Trial and Error: Exploration-Based Trajectory Optimization of LLM Agents

Anonymous ACL submission

Abstract

Large language models (LLMs) have emerged as the core controller for various autonomous agent systems. In this work, we introduce ETO, a method aimed at enhancing the capabilities of open-source LLM agents. Unlike previous work that solely trains on success expert trajectories, our approach enables agents to learn from exploration failures, leading to improved performance through an iterative exploration-training framework. During the exploration phase, the agent explores the environment, collecting failure trajectories to construct contrastive trajectory pairs. In the training phase, the agent leverages the trajectory contrastive information to update its policy. This iterative process of exploration and training facilitates further improvement for the agents. Experiments on three agent datasets show our method consistently outperforms baselines by more than 5% in final rewards. Moreover, analysis of task-solving efficiency and the potential in scenarios without expert trajectory further highlights the effectiveness of our method.[1]

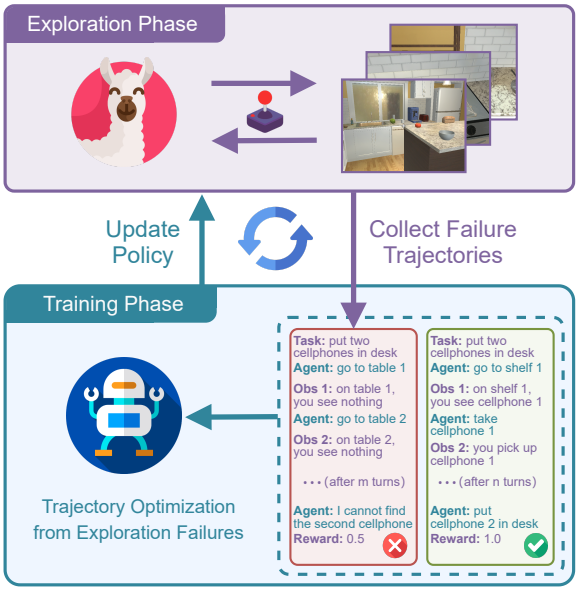

Figure 1: Exploration-based Trajectory Optimization (ETO) allows an LLM agent to iteratively collect failure trajectories and update its policy by learning from contrastive failure-success trajectory pairs.

1 Introduction

Large language model (LLM) agents leverage the powerful capabilities of LLMs to plan and interact with external tools or environments (Wang et al., 2023a; Xi et al., 2023). By utilizing ChatGPT (OpenAI, 2022) and GPT-4 (OpenAI, 2023) as the core controllers, various autonomous agent systems have been built to tackle complex interactive tasks, ranging from web browsing (Deng et al., 2023; Zhou et al., 2023), embodied house holding (Yao et al., 2022b), and even simulated science experiments (Lin et al., 2023). However, recent studies indicate that open-source LLMs lag significantly behind ChatGPT/GPT-4 in terms of agent capabilities (Liu et al., 2023; Wang et al., 2023d). Furthermore, even among the most advanced LLMs, performance still falls short of expectations (Zhou et al., 2023; Mialon et al., 2023)

To enhance the capabilities of LLM agents, one effective approach is through imitation learning. For example, behavioral cloning (BC) (Pomerleau, 1991) offers a straightforward method to acquire a policy by supervised learning on observation-action pairs from gold expert trajectories. Recently, there have been attempts (Chen et al., 2023; Zeng et al., 2023; Yin et al., 2023) to apply BC to open-source LLM-based agents by directly performing supervised fine-tuning (SFT) on expert trajectories. Taking a step further, Aksitov et al. (2023) refine the agent through iterative BC on success trajectories generated by the previous policy.

Existing research primarily concentrates on imitation learning from successful expert trajectories. However, relying solely on expert demonstrations

[1]The code is released at anonymous link.

for learning observation-action mappings can result in sub-optimal policies due to insufficient exploration beyond expert demonstrations. It is important to recognize that human learning encompasses not just successes but also failures. The underlying intuition is that humans can more effectively learn policies by exploring both successful and failed trajectories, contrasting successes with failures. Consequently, incorporating learning from failures into the training process holds the potential to further enhance the functionality of LLM agents.

Inspired by this, we take the spirit of learning by trial and error and propose an LLM agent learning paradigm called **Exploration-based Trajectory Optimization (ETO)**. Different from previous approaches which solely use success trajectories, our method leverages exploration failures of the current policy to facilitate the agent's learning process. Specifically, we first employ SFT behavioral cloning to establish a base agent. Subsequently, the agent undergoes further adaptation in an iterative exploration-training process. During the exploration phase, the base agent explores environments to gather new trajectories for tasks in the BC training data. Failed trajectories are then compared with expert trajectories, constructing contrastive trajectory pair data. In the subsequent training phase, we use the contrastive trajectory information and employ the DPO loss (Rafailov et al., 2023) to refine the LLM policy. The probabilities of success and failure trajectories are compared, and a constraint term is incorporated to ensure the agent maintains its basic policy. The exploration and training processes can be conducted iteratively, leading to further improvement of the LLM agents.

We evaluate our method on three representative datasets: WebShop (Yao et al., 2022a) for web navigation, ScienceWorld (Wang et al., 2022) for simulated science experiments, and ALFWorld (Shridhar et al., 2021) for embodied household tasks. Across these datasets, ETO consistently outperforms SFT behavioral cloning and other strong baselines by a large margin, demonstrating the effectiveness of learning from exploration failures. Moreover, our method shows a remarkable performance enhancement of 22% over SFT on the challenging out-of-domain test set in ScienceWorld, showcasing its strong generalization ability. The analysis also reveals the task-solving efficiency of our method, which achieves higher rewards with fewer action steps. Additionally, in extreme scenarios where expert trajectories are unavailable, ETO can still show impressive performance in a self-play manner, further highlighting the potential of our method.

To summarize, our contributions are as follows:

- We propose to enhance the capability of LLM agents by learning from exploration failures. To achieve this, we introduce ETO, a learning paradigm which iteratively collects failure trajectories and refines the agent policy through learning trajectory contrastive information.
- Extensive experiments on three distinct agent datasets show that our method outperforms SFT behavioral cloning and other strong baselines by a large margin.
- We conduct in-depth analysis to further validate the effectiveness of ETO, including the generalization on out-of-domain tasks, the task-solving efficiency, and the potential in scenarios without expert trajectory. The analysis further underscores the effectiveness of our method.

2 Task Formulation

The agent task with environment feedback can be formalized as a partially observable Markov decision process (POMDP) $(\mathcal{U}, \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R})$ with instruction space $\mathcal{U}$, state space $\mathcal{S}$, action space $\mathcal{A}$, observation space $\mathcal{O}$, transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$, and reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$. Note that in our LLM-based agent scenario, $\mathcal{U}, \mathcal{A}, \mathcal{O}$ are subsets of natural language space.

Given a task instruction $u \in \mathcal{U}$, the LLM agent with parameter θ generates the action $a_1 \sim \pi_\theta(\cdot|u) \in \mathcal{A}$ according to its policy π_θ. The action incurs a change in the latent state space $s_t \in \mathcal{S}$, and an execution feedback as observation $o_t \in \mathcal{O}$. Then the agent generates the corresponding action in the $t+1$ step $a_{t+1} \sim \pi_\theta(\cdot|u, a_1, o_1, ..., o_{t-1}, a_t) \in \mathcal{A}$. The interaction loop repeats until the task completes or exceeds the maximum steps, and the trajectory is denoted as:

$$e = (u, a_1, o_1, ..., o_{n-1}, a_n) \sim \pi_\theta(e|u), \quad (1)$$

$$\pi_\theta(e|u) = \prod_{j=1}^{n} \pi_\theta(a_j|u, a_1, o_1, ..., o_{j-1}), \quad (2)$$

where n is the trajectory length. Finally, the final reward $r(u, e) \in [0, 1]$ is computed, with 1 representing successful task completion.

3 Method

Our method, ETO, starts by training a base agent through behavioral cloning. Based on the base agent, our framework continually enhanced the policy from trial and error in an iterative manner.

3.1 Behavioral Cloning

Behavioral cloning (BC) has demonstrated promising results through supervised fine-tuning on the expert interaction trajectory data, serving as a solid starting point for building a powerful agent. In this work, we employ ReAct-style (Yao et al., 2022b) trajectory to conduct BC, which additionally generates Chain-of-Thought (CoT) rationales (Wei et al., 2022) before each action. Considering that the CoT and action are generated together in the ReAct framework, we use a to represent the action with CoT for simplicity.

Given an expert trajectory dataset $\mathcal{D} = \left\{(u, e)^{(i)}\right\}_{i=1}^{|\mathcal{D}|}$, where $|\mathcal{D}|$ is the number of trajectories, we fine-tune an LLM on auto-regressive loss to get the base agent π_{base}:

$$\mathcal{L}_{\text{SFT}}(\pi_\theta) = -\mathbb{E}_{e\sim\mathcal{D}}\left[\pi_\theta(e|u)\right], \tag{3}$$

where $e = (u, a_1, o_1, ...o_{n-1}, a_n) \sim \mathcal{D}$ is an expert interaction trajectory.

Since $\pi_\theta(e|u) = \prod_{j=1}^{n} \pi_\theta(a_j|u, ..., o_{j-1})$, in practice, we first concatenate the instruction, actions and observations in trajectory e as a text sequence t:

$$\begin{aligned} t &= \text{concat}(u, a_1, o_1, ..., o_{n-1}, a_n) \\ &= (t_1, t_2, ..., t_l), \end{aligned} \tag{4}$$

where t_k is the k-th token in the result sequence. Then the probability of the trajectory in Eq. (3) can be obtained by directly computing the probability of actions with observation tokens masked:

$$\pi_\theta(e|u) = -\sum_k \log \pi_\theta(t_j|t_{<j}) \times \mathbf{1}(t_j \in A), \tag{5}$$

where $\mathbf{1}(t_j \in A)$ is an indicator function about whether t_j is a token belonging to actions produced by the agent.

3.2 Learning From Exploration Failures

Behavioral cloning exclusively depends on expert trajectories and lacks the ability to explore the environment, leading to sub-optimal policies. To train a more powerful agent, it is important for the model to also explore failure trajectories. To achieve this, a viable approach is reinforcement learning, which empowers agents to actively explore the environment to get rewards and refine the policy through trial and error (Ouyang et al., 2022):

$$\begin{aligned} \max_{\pi_\theta} \mathbb{E}_{u\sim\mathcal{D}, e\sim\pi_\theta(e|u)}\left[r(u, e)\right] - \\ \beta \mathbb{D}_{\text{KL}}\left[\pi_\theta(e|u) \,||\, \pi_{\text{ref}}(e|u)\right], \end{aligned} \tag{6}$$

where the KL term with weighting parameter β controls the deviation from the base reference policy π_{ref}, *i.e.*, the base agent π_{base}. In practice, the agent to be trained π_θ is also initialized to π_{base}. Then the optimization problem in Eq. (6) can be solved via RL methods such as PPO (Schulman et al., 2017; Ouyang et al., 2022).

However, directly applying online RL on LLM agents will present practical challenges such as instability and low efficiency (Shen et al., 2023; Rafailov et al., 2023). Therefore, we instead design an iterative offline learning framework and train the agent with contrastive trajectory data. As shown in Algorithm 1, the training process can be formulated in an iterative exploration-training loop. In the exploration phase of ETO, the agent explores the environment to collect failure trajectories. During the training phase, the agent learns the contrastive information from the "failure-success" trajectory pairs to update the policy.

Exploration Phase In this phase, the base agent π_{base} explores the environment to get the trajectories on the instructions of training data for BC:

$$\hat{e} = (u, \hat{a}_1, \hat{o}_1, ..., \hat{o}_{m-1}, \hat{a}_m) \sim \pi_{\text{base}}(e|u). \tag{7}$$

The environments then return a reward $\hat{r}$ corresponding to the trajectory $\hat{e}$.

Then we construct failure-success trajectory pairs, denote as $e_w \succ e_l \mid u$, based on the final rewards. Here, e_w and e_l represent the trajectories with higher and lower rewards, chosen from the expert trajectory e and agent-generated trajectory $\hat{e}$ respectively. Note that we only collect pairs where two trajectories have different rewards. If both $\hat{e}$ and e successfully complete the task, we simply discard the pair. Finally, we get the contrastive trajectory dataset $\mathcal{D}_p = \left\{(u, e_w, e_l)^{(i)}\right\}_{i=1}^{|\mathcal{D}_p|}$.

Training Phase In this phase, the agent policy is updated by modeling the contrastive failure-success information in the trajectory pair data.

Algorithm 1: ETO

Input: $\mathcal{D} = \left\{(u, a_1, o_1, ...o_{n-1}, a_n)^{(i)}\right\}$: expert trajectory dataset for behavioral cloning, T_1: number of behavioral cloning steps, I: number of iterations for ETO, T_2: number of steps in training phase, π_θ: initial LLM policy.

Output: Final policy π_θ

```
// Behavioral cloning
```

for $i = 1$ **to** T_1 **do**

  Optimize θ on BC objective: $\mathcal{L}_{\text{SFT}}(\pi_\theta) = -\mathbb{E}_{e\sim\mathcal{D}}\left[\pi_\theta(e|u)\right]$

```
// Iteratively learning from exploration failures
```

for $i = 1$ **to** I **do**

  $\pi_{\text{base}} = \pi_\theta$; $\pi_{\text{ref}} = \pi_\theta$

  Get base agent trajectories on $\mathcal{D}$: $\hat{e} = (u, \hat{a}_1, \hat{o}_1, ..., \hat{o}_{m-1}, \hat{a}_m) \sim \pi_{\text{base}}(e|u)$

  Compare rewards of $\hat{e}$ with expert trajectory e to get the failure-success pair: $e_w \succ e_l \mid u$

  Construct contrastive trajectory dataset: $\mathcal{D}_p = \left\{(u, e_w, e_l)^{(i)}\right\}$

  for $j = 1$ **to** T_2 **do**

    Optimize θ on trajectory contrastive objective:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(u,e_w,e_l)\sim\mathcal{D}_p}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(e_w|u)}{\pi_\theta(e_l|u)} - \beta\log\frac{\pi_{\text{ref}}(e_w|u)}{\pi_{\text{ref}}(e_l|u)}\right)\right]$$

return π_θ

Given trajectory pair $e_w \succ e_l \mid u$, the failure-success relation can be modeled via Bradley-Terry (BT) (Bradley and Terry, 1952) model:

$$p(e_w \succ e_l|u) = \frac{\exp\left(r(u, e_w)\right)}{\exp\left(r(u, e_w)\right) + \exp\left(r(u, e_l)\right)}. \tag{8}$$

On the other hand, under the optimal policy $\pi_r(e|u)$ of Eq. (6), the reward function in the environment can be written as (Peng et al., 2019; Rafailov et al., 2023):

$$r(u, e) = \beta\log\frac{\pi_r(e|u)}{\pi_{\text{ref}}(e|u)} + \beta\log Z(x), \tag{9}$$

where $Z(u) = \sum_e \pi_{\text{ref}}(e|u)\exp\left(\frac{1}{\beta}r(u, e)\right)$ is the partition function. Substitute Eq. (9) into Eq. (8) to get the BT model over policy:

$$p(e_w \succ e_l|u) = \sigma\left(\beta\log\frac{\pi_\theta(e_w|u)}{\pi_\theta(e_l|u)} - \beta\log\frac{\pi_{\text{ref}}(e_w|u)}{\pi_{\text{ref}}(e_l|u)}\right), \tag{10}$$

where σ is the sigmoid function. Then the optimal policy π_θ can be achieved by applying maximum likelihood:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(u,e_w,e_l)\sim\mathcal{D}_p}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(e_w|u)}{\pi_\theta(e_l|u)} - \beta\log\frac{\pi_{\text{ref}}(e_w|u)}{\pi_{\text{ref}}(e_l|u)}\right)\right]. \tag{11}$$

This optimization objective aims to increase the likelihood of the success trajectories e_w and decrease the likelihood of failure trajectories e_l, with a constraint term to maintain the basic agent capabilities. Moreover, as a reformulation of RL objective Eq. (6), Eq. (11) is directly maximizing the final reward while avoiding the need to perform RL optimization.

Iteration To further improve the agent's performance, ETO adopts an iterative exploration-training manner. After the training phase, the agent policy can be used to gather new failure cases and create contrastive trajectory pairs. These new data are then used to further enhance the agent through trajectory contrastive learning.

4 Experiments

In this section, we conduct extensive experiments to validate the effectiveness of ETO. Our method demonstrates superior performance compared to baselines across three datasets, and it exhibits enhanced advantages when dealing with out-of-domain unseen tasks. The analysis further showcases the efficiency of our method. Furthermore, our method also holds promise in scenarios where expert trajectories are unavailable.

4.1 Experimental Settings

Datasets We conduct experiments on three representative agent datasets, WebShop (Yao et al., 2022a) for web navigation, ScienceWorld (Wang et al., 2022) for embodied science experiments, and ALFWorld (Shridhar et al., 2021) for embodied house holding tasks. Both WebShop and ScienceWorld environments provide dense final rewards ranging from 0 to 1, while ALFWorld only provides binary rewards indicating whether the task is completed. All three environments can be formally described as partially observable Markov decision processes. For details of the datasets and the expert trajectory collection process, please refer to Appendix A.

The statistical information of our datasets is presented in Table 1. It is important to mention that, in addition to the in-distribution test sets, both ScienceWorld and ALFWorld contain test sets that include out-of-distribution unseen variations. These additional test sets allow us to assess the generalization capabilities of different agents.

Training Setup We mainly use Llama-2-7B-Chat (Touvron et al., 2023) as the base model for building LLM agents. To provide more comprehensive results, we also conduct experiments on Llama-2-13B-Chat and Mistral-7B (Jiang et al., 2023). We utilize the AdamW optimizer. For the SFT phase, the batch size is 64 and the learning rate is set to 1e-5 with 3% warm up and a cosine scheduler. Then the base agent will explore once for each instance in the training set to collect failure trajectories. For the training phase of ETO, the batch size is 32 and the learning rate is set to 1e-6. The β in DPO loss is set to 0.1. The learning epochs of SFT phase and training phase in ETO are set to 3. The number of iterations of ETO is set to 2. All experiments are conducted on 8 NVIDIA A100 80G GPUs.

Baselines We compare ETO with SFT imitation learning and other post-imitation baseline methods. 1) SFT (Chen et al., 2023; Zeng et al., 2023) conducts imitation learning on expert trajectories, which is the base agent for ETO and other baselines. 2) Best-of-N sampling employs SFT base agent and selects the trajectory with the best reward within N samplings. Here we set N to 10. 3) RFT (Rejection sampling Fine-Tuning) (Yuan et al., 2023) is a strong baseline which adds the success trajectories to the expert trajectory dataset and trains the agent on new augmented trajectories. 4) PPO (Proximal Policy Optimization) (Schulman et al., 2017) is an RL method directly optimizing the SFT agents to maximize the final task reward. We also include GPT-3.5-Turbo (OpenAI, 2022), GPT-4 (OpenAI, 2023), and untuned Llama-2-7B-Chat for comparison.

Dataset	#Train	#Test-Seen	#Test-Unseen	#Turns
WebShop	1938	200	-	4.9
ScienceWorld	1483	194	241	14.4
ALFWorld	3321	140	134	10.1

Table 1: Statistics of datasets. "Test-Seen" and "Test-Unseen" are test set with seen and unseen scenarios respectively. "#Turns" denotes the average number of interaction turns for the expert trajectories.

Evaluation All methods are evaluated using the ReAct-style interaction format (Yao et al., 2022b), with CoT rationale generated before the action. See Appendix C for the detailed prompts. We add 1-shot in-context example in the instruction prompt for each task. The decoding temperature of the LLMs is set to be 0.0 for deterministic generation, except for Best-of-N method. We employ **Average Reward** as the metric, which represents the average reward of all task instances in the test set.

4.2 Results

Table 2 presents the performance comparison of ETO and baselines on three agent datasets. As shown, ETO demonstrates a significant improvement over SFT imitation learning, leading to an average reward increase of 8% and 9.5% for WebShop and ScienceWorld. Our method also outperforms all other baselines on all datasets. On WebShop dataset, ETO even outperforms GPT-4 on the average reward, showing the extraordinary performance of our method. Although the RFT method also exhibits improvement compared to SFT, its performance remains constrained as it is an augmented version of behavioral cloning and solely learns from success trajectories. This comparison indicates the comparison between failure and expert trajectories is essential to improve the performance of the agent. Meanwhile, though PPO gains improved performance on WebShop, it struggles to achieve satisfactory results on the other two datasets due to the inherent instability in RL optimization procedures, particularly on ALFWorld dataset which only provides binary final rewards.

Method	WebShop	ScienceWorld		ALFWorld	
		Seen	Unseen	Seen	Unseen
GPT-4	63.2	64.8	64.4	42.9	38.1
GPT-3.5-Turbo	62.4	16.5	13.0	7.9	10.5
Llama-2-7B-Chat	17.9	3.8	3.1	0.0	0.0
Llama-2-7B-Chat + SFT	63.1	67.4	53.0	60.0	67.2
Llama-2-7B-Chat + Best-of-N	63.8	70.2	57.6	62.1	69.4
Llama-2-7B-Chat + RFT	63.6	71.6	54.3	62.9	66.4
Llama-2-7B-Chat + PPO	64.2	59.4	51.7	22.1	29.1
Llama-2-7B-Chat + ETO (ours)	**67.4**	**73.8**	**65.0**	**68.6**	**72.4**

Table 2: The average reward of different methods on three agent datasets. "Seen" denotes the held-out test set with task types seen during training, while "Unseen" refers to the test set with critical unseen task variations.

Base LLM	Method	WebShop	ScienceWorld	
			Seen	Unseen
Llama-2-13B	SFT	66.3	68.1	57.6
	ETO	70.7	71.4	68.6
Mistral-7B	SFT	60.1	63.8	52.2
	ETO	66.2	68.5	62.5

Table 3: The average reward of different base LLMs on WebShop and ScienceWorld.

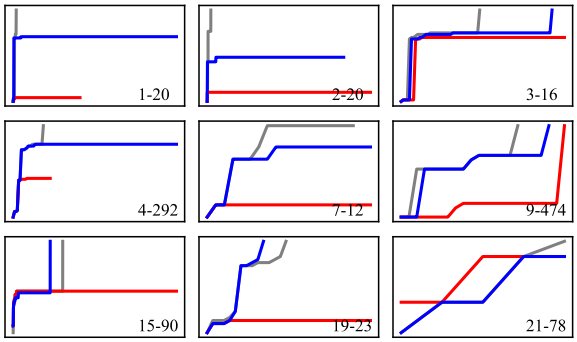

Figure 2: Cases of ScienceWorld reward trajectory for ETO, SFT Base Agent and Oracle. X: time steps $(0 \rightarrow T)$; Y: scores $(0 \rightarrow 100)$. Task IDs are shown at the bottom-right. Best viewed in color.

Notably, our approach showcases enhanced advantages in out-of-domain unseen scenarios, achieving an impressive performance boost of 20% on ScienceWorld-Unseen. Moreover, ETO exhibits strong effectiveness on the unseen scenarios in ALFWorld and outperforms the RFT and PPO baselines, both of which suffer from performance degradation. These results underscore that learning by trial and error can further enhance the agent's generalization capabilities, particularly in out-of-distribution unseen scenarios.

Results on Different LLMs To further demonstrate the effectiveness of our method, we present the results based on other base LLMs, including Llama-2-13B-Chat and Mistral-7B. Table 3 showcases the consistent improvement in agent performance achieved by ETO across different LLMs. Notably, when compared to Llama-2-7B, the 13B model displays a relatively smaller performance gain on both datasets, suggesting that our method can provide greater benefits to weaker agents. Despite Mistral-7B is a more powerful LLM than Llama-2-13B, it falls short of Llama-2-7B after either SFT or ETO. This finding indicates that there is not a strong correlation between the basic LLM capabilities and the agent capabilities.

Analysis on Efficiency We evaluate the task-solving efficiency of agents in ScienceWorld environment, which provides fine-grained subgoals for each task. The reward of a task is updated upon the accomplishment of a subgoal. By assessing the agent's ability to achieve higher rewards within fewer action steps, we can determine its efficiency. Figure 2 showcases the score trajectories of ScienceWorld-Seen test set, comparing ETO with the SFT base agent, and the oracle agent. As depicted, ETO can reach higher rewards in fewer action steps than the SFT base agent. Interestingly, in certain cases like 15-90 and 19-23, our method outperforms even the oracle agent, reaching a score of 100 earlier. These results demonstrate that by learning from failure trajectories, our method also acquires a more powerful task-solving efficiency.

4.3 Ablation of Iterations

In this section, we present a study on the impact of iteration numbers in ETO. The results are shown in Figure 3. As depicted, ETO demonstrates the

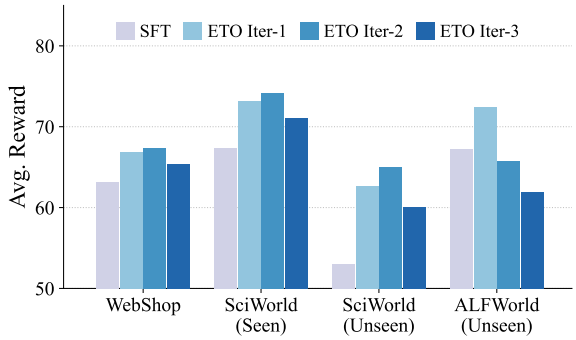

Figure 3: ETO performance on multiple iterations.

Method	Level	lr	β	Avg. Reward
SFT	-	-	-	63.1
ETO	Trajectory	1e-6	0.1	**67.4**
	Step	1e-6	0.1	8.3
	Step	1e-7	0.5	62.8
	Mixture	1e-6	0.1	64.3

Table 4: The average reward on WebShop of agents trained on different level contrastive data.

ability to enhance the performance of agents in the first two iterations on both the WebShop and ScienceWorld datasets. However, further increasing the iterations does not lead to continuous improvement. Instead, the performance starts to decline after the third iteration. Regarding the ALFWorld dataset, only the first iteration of ETO shows an improvement. Surprisingly, the performance on the second and third iterations even falls behind that of the SFT base agent.

To explain this, it is important to note that the learning process of ETO relies on a fixed expert trajectory set, and the exploration phase of the agent is performed on the same training set. Consequently, the diversity and quantity of failure-success contrastive trajectory data are constrained. Initially, the policy can be improved by learning from past mistakes. However, the model gets overfitting on the contrastive information in subsequent iterations, resulting in a decline in performance. In the case of ALFWorld, the coarse-grained binary rewards further hinder the agent from getting improvement from iterative training. As a potential solution, future work could explore the incorporation of GPT-4 to dynamically construct more diverse contrastive trajectory data.

4.4 Strategy of Contrastive Data Construction

In this section, we delve deeper into the contrastive trajectory pair construction strategy used in our method. In Section 3.2, we directly learn from failure-success trajectory pairs (Eq. (11)), referred to *trajectory-wise contrastive*. Alternatively, inspired by previous work (Lightman et al., 2023), we introduce a fine-grained variation of ETO that captures *step-wise contrastive* information by comparing "good-bad" action pairs. To achieve this, we use the expert trajectory to conduct teacher forcing for first $t-1$ steps, and then have the agent predict the action of t-th step. Then the quality of t-th action is determined by the final rewards. We also implement a *mixture* variation by combining the above two strategies. For further details regarding the step-wise variation of ETO, please refer to Appendix B.

The comparison of different methods is presented in Table 4. As the results demonstrate, trajectory-wise contrastive yields the best performance. On the other hand, we observed that step-wise contrastive modeling tends to be less stable, necessitating a lower learning rate and a higher constraint parameter β to maintain the basic capabilities of the agent. This instability may be attributed to the inaccurate estimation of the action quality, as we simply utilize the final rewards to construct step-wise contrastive pairs. Moreover, the performance of mixture strategy also falls short compared to trajectory-level contrastive.

4.5 Self-Play w/o Expert Trajectory

In this section, we explore a challenging scenario where no expert trajectory is available. In such cases, the agent is compelled to explore the environment and depend on self-play to enhance its capabilities. To achieve this, we eliminate the behavioral cloning phase of ETO and allow the LLM agent to explore the environments using a decoding temperature of 1.0. Subsequently, we compare different trajectories associated with the same instruction based on their final rewards, creating trajectory preference data. Finally, the agent is trained exclusively on the preference data generated by itself.

On WebShop, the untuned Llama-2-7B-Chat achieves relatively high rewards. Thus, we use this dataset to conduct the experiment. We also employ rejection sampling fine-tuning (RFT) as a baseline. The results in Table 5 show that ETO alone does not improve performance without behavioral cloning. In contrast, RFT shows promising ability to enhance the agent's capabilities without relying

Method	w/ BC?	Avg. Reward
SFT	✓	63.1
RFT	✓	63.6
ETO	✓	67.4
Llama-2-7B-Chat†	✗	17.9
RFT	✗	48.4
ETO	✗	12.5
RFT+ETO	✗	**51.2**

Table 5: Performance of self-play without behavioral cloning from expert trajectories. † means directly prompting an untuned Llama-2-7B-Chat.

on expert trajectories. However, when combining RFT with ETO, we observe a further enhancement in the agent's performance. These findings suggest that in scenarios without expert trajectories, it may be beneficial to first employ RFT and then allow the agent to learn from exploration failures. These results further highlight the potential of our method in scenarios where expert trajectories are unavailable.

5 Related Work

Imitation Learning Imitation learning is a learning paradigm where an agent learns a policy by mimicking expert demonstrations (Hussein et al., 2017; Fang et al., 2019). A prevalent approach in imitation learning is behavioral cloning (BC) (Pomerleau, 1991), which utilizes expert trajectories to learn a direct mapping from states to actions. There are various methods to mitigate the limitations of BC (Ross et al., 2011; Ross and Bagnell, 2014). Our method, ETO, shares a similar spirit with DAgger (Ross et al., 2011), an approach used to enhance the agent's performance by learning from failure cases. However, unlike DAgger which gathers additional expert trajectories on agent-failed cases, ETO improves the policy through learning from contrastive trajectory pairs.

LLM Agents With the various emergent abilities of LLMs, researchers have explored building agent systems based on LLMs (Xi et al., 2023). Recent projects such as AutoGPT (Richards, 2023), BabyAGI (Nakajima, 2023) have employed LLMs as core controllers, building powerful agent frameworks capable of solving realistic tasks. While GPTs have shown strong agent intelligence, open-source LLMs still lag far behind (Liu et al., 2023; Wang et al., 2023d). To bridge this gap, recent studies, including FireAct (Chen et al., 2023), Agent-Tuning (Zeng et al., 2023), and Lumos (Yin et al., 2023), construct expert trajectory data from teacher agents (*e.g.*, GPT-4) and perform BC on open-source LLMs. Taking a step further, Aksitov et al. (2023) refine the agent through iterative BC on success trajectories generated by the previous policy. Nonetheless, how to achieve agents with enhanced performance beyond simple behavioral cloning is still under-explore.

LLM Policy Learning Learning from preference has shown promise for learning an enhanced LLM policy, particularly in LLM alignment research. Reinforcement Learning from Human Feedback (RLHF) is a method that learns a reward model and then utilizes proximal policy optimization to update the policy model (Christiano et al., 2017; Ouyang et al., 2022). Despite its attractive advantages, RLHF presents limitations regarding training efficiency and instability. To address these issues, Rafailov et al. (2023) reformulate the optimization objective of RLHF, introducing the DPO loss to directly model preferences. Similar to our work, ReST (Gulcehre et al., 2023) iteratively generates new samples from the current policy and refines the policy using offline RL methods. Recent studies have explored the application of LLM policy learning in other domains (Lightman et al., 2023; Wang et al., 2023b). For example, Wang et al. (2023c) train a step-wise reward model to improve the performance of LLMs in mathematical reasoning.

6 Conclusion

In this work, we present ETO, a method aimed at enhancing the capabilities of LLM agents. Our approach allows the agent to learn by trial and error, thereby improving the performance of the base agent acquired through behavioral cloning. ETO uses an exploration-training iteration framework. During the exploration phase, the agent explores the environment, gathering failure trajectories and constructing trajectory preference pairs. Subsequently, in the training phase, the agent learns from the preference information using DPO loss. This iterative process of exploration and training enables further improvement in the agent's performance. Extensive experiments on three agent datasets demonstrate our method outperforms behavioral cloning and strong baselines by a large margin. Moreover, our method exhibits remarkable efficiency and shows great potential in scenarios where expert trajectories are unavailable.

Limitations

Our method, ETO, demonstrates effective learning of powerful LLM agents through trial and error. However, it is important to acknowledge several limitations of this work. 1) ETO simplifies the comparison of failure-success trajectories by assuming that the agent generates wrong actions right from the beginning. However, in realistic cases, the agent may start executing incorrect actions from some intermediate step. If we can identify when the agent makes a bad action (*e.g.*, $\hat{a}_3$ at 3-th step), we should then collect the expert trajectory for the remaining actions $a_{t>3}$. Unfortunately, most current environments do not contain such information, making it challenging to conduct action-wise or process-level reward modeling. A potential solution is to employ GPT-4 to identify the bad action and construct fine-grained contrastive trajectory data. 2) This work primarily focuses on developing specialized LLM agents for a specific agent task, with limited exploration into the construction of strong generalized agents. For future work, we will investigate the transferability of the policies trained by ETO and try to apply our method in a multi-task training scenario.

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

Renat Aksitov, Sobhan Miryoosefi, Zonglin Li, Daliang Li, Sheila Babayan, Kavya Kopparapu, Zachary Fisher, Ruiqi Guo, Sushant Prakash, Pranesh Srinivasan, et al. 2023. Rest meets react: Self-improvement for multi-step reasoning llm agent. *ArXiv preprint*, abs/2312.10003.

Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023. Fireact: Toward language agent fine-tuning. *ArXiv preprint*, abs/2310.05915.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pages 4299–4307.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*.

Bin Fang, Shidong Jia, Di Guo, Muhua Xu, Shuhuan Wen, and Fuchun Sun. 2019. Survey of imitation learning for robotic manipulation. *International Journal of Intelligent Robotics and Applications*, 3:362–369.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. 2023. Reinforced self-training (rest) for language modeling. *ArXiv preprint*, abs/2308.08998.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. 2017. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35.

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. 2023. Mistral 7b. *ArXiv preprint*, abs/2310.06825.

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*, abs/2305.20050.

Bill Yuchen Lin, Yicheng Fu, Karina Yang, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Prithviraj Ammanabrolu, Yejin Choi, and Xiang Ren. 2023. Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks. *Advances in Neural Information Processing Systems*.

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. In *The Twelfth International Conference on Learning Representations*.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. Gaia: a benchmark for general ai assistants. *ArXiv preprint*, abs/2311.12983.

Yohei Nakajima. 2023. Babyagi.

OpenAI. 2022. Introducing chatgpt.

OpenAI. 2023. Gpt-4 technical report. *arXiv*, pages 2303–08774.

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.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. 2019. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *ArXiv preprint*, abs/1910.00177.

Dean A Pomerleau. 1991. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *ArXiv preprint*, abs/2305.18290.

Toran Bruce Richards. 2023. Auto-gpt: An autonomous gpt-4 experiment.

Stephane Ross and J Andrew Bagnell. 2014. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *ArXiv preprint*, abs/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*, abs/2309.15025.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10737–10746. IEEE.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. 2021. Alfworld: Aligning text and embodied environments for interactive learning. In *9th International Conference on Learning Representations*.

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*, abs/2307.09288.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2023a. A survey on large language model based autonomous agents. *ArXiv preprint*, abs/2308.11432.

Peiyi Wang, Lei Li, Liang Chen, Feifan Song, Binghuai Lin, Yunbo Cao, Tianyu Liu, and Zhifang Sui. 2023b. Making large language models better reasoners with alignment. *ArXiv preprint*, abs/2309.02144.

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

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, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. 2023d. Mint: Evaluating llms in multi-turn interaction with tools and language feedback. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. 2023. The rise and potential of large language model based agents: A survey. *ArXiv preprint*, abs/2309.07864.

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. In *The Eleventh International Conference on Learning Representations*.

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*, abs/2311.05657.

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*, abs/2308.01825.

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*, abs/2310.12823.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. 2023. Webarena: A realistic web environment for building autonomous agents. *ArXiv preprint*, abs/2307.13854.

A Datasets

WebShop WebShop (Yao et al., 2022a) is an online shopping website environment where agents navigate the platform to make purchases based on user instructions. Once the agent selects the "buy" action, the environment provides a final reward, which is calculated based on the matching heuristics of the product's attributes and price.

ScienceWorld ScienceWorld (Wang et al., 2022) is a text-based virtual environment centered around accomplishing elementary science experiments, including 10 different task types such as thermodynamics and electrical circuits. The agents need to be grounded in embodied interactive environments to engage with and comprehend scientific concepts through practical experience. Each task in ScienceWorld includes several optional subgoals, and the overall final reward is computed based on the achievement of these subgoals.

The original test set in ScienceWorld consists of critical unseen task variations. For instance, in the training set, the task may involve boiling water, whereas in the test set, the task is to boil lead. Consequently, we employ the original test set to evaluate our model's generalization performance on unseen scenarios. We utilize the original development set as our test set with seen scenarios. We exclude Task-9 and Task-10 due to their excessively long task-solving trajectories. Following Lin et al. (2023), we use the first 10 instances for task types with more than 10 test variations for fair and cost-effective comparisons.

ALFWorld ALFWorld (Shridhar et al., 2021) consists of interactive TextWorld environments that parallel embodied worlds in the ALFRED (Shridhar et al., 2020) dataset. In this environment, agents are required to explore and complete high-level house-holding instructions. 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.

CoT Annotation Webshop and ALFWorld provide a few human-annotated trajectories for imitation learning. We also employ GPT-4 as the teacher agent to explore in the WebShop environment and select trajectories which have a reward greater than 0.7. ScienceWorld environment provides heuristic searching algorithms to generate golden trajectories for each sub-task. Since the original trajectories do not contain CoT information for each action step, we further utilize GPT-4 to generate the corresponding rationales.

B Details for Step-Wise Contrastive

We implement a variation of ETO which learns from contrastive good-bad action pairs. Specifically, for a task instruction u with expert trajectory $e = (u, a_1, ..., o_{n-1}, a_n)$, we utilize teacher forcing for the first $t-1$ steps $(a_1, o_1, ...a_{t-1}, o_{t-1})$, and let the agent predict the actions from t-th step to get the trajectory:

$$\hat{e} = (u, a_1, o_1, ..., o_{t-1}, \hat{a}_t, \hat{o}_t, ..., \hat{o}_{m-1}, \hat{a}_m) \tag{12}$$

The environments return a reward $\hat{r}$ for the trajectory $\hat{e}$. If we denote the golden trajectory for the first $t-1$ steps as $e_{(t-1)}$, then the good-bad action pairs $a_w \succ a_l \mid u, e_{(t-1)}$ is constructed based on the final rewards. Here, a_w and a_l represent the actions with higher and lower final rewards, chosen from $(a_t, \hat{a}_t)$ respectively. Then the contrastive relation of the action pair can also be utilized in DPO loss to improve the policy:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(a_w|u, e_{(t-1)})}{\pi_\theta(a_l|u, e_{(t-1)})} - \beta\log\frac{\pi_{\text{ref}}(a_w|u, e_{(t-1)})}{\pi_{\text{ref}}(a_l|u, e_{(t-1)})}\right)\right]. \tag{13}$$

C Prompt for Evaluation

We show the instruction prompts for WebShop, ScienceWorld, ALFWorld in this section.

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]

Instruction Prompt for ScienceWorld

You are a helpful assistant to do some scientific experiments 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.

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

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

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>