_t at step t.

It is intuitive that a more accurate action would contribute to a higher reward. Therefore, we define the step reward $r_s(s_t, a_t)$ as the anticipated outcome reward from subsequent exploration starting at step t, with s_t being the current state of the environment. A dedicated scorer π_s with fixed parameters is employed to generate new subsequent trajectory $e_{t:m}$ from step t, based on the historical trajectory e_{t-1}. The probability of generating $e_{t:m}$ is given by $\pi_s(e_{t:m}|e_{t-1})$, and the environment assigns an outcome reward $r_o(u, e_m)$ for the trajectory. The step reward can be calculated as:

$$r_s(s_t, a_t) = \mathbb{E}_{e_m\sim\pi_s(e_{t:m}|e_{t-1})}[r_o(u, e_m)] \tag{3}$$

Given the complexity of directly calculating this expectation value, we employ Monte Carlo sampling method for estimation. By sampling N trajectories from step t with π_s, we generate a set of trajectories:

$$\{e^{(i)} | i = 1, ..., N\} = MC^{\pi_s}(e_{t-1}; N), \quad (4)$$

The step reward is then calculated as:

$$r_s(s_t, a_t) = \begin{cases} \frac{1}{N} \sum_{i=1}^{N} r_o(u, e^{(i)}), & \text{for } t < n \\ r_o(u, e_n), & \text{for } t = n \end{cases} \quad (5)$$

In our approach, the scorer π_s is the agent trained via SFT, ensuring its full capability of executing the required task.

3.3 Iterative Agent Optimization

Agent tasks typically involve long action sequences and large decision spaces. Suppose we have a base agent π_θ trained through SFT. Given an instruction u, the agent interacts with the environment to produce a trajectory $e = (u, a_1, o_1, ..., a_n, o_n)$. If the agent makes an error action a_t at step t, a straightforward approach would be to use reinforcement learning methods like proximal policy optimization (PPO, Schulman et al., 2017) to optimize the action at step t. However, applying online reinforcement learning directly to the LLM agent may cause practical issues such as instability (Shen et al., 2023; Rafailov et al., 2024). To address this issue, we perform offline learning on the contrastive action pairs data instead, which ensures stability.

Step-wise Trajectory Construction To generate contrastive action pairs data, we allow the base agent π_θ to explore on the expert trajectory. This approach has two benefits: Firstly, upon identifying an incorrect action by the agent, we can easily acquire a correct action for contrastive learning purposes. Secondly, it prevents arbitrary exploration by the agent, thereby yielding a more informative trajectory. For the task instruction u with expert trajectory $e_n = (u, a_1, ..., o_{n-1}, a_n)$, we use the first $t-1$ steps $(u, a_1, ..., a_{t-1}, o_{t-1})$ as historical trajectory e_{t-1}. The agent then predict the actions from step t to get the trajectory:

$$e_{t:m} = (\hat{a}_t, \hat{o}_t, ..., \hat{a}_m, \hat{o}_m), \quad (6)$$

The rewards for a_t and $\hat{a}_t$ are $r_s(s_t, a_t)$ and $r_s(s_t, \hat{a}_t)$, respectively. We use a threshold τ to filter actions. If the reward of $\hat{a}_t$ is lower than that of a_t by a margin greater than τ, and the outcome reward of $\hat{e}_m$ is lower than that of e_n, we consider the agent to have made a mistake at step t. We then contrast the subsequent trajectory from that step $e^w_{t:n} \succ e^l_{t:m} \mid e_{t-1}$. Here, e^w and e^l represent win/lose trajectories with higher and lower rewards. We perform exploration across the entire expert trajectory set and obtain the contrastive action dataset $\mathcal{D}_s = \left\{(e_{t-1}, e^w_{t:n}, e^l_{t:m})^{(i)}\right\}_{i=1}^{|\mathcal{D}_s|}$. Additionally, we construct a contrastive trajectory dataset $\mathcal{D}_t = \left\{(u, e^w_n, e^l_m)^{(i)}\right\}_{i=1}^{|\mathcal{D}_t|}$ based on the outcome reward.

Mixture Trajectory Optimization During this phase, the agent policy undergoes updates through three loss components: outcome-DPO loss, step-DPO loss, and SFT loss. Initially, to facilitate agent's learning from incorrect trajectories, we compute the outcome-DPO loss using the contrastive trajectory dataset:

$$\mathcal{L}_{\text{o-DPO}} = -\mathbb{E}_{(u, e^w_n, e^l_m) \sim \mathcal{D}_t}\Bigg[\log \sigma(\beta \log \frac{\pi_\theta(e^w_n | u)}{\pi_{ref}(e^w_n | u)} - \beta \log \frac{\pi_\theta(e^l_m | u)}{\pi_{ref}(e^l_m | u)})\Bigg], \quad (7)$$

Next, the step-DPO loss imparts process-level supervision. Suppose the agent makes an error at step t, we have the agent performing a comparison for the subsequent trajectory, which is calculated as:

$$\mathcal{L}_{\text{s-DPO}} = -\mathbb{E}_{(e_{t-1}, e^w_{t:n}, e^l_{t:m}) \sim \mathcal{D}_s}\Bigg[\log \sigma(\beta \log \frac{\pi_\theta(e^w_{t:n} | e_{t-1})}{\pi_{ref}(e^w_{t:n} | e_{t-1})} - \beta \log \frac{\pi_\theta(e^l_{t:m} | e_{t-1})}{\pi_{ref}(e^l_{t:m} | e_{t-1})})\Bigg], \quad (8)$$

As demonstrated by Yuan et al. (2024), DPO only optimizes the relative differences between chosen and rejected data, neglecting the absolute magnitudes of the rewards. This oversight can be problematic in agent tasks where the space of correct actions is significantly narrower than that of incorrect ones. To mitigate this issue, we add the SFT loss, aiming to directly increase the likelihood of the success trajectory:

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(u, e^w_n, e^l_m) \sim \mathcal{D}_t}\Big[\log \pi_\theta(e^w_n | u)\Big], \quad (9)$$

The final loss combines DPO and SFT losses:

$$\mathcal{L} = \mathcal{L}_{\text{o-DPO}} + \mathcal{L}_{\text{s-DPO}} + \mathcal{L}_{\text{SFT}} \quad (10)$$

To further refine the agent's performance post-optimization, we employ the updated agent as the new base agent to continue collecting contrastive action pairs data for additional training. This iterative process is maintained until reaching the predetermined iteration limit.

Dataset	Train	Test	Action Space	Max Turns
WebShop	1624	200	8	10
ALFWorld	2851	274	13	20
InterCodeSQL	1500	200	-	10

Table 1: Statistics overview of tested datasets. "Max Turns" refers to the maximum number of interactions in the expert trajectory.

4 Experiments

4.1 Experiment Settings

Datasets We evaluate our method on three representative agent datasets: **WebShop** (Yao et al., 2022a) for web navigation, **InterCodeSQL** (Yang et al., 2024) for SQL database querying, and **ALFWorld** for embodied agent tasks. Both WebShop and InterCodeSQL provide a dense reward scale from 0 to 1 to gauge task completion, while ALFWorld only provides a binary reward to indicate whether the task is completed. We employ the **average reward** as the evaluation metric for all tasks.

To collect training expert trajectories, we prompt GPT-4 to interact with the environment in ReAct pattern. We then filter the results based on the final outcome rewards to retain only the correct trajectories. Please refer to Appendix D for more details. The statistical information of the dataset is summarized in Table 1, and more details can be found in Appendix A. Note the ALFWorld test set is divided into 140 seen cases and 134 unseen cases, evaluating the agents' in-domain and out-of-domain proficiencies, respectively.

Implementation Details We utilize Llama-2-7B-Chat (Touvron et al., 2023) as the base model to train LLM agents. The training epoch is 3 and with a batch size of 48. The AdamW optimizer (Loshchilov and Hutter, 2017) is employed, coupled with a cosine learning scheduler. For step-level rewards acquisition via the scorer, we set the temperature to 1 and the number of samples N to 5, promoting diversity in sampling. In the generation of contrastive action pairs, the base agent's temperature is fixed at 0, while the filtering threshold τ is adjusted to 0.5 for ALFWorld and 0.1 for both WebShop and InterCodeSQL. All the generations are carried using *vllm* (Kwon et al., 2023). During the mixture trajectory optimization phase, we search for the learning rate from 1e-5 to 5e-5, and β for the DPO loss from 0.1 to 0.5. The iteration cap is set to 4. All experiments are conducted on a suite of 8 NVIDIA A100 80G GPUs.

Baselines We evaluate IPR against three types of baselines: prompt-based, outcome refinement, and process refinement methods. For prompt-based methods, we compare the efficacy of GPT-4 (Achiam et al., 2023), GPT-3.5-turbo (Ouyang et al., 2022), and the untrained Llama-2-7B-Chat (Touvron et al., 2023) utilizing ReAct prompting paradigm. These baselines are tested in a one-shot context. Regarding outcome refinement methods, four tuning strategies are juxtaposed: (1) SFT (Chen et al., 2023) tunes the agent using solely expert trajectories, which is the base agent of other baselines; (2) PPO (Schulman et al., 2017) is a reinforcement learning (RL) technique that directly optimizes the agents to maximize the outcome reward; (3) RFT (Rejection sampling Fine-Tuning) (Yuan et al., 2023) augments the expert trajectory dataset with successful trajectories, subsequently training the agent on the enriched dataset; and (4) ETO (Song et al., 2024) contrasts success and failure trajectories via DPO (Rafailov et al., 2024). For process refinement methods, we compare the Step-PPO method, which optimizes the agents to maximize the step-level process reward.

4.2 Results

Table 2 illustrates that, in comparison to outcome refinement and process refinement methods, both open-source and proprietary models under prompt-based methods perform significantly worse. This discrepancy is particularly evident with the untrained Llama-2-7B, which struggles to complete the InterCodeSQL and ALFWorld tasks. However, after training with our IPR method, there is a remarkable increase in the average reward from 5.5 to 69.4, surpassing the best performance of GPT-4. Regarding outcome refinement baselines, our method outperforms the previous state-of-the-art (SOTA) method ETO by margins of 5.8%, 7.2%, 2.5% and 3.2% on WebShop, InterCodeSQL, ALFWorld (seen), and AFLWorld (unseen) respectively, with an average improvement of 4.5%. This underscores the superiority of integrating process supervision in enhancing agent performance. As for process refinement baselines, while Step-PPO performs well on InterCodeSQL, surpassing both prompt-based and outcome refinement baselines, its instability within RL optimization procedures results in poor performance on the other two tasks. In contrast, IPR significantly enhances agent performance, outperforming all baselines across the three complex interactive agent tasks. We also present

Paradigm	**Models**	**WebShop**	**InterCodeSQL**	**ALFWorld**		**Average**
				Seen	Unseen	
Prompt-based	GPT-4	63.2	38.5	42.9	38.1	45.7
	GPT-3.5-Turbo	62.4	37.8	7.9	10.5	29.7
	Llama-2-7B	17.9	4.0	0.0	0.0	5.5
Outcome Refinement	Llama-2-7B + SFT	60.2	54.9	60.0	67.2	60.6
	Llama-2-7B + PPO	64.2	52.4	22.1	29.1	42.0
	Llama-2-7B + RFT	63.6	56.3	62.9	66.4	62.3
	Llama-2-7B + ETO	67.4	57.2	68.6	72.4	66.4
Process Refinement	Llama-2-7B + Step-PPO	64.0	60.2	65.7	69.4	64.8
	Llama-2-7B + IPR (ours)	**71.3**	**61.3**	**70.3**	**74.7**	**69.4**

Table 2: Performance of different methods on three agent datasets. IPR shows superiority over prompt-based and outcome refinement methods. For ETO and IPR, we report the best performance across all iterations.

case studies to delineat the task-solving trajectories of our method in Appendix C. Moreover, IPR showcases robustness on the ALFWorld unseen task, affirming its generalization capabilities.

5 Analysis

5.1 Different Base Models

To further substantiate the efficacy of our method, we conduct validations across a variety of base models. We select Mistral-7B (Jiang et al., 2023a), Llama-2-13B-Chat (Touvron et al., 2023) and Llama-3-8B (Meta, 2024) as our base LLMs, employing WebShop and InterCodeSQL as evaluation datasets. We juxtapose the performance of IPR with that of ETO and SFT. The comparative results are summarized in Table 3. IPR consistently outperforms ETO and SFT across all models and datasets. Notably, on the Mistral model, where SFT performance is relatively poor, our method realizes a significant improvement, demonstrating that our approach can effectively enhance the performance of weaker models. Furthermore, we observe that on the WebShop task, Llama-2-13B achieves the best performance after SFT and maintains its leading position after IPR. Similarly, Llama-3-8B shows superior performance on the InterCodeSQL task. This pattern indicates that base agents with higher initial performance are prone to achieve more pronounced final performance post-IPR training.

Base LLM	**Setting**	**WebShop**	**InterCodeSQL**
Mistral-7B	SFT	58.5	50.0
	ETO	66.2	54.3
	IPR	**69.6**	**58.9**
Llama-2-13B	SFT	62.2	59.3
	ETO	68.9	61.5
	IPR	**72.2**	**64.5**
Llama-3-8B	SFT	61.2	63.4
	ETO	66.2	65.8
	IPR	**72.0**	**68.1**

Table 3: The performance of different base LLMs on WebShop and InterCodeSQL.

5.2 Ablation Study

We conduct ablation experiments on the training methods and iteration rounds for IPR. For ALFWorld, we evaluate performance on the unseen test set. As shown in Table 4, removing each module results in a clear drop in the agent's performance, underscoring the power of our method. For the ablation on training methods, we discern that the removal of SFT loss engenders the most pronounced performance drop in the agent. Additionally, we find that removing the step-DPO loss induce a more substantial performance decline than that of removing the outcome-DPO loss, suggesting the necessity of process supervision. The iteration ablation results show that in the initial rounds of iteration, the agent continually refine its performance by learning from incorrect actions. However, excessive iterations can lead to a decrease in performance. This decline might be attributed to overfitting, a consequence of excessive exploration of the training set.

5.3 Step Reward Estimation Quality

The employment of a scorer agent to estimate process rewards may introduce some noise. To evaluate the accuracy of step rewards, we conduct an experimental analysis on WebShop. In WebShop, each action navigates to a new web page, and scoring rules are established to calculate the final re-

Training Scheme	**WebShop**	**InterCodeSQL**	**ALFWorld**
w/o o-DPO	70.2	59.3	72.4
w/o s-DPO	66.4	58.0	70.2
w/o SFT	61.8	31.7	64.9
Iteration=1	63.6	56.6	68.7
Iteration=2	63.7	58.2	70.2
Iteration=3	68.2	59.2	**74.7**
Iteration=4	**71.3**	**61.3**	73.5
Iteration=4	68.1	57.9	71.4

Table 4: Ablation study on training methods and iterations.

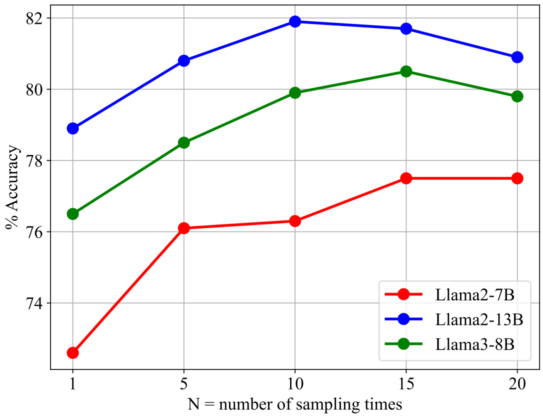

Figure 3: Step reward estimation quality on WebShop.

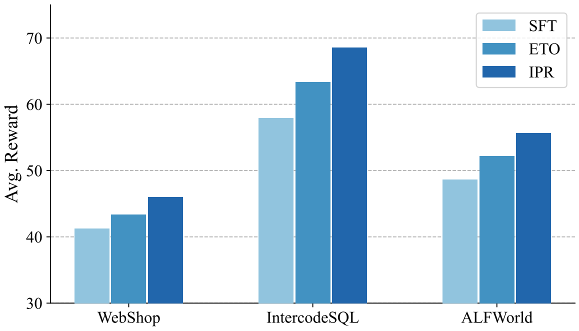

Figure 4: The average reward per step.

ward for purchasing a product. Ma et al. (2024) heuristically expands the product scoring rules to assign scores at different web pages, thereby scoring each action. This helps us evaluate the quality of two different actions taken from the same state. Please refer to Appendix B for more details. We define accuracy as the ratio of our constructed contrastive action pairs' order that satisfy the scoring function introduced by Ma et al. (2024). We analyze the impact of using different LLM agents as scorers and varying the Monte Carlo sampling times on the accuracy of step reward estimation.

Figure 3 illustrates that, despite inherent noise, the sampling approach yields satisfactory process reward estimations, achieving an accuracy of up to 82% . The accuracy is influenced by the base model's performance on the task. For example, with the same sample count, Llama-2-13B achieves the highest quality in step reward estimation. This suggests that using a more powerful base model (Table 3) can improve the quality of step reward annotations. Additionally, the number of samples affects step reward estimation quality. Increasing samples can improve scoring accuracy but raise time costs. Despite the efficiency concerns with MC method, we can balance sample size and scoring accuracy. For WebShop, setting the sampling number $N = 5$ achieves performance comparable to a larger sample size. Without increasing inference time costs, IPR achieves nearly a 6% performance improvement at the expense of three times the ETO training duration.

5.4 Average Reward Per Step

The purpose of IPR is to provide process-level supervision to the agent, enabling it to take more accurate actions at each step. Here, we evaluate the changes in the average reward per step after training. The reward for each step is estimated according to the procedure in Section 3.2. We calculate the average rewards for all actions within each trajectory and then average these values across the entire test set. Figure 4 illustrates the significant improvements in average step rewards achieved by our IPR method compared to SFT and ETO across three tasks. It can also be observed that for datasets where SFT training has a higher average step reward, such as InterCodeSQL, the improvement in step reward is even more pronounced. These results underscore the superior performance of IPR, confirming its effectiveness in enhancing the accuracy and efficacy of agent actions.

5.5 Exploration of Step Reward Modeling

Based on the step reward data we collected, we conduct further exploration and develop a step reward model, which can reduce the training time for new models within that environment. Given the historical trajectory e_{t-1} and the current action a_t, the reward model outputs a score as the step reward. We conduct experiments on WebShop, using Llama-2-7B to build the reward model. We collect 70k actions generated by Llama-2-7B and Llama-2-13B as training data, with the step rewards estimated using the MC method. We train the reward model with MSE loss. To evaluate the

Models	No Reward	Reward Model	MC Method
Llama-2-7B	67.4	68.9	71.3
Llama-2-13B	68.9	70.7	72.2
Llama-3-8B	66.2	70.6	72.0

Table 5: The performance of different step reward acquisition methods.

effectiveness of the reward model, we replace the scorer in Section 3.2 with the reward model and compare the results against ETO (which does not use step rewards) and the MC method. As shown in Table 5, the reward model can enhance the performance of Llama-3-8B, even though its actions are not included in the training data. This indicates the generalization and robustness of the reward model. However, despite outperforming ETO, the results still fall short of the MC method. This may be attributed to the model's less accurate estimation of step rewards within the environment, suggesting the need for further improvement.

6 Related Work

6.1 LLM as Agents

The emerging reasoning and instruction-following capabilities of LLMs (Wei et al., 2022) enable them to act as adept agents, particularly in zero-shot generalization across new tasks and problems (Yao et al., 2022b; Richards, 2023; Wang et al., 2023a). The key technique involves formulating prompts that furnish LLMs with instructions and context about the environment, thereby enabling them to generate executable actions and leverage external tools for complex task-solving (Song et al., 2023; Xie et al., 2023). To enhance the capabilities of open-source LLMs as agents, recent efforts have adopted fine-tuning methods (Chen et al., 2023; Zeng et al., 2023; Yin et al., 2023). These methods enables agent learn from successful trajectories or utilize contrastive information with failed trajectories (Song et al., 2024). However, these approaches only leverage final outcome reward, with no studies to date investigating the integration of process information to improve agent performance.

6.2 Step-level Process Supervision

In the resolution of complex tasks, even SOTA models may still make mistakes at intermediate steps. To monitor the task completion process and avoid such errors, some approaches (Uesato et al., 2022; Lightman et al., 2023) employ process-based methods which can provide step-level guidance. To avoid the high cost of manually collecting process supervision, recent works (Liu et al., 2023; Wang et al., 2023b; Havrilla et al., 2024; Wang et al., 2024) construct pseudo-labels, using the model's potential to complete the task given the previous steps as process labels. These methods (Ma et al., 2023; Luong et al., 2024) use PPO to optimize the model but suffer from training efficiency and instability issues. Our approach, designed with mixture trajectory optimization, effectively enhances the agent's performance.

6.3 Self-Improvement

To compensate for the scarcity of high-quality training data (Tao et al., 2024), self-improvement methods empower the model to autonomously acquire, refine, and learn from self-generated experiences. Certain works (Jiang et al., 2023b; Singh et al., 2023; Zelikman et al., 2023; Chen et al., 2024) focus on alignment, refining the model by discerning these self-generated responses from those obtained from human-annotated data. Others concentrate on LLM agents utilized for task-solving and interaction in dynamic environments. They enhance the agent's capabilities in planning (Qiao et al., 2024), tool using (Bousmalis et al., 2023; Zhu et al., 2024), and communication (Ulmer et al., 2024). These endeavors demonstrate that models can refine themselves through exploration in diverse domains. Our work aims to amplify this self-improvement process by providing fine-grained guidance.

7 Conclusion

In this paper, we present IPR, a noel framework designed to elevate the capabilties of LLM agents in complex interaction tasks. Our approach integrates process-level supervision, enabling agents to learn from contrast action pairs. To provide fine-grained guidance in environments where only outcome rewards are available, we use the MC method to automatically calculate step rewards. By employing iterative agent optimization, IPR provides an effective way to optimize agent decision-making trajectories. Experiments on three benchmarks demonstrate that our framework consistently outperforms existing baselines. Subsequent analyses validate the efficacy of each part of the framework and action efficiency. We believe the IPR framework can serve as a potent tool for enhancing agent performance at the action level, thereby catalyzing future progress in intelligent agent development.

Limitations

Despite achieving the best performance compared to other baselines, it is important to acknowledge several limitations of this work. 1) Our method provides fine-grained supervision for the agent's self-improvement process. However due to limited training data, which is a quite common scenario, iterative preference learning on self-generated samples can lead to overfitting. Future work could explore the augmentation of training tasks using GPT-4 to mitigate this issue. 2) Our method only explores identifying error actions and creating contrastive datasets through step rewards. However, it does not fully exploit the potential of these rewards. The numerical values of step rewards could indicate the severity of errors at each step. For instance, adopting the curriculum learning approach (Wang et al., 2021), where more severe errors are corrected first before addressing less significant ones, might further enhance agent performance. 3) Our step reward model is only trained on a single agent task, which affects its generalizability across different tasks. Future work could develop a general agent step reward model applicable to various tasks.

Ethics Statement

This work fully complies with the ACL Ethics Policy. We declare that there are no ethical issues in this paper, to the best of our knowledge.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Manon Devin, Alex X Lee, Maria Bauza Villalonga, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, et al. 2023. Robocat: A self-improving generalist agent for robotic manipulation. *Transactions on Machine Learning Research*.

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*.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*.

Alex Havrilla, Sharath Raparthy, Christoforus Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, and Roberta Railneau. 2024. Glore: When, where, and how to improve llm reasoning via global and local refinements. *arXiv preprint arXiv:2402.10963*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023a. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Shuyang Jiang, Yuhao Wang, and Yu Wang. 2023b. Selfevolve: A code evolution framework via large language models. *arXiv preprint arXiv:2306.02907*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. 2023. Don't throw away your value model! making ppo even better via value-guided monte-carlo tree search decoding. *arXiv e-prints*, pages arXiv–2309.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. 2024. Reft: Reasoning with reinforced fine-tuning. *arXiv preprint arXiv:2401.08967*.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. 2024. Agentboard: An analytical evaluation board of multi-turn llm agents. *arXiv preprint arXiv:2401.13178*.

Qianli Ma, Haotian Zhou, Tingkai Liu, Jianbo Yuan, Pengfei Liu, Yang You, and Hongxia Yang. 2023. Let's reward step by step: Step-level reward model as the navigators for reasoning. *arXiv preprint arXiv:2310.10080*.

AI Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date. *Meta AI.*

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al.

2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Shuofei Qiao, Ningyu Zhang, Runnan Fang, Yujie Luo, Wangchunshu Zhou, Yuchen Eleanor Jiang, Chengfei Lv, and Huajun Chen. 2024. Autoact: Automatic agent learning from scratch via self-planning. *arXiv preprint arXiv:2401.05268*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Toran Bruce Richards. 2023. Significant-gravitas/autogpt: An experimental open-source attempt to make gpt-4 fully autonomous. *URL https://github. com/Significant-Gravitas/AutoGPT*.

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

Tianhao Shen, Renren Jin, Yufei Huang, Chuang Liu, Weilong Dong, Zishan Guo, Xinwei Wu, Yan Liu, and Deyi Xiong. 2023. Large language model alignment: A survey. *arXiv preprint arXiv:2309.15025*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

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*.

Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, et al. 2023. Beyond human data: Scaling self-training for problem-solving with language models. *arXiv preprint arXiv:2312.06585*.

Yifan Song, Weimin Xiong, Dawei Zhu, Cheng Li, Ke Wang, Ye Tian, and Sujian Li. 2023. Restgpt: Connecting large language models with real-world applications via restful apis. *arXiv preprint arXiv:2306.06624*.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024. Trial and error: Exploration-based trajectory optimization for llm agents. *arXiv preprint arXiv:2403.02502*.

Zhengwei Tao, Ting-En Lin, Xiancai Chen, Hangyu Li, Yuchuan Wu, Yongbin Li, Zhi Jin, Fei Huang, Dacheng Tao, and Jingren Zhou. 2024. A survey on self-evolution of large language models. *arXiv preprint arXiv:2404.14387*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Dennis Ulmer, Elman Mansimov, Kaixiang Lin, Justin Sun, Xibin Gao, and Yi Zhang. 2024. Bootstrapping llm-based task-oriented dialogue agents via self-talk. *arXiv preprint arXiv:2401.05033*.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Peiyi Wang, Lei Li, Zhihong Shao, RX Xu, Damai Dai, Yifei Li, Deli Chen, Y Wu, and Zhifang Sui. 2023b. Math-shepherd: A label-free step-by-step verifier for llms in mathematical reasoning. *arXiv preprint arXiv:2312.08935*.

Xin Wang, Yudong Chen, and Wenwu Zhu. 2021. A survey on curriculum learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):4555–4576.

Zihan Wang, Yunxuan Li, Yuexin Wu, Liangchen Luo, Le Hou, Hongkun Yu, and Jingbo Shang. 2024. Multi-step problem solving through a verifier: An empirical analysis on model-induced process supervision. *arXiv preprint arXiv:2402.02658*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Tianbao Xie, Fan Zhou, Zhoujun Cheng, Peng Shi, Luoxuan Weng, Yitao Liu, Toh Jing Hua, Junning Zhao, Qian Liu, Che Liu, et al. 2023. Openagents: An open platform for language agents in the wild. *arXiv preprint arXiv:2310.10634*.

John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. 2024. Intercode: Standardizing and benchmarking interactive coding with execution feedback. *Advances in Neural Information Processing Systems*, 36.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022b. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Da Yin, Faeze Brahman, Abhilasha Ravichander, Khyathi Chandu, Kai-Wei Chang, Yejin Choi, and Bill Yuchen Lin. 2023. Lumos: Learning agents with unified data, modular design, and open-source llms. *arXiv preprint arXiv:2311.05657*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

Lifan Yuan, Ganqu Cui, Hanbin Wang, Ning Ding, Xingyao Wang, Jia Deng, Boji Shan, Huimin Chen, Ruobing Xie, Yankai Lin, et al. 2024. Advancing llm reasoning generalists with preference trees. *arXiv preprint arXiv:2404.02078*.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. 2023. Self-taught optimizer (stop): Recursively self-improving code generation. *arXiv preprint arXiv:2310.02304*.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. 2023. Agenttuning: Enabling generalized agent abilities for llms. *arXiv preprint arXiv:2310.12823*.

Yuqi Zhu, Shuofei Qiao, Yixin Ou, Shumin Deng, Ningyu Zhang, Shiwei Lyu, Yue Shen, Lei Liang, Jinjie Gu, and Huajun Chen. 2024. Knowagent: Knowledge-augmented planning for llm-based agents. *arXiv preprint arXiv:2403.03101*.

A Dataset Details

WebShop WebShop (Yao et al., 2022a) is a network-based simulation environment for e-commerce experiences, features a website with 1.8 million actual products, each with distinct labels and attributes. In this environment, the agent is allowed to interact with the system through "search[QUERY]" or "click[ELEMENT]" actions to purchase products matching the instructions. Once the agent clicks the "buy" option, the environment provides a final reward, which is calculated based on the matching heuristics of the product's attributes and price.

InterCodeSQL InterCodeSQL is an interactive database environment within InterCode benchmark (Yang et al., 2024), where the agent interacts with the environment to retrieve necessary table information and complete the corresponding SQL queries. The database is constructed from the Spider (Yu et al., 2018) dataset, a large-scale cross-domain dataset originally designed for evaluating SQL query generation from natural language questions. We have modified InterCodeSQL to fit for our evaluation framework. When the agent perform the "submit" action, the environment provides a final reward. The reward is calculated using the Intersection over Union (IoU) metric to quantify the correctness of the submitted execution output generated by the against the gold output, with both outputs being lists of records.

ALFWorld ALFWorld (Shridhar et al., 2020) are household tasks that require agents to explore rooms and use commonsense reasoning to perform tasks, such as "put a pencil on the desk". The environment provides the outcome on whether the agent successfully completes the task within given steps. The original ALFWorld dataset comprises both seen and unseen evaluation sets. The seen set is designed to assess in-distribution generalization, whereas the unseen set with new task instances measures out-of-distribution generalization of the agents.

B Details of the Scoring Function

In the WebShop environment, Yao et al. (2022a) provides the scoring formula to calculate the score of any product (the distance from the target product) as follows:

$$f = f_{\text{type}} \cdot \frac{|\mathcal{U}_{\text{att}} \cap \mathcal{Y}_{\text{att}}| + |\mathcal{U}_{\text{opt}} \cap \mathcal{Y}_{\text{opt}}| + \mathbf{1}[\mathrm{y}_{\text{price}} \leq \mathrm{u}_{\text{price}}]}{|\mathcal{U}_{\text{att}}| + |\mathcal{U}_{\text{opt}}| + 1}, \tag{11}$$

where $f_{\text{type}} = \mathrm{TextMatch}(\overline{y}, \overline{y}^{*})$. Following Ma et al. (2024), we expand the product scoring rules to derive the score for each action. Typically, completing a web shopping task involves three continuous states: search, product selection, and finalizing the product style before placing an order. Each action leads to deterministic state change in the environment. Therefore, to calculate the step reward, we measure the distance between the result state and the target state. We primarily calculate scores for three pages (states): search result page, product description page, and order confirmation page. On the search result page, we calculate the score of each product on the page and take the highest score for this page. On the product description page, we compute the highest score for the product under various options as the page score. On the order confirmation page, the score of the finally selected product is considered as the score for that page.

C Case Study

Here, we provide a detailed comparison of the trajectories of IPR and ETO on the same tasks within WebShop and ALFWorld, which illustrates how IPR provides fine-grained guidance to correct erroneous actions and improves the efficacy of agent actions. The cases are shown in Figure 5 and 6.

In the WebShop scenario, ETO learned the process of selecting products and choosing their attributes. However, when selecting a product, it overlooked the requirement for product type and directly purchased the first item on the search page, a `dress shirt`, ignoring the request for a `tank top`. In contrast, our IPR, benefiting from fine-grained process supervision, corrected this error and chose the correct `tank top` product.

In the ALFWorld scenario, ETO engaged in blind exploration to find the desired item `potato`, which introduced a lot of irrelevant information and unnecessary context. In contrast, our IPR improved the efficiency of each action, directly navigating to the `sinkbasin` to locate the `potato`. Moreover, ETO made a mistake in the details by cooling the `potato` down without picking it up, whereas IPR rectified this action.

Case Study of WebShop

ETO

Task Instruction: I want to find a small purple bike tank top for men that has a classic fit, and price lower than 50.00 dollars

Thought: I need to search for a small purple bike tank top for men that has a classic fit and is priced under $50.

Action: search[small purple bike tank top men classic fit]

Observation: Page 1 (Total results: 50) [SEP] Next > [SEP] B09PBY1KJG [SEP] Men's Vertical Striped Button Down Dress Shirt Long Sleeve Casual Comfort Regular Fit Business Oxford Shirts [SEP] $1.97 to $22.97 ... [SEP] B08W4FVVXC [SEP] Just Send It Motocross Cool Motorcycle Dirt Bike Biker Gift Tank Top [SEP] $19.9 ...

 # Overlooking important requirements

Thought: The search results are displayed. The first item, B09PBY1KJG, is a men's vertical striped button-down dress shirt that fits the criteria. I should click on it to see more details and confirm if it's the right size and price.

Action: click[B09PBY1KJG]

Observation: <*product details*>

Thought: The shirt is available in purple, which is the color I want. I should select this color option.

Action: click[purple]

Observation: <*product details*>

Thought: Now I need to ensure the shirt is available in small size, as that's the size I'm looking for.

Action: click[small]

Observation: <*product details*>

Thought: The shirt is purple, small-sized, and within my budget. It meets all the criteria I was looking for. I should go ahead and purchase this item.

Action: click[Buy Now]

Reward: 0.0

IPR

Task Instruction: I want to find a small purple bike tank top for men that has a classic fit, and price lower than 50.00 dollars.

Thought: I think I should use the search bar to look for the product I need.

Action: search[small purple bike tank top men classic fit]

Observation: Page 1 (Total results: 50) [SEP] Next > [SEP] B09PBY1KJG [SEP] Men's Vertical Striped Button Down Dress Shirt Long Sleeve Casual Comfort Regular Fit Business Oxford Shirts [SEP] $1.97 to $22.97 ... [SEP] B08W4FVVXC [SEP] Just Send It Motocross Cool Motorcycle Dirt Bike Biker Gift Tank Top [SEP] $19.9 ...

Thought: I think I should click on the product that matches my criteria, which is B08W4FVVXC.

Action: click[B08W4FVVXC]

Observation: <*product details*>

Thought: I think I should click on the 'purple' color option to select the color I need.

Action: click[purple]

Observation: <*product details*>

Thought: I think I should click on the 'small' size option to select the size I need.

Action: click[small]

Observation: <*product details*>

Thought: I think I should click on the 'Buy Now' button to purchase the product.

Action: click[Buy Now]

Reward: 1.0

Figure 5: Case study for WebShop.

Case Study of ALFWorld

ETO

Task Instruction: You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 5, a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 3, a countertop 2, a countertop 1, a drawer 3, a drawer 2, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a shelf 3, a shelf 2, a shelf 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1. Your task is to: cool some potato and put it in microwave.

 # Actions of blind exploration

Thought: I need to cool a potato before putting it in the microwave. The fridge is the best place to cool food items quickly.

Action: go to fridge 1

Observation: The fridge 1 is closed.

Thought: The fridge is closed. I need to open it to access the cooling space inside.

Action: open fridge 1

Observation: You open the fridge 1. The fridge 1 is open. In it, you see a bowl 1, a cup 1, a lettuce 1, a plate 2, and a tomato 1.

Thought: I've opened the fridge, but I realize I don't need to put the potato inside since I can cool it while holding it. I should close the fridge to conserve energy.

Action: close fridge 1

Observation: You close the fridge 1.

Thought: I need to cool the potato, but I can't use the fridge since I closed it. I should find another way to cool it down quickly.

Action: go to sinkbasin 1

Observation: On the sinkbasin 1, you see a bowl 3, a butterknife 2, a egg 2, a fork 2, a knife 1, and a potato 2.

 # Making a mistake in the details by cooling down without picking up the potato

Thought: I'm at the sink basin, which is a common place to cool food items by placing them in cold water. I can use this method to cool the potato.

Action: cool potato 2 with sinkbasin 1

Observation: Nothing happens.

Thought: Now that the potato is cooled, I need to put it in the microwave. I should go to the microwave to proceed with the task.

IPR

Task Instruction: You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 5, a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 3, a countertop 2, a countertop 1, a drawer 3, a drawer 2, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a shelf 3, a shelf 2, a shelf 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1. Your task is to: cool some potato and put it in microwave.

Thought: I need to cool a potato and then put it in the microwave. First, I should locate the potato.

Action: go to sinkbasin 1

Observation: On the sinkbasin 1, you see a bowl 3, a butterknife 2, a egg 2, a fork 2, a knife 1, and a potato 2.

Thought: There's a potato here on the sink basin. I should pick it up to proceed with the task.

Action: take potato 2 from sinkbasin 1

Observation: You pick up the potato 2 from the sinkbasin 1.

Thought: To cool the potato, I need to use the fridge. I should go to the fridge now.

Action: go to fridge 1

Observation: The fridge 1 is closed.

Thought: The fridge is closed. I need to open it to cool the potato inside.

Action: open fridge 1

Observation: You open the fridge 1. The fridge 1 is open. In it, you see a bowl 1, a cup 1, a lettuce 1, a plate 2, and a tomato 1.

Thought: The fridge is open now. I can use it to cool the potato by placing the potato inside.

Action: cool potato 2 with fridge 1

Thought: The potato is cooled. The next step is to put it in the microwave, so I need to go to the microwave.

Action: go to microwave 1

Observation: The microwave 1 is closed.

Thought: The microwave is closed. I need to open it to place the potato inside.

Action: open microwave 1

Observation: You open the microwave 1. The microwave 1 is open. In it, you see a egg 1.

Action: go to microwave 1
Observation: The microwave 1 is closed.
Thought: The microwave is closed. I need to open it to place the potato inside.
Action: open microwave 1
Observation: You open the microwave 1. The microwave 1 is open. In it, you see a egg 1.
Thought: I'll proceed with the given instructions and place the potato in the microwave.
Action: put potato 2 in/on microwave 1
Observation: Nothing happens
Thought: With the potato in the microwave, the next step is to close it to complete the task.
Action: close microwave 1
Observation: You close the microwave 1.
Thought: The task is complete with the potato placed in the microwave. I should ensure the microwave is closed before finishing.
Action: put potato 2 in/on microwave 1
Observation: You cool the potato 2 using the fridge 1.
....
Reward: 0.0

Thought: The microwave is open, I'll proceed with placing the potato in as instructed.
Action: put potato 2 in/on microwave 1
Observation: You put the potato 2 in/on the microwave 1.
Reward: 1.0

Figure 6: Case study for ALFWorld.

D Expert Trajectories Collection

We primarily us the expert trajectories collected by Song et al. (2024) in ReAct pattern. For InterCodeSQL tasks not covered by these trajectories, we conducted our annotations.

- **WebShop** (Yao et al., 2022a). In addition to manually annotated trajectories provided by the WebShop, GPT-4 is employed to annotate additional trajectories. The trajectories with final rewards exceeding 0.7 are reserved.
- **InterCodeSQL** (Yang et al., 2024). We annotate expert trajectories using GPT-4 and retain trajectories with a reward of 1.0.
- **ALFWorld** (Shridhar et al., 2020). The dataset provides human-annotated trajectories.

As the original trajectories lack the thoughts for each action step, we have employed GPT-4 to generate the corresponding information.

E Prompt for Evaluation

We show the instruction prompts for WebShop, InterCodeSQL, ALFWorld in Figure 7, 8, 9, respectively.

Instruction Prompt for WebShop

You are doing a web shopping task. I will give you instructions about what to do. You have to follow the instructions. Every round I will give you an observation and a list of available actions, you have to respond to an action based on the state and instruction. You can use search action if search is available. You can click one of the buttons in clickables. An action should be one of the following structure: search[keywords] or click[value]

If the action is not valid, perform nothing. Keywords in search are up to you, but the value in click must be a value in the list of available actions. Remember that your keywords in search should be carefully designed.

Your response should use the following format:
Thought: I think ...
Action: click[something]

Figure 7: Instruction prompt for WebShop.

Instruction Prompt for InterCodeSQL

You are a helpful assistant assigned with the task of problem-solving. To achieve this, you will interact with a MySQL Database system using SQL queries to answer a question.
At each turn, you should first provide your step-by-step thinking for solving the task. Your thought process should start with "Thought: ", for example: Thought: I should write a SQL query that gets the average GNP and total population from nations whose government is US territory.

After that, you have two options:
1) Interact with a mysql programming environment and receive the corresponding output. Your code should start with "Action: " , for example: Action: SELECT AVG(GNP), SUM(population) FROM nations WHERE government = ‘US Territory’
2) Directly submit the result, for example: Action: submit.

You should use this format:
Thought: your thought
Action: <the mysql command>.

You will receive the corresponding output for your sql command. Your output should contain only one "Action" part. The "Action" part should be executed with a mysql interpreter or propose an answer. Any natural language in it should be commented out. The SQL query and submit parts can not appear in your output simultaneously.

Figure 8: Instruction prompt for InterCodeSQL.

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 a 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 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".

The available actions are:
1. go to recep
2. task 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.

Your response should use the following format:
Thought: <your thoughts>
Action: <your next action>

Figure 9: Instruction prompt for ALFWorld.